UCLouvain
École de statistique, biostatistique
et sciences actuarielles

FACULTE DES SCIENCES

LSTAT2150

NON PARAMETRIC CURVE ESTIMATION: SMOOTHING METHODS

Project 34 :

Confdence intervals for non-parametric regression

Project report

Author :

Mathias Dah Fienon, *noma :04452100*

Academic year: *2022-2023*

UCLouvain
École de statistique, biostatistique
et sciences actuarielles

Confdence intervals for non-parametric regression

Contents

Mathias Dah Fienon

Introduction

Confidence intervals play a central inferential role in non-parametric function estimation. Many attempts have been made to construct confidence intervals based on different estimation methods. The objective of this project is, based on a kernel estimator $\hat{m}(x)$ with *global bandwith*, to construct a pointwise confidence interval for for $m(x)$ in the model:

$$Y_i = m(x_i) + 0.5\varepsilon_i, \qquad i = 1, ..., n$$

where $x_i = i/n \ \in (0, 1]$, $\varepsilon_i \sim \mathcal{N}(0, 1)$, and

$$m(x) = (sin(2\pi x^3))^3$$

1 Construction of the Confidence Interval

The confidence interval we will use in this project is the one such that :

$$P\left(m(x) \in \left[\hat{m}_h(x) - z_{\alpha/2}\left(\frac{V(x)}{nh}\right)^{1/2}, \ \hat{m}_h(x) + z_{\alpha/2}\left(\frac{V(x)}{nh}\right)^{1/2}\right]\right) \approx 1 - \alpha$$

where

- $\hat{m}(x)$ is a non-parametric kernel estimator of $m(x)$ with smoothing parameter h
- $z_{\alpha/2}$ the two-sided α-quantile of the standard normal distribution
- $V(x)$ the asymptotic variance $\frac{\sigma^2(x)}{f(x)}R(K)$

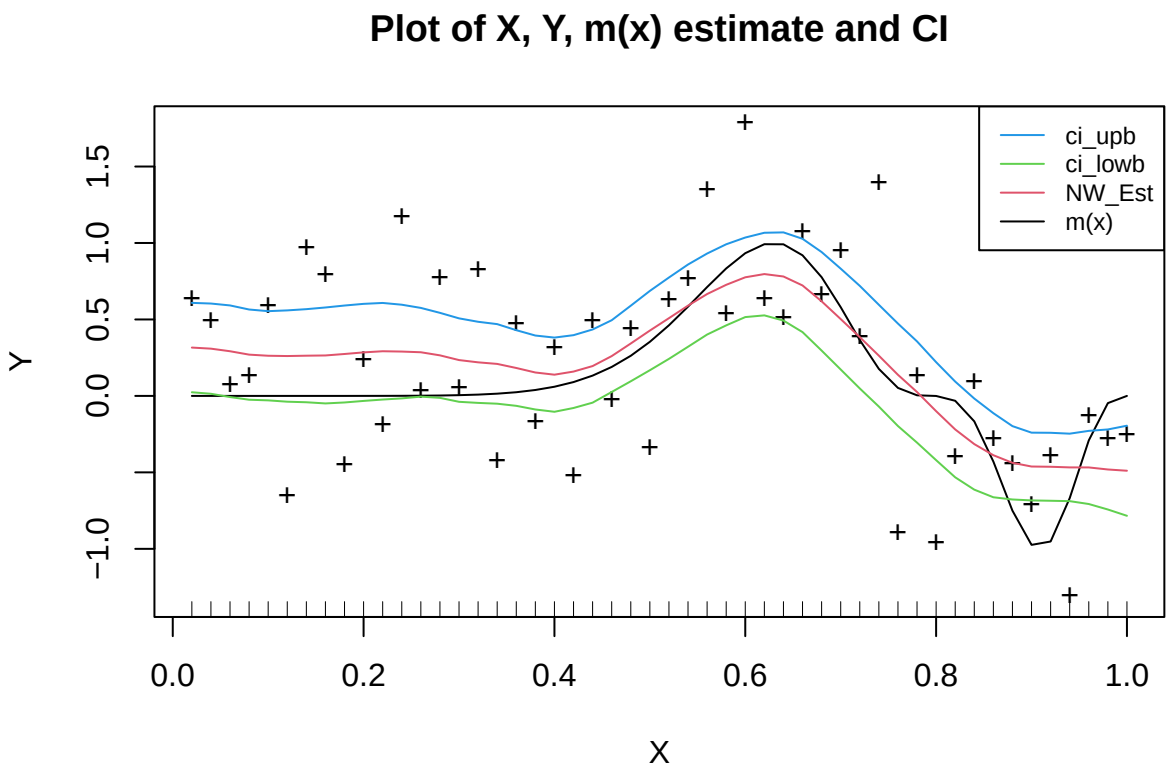
Recall that in this special case, we are in a *fixed design*

