Université Catholique de Louvain
Institut de Statistique

LSTAT2150
Non-parametric statistics
Smoothing methods

Project number 7

December 4, 2020

LAMY Lionel

Contents

1 Introduction

Let $\{(X_i, Y_i)\}_{i=1}^n$, $n = 100$, follow the regression model :

$$Y_i = m(X_i) + 0.5\,\varepsilon_i, \quad i = 1, \ldots, n$$

$$\text{where } X_i \text{ i.i.d} \sim \text{Uniform}\,[0{,}1],\ \{\varepsilon_i\} \text{ i.i.d} \sim \mathcal{N}(0,1) \text{ and } m(x) = (\sin(2\,\pi\,x^3))^3$$

The objective of this project is to compare the performance between a parametric polynomial regression estimator and a nonparametric Nadayara-Watson regression estimator. To do so, we will first try to determine the optimal polynomial order and bandwidth by varying over a grid of possible values and selecting those values that minimize the MISE (or the MSSE) with respect to the true function $m(x)$. Then, we will inquire about the bias and the variance. Finaly, we will investigate the MSE at locations we feel are relevant in order to conclude from the performance of our regressions.

2 Quality assessment

As announced in the introduction above, we will use estimators to judge and compare the quality of our models. As such, it seems important to us to define them beforehand.

2.1 Bias

The bias mesures the systematic deviation of the estimator from the true density. If the estimator tends to zero when the number of observations tends towards infinity, then it is considered asymptotically unbiased.

$$\text{Bias}\,[\widehat{m}(x)] = \text{E}[\widehat{m}(x)] - m(x)$$

2.2 Variance

The variance represents the mean of the squares of the deviations from the true mean and is used to quantify the dispersion of the estimates.

$$\text{Var}\,[\widehat{m}(x)] = \text{E}\,[(\widehat{m}(x) - \text{E}\,[\widehat{m}(x)])^2]$$

2.3 Mean squared error

The MSE is the average squared difference between the estimated values and the actual value which means that it characterizes the accuracy of an estimator.

$$\text{MSE}\,[\widehat{m}(x)] = \text{E}\,[(\widehat{m}(x) - m(x))^2]$$

2.4 Mean integrated squared error

We will mainly try to use this measurement to define the quality and accuracy of our models. It is the integration of the MSE on all possible values of x (in our case 0 to 1 as X_i follows an Uniform[0,1]).
It can also be computed as the mean of integrated squared errors (ISE).

$$\begin{aligned}\text{MISE}\,[\widehat{m}(\cdot)] &= \int \text{MSE}\,[\widehat{m}(x)]\;dx \\ &= \int \text{E}\,[(\widehat{m}(x) - m(x))^2]\;dx \;= \text{E}\int (\widehat{m}(x) - m(x))^2\;dx\end{aligned}$$

2.5 Mean sum of squared error

As an integral can be approximated by $\frac{1}{n}\sum_{i=1}^n$, we have a simpler equivalence of the MISE which saves us from having to solve an integral, which can sometimes be tricky.

$$\text{MSSE}[\widehat{m}(\cdot)] = \frac{1}{n}\sum_{i=1}^{n} \text{MSE}\,[\widehat{m}(x_i)]$$

3 Regression estimators

3.1 Nadayara-Watson (nonparametric)

3.1.1 Description

This estimator, published in 1964 by Nadaraya and Watson, proposes to consider the function m(x) as a locally weighted average of Y_i given observations X_i and try to estimate it using a kernel as a weighting function. The motivation behind it comes from :

$$m(x) = E[Y|X = x] = \int y \, \frac{f(x, y)}{f(x)} \, dy$$

Where the numerator $f(x, y)$ and denominator $f(x)$ can be estimated via a kernel density estimator :

$$f(x, y) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) Y_i$$

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)$$

Combining all in one, we therefore have the NW estimator, with h as the bandwidth of the kernel K:

$$\widehat{m}_{NW}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) Y_i}{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)}$$

The Kernel used is this work is the Gaussian, defined as $K_G = \exp\left(-\frac{1}{2}u^2\right)(\sqrt{2\pi})^{-1}$ or simply **dnorm**(u) in R. Notice that the equations above show that the bandwidth is what affects the estimator the most, so the choice of a Kernel isn't that much of a matter.

