BIOS 7720: Applied Functional Data Analysis

Leture 2: Basis Expansions, Smoothing Splines

Andrew Leroux

March 4, 2021

Roadmap

1. Basis functions
2. Choosing the number of basis functions
3. Enforcing smoothness via penalization

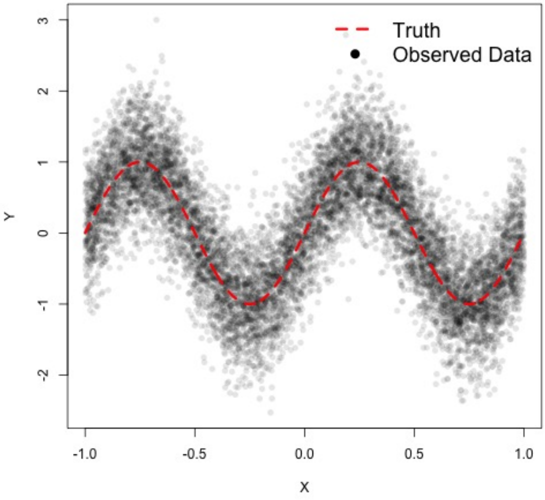
Data Context: Set-Up

$$
\begin{aligned}
y_i &= f(x_i) + \epsilon_i \\
\epsilon_i &\sim N(0, \sigma_\epsilon^2) \\
x_i &\sim \text{Unif}[-1, 1] \\
f(x) &= \sin(2\pi x)
\end{aligned}
$$

Data Context: Simulating in R


```r
# number of (X,Y) pairs to simulate
N <- 10000
# variance for Gaussian noise
sig2_e <- 0.5
# conditional mean of Y given X
f <- function(x) sin(2*pi*x)
# simulate predictor X ~ Unif(-1,1)
X <- runif(N, min=-1, max=1)
# simulate Y = f(X) + \epsilon
Y <- f(X) + rnorm(N, mean=0, sd=0.5)
```

Data Context: Plotting the Data



What are Basis Expansions?

- Recall the model: $y_i = f(x_i) + \epsilon_i$
- Goal: Turn this into a linear model (use (G)LM(M) tools)
- Assume $f(\cdot)$ can represented using a linear combination known functions
- These functions constitute a set of basis functions

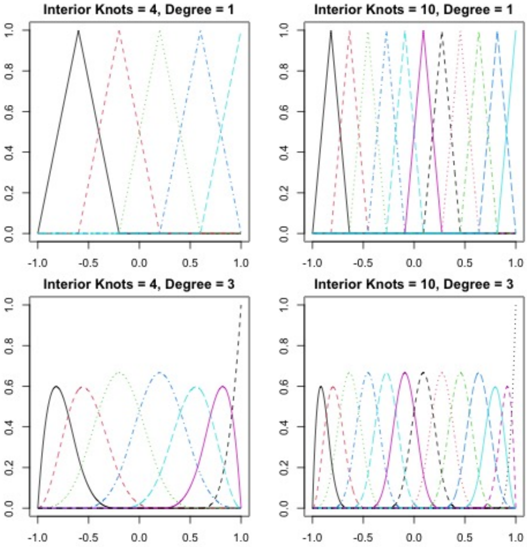
$$\begin{aligned} y_i &= f(x_i) + \epsilon_i \\ &= \sum_{k=1}^{K} \xi_k \phi_k(x_i) + \epsilon_i && \text{apply basis expansion} \\ \boldsymbol{y} &= \boldsymbol{\Phi}\boldsymbol{\xi} + \boldsymbol{\epsilon} && \text{vector/matrix notation} \end{aligned}$$

where $\boldsymbol{y} = [y_1, \ldots, y_N]^t$, $\boldsymbol{\phi}_i = [\phi_1(x_i), \ldots, \phi_K(x_i)]$, $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N]^t$ and $\boldsymbol{\epsilon} = [\epsilon_1, \ldots, \epsilon_N]^t$