Little set up

To compute the confidence interval, we will need some additional parameters (such as the second order integral of the kernel, the variance at each estimate $\hat{m}(x)$). Since the *lokerns* package gives no information about the kernel used in the implementation, we decide to make the implementation ourselves. Thus our implementation may suffer a bit of accuracy or optimality.

To construct the estimate, we will be using :

- the *Epanechnikov* kernel, and
- for the estimation of the non parametric function, we will be using the $Nadaraya - Watson$ estimator.
- the bandwidth will be computed using the cross-validation rule. The function *bw.ucv* available in R will be used to that purpose
- to compute the variance of chosen x, will use h_2 from the plug-in method of Sheather and Jones
- to estimate the function $f(x)$, will still use the *Epanechnikov*



Looking at the plot, we can see that the confidence interval doesn't cover all the points. Actually, the confidence interval we construct, suffers from the dispersion of the points around some particular area. Henceforth, one can divide the plot into three area:

- going from $x = 0$ to around $x = 0.4$. In this area, we can remark that the point are some kind of align. But still difficult to find a particular trend in the points.
- from $x = 0.5$ to $x = 0.7$, where we observe a kind of mountain and all x get a positive y (so to say).
- and the last part from $x = 0.8$ to the end, where the corresponding y are now mostly negative.

Choice of reference points

Since our confidence interval doesn't catch most of the dispersion in the cloud of points, let's perform a Monte-Carlo simulations to see the coverage of the confidence interval.

To do that and based on what we know, we decide to choose six different points (x_1, x_2, x_3, x_4, x_5 and x_6) : two from each region we have defined previously. The choice of this points are made based on the structure of the dispersion of the points on the plot.

- $x_1 = 0.18$ and $x_2 = 0.39$ (chosen from the slightly linear part of the plot)
- $x_3 = 0.52$ and $x_4 = 0.68$ (chosen from the first mountain part of the plot)
- $x_5 = 0.81$ and $x_6 = 0.92$ (chosen from the valley part of the plot)

Now, our objective is to perform Monte-Carlo simulations around this points to evaluate the coverage of the Monte-Carlo confidence interval.

2 Monte-Carlo Simulations

For this simulations, we will take $R = 1000$ repetitions of sample of size : 50, 100, 500. Draw the confidence interval for $\hat{m}(x)$ for $x_i,\ i \in [1, .., 6]$ for 95% and 90% of confidence level and evaluate their coverage regarding $m(x)$.

The confidence intervals will be computed using two approaches :

- one based on the asymptotic formula provided in the previous session, and
- the quantile-based confidence interval

These are resumed in the following tables :

Table 1: Result of confidence intervals coverage for the asymptotic formula

	$x_1 =$ 0.18			$x_2 =$ 0.39			$x_3 =$ 0.52			$x_4 =$ 0.68			$x_5 =$ 0.81			$x_6 =$ 0.92		
	50	100	500	50	100	500	50	100	500	50	100	500	50	100	500	50	100	500
95%	0.92	0.93	0.94	0.9	0.92	0.93	0.88	0.89	0.94	0.67	0.68	0.87	0.73	0.67	0.86	0.14	0.06	0.25
90%	0.87	0.89	0.88	0.83	0.86	0.89	0.81	0.83	0.87	0.62	0.56	0.8	0.61	0.58	0.79	0.07	0.03	0.14