3.1.2 Optimal bandwidth selection

In order to find the best bandwidth, we code a R function that compute M Nadayara-Watson regressions for each value of a sequence of possible bandwidths and returns the one that minimize the MISE/MSSE. First, we tried bandwidths ranging from 0.01 to 0.50 in increments of 0.01 and with M=2000. We obtained $h = 0.04$ as result. For the sake of accuracy, we started the experiment again with possible values ranging from 0.03 to 0.05 per 0.001 increase. We thus found that 0.035 seems to be the best bandwidth. However, on the graph below we can easily see that the estimation does not quite fit the true function $m(x)$.

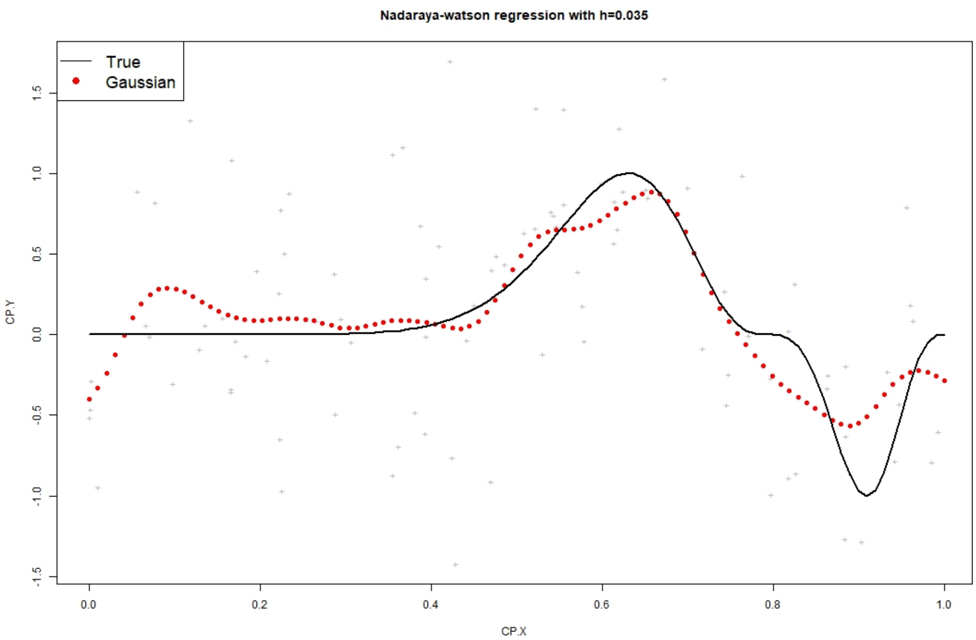

Figure 1: Fit of the NW estimator with the optimal h and n=100
An animated version is available here

We also wanted to know what would be the best h with a smaller sample size, and to verify in the process that the MSSE is a good approximation of the MISE. This appears to be confirmed in the table below. We also observe that as the sample size increases, the estimators and bandwidths appear to decrease.

n	Minimized		Bandwidth	
	MISE	MSSE	MISE	MSSE
25	0.6493	0.6411	0.145	0.145
50	0.1073	0.1122	0.127	0.124
100	0.0306	0.0308	0.035	0.035

Table 1: Evolution of the MISE/MSSE and the optimal bandwidth by n

3.1.3 Bias and variance

TODO

3.2 Polynomial (parametric)

We will use the following R command : **lm**(Y $\sim$ **poly**(X, degree)). The latter allows us to avoid using a long formula with multiple powers, thanks to the poly function that returns a matrix of orthogonal polynomials. Our goal right now is to find out what is the best degree to use to fit the real $m(x)$ function.

