Estimating Functional Single Index Models with Compact Support

Barinder Thind

Simon Fraser University

STAT 843

February 11, 2019

Overview

1 Introduction

2 Functional Linear Models

3 Functional Single Index Models

4 FuSIM - Compact Support

5 Conclusions

Preamble

- For completeness, an introduction is given to the basics of FDA
- Then enter: functional linear models
- Next, the functional single index model is introduced
- Then, a short break
- Finally, the single index model with compact support is presented!

Section I

Introduction

What is Functional Data?

- You have some observations, perhaps temperature data
  - ⋆ Traditionally, you look at them as though they were discrete
  - ⋆ What if there was an underlying data generating function?

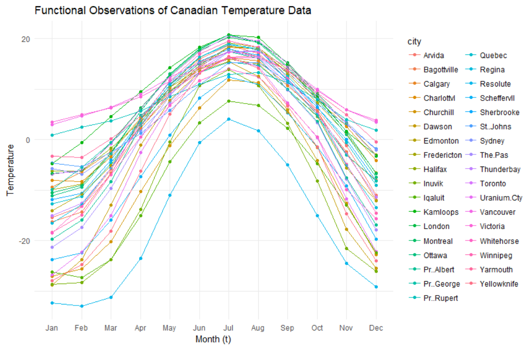


Figure 1: Temperature Data

What is Functional Data? Cont. (1)

- Here are some questions:
  - ⋆ How do you estimate these functions?
  - ⋆ Why do we smooth these functions?
  - ⋆ What models can we build?
- Essentially, we can always establish that there is a paradigm for how we treat collections of data and moreover, successfully argue against that paradigm but what advantage does FDA give? Where is it useful?

What is Functional Data? Cont. (2)

- And, here are some answers:
  - ⋆ There are multiple ways to estimate these functions. A derivation of the penalized least squares approach is provided later
  - ⋆ Smoothing allows us to work with derivatives which can reveal information from the data that may not be obvious from a regular multivariate analysis
  - ⋆ We can build models that help us see how some response is affected over a continuum like time!

Basis Functions

- Basis functions are used to represent functions
  - ⋆ Functional data is also observed and recorded as a pair: (t_j, y_j) - this can be seen as a snapshot of the function at some time t

$$x_i = (t_{ij}, y_{ij}) = \begin{bmatrix} (t_{11}, y_{11}) & (t_{12}, y_{12}) & \dots & (t_{1n}, y_{1n}) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ (t_{m1}, y_{m1}) & \dots & \dots & (t_{mn}, y_{mn}) \end{bmatrix}$$

- Each row can be thought of as a separate functional observation to be estimated.

Basis Functions Cont. (1)

- Different types of basis functions:
  - ⋆ Use Fourier basis for periodic data
  - ⋆ Use b-spline basis for non-periodic data
- Dr. Cao went over this!
- Functions represented as:

$$\begin{aligned} x(t) &= \sum_{k=1}^{K} c_k \phi_k(t) \\ &= c_1\phi_1(t) + c_2\phi_2(t) + \ldots + c_K\phi_K(t) \\ &= c^T\phi \end{aligned}$$

Basis Functions Cont. (2)

- We are essentially representing the infinite-dimensional $x(t)$ in a finite space as this basis
  - ⋆ When $K = n$, we get an exact interpolation (That is: $x(t_j) = y_j$ and K being the number of basis functions)
  - ⋆ With a smaller K and a better choice in basis, we are given the privilege of more degrees of freedom and more computational efficiency
- But how do we calculate c^T, the vector of coefficients?