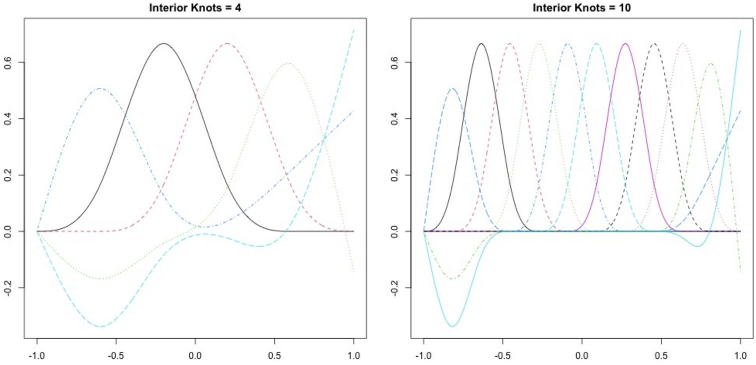
Bases

- There are many different bases
  - B-splines
  - Natural cubic splines
  - Thin plate (regression) splines
  - Fourier
- In many applications the choice basis not particularly important*
- For spline bases, need to choose
  - Number of knots/basis functions
  - Knot location
- Lots of packages in *R*
  - *mgcv*
  - *fda*
  - *splines*
  - $\vdots$