Unlike non-parametric regression, we were unable to find a way to calculate the MISE for linear regression and therefore sought to minimize the MSSE. Although this does not affect the results - as we saw earlier -, we wanted to make the point.

n	MSSE	Polynomial degree	Decimal degree
25	0.0697	5	5.40
50	0.0436	6	6.66
100	0.0286	6	6.77

Table 2: Evolution of the MSSE and optimal polynomial degree by n

TODO

4 Investigation

It is interesting to note that since the error is distributed according to a gaussian distribution with $E(\epsilon) = 0$ and $Var(\epsilon) = 1$, we know by theory that with a number of observations tending towards infinity, the error will tend towards zero, thus leading the Mean Squared Error towards zero as well. However, we are going to verify this numerically with 5000 Monte-Carlo simulations.

The points that we find interesting to explore are [0, 0.2, 0.45, 0.63, 0.78, 0.91, 1] and are represented on the graph below by a red dashed line. We chose the two points at the extremities, the point between the maximum and the minimum as well as these last two and an additional point near the center.

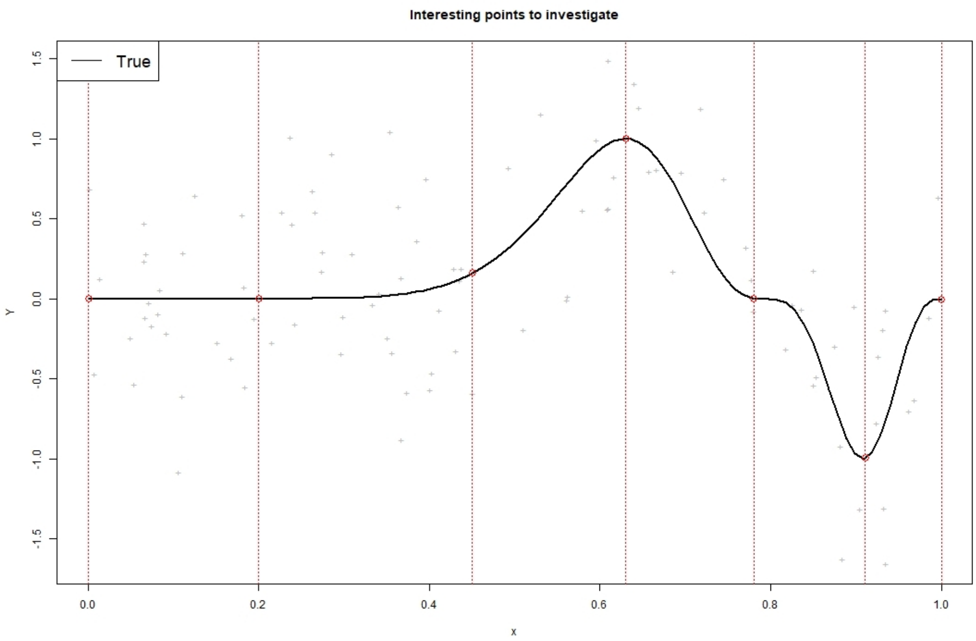

Figure 2: Interesting points marked by a red line

4.1 Nadayara-Watson

TODO

Point	n=25	n=50	n=100	n=1000	Mean
0	0.1441	0.0889	0.0424	0.0040	0.0699
0.2	0.0909	0.0440	0.0212	0.0020	0.0395
0.45	0.0992	0.0450	0.0220	0.0026	0.0422
0.63	0.1120	0.0572	0.0308	0.0111	0.0528
0.78	0.0996	0.0458	0.0235	0.0034	0.0431
0.91	0.2459	0.1562	0.1146	0.0864	0.1508
1	0.3630	0.1951	0.1033	0.0454	0.1767
Mean	**0.1650**	**0.0903**	**0.0511**	**0.0221**	0.0821

Table 3: NW regression MSE evolution

Point	n=25	n=50	n=100	n=1000	Mean
0	0.1941	0.1643	0.1191	0.0270	0.1261
0.2	0.0713	0.0338	0.0179	0.0032	0.0316
0.45	0.1261	0.0769	0.0513	0.010	0.0662
0.63	0.2339	0.1483	0.0819	0.0327	0.1392
0.78	0.3828	0.2292	0.1406	0.0224	0.1938
0.91	0.7049	0.2811	0.1394	0.0641	0.2974
1	0.4177	0.2481	0.1435	0.0931	0.2256
Mean	**0.3130**	**0.1688**	**0.0991**	**0.0362**	0.1543

