BIOS 7720: Applied Functional Data Analysis

Leture 4b: Review

Andrew Leroux

March 26, 2021

Roadmap

1. Class projects
2. Homework 1
3. In-class exercises from Lecture 4

Class Projects

Homework 1: Problem 1

- (a) Leave in general terms

$$f(x) = \sum_{k=1}^{K} \xi_k \phi_k(x) = \boldsymbol{\xi}^t \boldsymbol{\phi}(x)$$
$$f''(x) = \sum_{k=1}^{K} \xi_k \phi_k''(x) = \boldsymbol{\xi}^t \boldsymbol{b}(x)$$

  where $b_k(x) = \phi_k''(x)$. Hints: 1) a scalar is its own transpose; 2) integration is done over the range of observed x.
- (b) use the formula you get from (a)
- (c) take the derivative of PENSSE_λ w.r.t $\boldsymbol{\xi}$, set equal to 0, solve, check second derivative

Homework 1: Problem 1

- (d)

$$\begin{aligned}\mathrm{Var}(\hat{f}(x)) &= \mathrm{Var}(\sum_{k=1}^{K} \phi_k(x)\hat{\xi}_k) \\ &= \mathrm{Var}(\boldsymbol{\phi}(x)^t \hat{\boldsymbol{\xi}})\end{aligned}$$

- (e) Find $\boldsymbol{a}$, $\boldsymbol{B}$, for the hint, $\boldsymbol{D}$ is the matrix square root of $\boldsymbol{S}$. You can take the existence of $\boldsymbol{D}$ as given.

Homework 1: Problem 2

- Many acceptable ways to answer this problem
- the "direct" way is to write the function as written is:
  1. Set up vector of candidate smoothing parameters (λ)
  2. Loop over the candidate λ
     a. Loop over the $N = 100$ observations, excluding one at a time
     b. Apply your function which should return $\hat{\xi}$
     c. Calculate the squared error for predicting the one excluded observation
  3. Find the smoothing parameter which has the lowest MSE over all 100 observations
- Or, within your function, calculate OCV/GCV using the shortcut formulas presented in lecture, basically just wrapping the code from lecture 2, slide 26 in a function

Homework 1: Problem 3

- As originally written this problem was not sufficiently clear (expected too large a “jump" from course material):
  - The penalized portion of the log-likelihood
  - deriving versus numerical optimization
- The original goal was to have you derive the MLE and then use numeric optimization (the λ here is slightly different from that obtained using PLS in problem 2)
- Because of the lack of clarity, I'll accept for full credit the use of the λ from problem 2 directly

Homework 1: Problem 3

- Likelihood

$$\begin{aligned} \mathrm{L}(\boldsymbol{\xi}, \sigma_\epsilon^2; \boldsymbol{y}, \boldsymbol{x}) &= (2\pi\sigma_\epsilon^2)^{-N/2} e^{-\sum_{i=\mathbf{1}}^{N} (y_i - \sum_{k=\mathbf{1}}^{K} \xi_k \phi_k(x_i))^{\mathbf{2}} / 2\sigma_\epsilon^{\mathbf{2}}} \\ &= (2\pi\sigma_\epsilon^2)^{-N/2} e^{-(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\xi})^t (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\xi}) / 2\sigma_\epsilon^{\mathbf{2}}} \end{aligned}$$

- Take log, add penalty term

$$\mathrm{pl}(\boldsymbol{\xi}, \sigma_\epsilon^2; \boldsymbol{y}, \boldsymbol{x}) = \frac{1}{2}\left[-N \log(2\pi\sigma_\epsilon^2) - (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\xi})^t (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\xi}) / \sigma_\epsilon^2 - \lambda \boldsymbol{\xi}^t \boldsymbol{S} \boldsymbol{\xi}\right]$$

- Take derivative w.r.t. $\boldsymbol{\xi}$, set equal to 0, solve, check second derivative
- Result will look similar to PLS result from problem 1c

Homework 1: Problem 4

- (a) You are being asked to create your own bases here. For example, for $K = 2$

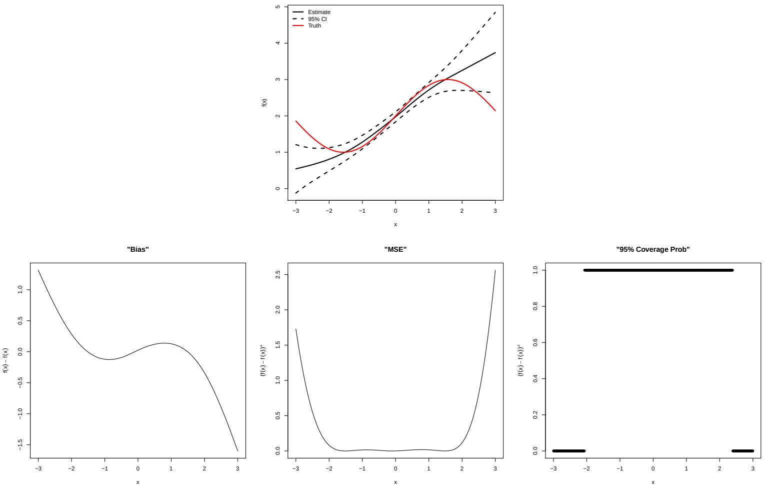
```r
x1  <- runif(100, 0, 1)
Phi <- cbind(1, x1, x1^2)
```

- (b) Unpenalzied least squares = linear regression
- (c) Find $\boldsymbol{S}$ using your results from problem 1a
- (d) Use formula from problem 1d for $\mathrm{SE}(\hat{f}_p(x))$

