Beta-Bernstein smoothing for Regression curve with compact support

ZiXia Huang

2024-05-27

1. Main Research Problem

In the traditional kernel methods for curve estimation, it has be regarded that the performance of the kernel methods depends largely on the smoothing bandwidth, and depends very little on the form of the kernels. Most kernels used are symmetric kernels and, once chosen, are fixed. This is efficient for estimating curves with unbounded supports, but not for curves which have compact support and are discontinuous at boundary points. For this type, a fixed form of kernel leads to boundary bias. For instance, if a symmetric kernel is used, the estimate for the curve at the boundary points has an expected value only half the true curve value as half of the kernel weight is outside the data support.

The main research topic is to follow the original paper(https://onlinelibrary.wiley.com/doi/epdf/10.1111/1467-9469.00136), which proposes a variable kernel method of non-parametric curve estimation, for functions having compact support, which does not suffer from the problem of boundary bias. The method is closely connected to classical Bernstein polynomials.

First we investigates kernel smoother for curves with compact support based on certain beta density functions. Its mean squared error and optimal bandwidth value are considered.

Then introduces the curve estimation model and the form of weights in the extended Bernstein smoothing scheme.We try to assess the performance of this proposed curve estimation model by considering again the integrated mean squared error as the original paper did. And reproduce some of the calculations from the original paper.

Finally carry out a simulation study and perform this Beta-Bernstein estimator together with other methods like local linear regression and local polynomial regression we learnt in the class.

2. A Beta Kernel Estimator

2.1 General settings

Suppose n obervations $y_1, \cdots, y_n$, are the responses design points $x_1, \cdots, x_n$, within $[0,1]$ from the model:

$$y_i = m(x_i) + \epsilon_i, i = 1, \cdots, n,$$

where $m(x)$ is an unknown function defined in $x \in [0,1]$ and the residuals ϵ_i are uncorrelated random error term with zero mean and variance σ^2. The design points are generated by a probability density function f and are pre-ordered such that $0 \leq x_1 \leq \cdots \leq x_n \leq 1$.

Let $K_{\alpha,\beta}$ denote the density function of a $Beta(\alpha, \beta)$ random variable. We use:

$$K_{\frac{x}{b}+1, \frac{1-x}{b}+1}(t) = \frac{t^{x/b}(1-t)^{(1-x)/b}}{B\{\frac{x}{b}+1, \frac{1-x}{b}+1\}} \times I_{0 \leq t \leq 1}$$

, as the kernel smooth at x, where B is the beta function and b is a smoothing parameter satisfying $b \to 0$ as $n \to \infty$.

2.2 The beta kernel estimator for m

The commonly used beta kernel estimator for m at $x \in [0,1]$ is:

$$\hat{m}_1(x) = \sum_{i=1}^{n} y_i \int_{x_i}^{x_i+1} K_{\frac{x}{b}+1, \frac{1-x}{b}+1}(t)dt$$

By the following assumptions:

(1) $|m(x)| < m_{max} < \infty$

(2) m is sufficiently smooth

(3) $|m'(x)| < m'_{max} < \infty$

(4) $b \to 0$ and $nb^2 \to \infty$ as $n \to \infty$.

The MSE decomposition:

$MSE = \{bias(\hat{m}(x))\}^2 + var(\hat{m}(x)) = [E\{\hat{m}(x)\} - m(x)]^2 + var(\hat{m}(x))$.

It can be shown that for any fixed x_0, $MSE(x_0) \leq C_1 b^2 + \frac{C_2\sqrt{b}}{n}$ within the interior of [0,1].

Therefore the mean integrated squared error:

$$MISE(x_0) = C_1^* b^2 + \frac{C_2^* \sqrt{b}}{n}$$

Choosing the optimal $b^* = C^{**} n^{-2/5}$, the convergence rate is $n^{-4/5}$, which is consistent with the convergence rate we are given in the class.

3. The Beta-Bernstein Smoothing for m

3.1 General settings

Suppose we have n observations $y_1, \cdots, y_n$ which are responses at n fixed equi-spaced design points $\{\frac{i}{n}; i = 1, 2, \cdots, n\}$ in [0,1] according to the regression model $y_i = m(\frac{i}{n}) + \epsilon_i$, where m is an unknown smooth function on [0,1], and the errors are independent with 0 mean and constant variance σ^2.

