STA547: HW4 *

Zhiling Gu †

Contents

*Instructor: Xiongtao Dai, Iowa State University
†I discussed with Haihan YU about this homework.

1 Nadaraya–Watson Estimator Asymptotic Behaviour

Consider the univariate nonparametric regression setting where we have a sample $(X_i, Y_i), i = 1, \ldots, n$, which satisfies $Y_i = \mu(X_i) + \epsilon_i$, and the error variance $\text{var}(\epsilon_i) \equiv \sigma^2 > 0$ is a constant. Assume the density of X_i is positive, continuous, and supported on $[0, 1]$. The kernel $K(\cdot)$ is a symmetric continous denction supported on $[-1, 1]$ with $\int_{-1}^{1} K^2(x)dx < \infty$. The regression function μ is assumed to be twice differentiable with a bounded second derivative. Derive the asymptotic bias and variance for the Nadaraya-Watson estimator at a left boundary point $x_0 = ch$, where $c \in [0, 1)$, as $h \to 0$ and $nh \to \infty$. [For example, $c = 0$ implies $x_0 = 0$, so only design points falling within $[0, h]$ will be utilized.

Proof. Recall the definition of Nadaraya Watson (NW) estimator:

$$\widehat{\mu}_{NW}(X_0) = \frac{\sum_{i=1}^{n} K_h(X_i - X_0)Y_i}{\sum_{j=1}^{n} K_h(X_j - X_0)}, X_0 \in [a, b]$$

where K_h is a kernel with a bandwidth h. The denominator is a weighting term with sum 1. $K_h(\cdot) = \frac{1}{h}K(\frac{\cdot}{h})$. The $K(\cdot)$ is ususally a symmetric pdf. Our objective here is to find the bias and variance of $\widehat{\mu}_{NW}(X_0)$ at $X_0 = a = 0, X_0 \to 0, h \to 0, nh \to \infty$.

Refer to Fan, J. and Gijbels, I. (1996)[1]. Consider $p = 1, 2$, and $q \geq 0$

$$\begin{aligned}
E\left(\sum_{i=1}^{n} \left[K_h\left(X_i - x_0\right)\right]^p \left(X_i - x_0\right)^q\right) &= nE\left(\left[K_h\left(X_i - x_0\right)\right]^p \left(X_i - x_0\right)^q\right) \\
&= n\int_0^{x_0+h} \left[\frac{1}{h}K\left(\frac{x - x_0}{h}\right)\right]^p (x - x_0)^q f(x)dx \\
&= nh^{1-p+1}\int_{-c}^{1} [K(u)]^p (uh)^q du,\ 0 \leq X_i \leq 1, 0 \leq c \leq 1 \\
&= nh^{q-p+1}\int_{-c}^{1} u^q [K(u)]^p f\left(x_0 + uh\right) du, \text{Change of variable} \\
&= nh^{q-p+1}\int_{-c}^{1} u^q [K(u)]^p du\left(f\left(x_0\right) + o(1)\right) \text{ by taylor expansion}
\end{aligned}$$

$$\begin{aligned}
\text{Var}\left(\sum_{i=1}^{n} K_h\left(X_i - x_0\right)^p \left(X_i - x_0\right)^q\right) &\leq \sum_{i=1}^{n} E\left(\left[K_h\left(X_i - x_0\right)\right]^{2p} \left(X_i - x_0\right)^{2q}\right) \\
&= nh^{2q-2p+1}\int_{-c}^{1} u^{2q} [K(u)]^{2p} f\left(x_0 + uh\right) du = O\left(nh^{2q-2p+1}\right)
\end{aligned}$$

By Chebyshev's inequality, we have any random variable X satisfies $X = E(X) + O(\sqrt{Var(X)})$. Apply this to $\sum_{i=1}^{n} K_h\left(X_i - x_0\right)^p \left(X_i - x_0\right)^q$, we have

$$\begin{aligned}
\sum_{i=1}^{n} K_h\left(X_i - x_0\right)^p \left(X_i - x_0\right)^q &= nh^{q-p+1}\int_{-c}^{1} u^q [K(u)]^p du\left(f\left(x_0\right) + o(1)\right) + O_p(\sqrt{nh^{2q-2p+1}}) \\
&= nh^{q-p+1} f\left(x_0\right)\int_{-c}^{1} u^q [K(u)]^p du\left(1 + o(1) + O_p(1/\sqrt{nh}\right) \\
&= nh^{q-p+1} f\left(x_0\right)\int_{-c}^{1} u^q [K(u)]^p du\left(1 + o_p(1)\right)
\end{aligned}$$

[1] Local Polynomial Modelling and Its Applications. Chapman and Hall, London.