Estimating Coefficients

- One approach is to use least squares:

$$\hat{c} = (\Phi^T \Phi)^{-1} \Phi^T y$$

  - ⋆ Where Φ is the n by K matrix containing our K basis functions each evaluated at the values of t

- Therefore, the functional observation fitted value estimation becomes (essentially, these are points along the curve):

$$\hat{y} = \Phi(\Phi^T \Phi)^{-1} \Phi^T y$$

  - ⋆ Assuming the original model was: $y_j = \sum_k c_k \phi_k(t_j)$

Estimating Coefficients Cont. (1)

- Assumption violation issues using that approach, so we can use a weighted approach
  - ⋆ The assumption violated being that the there is autocorrelated errors
- The estimate then becomes:

$$\hat{c} = (\Phi^T \mathbf{W} \Phi)^{-1} \Phi^T \mathbf{W} y$$

- Where **W** is symmetric positive definite matrix that allows for unequal weighting of squares
- **W** can be the inverse of the variance-covariance matrix of the residuals if it is known

Smoothing

- While the above methods have their merits, they have issues with discontinuity control
  - ⋆ We want smoothing to be more explicit
  - ⋆ Hence, we make it explicit in the least squares approach
  - ⋆ That is, we add a roughness penalty
- Previously, we would control smoothing by the number of basis functions picked = implicit!

Smoothing Cont. (1)

- Remember the bias-variance trade-off?
  - ⋆ In the usual least squares coefficient estimate, we can see how well it does based on its MSE
  - ⋆ A high bias means that the curve deviates from the observed underlying data greatly
  - ⋆ On the other hand, the smaller the bias, the higher the risk of overfitting
- This trade-off was seen implicitly in previous methodology - in smoothing approaches, it is made explicit
- How though?

Smoothing Cont. (2)

- Remember curvature? It's defined as:

$$\text{curvature} = [D^2 x(s)]^2$$

- This is one good measure because... imagine a line - perfectly smooth which implies no curvature
- Turns out, the second derivative of a line is 0!
- One measure of curvature we can use is defined as:

$$\text{PEN}_2(x) = \int [D^2 x(s)]^2 ds$$

Smoothing Cont. (3)

- Now, we can redefine the least squares from earlier in this context as follows:

$$\text{PENSSE}_\lambda(x|y) = [\mathbf{y} - x(t)]^T \mathbf{W}[\mathbf{y} - x(t)] + \lambda \cdot \text{PEN}_2(x)$$

- We minimize the above by finding a function $x(t)$ from the space of functions for which the penalty term is defined
- As $\lambda \to 0$, we get closer and closer to an interpolation
- On the other hand, as $\lambda \to \infty$, we get closer and closer to the standard regression model (i.e. a line) because non-linear functions incur a bigger roughness penalty

Smoothing Cont. (4)

- Aside: assume we make no assumptions about $x(t)$ other than that it has a second derivative
  - ⋆ Theorem in de Boor (2001) shows that the curve which minimizes the above criterion is a cubic spline!

Smoothing Cont. (5) - Derivation of Smoothing Spline Estimator

- Let's now re-express the penalization term as follows:

$$\begin{aligned} PEN_m(x) &= \int [D^m x(s)]^2 ds \\ &= \int [D^m c' \phi(s)]^2 ds \\ &= c' \int [D^m \phi(s) D^m \phi'(s) ds] c \\ &= c' \mathbf{R} c \end{aligned}$$

- Then, the minimization criteria becomes:

$$\text{PENSSE}_\lambda(x|y) = [\mathbf{y} - x(t)]^T \mathbf{W} [\mathbf{y} - x(t)] + \lambda \cdot c' \mathbf{R} c$$

- Where $x(t)$ can be expressed as Φc

Smoothing Cont. (6) - Derivation of Smoothing Spline Estimator

- Taking the derivative with respect to our parameter c and setting to 0 yields:

$$-2\Phi^{'}\mathbf{W}y + \Phi^{'}W\Phi c + \lambda\mathbf{R}c = 0$$

- And therefore, the estimate for c becomes:

$$\hat{c} = (\Phi^{T}\mathbf{W}\Phi + \lambda\mathbf{R})^{-1}\Phi^{T}\mathbf{W}y$$

- Now we know what is functional data, how its "made", and how to control its roughness - we are now ready to build models!