Starting from a linear smoother: $\hat{m}(x) = \sum_{i=1}^n y_i w_i(x)$,

we consider the weights $\{w_i(x)\}$ formed by splitting the beta pdf $K_{\lambda x+1,\lambda(1-x)+1}$ which allocates "pieces" to observations $\{y_i\}_{i=1}^n$:

$$w_i(x) = \int_0^1 K_{\lambda x+1,\lambda(1-x)+1}(t) f_i(t) dt$$

, where $\sum_{i=1}^n f_i(t) = 1$ for all $t \in [0,1]$ and λ is a constant smoothing parameter.

An approach linking Bernstein polynomials with a family of beta probability density functions is considered. We choose $f_i(t) = C_n^i t^i (1-t)^{n-i}$.

Thus treating i as a binomial random variable $Bi(n,t)$, and establishing links with classical Bernstein polynomials. Clearly $\sum_0^n f_i(t) = 1$ for all t, as required. For convenience, assume λ to be an integer. Then the weight:

$$\begin{aligned} w_i(x) &= \int_0^1 \frac{t^{\lambda x}(1-t)^{\lambda(1-x)}}{B(\lambda x+1, \lambda(1-x)+1)} C_i^n t^i (1-t)^{n-i} dt \\ &= \frac{1}{B(\lambda x+1, \lambda(1-x)+1)} \times \frac{n!}{i!(n-i)!} \times \int_0^1 t^{i+\lambda x}(1-t)^{\lambda(1-x)+n-i} dt \\ &= \frac{(\lambda+1)!n!}{(\lambda+n+1)!i!(n-i)!} \frac{\tau(\lambda(1-x)+n-i+1)}{\tau(\lambda(1-x)+1)} \frac{\tau(i+\lambda x+1)}{\tau(\lambda x+1)} \end{aligned}$$

and this form is easy to deal with numerically as long as a gamma function routine is available.

3.2 MISE for the proposed estimator

To assess the performance of $\hat{m}$, we again consider the mean integrated squared error as the paper did: $MISE(\hat{m}) = \int_0^1 E\{\hat{m}(x) - m(x)\}^2 dx = \int_0^1 [E(\hat{m}(x)) - m(x)]^2 + var(\hat{m}(x)) dx$.

3.2.1 Bias

First we consider the bias of this proposed estimator. To access the bias, we will need the following lemma.

Lemma 1: Let $|g''(x)| \le K$ for $0 \le x \le 1$. Then $|B_n g(x) - g(x)| \le \frac{1}{2} n^{-1} x(1-x) K$ where $B_n g(x) = E\{g(X_x/n)\}$, and $X_x \sim Bi(n,x)$.

Lemma 2: Let $m'' \in C[0,1]$. Then as h tends 0, we have: $m(x+h) - m(x) = hm'(x) + \frac{1}{2}h^2 m''(x) + 0(h^2)$, uniformly in $x, x+h \in [0,1]$.

Now with the above lemma, we can observe that:

$$E\{\hat{m}(x)\} = \sum_{i=1}^n w_i(x) m(i/n) = E\{B_n m(X)\}$$

, where $X \sim beta(\lambda x+1, \lambda(1-x)+1)$, which the mean is $E(X) = (\lambda x+1)/(\lambda+2)$, and variance is $\omega^2 = \frac{(\lambda x+1)(\lambda(1-x)+1)}{(\lambda+2)^2(\lambda+3)}$.

We can write $E(\hat{m}(x)) = E\{B_n m(\frac{\lambda x+1}{\lambda+2} + \omega U)\}$, $E(U) = 0, var(U) = 1$.

Then the bias can be break into 3 parts:

$$\begin{aligned} E(\hat{m}(x)) - m(x) = E\{B_n m(\frac{\lambda x+1}{\lambda+2} + \omega U) - m(\frac{\lambda x+1}{\lambda+2} + \omega U)\} \\ + E\{m(\frac{\lambda x+1}{\lambda+2} + \omega U)\} - m(\frac{\lambda x+1}{\lambda+2}) \\ + m(\frac{\lambda x+1}{\lambda+2}) - m(x) \end{aligned}$$

For the first term we can apply lemma 1:

$E\{B_n m(\frac{\lambda x+1}{\lambda+2} + \omega U) - m(\frac{\lambda x+1}{\lambda+2} + \omega U)\} \le \frac{1}{2} n^{-1} (\frac{\lambda x+1}{\lambda+2} + \omega U)(1 - \frac{\lambda x+1}{\lambda+2} - \omega U) K \le \frac{1}{2} n^{-1} x(1-x) K$.