1.1 Bias

$$
\begin{aligned}
Bias(\widehat{\mu}_{NW}(x_0)) &= E(\widehat{\mu}_{NW}(x_0)) - \mu(x_0) \\
&= E\left(\frac{\sum_{i=1}^n K_h(X_i - x_0)Y_i}{\sum_{j=1}^n K_h(X_j - x_0)}\right) - \mu(x_0) \\
&= \frac{\sum_{i=1}^n K_h(X_i - x_0)[E(Y_i) - \mu(x_0)]}{\sum_{j=1}^n K_h(X_j - x_0)} \\
&= \frac{\sum_{i=1}^n K_h(X_i - x_0)[\mu(X_i) - \mu(x_0)]}{\sum_{j=1}^n K_h(X_j - x_0)} \\
&= \frac{\sum_{i=1}^n K_h(X_i - x_0)(\mu'(x_0)(X_i - x_0) + O((X_i - x_0)^2))}{\sum_{j=1}^n K_h(X_j)} \\
&= \frac{\sum_{i=1}^n K_h(X_i - x_0)\mu'(x_0)(X_i - x_0)}{\sum_{j=1}^n K_h(X_j)} + O((X_i - x_0)^2)
\end{aligned}
$$

Apply the lemma twice in the denominator and nominator

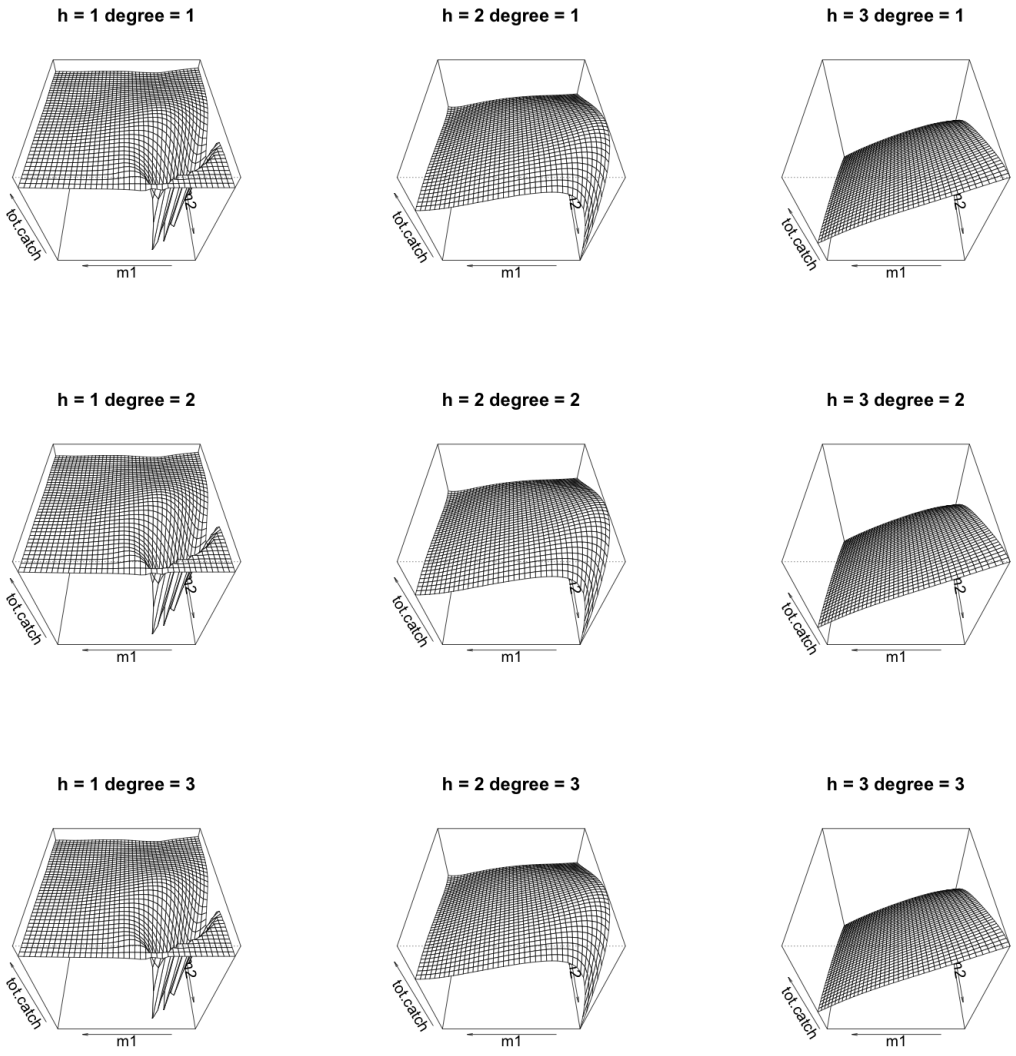
$$
\begin{aligned}
&= \frac{\mu'(x_0)nhf(x_0)\int_{-c}^1 u[K(u)]du\,(1 + o_p(1))}{nf(x_0)\int_{-c}^1 udu\,(1 + o_p(1))} + O(h^2) \\
&= \frac{\mu'(x_0)h\int_{-c}^1 u[K(u)]du\,(1 + o_p(1))}{\int_{-c}^1 udu} + O(h^2) \\
&= \frac{\mu'(x_0)h\int_{-c}^1 u[K(u)]du}{\int_{-c}^1 udu} + o_p(h) \quad \square
\end{aligned}
$$