Functional Linear Models

Methodology

Functional Linear Model

- Now that we have our functional data, what do we do with it?
  - ⋆ We can build models!
- Three types of functional linear models:
  - ⋆ Scalar response, functional covariate
  - ⋆ Functional response, scalar covariate
  - ⋆ Functional response, functional covariate
- This talk focuses on the first because that's the context under which single index models are made

Functional Linear Model Cont. (1)

- The general functional linear model can be written as:

$$E(Y|X) = \beta_0 + \int_0^T \beta(t)x_i(t)dt$$

- Here, β_0 is the intercept term and you could think of $\beta(t)$ as a function which generates the coefficients associated with the covariate at any point from 0 to T
- $X(t)$ is the functional data used to predict the scalar response Y
- How do we estimate $\beta(t)$?

Functional Linear Model Cont. (2)

- We can estimate $\beta(t)$ by minimizing error:

$$\beta(t) = \text{argmin} \sum \left(y_i - \alpha - \int \beta(t) x_i(t) dt \right)^2$$

- Results in very volatile estimates for $\beta(t)$
- Can always get the exact solution (overfit, infinitely many solutions, identifiability issues)
- Constraint $\beta(t)$ (similar idea to smoothing) i.e. minimize the following:

$$\text{PENSSE}_\lambda(\beta) = \sum_{i=1}^{n} \left(y_i - \alpha - \int \beta(t) x_i(t) dt \right)^2 + \lambda \int [L\beta(t)]^2 \, dt$$

Functional Linear Model Cont. (3)

- We still represent $\beta(t)$ as basis functions:

$$\beta(t) = \sum c_i \phi_i(t)$$

- Now, let $Z_i = [1, \mathbf{x}_i]$ and $\mathbf{x}_i = \int \Phi(t) x_i(t) dt$
- Then, we can rewrite the model as:

$$\mathbf{y} = \mathbf{Z} \left[\alpha \quad \mathbf{c}\right]^T + \epsilon$$

- Essentially then, the vector in the above equation is what we need to estimate. Under the smoothing conditions, the estimate is:

$$[\hat{\alpha} \; \hat{\mathbf{c}}^T]^T = \left(Z^T Z + \lambda R_L\right)^{-1} Z^T \mathbf{y}$$

Functional Linear Model Cont. (4)

- Finally, this means that our predictions for the scalar response become:

$$\hat{\mathbf{y}} = \int \hat{\beta}(t) x_i(t) dt = \mathbf{Z} \left[\hat{\alpha} \quad \hat{\mathbf{c}} \right]^T$$

- Again, smoothing parameter, λ can be chosen on the minimization of the MSE
- Now prepared to move onto functional single index models!

Functional Single Index Models

Methodology

Functional Single Index Model

- A single index model is some conditional density or mean $E(y|x)$ that depends on some linear combination $x^T\beta$ through a function $g(.)$

$$E(Y|X = x) = g(x^T\beta)$$

- $x^T\beta$ is called an index
- Problem: estimate β and G from observations of (Y, X)
- Useful model in many contexts (for example, wage problem)

Functional Single Index Model Cont. (1)

- Semi-parametric method
- A single-index model reduces the risk of misspecification relative to a parametric models
- Avoids some drawbacks of fully non-parametric methods such as the curse of dimensionality, difficulty of interpretation, and lack of extrapolation capability

Functional Single Index Model Cont. (2)

- Some issues - need restrictions on $g(.)$ and $\beta(x)$ otherwise model is not identifiable
- Consider the example where $X = (X_1, X_2)$. This is a two dimensional problem.
- Let the support be: (0, 0), (1, 0), (0, 1), and (1, 1)
- Let Y be value of interest, say $Y = [0, 0.1, 0.3, 0.4]$
- The single index model then becomes:

$$E(Y|X = x) = g(x_1 + \beta_2 x_2)$$

- Suppose the values after evaluating $g(.)$ are: $[a, b, c, d]$
- Need to find $g(.)$ such that $[a, b, c, d]$ are equal to $[0, 0.1, 0.3, 0.4]$
- Many solutions! Need to set restrictions