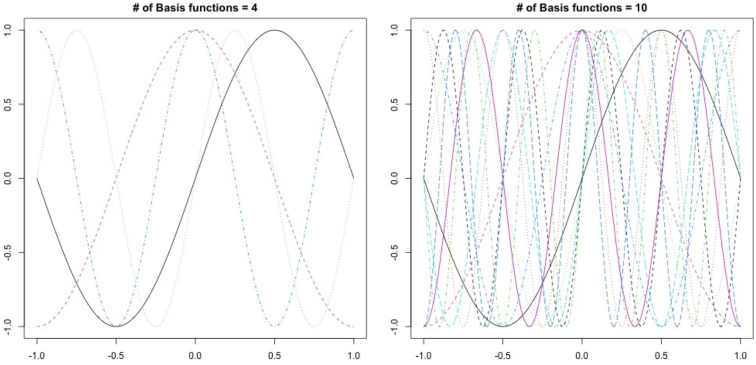
Some Different Bases: B-splines



Some Different Bases: Natural Cubic Splines



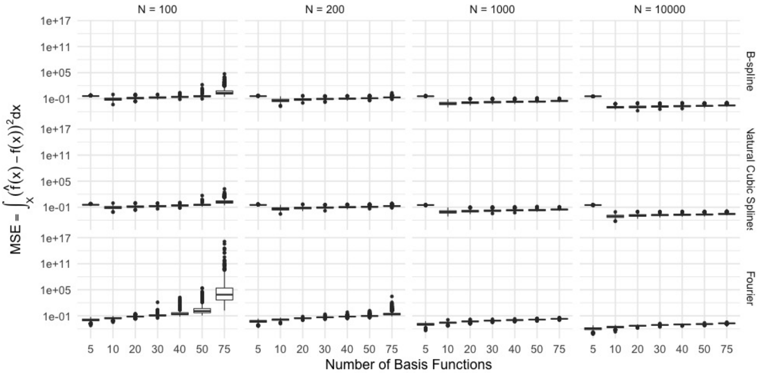
Some Different Bases: Fourier



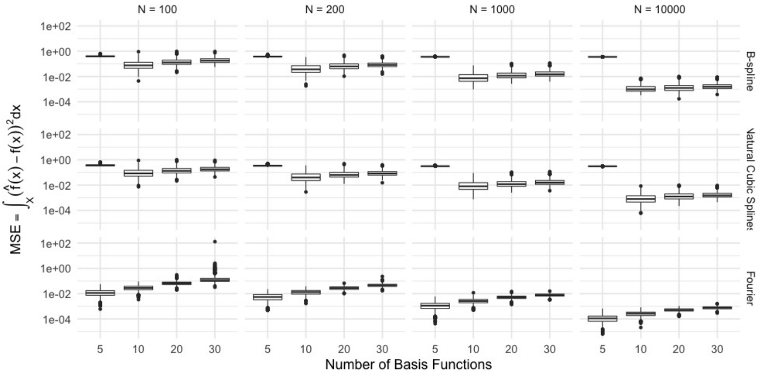
Simulation Set-up

- $N \in \{100, 200, 1000, 10000\}$
- Number of basis functions $K \in \{5, 10, 20, 30, 40, 50, 75\}$
- 1000 simulated datasets for each
- $\text{MSE} = \int_X (f(x) - \hat{f}(x))^2 dx$

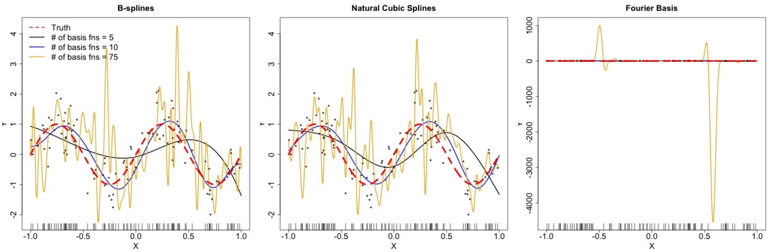
Simulation Results



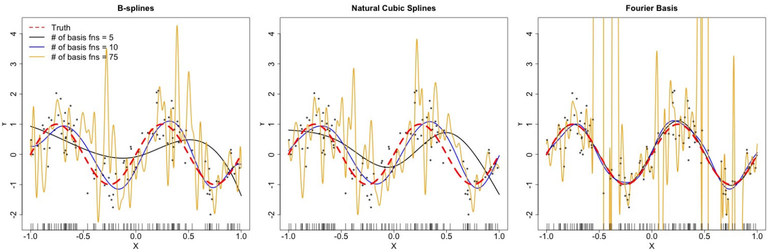
Simulation Results



A Single Simulated Dataset (N=100)



A Single Simulated Dataset (N=100)



Simulation Takeaways

- Overfitting is a problem
- Basis functions that are not flexible enough are a problem
- Increasing sample size helps, but doesn't solve the problem

Controlling Smoothness

- Choose “K"
- Penalization

Cross-Validated "Choose K"

- Want to avoid overfitting to the data
- (leave-one-out) Cross-validated prediction error
- Computationally expensive, convenient shortcut

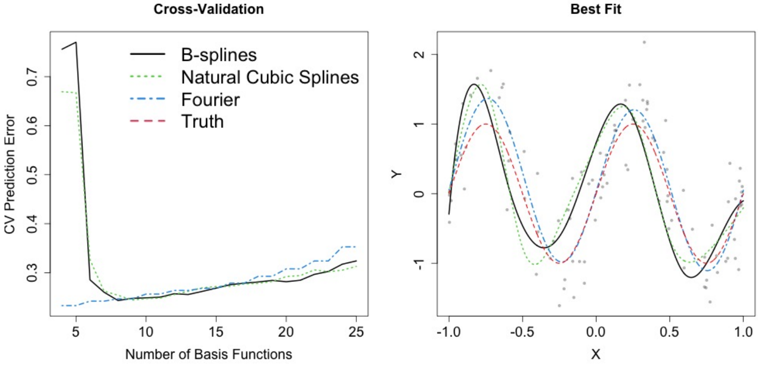
$$\begin{aligned}\mathcal{V}_0 &= N^{-1}\sum_{i=1}^{N}(\hat{y}_i^{[-i]} - y_i)^2 \\ &= N^{-1}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2/(1 - H_{ii})^2\end{aligned}$$