1.2 Variance

$$
\begin{aligned}
Var(\widehat{\mu}_{NW}(x_0)) &= E(\widehat{\mu}_{NW}(x_0) - \mu(x_0))^2 \\
&= E\left(\frac{\sum_{i=1}^n K_h(X_i - x_0)Y_i}{\sum_{j=1}^n K_h(X_j - x_0)} - \mu(x_0)\right)^2 \\
&= E\left(\frac{\sum_{i=1}^n K_h(X_i - x_0)(\mu(X_i) + \epsilon_i - \mu(x_0))}{\sum_{j=1}^n K_h(X_j - x_0)}\right)^2 \\
&= E\left(\frac{\sum_{i=1}^n K_h(X_i - x_0)(\mu'(x_0)(X_i - x_0) + O(X_i - x_0)^2 + \epsilon_i)}{\sum_{j=1}^n K_h(X_j - x_0)}\right)^2 \\
&= E\left(\frac{\sum_{i=1}^n K_h(X_i - x_0)(\mu'(x_0)(X_i - x_0) + O(h^2) + \epsilon_i)}{\sum_{j=1}^n K_h(X_j - x_0)}\right)^2 \\
&= \frac{1}{\left(\sum_{j=1}^n K_h(X_j - x_0)\right)^2}\left\{(\sum_{i=1}^n K_h(X_i - x_0)(X_i - x_0))^2\mu'(x_0)^2 + \right. \\
&\quad \left. (\sum_{i=1}^n K_h(X_i - x_0)(X_i - x_0))O(h^2) + (\sum_{i=1}^n K_h(X_i - x_0))^2\sigma^2\right\} \\
&= \frac{1}{(nf(x_0)\int_{-c}^1 [K(u)]du\,(1 + o_p(1)))^2}\left\{(nhf(x_0)\int_{-c}^1 u[K(u)]du\,(1 + o_p(1))^2\mu'(x_0)^2 + \right. \\
&\quad \left. nhf(x_0)\int_{-c}^1 u[K(u)]du\,(1 + o_p(1))\,O(h^2) + nh^{-1}f(x_0)\int_{-c}^1 [K(u)]^2du\,(1 + o_p(1))\,\sigma^2\right\} \\
&= o_p(h^2) + \frac{\sigma^2}{nh}\frac{\int_{-c}^1 [K(u)]^2du}{(\int_{-c}^1 [K(u)]du)^2} \quad \square
\end{aligned}
$$

2 Bivariate smoothing

```
library(SemiPar)
data("scallop")
library(locfit)
dv <- c(1,2,3)
hv <- c(1,2,3)
n <- nrow(scallop)
m <- 100
png("q2.png",width = 1500, height = 1500, units = "px", pointsize = 30)
par (mfrow = c(3,3))
for (deg in dv) {
  for(h in hv){
    res <- locfit(tot.catch~ lp(latitude, longitude, deg=1, h=h), scallop, ev=lfgrid(mg=m))
    plot(res, type = 'persp', main=paste0('h = ',h, ' degree = ', deg), theta=180, phi=50 )
  }
}
dev.off()
```