Functional Single Index Model Cont. (3)

- β is not identified if $g(.)$ is a constant function
- β is not identified if the data exhibits multicollinearity
- $g(.)$ needs to be differentiable

Functional Single Index Model Cont. (4)

- If we have some estimate of β...
  - ⋆ We can estimate say, $G_n(z)$ using kernel estimators
  - ⋆ Let $Z_{ni} = X_i^T b_n$ where b_n is an estimate of β, then:

$$G_n(z) = \frac{1}{nh_n p_n(z)} \sum_{i=1}^{n} Y_i K\left(\frac{z - Z_{ni}}{h_n}\right),$$

  - ⋆ Where:

$$p_n(z) = \frac{1}{nh_n} \sum_{i=1}^{n} K\left(\frac{z - Z_{ni}}{h_n}\right).$$

Functional Single Index Model Cont. (5)

- The functional single index (FuSIM) model is an extension to the usual functional linear model in that it asserts that the scalar response Y is related to the functional covariate $x(t)$ through some function $g(.)$
- The model is defined as:

$$E(Y|X) = g(\int_0^T \beta(t)X(t)dt),$$

- This approach helps model non-linear relationships

Functional Single Index Model Cont. (6)

- If we assume some parametric form on $g(.)$, we can get back familiar models!
  - ⋆ Logit Function = Generalized Functional Linear Model
  - ⋆ Identity Function = Functional Linear Model
- In order to estimate the link function $g(.)$ and the index function $\beta(t)$, we use least squares in conjunction with local-linear smoothing

FuSIM - Compact Support

Methodology

FuSIM [Compact Support]

- What if we wanted our $\beta(t)$ to be non-zero only when $x(t)$ specifically relates to the scalar response Y?
- Enter: Functional Single Index Models with Compact Support

FuSIM [Compact Support] Cont. (1)

- The new model now becomes as follows:

$$E(Y|X) = g(\int_S \beta(t)X(t)dt)$$

- The main difference: S is a different domain than the usual domain which spans from 0 to T.
    - ⋆ More specifically, $S \in [0, T]$
    - ⋆ Which is to say S is some subregion of the domain of the functional covariate

FuSIM [Compact Support] Cont. (2)

- However, a couple of problems:
  - ⋆ S is unknown
  - ⋆ No explicit claim on the link function, g
- The goal of this paper:
  - ⋆ Simultaneously estimate the link function g, the index function $\beta(t)$, and the support region S on which $X(t)$ is related to Y

FuSIM [Compact Support] Cont. (3)

- In order to do this, the authors organized the paper as follows (this will also be the direction for the rest of the talk)
  - ⋆ First, more mathematical details are provided on the specifics associated with the proposed model
  - ⋆ Next, an efficient algorithm is presented that estimates the required unknowns
  - ⋆ Lastly, a simulation is provided that shows this approaches competency

FuSIM [Compact Support] Cont. (4) - General Estimation

- Let $y_i, i = 1, 2, ..., n$ be the scalar response
- $x_i(t)$ is the corresponding functional predictor
- Then, the model is estimated by minimizing the following:

$$Q(\beta, g) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - g(\int_0^T \beta(t) x_i(t) dt) \right)^2 + \textbf{PEN}_\lambda(\beta)$$

- Note: Let the norm of $\beta(t) = 1$ for identifiability purposes

FuSIM [Compact Support] Cont. (5)

- In order to obtain a compact supported estimator for $\beta(t)$, we add a second term which penalizes the L_1 norm of the index function $\beta(t)$
- The tuning parameter λ controls the compactness of the resulting $\beta(t)$
  - ⋆ A large value of λ will shrink the magnitude of $\beta(t)$ towards zero in some subintervals
  - ⋆ The union of the non-zero subintervals will be S
  - ⋆ When $\lambda = 0$, we get the usual FuSIM model