For the second term we can apply lemma 2:

$E\{m(\frac{\lambda x+1}{\lambda+2} + \omega U)\} - m(\frac{\lambda x+1}{\lambda+2}) = E(\omega U) m'(\frac{\lambda x+1}{\lambda+2}) + \frac{1}{2}\omega^2 m''(\frac{\lambda x+1}{\lambda+2}) + o(\lambda^{-1}) = \frac{1}{2}\omega^2 m''(\frac{\lambda x+1}{\lambda+2}) + o(\lambda^{-1})$.

For the third term we again apply lemma 2:

$m(\frac{\lambda x+1}{\lambda+2}) - m(x) = m(x + \frac{1-2x}{\lambda+2}) - m(x) = \frac{1-2x}{\lambda+2} m'(x) + o(\lambda^{-2})$.

Therefore,

$$\begin{aligned} Bias^2 \le \frac{1}{2} \frac{(\lambda x+1)(\lambda(1-x)+1)}{(\lambda+2)^2(\lambda+3)} m''(\frac{\lambda x+1}{\lambda+2}) + \frac{1-2x}{\lambda+2} m'(x) + o(\lambda^{-1}) + o(n^{-1}) \\ \le \frac{1}{2} \frac{\lambda x+1}{\lambda+2} \frac{\lambda(1-x)+1}{\lambda+2} m''(\frac{\lambda x+1}{\lambda+2}) \lambda^{-1} + (1-2x) m'(x) \lambda^{-1} + o(\lambda^{-1}) + o(n^{-1}) \\ \le [\frac{1}{2} x(1-x) m''(x) + (1-2x) m'(x)] \lambda^{-1} \end{aligned}$$

3.2.2 Variance

To access the integrated variance, we will need the following lemma.

Lemma 3: If $p_i = C_k^i \theta^i (1-\theta)^{k-i}$, then $\sum_{i=1}^k p_i^2 \to \frac{1}{2}\{k\pi\theta(1-\theta)\}^{-1/2}$ as k tends to infinity.

First we can write:

$$Var(\hat{m}(x)) = Var(\sum_{i=1}^n w_i(x) y_i) = Var(\sum w_i(x) m(x_i)) + Var(\sum w_i \epsilon_i) = Var(\epsilon_i) \sum w_i(x)^2 (\text{fixed design}) = \sigma^2 \sum w_i(x)^2$$

Denote the integrated variance IV, we have $IV = \sigma^2 \int_0^1 \sum_{i=1}^n w_i^2(x)$ is approximately equal to

$IV \approx \frac{\sigma^2}{\lambda} \sum_{j=1}^{\lambda} \sum_{i=1}^n w_i^2(\frac{j}{\lambda})$.

Now from the weight formula we derived from section 3.1, we can have:

$w_i(\frac{j}{\lambda}) = (\frac{\lambda+1}{\lambda+n+1}) \frac{C_\lambda^j C_n^{a-j}}{C_{\lambda+n}^a}$

Observe that the very right hand side expression is the pdf for hypergeometric distribution $Hg(\lambda + n; \lambda, a)$. Now λ and n tends to infinity with λ/n tends to 0, this distribution is equivalent to $Bi(\lambda, a/(\lambda + n))$. Thus we can apply Lemma 4.

$$IV \sim \sigma^2 \frac{\lambda}{(\lambda+n)^2} \sum_a \frac{1}{2} \{\pi\lambda \frac{a}{\lambda+n}(1 - \frac{a}{\lambda+n})\}^{-1/2} \sim \frac{\sigma^2\sqrt{\lambda\pi}}{2(\lambda+n)}$$

, as n, λ tend to infinity.

3.3 Main result

Based on the results from previous 2 subsections, we now have the mean integrated squared error:

$$\begin{aligned} MISE(\hat{m}) &= \lambda^{-2} \int_0^1 \{(1-2x)m'(x) + \frac{1}{2}x(1-x)m''(x)\}^2 dx + \frac{\sigma^2\sqrt{\lambda\pi}}{2(\lambda+n)} \\ &= C_1\lambda^{-2} + \frac{C_2}{n\sqrt{\lambda}} \end{aligned}$$

Set the first order derivative to 0 and we can get the optimal lambda: $\lambda^* = C^{**}n^{2/5}$, and the convergence rate with optimal lambda is $n^{-\frac{4}{5}}$, which is the same as the common Beta kernel estimator.