Table 2: Result of coverage of the quantile-based confidence intervals for Monte-Carlo distribution of $\hat{m}(x)$

	$x_1 =$ 0.18			$x_2 =$ 0.39			$x_3 =$ 0.52			$x_4 =$ 0.68			$x_5 =$ 0.81			$x_6 =$ 0.92		
	50	100	500	50	100	500	50	100	500	50	100	500	50	100	500	50	100	500
95%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
90%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

From the table 1, we can see that in the asymptotic world:

- the coverage gets better when the sample size increases.
- but still the coverage depend on the reference point: reference point around "mountain" or "valley" have lower coverage.
- the confidence level tends to have an impact on the coverage level. 95% confidence level tends to have high coverage level compares to 90% confidence level.

From table 2, what is observe is that the quantile based method tends to have a high coverage.

3 Possible improvements

Note that for this project, we have used from set up : choice of the kernel, of the bandwidth and some order parameters. Thus, our implementation produce results based on those sets up. One may change the set up :

- choice another kernel instead of *Epanechnikov*
- choice another bandwidth maybe $1.06\sigma n^{-1/5}$

- use Gasser-Muller kernel estimation or other non-parametrics regression methods

All this different set up may lead to different results. Also, for the Monte-Carlo estimation and specially to evaluate the coverage of the confidence intervals, many methods can be used to construct the simulated confidence interval. All this may have an impact of the results.

Conclusion

Based on the previous analysis, the coverage of the confidence interval of $m(x)$ from a kernel estimator with global bandwidth, depends strongly on the the choice of the reference point. Point in region with high variabilities tends to be difficult to estimate accurately. Nevertheless, increasing the sample size may tends to reduce this effect but still to a lesser extent.