where $H = X(X^tX)^{-1}X^t$

Cross-Validated "Choose K"

```
set.seed(100)
N <- 1000
p <- 10
X <- cbind(1, matrix(rnorm(N*p), N, p))
Y <- X %*% seq(-1,1,len=p+1) + rnorm(N)
## do ordinary cross validation
system.time({
  OCV_slow <-
    vapply(1:N, function(x)
      Y[x] - X[x,] %*% (.lm.fit(X[-x,], Y[-x])$coef),
    numeric(1))
  OCV_slow <- OCV_slow^2
})

##    user  system elapsed
##   0.215   0.037   0.252

## do "fast" ordinary cross validation
system.time({
  H <- X %*% solve(crossprod(X)) %*% t(X)
  OCV_fast <- .lm.fit(X,Y)$residuals^2/(1-diag(H))^2
})

##    user  system elapsed
##   0.007   0.000   0.007

all.equal(as.vector(OCV_fast), OCV_slow)

## [1] TRUE
```

Cross-Validated "Choose K" in Our Simulation



Cross-Validated "Choose K" in Our Simulation

- Works fairly well! However...
- Natural and B-splines aren't flexible enough without overfitting
- Knot location selection?
  - Some methods developed
  - Computationally expensive
  - Hard to generalize to more complicated problems, multiple smooths

Penalization

- Want a flexible function (lots of basis functions)
- But need to balance flexibility and tendency to overfit
- Idea: place a penalty on the curvature of $f(\cdot)$
- Penalized least squares

$$\sum_{i=1}^{N}(y_i - f(x_i))^2 + P_\lambda(f)$$

- How to choose $P_\lambda(f)$?
- Problem dependent, for splines generally second derivative

Penalization

Penalized least squares problem:

$$
\begin{aligned}
\text{PENSSE}_\lambda &= \sum_{i=1}^{N}(y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx \\
&= \sum_{i=1}^{N}\Big(y_i - \sum_{k=1}^{K}\xi_k \phi_k(x_i)\Big)^2 + \lambda \int f''(x)^2 dx \\
&= ||\boldsymbol{y} - \Phi\boldsymbol{\xi}||^2 + \lambda \boldsymbol{\xi}^t \boldsymbol{S} \boldsymbol{\xi}
\end{aligned}
$$

Closed form solution:

$$
\hat{\boldsymbol{\xi}} = (\boldsymbol{X}^t\boldsymbol{X} + \lambda\boldsymbol{S})^{-1}\boldsymbol{X}^t
$$

Penalization

- $\text{PENSSE}_\lambda = ||\boldsymbol{y} - \Phi\xi||^2 + \lambda\xi^t S\xi$
- λ is known as a smoothing parameter which we need to estimate
- Other choices for $P_\lambda(\xi)$
  - Ridge regression: $\lambda\xi^t\xi$
  - Lasso: $\lambda\sum_k |\xi_k|$
- How to choose λ?
  - Choose large K
  - Cross-validation
- How to find $\boldsymbol{S}$?
  - Manually (straightforward, but tedius)
  - Let software do it for us (preferred)

Penalization and *mgcv*

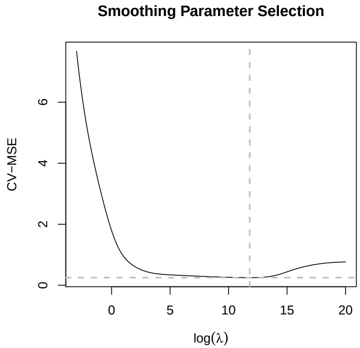

```
library("mgcv"); set.seed(5520); N <- 100
f <- function(x) sin(2*pi*x)
X <- runif(N, min=-1, max=1)
Y <- f(X) + rnorm(N, mean=0, sd=0.5)
xind <- seq(0,1,len=100)
sm <-smoothCon(s(X, bs="cr", k=75), data=data.frame(X=X))
str(sm)
```