However, as the size of the area where the Beta-Bernstein estimator has an increased variance is one order of magnitude smaller than the boundary bias area, it has no effect on the overall convergence rate of the MISE.

4. Simulation Study

In this section we present some empirical results designed to investigate the performance of the proposed beta kernel estimator for a regression function. We consider the following quadratic regression model:

$$y_i = sin(2\pi x_i) + \epsilon_i, i = 1, \cdots, n,$$

where the fixed design points x_i are taken at equally spaced points in [0,1] and ϵ_i are uncorrelated normal random variables with zero mean and standard deviation $\sigma = 0.1$.

The local polynomial regression with order one and two are also considered as comparison.

We simulate the data with sets of values $n = 100$, fit the model with a sequence of bandwidth and return empirical MSE.The bandwidth of thebeta-Bernstein estimator λ was chosen as h^2 to bring the amount of smoothing on the same scale.

```r
library(KernSmooth)
# Define the Beta-Bernstein basis function
beta_bernstein_basis <- function(i, n, x,lambda) {
  choose(n, i) * (x^(i+lambda)) * ((1 - x)^(n+lambda - i))/beta(lambda*x+1,lambda*(1-x)+1)
}

# Define the Beta-Bernstein smoother function
beta_bernstein_smoother <- function(x, y, n,lambda) {
  # Calculate the Beta-Bernstein basis functions
  basis <- sapply(0:n, function(i) beta_bernstein_basis(i, n, x,lambda=lambda))

  # Fit the model
  fit <- lm(y ~ basis - 1)

  # Return the fitted values
  return(fit$fitted.values)
}
```

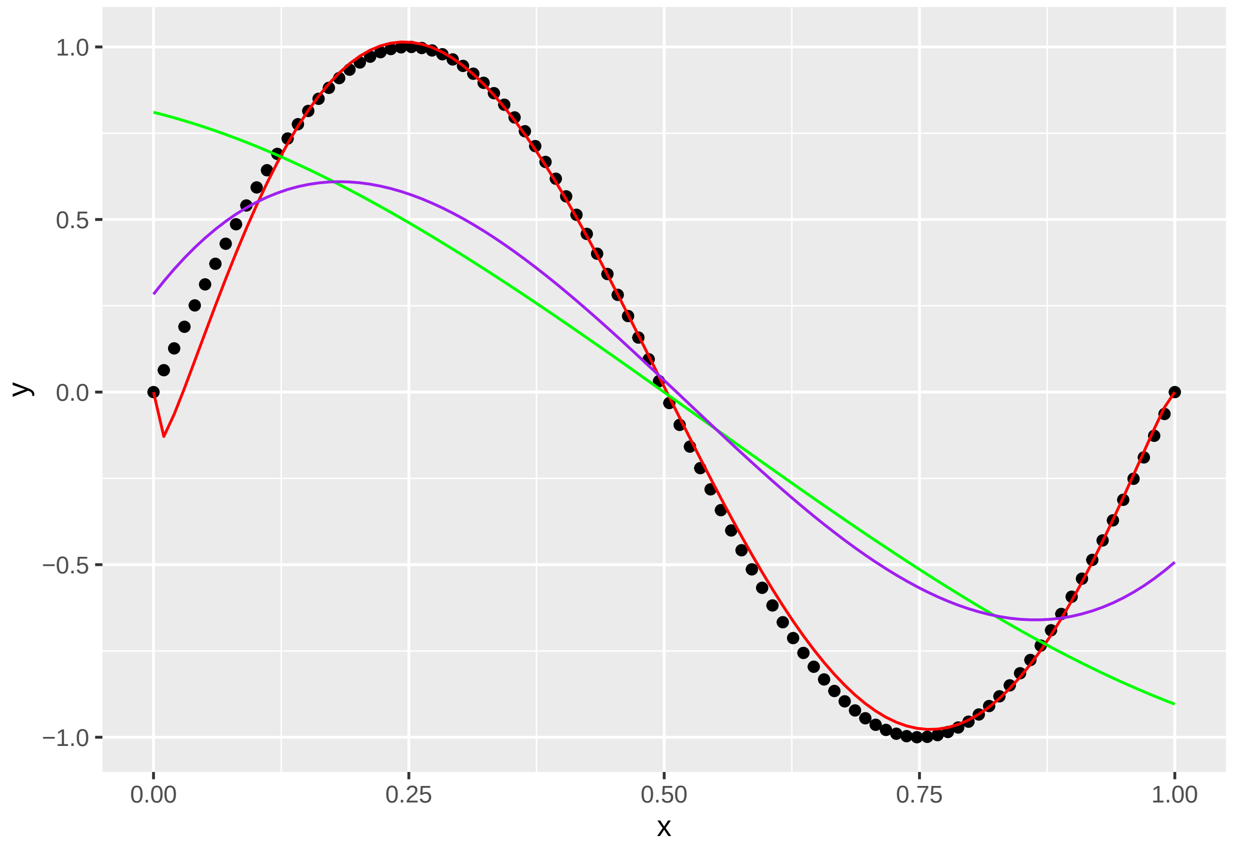
```r
set.seed(12)
n <- 100
X <- seq(from=0,to=1,length.out=n)
g_true <- sin(2*pi*X)
epsilon <- rnorm(100,mean = 0,sd=0.3)
Y <- g_true+epsilon
```