Appendix: All code for this report

```
knitr::opts_chunk$set(echo = FALSE, cache = TRUE, comment = "", include = TRUE, warning = FALSE, messag
knitr::opts_chunk$set(echo = FALSE, comment = NA, warning = FALSE, cache = TRUE, message = FALSE)
library(ggplot2)
theme_set(theme_bw())
library(lokern)
set.seed(28112022)

# sample size
n <- 50

# m(x) the known mean the regression
m <- function(x) (sin(2*pi*x^3))^3

# X from uniform distribution
X <- seq(from =1, to = n)/n

# Y obtain from formula
Y <- m(X) + 0.5*rnorm(n, 0,1)

# chosen points to estimate m(x)
# x <- sort(sample(x = X, size = n/2))

x <- seq(from = min(X), to = max(X), length = n)

# global bandwidth to use
h <- bw.ucv(X)

# kernel to use : epanechnikov
epanKer <- function(u) 0.75*(1-u^2)*(abs(u) <=1)

# Nadaraya-Watson estimate to use
nwregEst <- function(x, X, Y, h, epanKer) sum(Y*epanKer((x-X)/h))/sum(epanKer((x-X)/h))

# NW on x
nwEst <- sapply(X, function(x) nwregEst(x, X, Y, h, epanKer))


# residu for whole data
residu <- Y - sapply(X, function(x) nwregEst(x, X, Y, h, epanKer))

# integral of square of kernel
R_K <- integrate(function(u) epanKer(u)^2, -Inf, Inf)$value

# sigma_hat for chosen x
h2 <- bw.SJ(X)
sigma_hat <- function(x) sum(epanKer((X-x)/h2)*residu^2)/sum(epanKer((X-x)/h2))


# estimation of f(x)
fest <- function(x) (1/50)*sum((1/h2)*epanKer((X-x)/h2))
```

```r
# lowerbound of CI at 95%
lowb <- function(xi) nwregEst(xi, X, Y, h, epanKer) - 1.96*(sqrt(1/(n*h))*sqrt(R_K * sigma_hat(xi) /fes

lowerbound <- rep(NA, 50)

for (j in 1:50){
  lowerbound[j] <- lowb(x[j])
}


# upperband of CI at 95%

upb <- function(xi) nwregEst(xi, X, Y, h, epanKer) + 1.96*(sqrt(1/(n*h))*sqrt(R_K * sigma_hat(xi) /fest


upperBound <- rep(NA, 50)
for (j in 1:50){
  upperBound[j] <- upb(x[j])
}

plot(X, Y, pch = "+")
lines(X, m(X), type = "l", col = 1)
lines(X, nwEst, type = "l", col = 2)
lines(X, lowerbound, col = 3)
lines(x, upperBound, col = 4)
rug(X)
title(main = "Plot of X, Y, m(x) estimate and CI")
legend("topright", legend = c("ci_upb", "ci_lowb", "NW_Est", "m(x)"), col = c(4,3,2,1), lty = c(1,1,1,1
x1 <- 0.18; x2 <- 0.39; x3 <- 0.52; x4 <- 0.68; x5 <- 0.81; x6 <- 0.92

m1 <- m(0.18) ; m2 <- m(0.39); m3 <- m(0.52); m4 <- m(0.68) ; m5 <- m(0.81); m6 <- m(0.92)
ci_construct <- function(x_cho, m_cho, n, level){

  X50 <- seq(from =1, to = n)/n
  h_to_use <- bw.ucv(X50)
  Y50 <- m(X50) + 0.5*rnorm(n, 0,1)


  # integral of square of kernel
  R_K <- integrate(function(u) epanKer(u)^2, -Inf, Inf)$value

  # sigma_hat for chosen x
  h2 <- bw.SJ(X50)
  sigma_hat <- function(x) sum(epanKer((X50-x)/h2)*residu^2)/sum(epanKer((X50-x)/h2))


  # estimation of f(x)
  fest <- function(x) (1/n)*sum((1/h2)*epanKer((X50-x)/h2))

  coverage <- rep(NA, 1000)
  # mesti <- rep(NA, 1000)

  for (j in 1:1000){
```

```
  # resampling from X50
  Xsa <- sample(X50, replace = TRUE)


  Ysa <- m(Xsa) + 0.5*rnorm(n, 0,1)

  mhat <- nwregEst(x_cho, Xsa, Ysa, h_to_use, epanKer)

  # mesti[j] <- mhat

  # residu for whole data
  residu <- Ysa - sapply(Xsa, function(x) nwregEst(x, Xsa, Ysa, h_to_use, epanKer))

  lb <- mhat - level*(sqrt(1/(n*h_to_use))*sqrt(R_K * sigma_hat(x_cho) /fest(x_cho)))

  ub <- mhat + level*(sqrt(1/(n*h_to_use))*sqrt(R_K * sigma_hat(x_cho) /fest(x_cho)))

  coverage[j] <- (lb < m_cho & m_cho < ub)*1
  }

  return(round(mean(coverage),2))

}
ci2_construct <- function(x_cho, m_cho, n, level){

  X50 <- seq(from =1, to = n)/n
  h_to_use <- bw.ucv(X50)
  Y50 <- m(X50) + 0.5*rnorm(n, 0,1)

  coverage <- rep(NA, 1000)

  for (i in 1:1000){
    mesti <- rep(NA, 500)

    for (j in 1:500){

      # resampling from X50
      Xsa <- sample(X50, replace = TRUE)

      Ysa <- m(Xsa) + 0.5*rnorm(n, 0,1)

      mesti[j] <- nwregEst(x_cho, Xsa, Ysa, h_to_use, epanKer)

     }

    lb <- quantile(mesti, level, na.rm = TRUE)

    ub <- quantile(mesti, 1-level, na.rm = TRUE)

    coverage[i] <- (lb <= m_cho & m_cho <= ub)*1

  }
```

```
  return(round(mean(coverage),2))

}
```