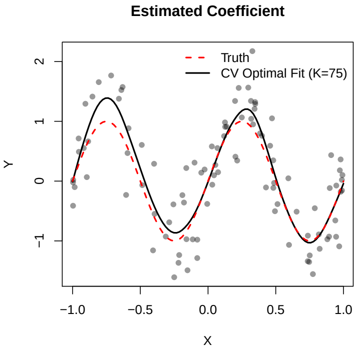
```
## List of 1
##  $ :List of 22
##   ..$ term          : chr "X"
##   ..$ bs.dim        : num 75
##   ..$ fixed         : logi FALSE
##   ..$ dim           : int 1
##   ..$ p.order       : logi NA
##   ..$ by            : chr "NA"
##   ..$ label         : chr "s(X)"
##   ..$ xt            : NULL
##   ..$ id            : NULL
##   ..$ sp            : NULL
##   ..$ X             : num [1:100, 1:75] 7.19e-20 -4.84e-23 -5.14e-33 4.77e-28 0.00 ...
##   ..$ S             :List of 1
##   .. ..$ : num [1:75, 1:75] 13.7538 -14.8363 1.1197 -0.0631 0.0284 ...
##   ..$ rank          : num 73
##   ..$ null.space.dim: num 2
##   ..$ df            : num 75
##   ..$ xp            : Named num [1:75] -0.997 -0.996 -0.988 -0.955 -0.941 ...
##   .. ..- attr(*, "names")= chr [1:75] "0%" "1.351351%" "2.702703%" "4.054054%" ...
##   ..$ F             : num [1:5625] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ noterp        : logi TRUE
##   ..$ plot.me       : logi TRUE
##   ..$ side.constrain: logi TRUE
##   ..$ C             : num [1, 1:75] 0.0238 0.00195 0.01368 0.01869 0.00379 ...
##   ..$ S.scale       : num 82871195
##   ..- attr(*, "class")= chr [1:2] "cr.smooth" "mgcv.smooth"
```

Penalization: Selecting λ Using Cross-Validation

```
nlambda   <- 1000                   # number of smoothing parameters to consider
loglambda <- seq(-3,20,len=nlambda) # sequence of log smoothing parameters
Phi <- sm[[1]]$X           # get the spline basis matrix
S   <- sm[[1]]$S[[1]]      # get the "S" matrix
MSE_CV <-rep(NA, nlambda) # empty container for storing CV-MSE
for(i in 1:nlambda){       # do the cross-validation
  H    <- Phi %*% solve(crossprod(Phi) + exp(loglambda[i])*S) %*% t(Phi)
  yhat <- H %*% Y
  MSE_CV[i] <- mean((yhat - Y)^2/(1-diag(H))^2)
}
# get the estimated coefficient for the optimal smoothing parameter
lambda_min <- exp(loglambda[which.min(MSE_CV)])
# get estimated spline coefficients for optimal lambda
xi_hat     <- solve(crossprod(Phi) + lambda_min*S) %*% t(Phi) %*% Y
# set of "X" values to estimate \hat{f} on (for plotting)
xind_pred  <- seq(-1,1,len=100)
# spline basis for associated with "xind_pred"
Phi_hat    <- PredictMat(sm[[1]], data=data.frame(X=xind_pred))
# estimated coefficient
f_hat      <- Phi_hat %*% xi_hat
```

Penalization: Selecting λ Using Cross-Validation





In-Class Exercises: Set Up

- Download the NHANES data uploaded to Canvas ("/files/data/NHANES_AC_processed.rds")
- File contains data on activity counts (1 row per participant-day)
- Within participants, rows are ordered chronologically
- Columns contain information on
  - Demographic, lifestlye, comorbidity, etc.
  - Activity counts at each minute of the day (MIN1 - MIN1440)
  - Indicator for whether that particular day is a "good" day of data (good_day)

In-Class Exercises

1. Single day of activity
   - Take the minute level activity data for the first "good" day in the data
   - Fit the model $E[\text{AC}(t)] = f(t)$ using penalized lease squares
   - Use penalized B-splines (bs="ps") with 75 basis functions
   - Select the smoothing parameter using leave-one-out cross-validation
   - Plot the data versus the fit, comment on results
2. Age versus wear time ("wear_time" column)
   - Use the same procedure above to fit the model $E[\text{Age}_i] = f(\text{wear time}_i)$
   - Plot the data versus the fit, comment on results