With h=0.4, $\lambda = 0.16$

```r
#Local polynomial regression with order 1
P1_fitted <- locpoly(X, Y, bandwidth=0.4,degree = 1)

P1_pred <- approx(P1_fitted$x, P1_fitted$y, xout=X)$y
P1_mse1 <- mean((g_true - P1_pred)^2)
#Local polynomial regression with order 2
P2_fitted <- locpoly(X, Y, bandwidth=0.4,degree = 2)

P2_pred <- approx(P2_fitted$x, P2_fitted$y, xout=X)$y
P2_mse1 <- mean((g_true - P2_pred)^2)
#Beta-Bernstein
BB_fitted<- beta_bernstein_smoother(x=X,y=Y,5,lambda = 0.16)
BB_mse1 <- mean((g_true-BB_fitted)^2)

df1 <- data.frame(x=X,y=g_true,BB=BB_fitted,poly1=P1_pred,ploy2=P2_pred)
```

```r
library(ggplot2)

# Create the base plot
p <- ggplot(df1, aes(x=x)) +
  geom_point(aes(y=y), color="black") +
  geom_line(aes(y=BB), color="red") +
  geom_line(aes(y=poly1), color="green") +
  geom_line(aes(y=ploy2), color="purple")
```

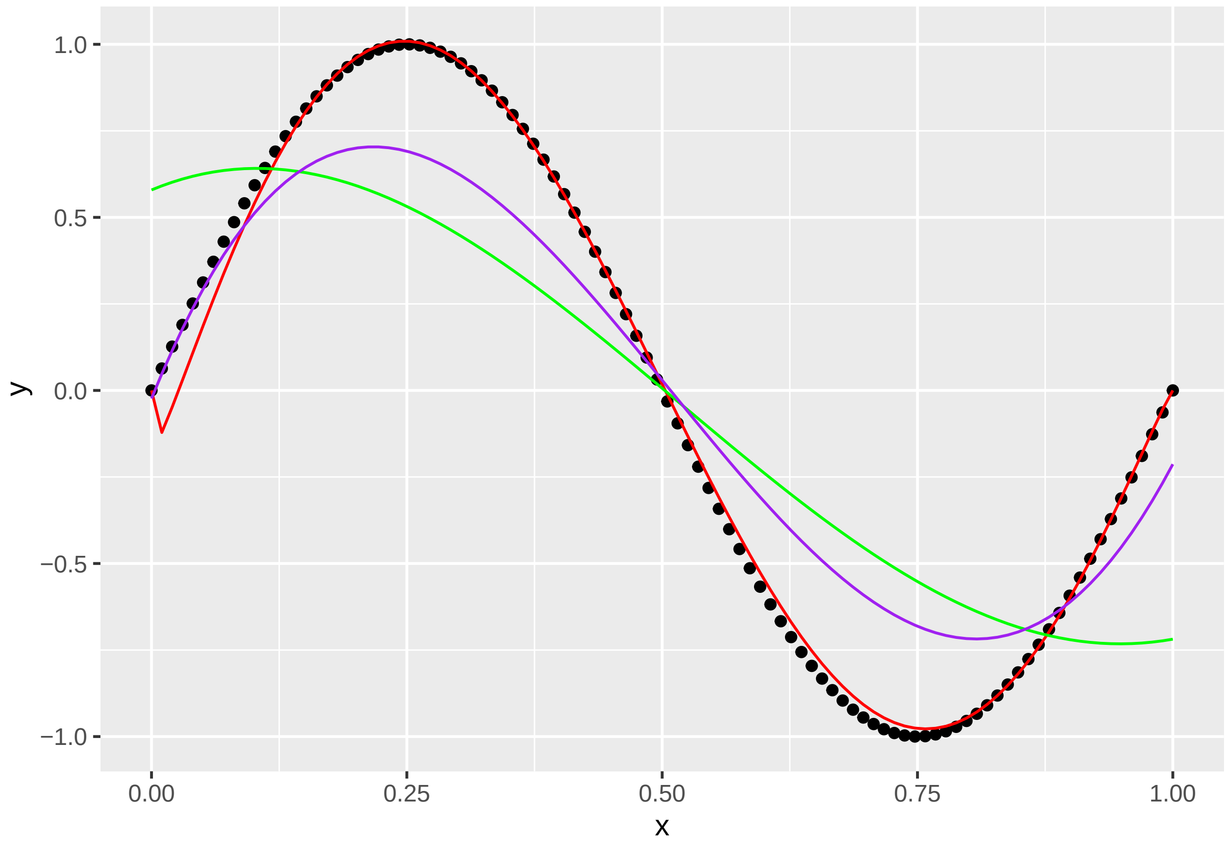
```
# Display the plot
print(p)
```



With h=0.28, $\lambda = 0.078$

```
#Local polynomial regression with order 1
P1_fitted <- locpoly(X, Y, bandwidth=0.28,degree = 1)

P1_pred <- approx(P1_fitted$x, P1_fitted$y, xout=X)$y
P1_mse2 <- mean((g_true - P1_pred)^2)
#Local polynomial regression with order 2
P2_fitted <- locpoly(X, Y, bandwidth=0.28,degree = 2)

P2_pred <- approx(P2_fitted$x, P2_fitted$y, xout=X)$y
P2_mse2 <- mean((g_true - P2_pred)^2)
#Beta-Bernstein
BB_fitted<- beta_bernstein_smoother(x=X,y=Y,5,lambda = 0.078)
BB_mse2 <- mean((g_true-BB_fitted)^2)

df2 <- data.frame(x=X,y=g_true,BB=BB_fitted,poly1=P1_pred,ploy2=P2_pred)
```