Homework 1: Problem 5

- Again, many ways to answer this problem in many ways
- Intended to give practice coding up simulation studies and presenting the results
- You can use your code from problem 2, or modify to be more compact
- You can fix the distribution of x, ϵ across simulation scenarios
- Coverage probability generally assessed using 95% CIs, but any % CI is fine
- Assessing bias, coverage probability, MSE can be done over the range of x (preferred), or averaged over the range of observed x values

Homework 1: Problem 5



Homework 1: Problem 5

```r
## set number of simulated datasets to create for each scenario
nsim <- 1000
## set up simulation scenarios
Ns   <- c(100,500,1000)
fs   <- list(f1=function(x) x^2,
             f2=function(x) sin(x),
             f3=function(x) cos(x)^2+ x^3)
Ks   <- c(5, 20, 50)
## set up empty containers to store results
# range of x values to assess simulations on, -3,3 assumes
# X ~ N(0,1), may need to modify range depedning on how you simulate X
nx_pred <- 100
xind    <- seq(-3,3,len=nx_pred)
arr_MSE <- arr_bias <- arr_coverage <-
  array(NA, dim=c(nsim, length(Ns), length(fs), length(Ks), nx_pred))
for(N in seq_along(Ns)){      # loop over number of observations
  for(f in seq_along(fs)){    # loop over association structures
    for(K in seq_along(Ks)){  # loop over number of basis functions
      for(n in 1:nsim){       # loop over simulated datasets
        # simulate covariate data
        # simulate outcome data
        # set up basis functions, get S matrix
        # find optiomal smoothing parameter
        # given the optimal smoothing parameter, get final fit
        # calculate bias, MSE, coverage probability
        # store your results
      }
    }
  }
}
## summarize and plot
```

Homework 1: Problem 6

- You're asked to show (prove) this
- I'll accept for full credit a proof where the constant vector is explicitly included in $\mathbf{\Phi}$
- Standard $\mathbf{\Phi}$ is full column rank assumption

Homework 1: Problem 7

- (a) Hint: don't just use *plot.gam()* with the default arguments
- (b) SEQN column is the subject identifier. Loop over candidate smoothing parameters, loop over individuals, estimate squared prediction error using leave one **subject** out cross validation. Choose the optimal smoothing parameter, plot the results, compare with part (a).

Homework 1: Problem 8

- (a) Fit the model to the data using the code provided. Same hint from 7a applies
- (b) Take repeated subsamples of various size (N) from the data (with or without replacement is fine here), fit the model using the code from part (a), calculate a **numeric summary of the "wiggliness"** of the estimated function. Compare the results for the various N

In-Class Exercises from Lecture 4

[Switch over to solutions, to be uploaded to Canvas]

In-Class Exercises

1. Simulate $y_1 \sim N(\mu, \sigma^2)$ and $y_2 \sim$ Poisson(λ) data. Find the MLEs for μ and σ^2 (for y_1) and λ (for y_2) using the *optim* function in *R*, obtain confidence intervals using the hessian matrix.
2. Note that the *optim* function minimizes by default. We can change this by specifying the argument: control=list(fnscale=-1).
3. Below, I write a function (fn_pois) which caculates the MLE of λ for N iid Poisson RVs. The function's first argument, params, is a vector (in this case, length 1) of parameters to optimize over. The second and third arguments, y and min, respectively, are for passing through the data and specifying whether to minimize or maximize the (negative) log likelihood.
4. I do the same thing for the normal data (fn_norm)
5. The optim function's first two arguments are starting values for the parameters and the function to be optimized. The method argument specifies the method to be used. The hessian argument specifies whether to return the Hessian matrix for the function being optimized.

In-Class Exercises

```
fn_pois <- function(params, y, min=TRUE){
  const <- ifelse(min, -1, 1)
  const*sum(dpois(y, lambda=params[1], log=TRUE))
}
fn_norm <- function(params, y){
  -sum(dnorm(y, mean=params[1], sd=params[2], log=TRUE))
}
```

In-Class Exercises

```
N  <- 1000; mu <- 5; y  <- rpois(N, lambda=mu)
opt_pois <- optim(par=c(1), fn=fn_pois,  y=y, min=TRUE, hessian=TRUE,
                  method="BFGS")
est_pois <- opt_pois$par
var_pois <- 1/opt_pois$hessian
## the the estimate of \lambda
est_pois

## [1] 4.88

## the 95% CI for \lambda
est_pois + c(-2,2)*sqrt(rep(var_pois,2))

## [1] 4.740286 5.019714
```

In-Class Exercises

```
N     <- 1000; mu <- 5; sig2 <- 2
y     <- rnorm(N, mean=mu, sd=sqrt(sig2))
opt_norm <- optim(par=c(1,1), fn=fn_norm,  y=y, hessian=TRUE, method="BFGS",
                  control=list(maxit=1000))
est_norm <- opt_norm$par
var_norm <- diag(solve(opt_norm$hessian))
## the estimates for \mu and \sigma
est_norm

## [1] 5.025524 1.387451

## 95% CI for \mu
est_norm[1] + c(-2,2)*sqrt(rep(var_norm[1],2))

## [1] 4.937774 5.113275
```

Additional Info For Homework

The code below can be used for problem 8b

```
library("mgcv")
N <- 100
x <- runif(N, -3,3)
y <- sin(pi*x) + rnorm(N, sd=1)
fit <- gam(y ~ s(x, k=50))
## extract the smoothing parameter
fit$sp

##     s(x)
## 6.699608

## get \xi^t S \xi
t(coef(fit)[-1]) %*% fit$smooth[[1]]$S[[1]] %*% coef(fit)[-1]

##          [,1]
## [1,] 1.136804
```