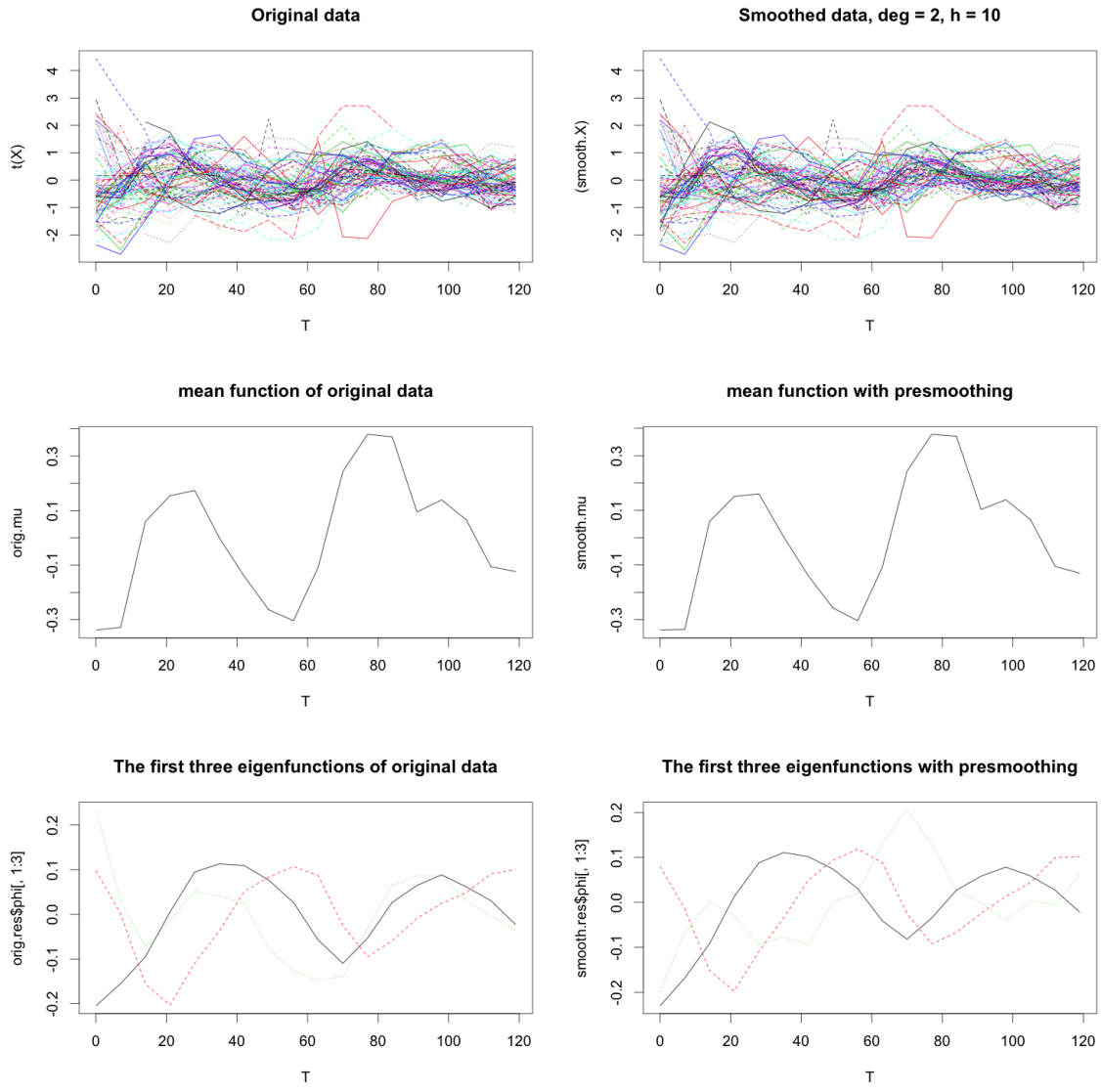


Note from the graph that, the increase in bandwidth and the increase in degree both drive the approximation to be smooth.