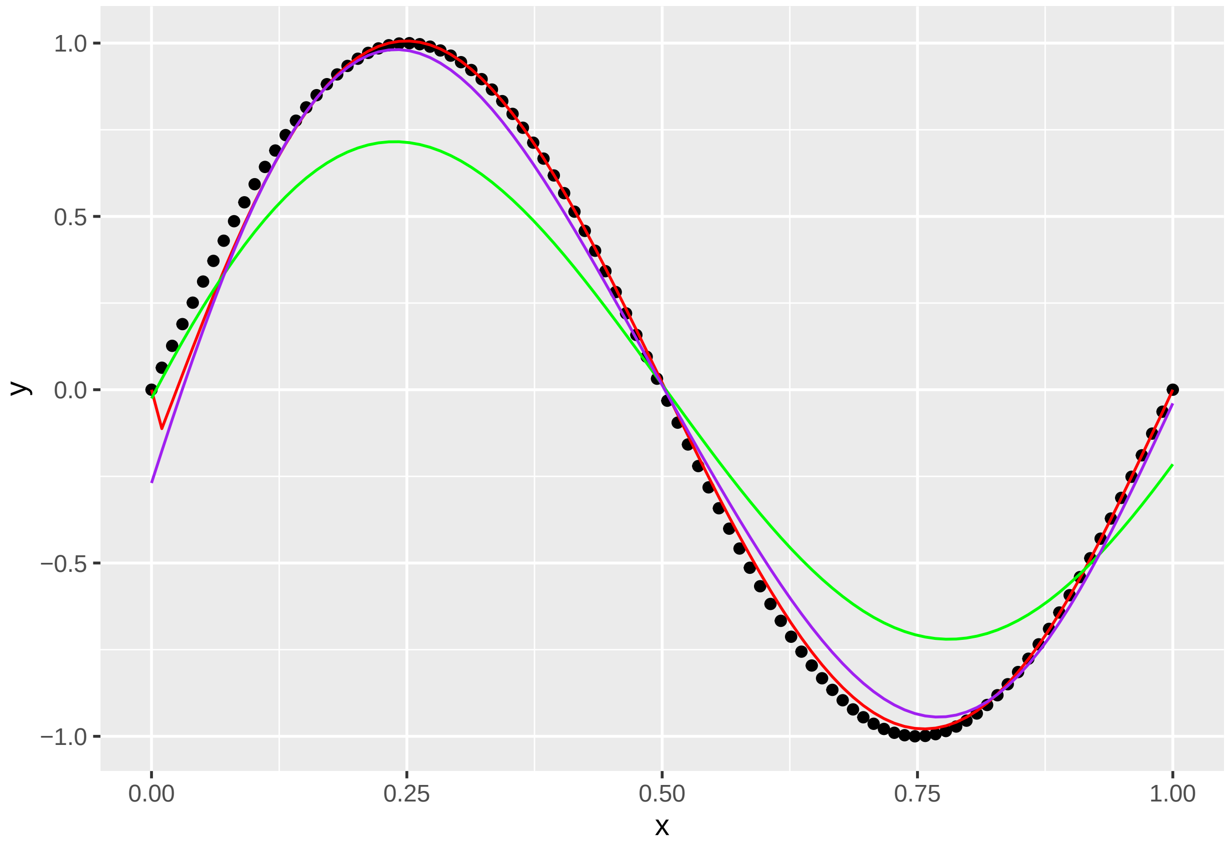
```
# Create the base plot
p <- ggplot(df2, aes(x=x)) +
  geom_point(aes(y=y), color="black") +
  geom_line(aes(y=BB), color="red") +
  geom_line(aes(y=poly1), color="green") +
  geom_line(aes(y=ploy2), color="purple")

# Display the plot
print(p)
```



With h=0.14, $\lambda = 0.019$

```
#Local polynomial regression with order 1
P1_fitted <- locpoly(X, Y, bandwidth=0.14,degree = 1)

P1_pred <- approx(P1_fitted$x, P1_fitted$y, xout=X)$y
P1_mse3 <- mean((g_true - P1_pred)^2)
#Local polynomial regression with order 2
P2_fitted <- locpoly(X, Y, bandwidth=0.14,degree = 2)

P2_pred <- approx(P2_fitted$x, P2_fitted$y, xout=X)$y
P2_mse3 <- mean((g_true - P2_pred)^2)
#Beta-Bernstein
BB_fitted<- beta_bernstein_smoother(x=X,y=Y,5,lambda = 0.019)
BB_mse3 <- mean((g_true-BB_fitted)^2)

df3 <- data.frame(x=X,y=g_true,BB=BB_fitted,poly1=P1_pred,ploy2=P2_pred)
```