FuSIM [Compact Support] Cont. (6)

- To be more explicit, the penalty term penalizes the L_1 norm of $\beta(t)$ to obtain a compact supported estimate
  - ⋆ This is done through the implementation of the SCAD method
  - ⋆ This finds a locally sparse estimator for the coefficient function in functional linear regression models.
  - ⋆ The nice shrinkage property of functional SCAD allows the proposed estimator to locate null subregions of the coefficient function without over shrinking nonzero values of the coefficient functions.

FuSIM [Compact Support] Cont. (7) - SCAD Penalty Term Definition

- Explicity, the SCAD penalty term is defined as:

$$\mathrm{PEN}_\lambda(\beta) = \frac{1}{T}\int_0^T p_\lambda(|\beta(t)|)dt,$$

- With $p_\lambda(.)$ as:

$$p_\lambda(u) = \begin{cases} \lambda u & \text{if } 0 \leq u \leq \lambda, \\ -\frac{u^2 - 2a\lambda u + \lambda^2}{2(a-1)} & \text{if } \lambda < u < a\lambda, \\ \frac{(a+1)\lambda^2}{2} & \text{if } u \geq a\lambda, \end{cases}$$

FuSIM [Compact Support] Cont. (8)

- Note: *a* is suggested to be 3.7 by *Fan and Li (2001)*
- λ = tuning parameter = sparsity parameter
  - ⋆ Which is to say, the larger the value of λ, the more sparse the space within $[0, T]$ which is non-zero for the estimator, $\beta(t)$
- Now, we turn our attention to the general algorithm...

FuSIM [Compact Support] Cont. (9) - General Algorithm

- We provide the sparsity parameter, λ, then the algorithm proceeds as follows:
  - ⋆ Step I: Set some initial value for $\beta(t)$, denote as $\beta^{(0)}(t)$
  - ⋆ Step II: Given the current $\beta^{(j)}(t)$, we estimate $g^{(j)}(.)$ by minimizing:

$$Q_1(g|\beta^{(j)}) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - g(\int_0^T \beta^{(j)}(t)x_i(t)dt)\right)^2$$

- ...

FuSIM [Compact Support] Cont. (10) - General Algorithm

- ...
  - ⋆ Step III: Given the current value of $g^{(j)}(.)$, we update the estimate of $\beta(t)$ to $\beta^{(j+1)}(t)$ by minimizing:

$$Q_2(\beta|g^{(j)}) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - g^{(j)}\left(\int_0^T \beta(t)x_i(t)dt\right)\right)^2$$

$$+ \mathbf{PEN}_\lambda(\beta)$$

- Which, if you recall is the original minimization criteria
- Now, the discussion turns to steps II and III of this algorithm

FuSIM [Compact Support] Cont. (11) - Algorithm Step II

- For step II of the algorithm, we need to find the link function, $g(.)$ that minimizes Q_1
  - ⋆ Local linear regression estimator is used
  - ⋆ This is given as:

$$\hat{g}(z) = \sum_{i=1}^{n} [\hat{m}_0 - \hat{m}_1(z_i - z)] K_h(z_i - z),$$

FuSIM [Compact Support] Cont. (12) - Algorithm Step II

- ... where $\hat{m}_0$ and $\hat{m}_1$ is obtained by minimizing the following:

$$\frac{1}{n}\sum_{i=1}^{n}\left([y_i - m_0 - m_1(z_i - z)]K_h(z_i - z)\right)^2$$

- Where $K_h(.)$ is a kernel function with some bandwidth h
- This bandwidth is selected using cross-validation
- Note: the resulting $g(.)$ is a non-linear function