Check the design plot

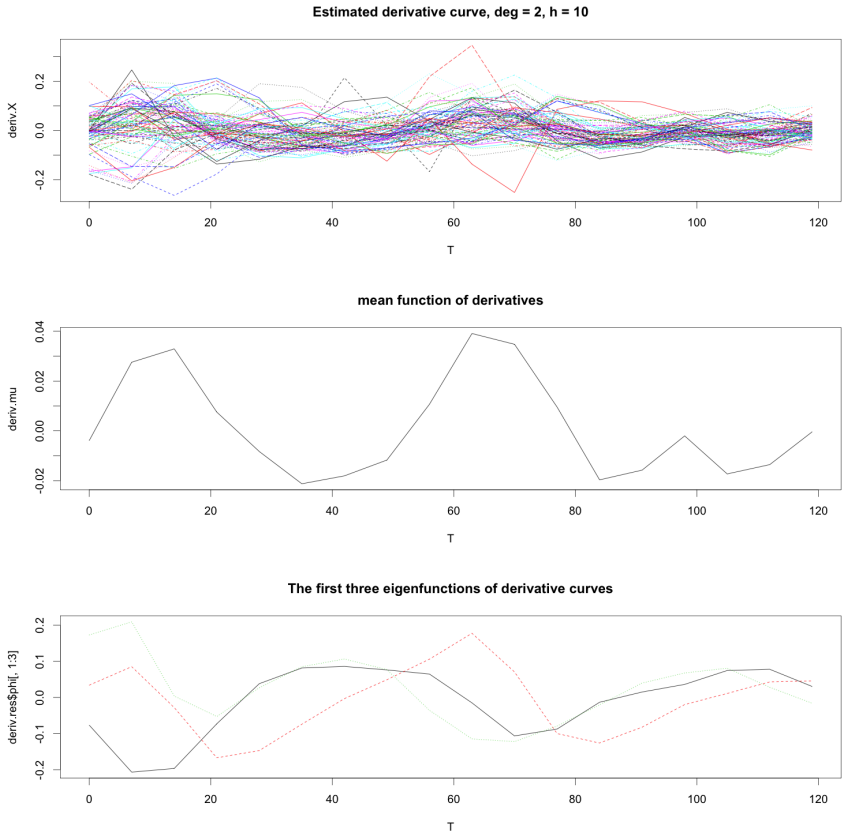
3 Presmoothing

3.1 The influence of Presmoothing on FPCA



In this question, local polynomial is applied with degree 2 and bandwidth 10. This choice is based on the similarity of the original data and smoothed data, along with the data precision of T which is observation per difference of 7.

3.2 The first derivative curve



Note the choice of bandwidth is essential when estimating the derivative compared to the smoothing on the original data.

```r
# 3.1 Presmoothing
data <- read.table("yeast.txt", skip  = 4)
X <- as.matrix(data)
T <- as.numeric(substr(names(data), 6, 8))

png("q3a.png",width = 1500, height = 1500, units = "px", pointsize = 30)
par (mfrow = c(3,2))
# original data & smoothing
matplot(T, t(X), type = 'l', main = "Original data")
m <- 30
smooth.X = apply(X, 1,function(y){
  res=locfit(y ~ lp(T, deg = 2, h = 10), ev=lfgrid(mg=m))
  predict(res, newdata=T)
})
matplot(T, (smooth.X), type = 'l', main = "Smoothed data, deg = 2, h = 10")


# fpca
fpca <- function(T, X) {
  mu <- colMeans(X, na.rm = TRUE) # In case missing data points in original datasets
  G <- cov(X, use = 'complete.obs') # In case missing data points in original datasets
  eig <- eigen(G)
  n <- nrow(X)
  m <- ncol(X)
  lam <- eig$values * diff(range(T)) / m
  phi <- eig$vectors / sqrt(diff(range(T) / m))
  Xcenter <- X - matrix(mu, nrow=n, ncol=m, byrow=TRUE)
  xi <- Xcenter %*% phi * diff(range(T)) / m
  list(mu=mu, G=G, phi=phi, lam=lam, xi=xi)
}

orig.res <- fpca(T, (X))
smooth.res <- fpca(T, t(smooth.X))
```

```r

orig.mu <- orig.res$mu
smooth.mu<- smooth.res$mu

plot(T, orig.mu, type = 'l', main="Mean function of original data")
plot(T, smooth.mu, type = 'l', main="Mean function with presmoothing")
matplot(T, orig.res$phi[, 1:3], type='l', main='The first three eigenfunctions of original data')
matplot(T, smooth.res$phi[, 1:3], type='l', main='The first three eigenfunctions with presmoothing')
dev.off()

# 3.2 first derivative
png("q3b.png",width = 1500, height = 1500, units = "px", pointsize = 30)
par(mfrow = c(3,1))
deriv.X = apply(X, 1, function(y){
  res=locfit(y ~ lp(T, deg = 2, h = 10), deriv=1, ev=lfgrid(mg=m))
  predict(res, newdata=T)
})
matplot(T, deriv.X, type = 'l', main= 'Estimated derivative curve, deg = 2, h = 10')
deriv.res= fpca(T, t(deriv.X))
deriv.mu=deriv.res$mu
plot(T, deriv.mu, type = 'l', main='mean function of derivatives')
matplot(T, deriv.res$phi[, 1:3], type='l', main='The first three eigenfunctions of derivative curves
    ')
dev.off()
```