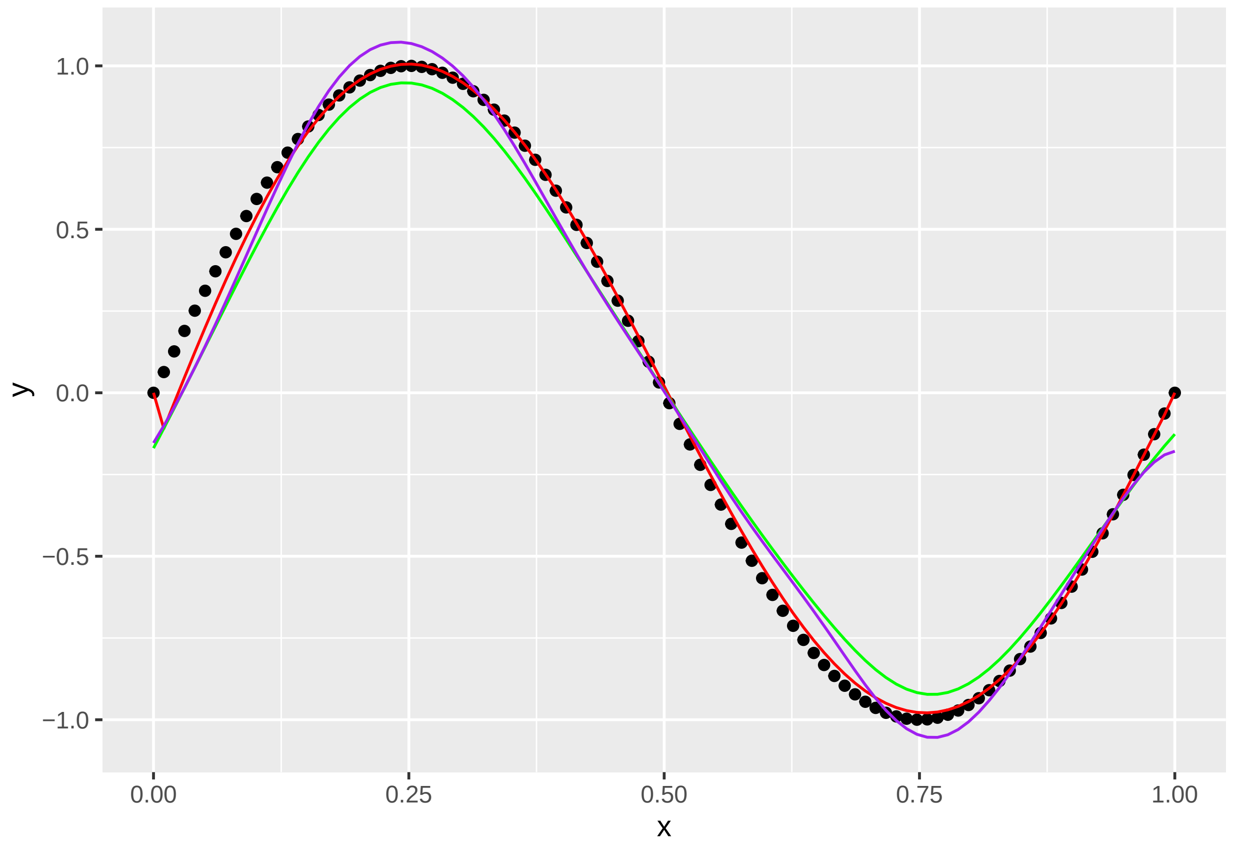
```
# Create the base plot
p <- ggplot(df3, aes(x=x)) +
  geom_point(aes(y=y), color="black") +
  geom_line(aes(y=BB), color="red") +
  geom_line(aes(y=poly1), color="green") +
  geom_line(aes(y=ploy2), color="purple")

# Display the plot
print(p)
```



With h=0.07, $\lambda = 0.005$

```
#Local polynomial regression with order 1
P1_fitted <- locpoly(X, Y, bandwidth=0.07,degree = 1)

P1_pred <- approx(P1_fitted$x, P1_fitted$y, xout=X)$y
P1_mse4 <- mean((g_true - P1_pred)^2)
#Local polynomial regression with order 2
P2_fitted <- locpoly(X, Y, bandwidth=0.07,degree = 2)

P2_pred <- approx(P2_fitted$x, P2_fitted$y, xout=X)$y
P2_mse4 <- mean((g_true - P2_pred)^2)
#Beta-Bernstein
BB_fitted<- beta_bernstein_smoother(x=X,y=Y,5,lambda = 0.005)
BB_mse4 <- mean((g_true-BB_fitted)^2)

df4 <- data.frame(x=X,y=g_true,BB=BB_fitted,poly1=P1_pred,ploy2=P2_pred)
```

```r
# Create the base plot
p <- ggplot(df4, aes(x=x)) +
  geom_point(aes(y=y), color="black") +
  geom_line(aes(y=BB), color="red") +
  geom_line(aes(y=poly1), color="green") +
  geom_line(aes(y=ploy2), color="purple")

# Display the plot
print(p)
```



Here we have the MSE table:

5. Results

The proposed Beta-Bernstein estimator has optimal mean integrated squared error at an order of magnitude $n^{-4/5}$, equivalent to that of standard kernel estimators when the curve has an unbounded support.

From the MSE table in the previous section, we can tell, the Beta-Bernstein estimates were close to their simulated counterparts. We find the Beta-Bernstein estimator performed better for small to medium sample sizes regardless of what value was used for the smoothing bandwidth.

From four plots in the previous section, we can tell that the Beta-Bernstein estimates indeed are free of boundary bias. And the local polynomial regression performance is largely dependent on the smoothing bandwidth.