FuSIM [Compact Support] Cont. (13) - Algorithm Step III

- Now, we turn our attention to third step - estimating the index function, $\beta(t)$
- Let $\hat{g}(.)$ be the current estimate, then step III becomes:

$$Q_2(\beta|\hat{g}) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{g}(\int_0^T \beta(t)x_i(t)dt)\right)^2 + \mathbf{PEN}_\lambda(\beta)$$

- Let $\beta(t) = \mathbf{b}^T\mathbf{B}(t)$ where $\mathbf{B}(t) = (B_1(t), ..., B_L(t))^T$ are basis functions and $\mathbf{b}$ is the vector of basis coefficients
- Use b-splines as the basis functions due to computational efficiency

FuSIM [Compact Support] Cont. (14) - Algorithm Step III

- Using this approximation for $\beta(t)$, the least squares term can be rewritten as:

$$\int_0^T \beta(t)x_i(t)dt = \mathbf{b}^T \int_0^T \mathbf{B}(t)x_i(t)dt = \mathbf{Z}_i^T\mathbf{b},$$

- $g(.)$ is non-linear and because the above term is nested in the link function, this is a non-linear estimation problem
- Iterative approach used

FuSIM [Compact Support] Cont. (15) - Algorithm Step III

- Letting $\beta^{(j)}(t) = \mathbf{b}^{(j)}\mathbf{B}(t)$ be the current estimate in the j^{th} iteration. The least squares term is approximated as follows:

$$
\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{g}(\int_0^T \beta(t)x_i(t)dt)\right)^2 \\
\approx&\frac{1}{n}\sum_{i=1}^{n}\left(\left[y_i - \hat{g}(\mathbf{Z}_i^T\mathbf{b}^{(j)}) - \hat{g}'(\mathbf{Z}_i^T\mathbf{b}^{(j)})\right]\left[\mathbf{b}^T\mathbf{Z}_i - \mathbf{Z}_i^T\mathbf{b}^{(j)}\right]\right)^2 \\
=&(\mathbf{b}^{(j)} - \mathbf{b})^T\mathbf{G}^{(j)}(\mathbf{b}^{(j)} - \mathbf{b}),
\end{aligned}
$$

- Where $\mathbf{G}^{(j)} = \frac{1}{n}\sum_{i=1}^{n}(y_i^{(j)})^2\mathbf{Z}_i^T\mathbf{Z}_i$ and $y_i^{(j)} = y_i - \hat{g}(\mathbf{Z}_i^T\mathbf{b}^{(j)}$ - $\hat{g}'(\mathbf{Z}_i^T\mathbf{b}^{(j)})$

FuSIM [Compact Support] Cont. (16) - Algorithm Step III

- In order to estimate the penalty term (SCAD), we can use a local quadratic approximation (method proposed by *Fan and Li (2001)*):

$$\frac{1}{T}\int p_\lambda(|\beta(t)|)dt \approx \frac{1}{L-d}\sum_{\ell=1}^{L-d} p_\lambda\left(\sqrt{\int_{t_{\ell-1}}^{t_\ell} \beta^2(t)dt}\right),$$

- Given some current estimate of $\beta(t)$, $\mathbf{t}$ represents the sequence of knots as defined by the b-spline basis, and d represents the order

FuSIM [Compact Support] Cont. (17) - Algorithm Step III

- Okay, so this gets very mathy from here on out...
- Just know that the function Q_2 simplifies to:

$$
\begin{aligned}
Q_2(\mathbf{b}) \approx & (\mathbf{b}^{(j)} - \mathbf{b})^T \mathbf{G}^{(j)} (\mathbf{b}^{(j)} - \mathbf{b}) \\
& + \frac{1}{L-d} \left[\mathbf{b}^T \boldsymbol{W}^{(j)} \mathbf{b} + C(\mathbf{b}^{(j)}) \right] + \gamma \mathbf{b}^T \Gamma \mathbf{b}.
\end{aligned}
$$

- Where **W** and **C** are some complicated expressions.
- Recall that $\beta(t) = \mathbf{b}^T \mathbf{B}(t)$
- Taking the derivative with respect to **b** of Q_2 and equating to 0 gives us the minimizing expression for **b** denoted as $\hat{\mathbf{b}}$

FuSIM [Compact Support] Cont. (18) - Algorithm Step III

- The derivative is:

$$\frac{\partial Q_2(\mathbf{b})}{\partial \mathbf{b}} = -2\mathbf{G}^{(j)}(\mathbf{b}^{(j)} - \mathbf{b}) + 2\frac{1}{L-d}\boldsymbol{W}^{(j)}\mathbf{b} + 2\gamma\Gamma\mathbf{b}.$$

- And hence, the **b** estimate is:

$$\widehat{\mathbf{b}} = \left(\mathbf{G}^{(j)} + \tfrac{1}{L-d}\boldsymbol{W}^{(j)} + \gamma\Gamma\right)^{-1}\mathbf{G}^{(j)}\mathbf{b}^{(j)}$$

- And so finally, $\hat{\beta}(t) = \hat{\mathbf{b}}^T \mathbf{B}(t)$

FuSIM [Compact Support] Cont. (19) - Sparsity Parameter Criterion

- Let's consider λ again - how do we pick it?
  - One approach is to minimize AIC or BIC
  - BIC defined as:

$$\text{BIC} = n\log\left(\frac{RSS}{n}\right) + \log(n)(p+1)$$

  - Where n is the sample size, RSS is the sum of the squared residuals, and p is the number of non-zero elements in $\hat{\mathbf{b}}$ (the basis coefficients)
- Okay, now we are ready to do some analysis!

FuSIM Compact

Results

Data Set Overview

- More bike rentals demanded in recent years
- Want to ensure a sufficient bike supply
- Want to learn about the customer's rental behaviour as it relates to weather conditions during the weekend
- The total counts of casual bike rentals are recorded from January 1^{st}, 2011, to December 31^{st}, 2012, for a total of 105 weeks
- Weather temperature collected hourly

Data Set Overview

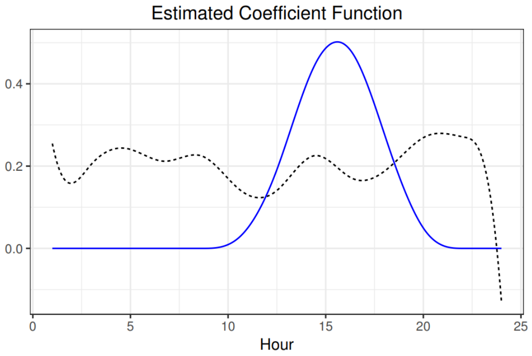
- Goal: to study how Saturday rentals relate to the hourly temperature
- Y is the total number of Saturday rentals
- $X(t)$ is the hourly temperature

Analysis

- The optimal value of the sparsity parameter λ is selected from $\{10^{-1}, 10^1, 10^3, 10^5, 10^8\}$ by 10-fold cross-validation
- Given the value of λ, the bandwidth of the local linear regression is selected by the leave-one-out crossvalidation
- Optimal value of the sparsity parameter: 10^5 and bandwidth $= 29$

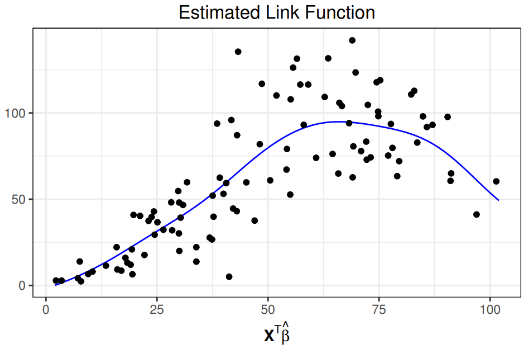
Analysis Cont. (1)

- The estimated index function is:



Analysis Cont. (1)

- The estimated link function is:



Comparison with FuSIM

- Want to compare the two models
- The response y_i, $i = 1, 2, ..., n$ is generated using the following model:

$$y_i = g_0(\int \beta_0 X(t)dt) + \epsilon_i$$

- Where g_0 and β_0 are taken from the previous estimates and ϵ is normally distributed with a mean of 0 and some standard deviation σ_ϵ
- In each simulation repetition, we randomly select 80% of the samples from the observed hourly temperature curves as the training data set and treat the remaining samples data as the test data set.

Comparison with FuSIM Cont. (1)

- Estimate the coefficient function $\beta(t)$ and the link function $g(.)$ using the training data set only
- Predict the response variable using the test data set and obtain the mean squared prediction errors (MSPEs)
    - ⋆ This is done 100 times

Comparison with FuSIM Cont. (2)

- The average MSPE produced by the compact FuSIM is 16.40, which is 5 times smaller than the average MSEP using the conventional FuSIM.
- The compact FuSIMs performance is more stable compared to the conventional FuSIM, because the standard deviation of the compact FuSIM method is much smaller than that of to the conventional FuSIM

Comparison with FuSIM Cont. (3)

- Compared the estimated $\beta(\hat{t})$ with the true coefficient $\beta_0(t)$ by integrated mean square error (IMSE) defined as:

$$\text{IMSE} = \int_0^T (\hat{\beta}(t) - \beta_0(t))^2 dt.$$

- The average IMSE using using compact FuSIM is 0.02 whereas using the cconventional FuSIM, it is 1.88

Section IV

Conclusions

Conclusions

- This talk went from the very basics of functional data analysis to a relatively in-depth introduction in functional single index models
  - ⋆ Information was given on how to "create" functional data including coefficient estimation and basis selection
  - ⋆ Smoothing methods were shown
- Functional linear models for scalar responses were presented
  - ⋆ However, this type of model is not always suitable or helpful in solving questions

Conclusions Cont. (1)

- Functional linear models for scalar responses were presented
  - ⋆ However, this type of model is not always suitable or helpful in solving questions
- In order to remedy this, a paper introduced functional single index models with compact support!
  - ⋆ Shrinks coefficient towards 0 in some subdomains of the continuum
  - ⋆ In some questions, agrees with our intuition on what ought to happen

Future Considerations

- There are many moving parts when it comes to solving for the various parts that make up the single index model
  - ⋆ The link function
  - ⋆ The index function
  - ⋆ The subdoing in compact models
- Using alternatives to the SCAD method that allow for an increase in computational efficiency could be useful
- Alternatives to the local linear regression estimator such as the Weighted Nadaraya-Watson Regression estimator (this is a non-parametric technique in the same realm as local linear regression) could be worth exploring

Future Considerations Cont. (1)

- Skeptical of simulation method
  - ⋆ Is there some analogue of confirmation bias here?
  - ⋆ The index and link functions estimated by the compact method were used to get the true estimates or generate the estimates for y
  - ⋆ Issue: the support is restricted using those estimates

References

- Chen, D., P. Hall, and H.-G. Muller *Single and multiple index functional regression models with nonparametric link.* 2011
- Yunlong Nie *Functional Single Index Models with Compact Support* 2018
- Jiang, C.-R. and J.-L. Wang , *Functional single index models for longitudinal data The Annals of Statistics* 2011
- Lin, Z., J. Cao, L. Wang, and H. Wang, *Locally sparse estimator for functional linear regression models.* 2017

References Cont. (1)

- Ramsay, J. and B. Silverman *Functional data analysis* 2005
- Zambom, A. Z. and M. G. Akritas *NonpModelCheck: An R package for nonparametric lack-of-fit testing and variable selection* 2017
- Han Lin Shang , *Estimation of a functional single index model with dependent errors and unknown error density The Annals of Statistics* 2018
- Fei Jiang, Seungchul Baek, Jiguo Cao and Yanyuan Ma *A Functional Single Index Model*

Thanks for Listening!