Table 4: Linear regression MSE evolution

TODO

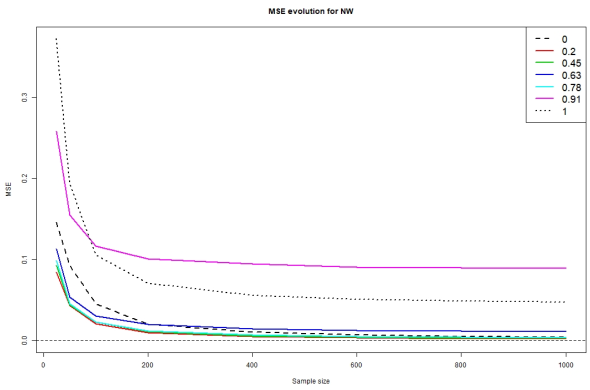

Figure 3: NW-MSE evolution

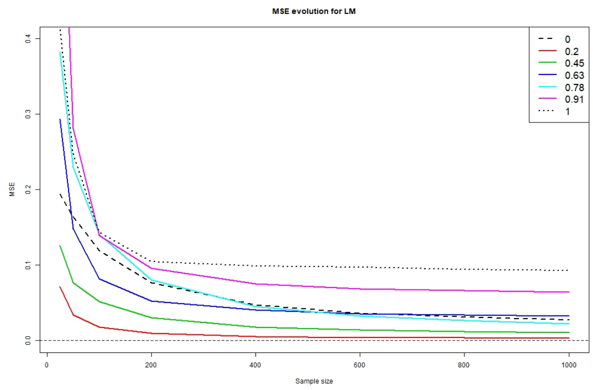

Figure 4: LM-MSE evolution

4.2 Polynomial

TODO

5 Conclusion

TODO

A Appendix

Info: All the files (including figures) and the code can be found on GitHub :
https://github.com/lamylio/LSTAT2150-Project

A.1 Figures

A.2 Code

A.2.1 Main

```
set.seed(2020)

# Import custom R files
source("./sources/setup.r")
source("./sources/minimization.r")

# Global constant X and Y (used for comparison)
CP.X = X(n)
CP.Y = Y(CP.X)
CP.x = seq(0, 1, length = n)

# Kernels
Knorm = function(u) dnorm(u) #Gaussian kernel
Kunif = function(u) (abs(u) <= 1) * 0.5 #Uniform kernel

# NW Estimator
NW.regEst <- function(x, X, Y, h, K) sum(Y * K((x - X)/h))/sum(K((x - X)/h))


# Please see Minimization.r
# Use ni=100, M=100 by default but better results with higher values

# NW.MISE = NW.minimizeMISE() # 0.041
# NW.MSSE = NW.minimizeMSSE() # 0.036

NW.MISE = NW.minimizeMISE(h.test = seq(0.035, 0.037, 0.0001), M=200) # more precise
LM.MSSE = LM.minimizeMSSE()

# Optimal bandwidth
NW.h = 0.0355 # NW.MISE[2]
# Optimal degree
LM.poly = 6 # round(LM.MSSE[2])


CP.to_check = c(0,0.2,0.45,0.63,0.78,0.91,1)

# Obtain the MSE for specific points
NW.investigation = function(M=5000, n=100){

  SE = matrix(NA, M, length(CP.to_check))

  for (i in 1:M){
    boot.X = runif(n)
    boot.Y = Y(boot.X)
    SE[i,] = (sapply(
      CP.to_check,
      function(xi) NW.regEst(xi, boot.X, boot.Y, NW.h, Knorm)) - m(CP.to_check)
    )^2
  }

  MSE = colMeans(SE)
```

```
  return(MSE)
}
```

A.2.2 Setup

```
###
# Given in the project statement
###

n=100

X = function(n=100) runif(n)
m = function(x) (sin(2*pi*x^3))^3
Y = function(X=runif(n)) m(X) + rnorm(length(X), 0, 0.5) # transformed into a function
```

A.2.3 Minimize

```
###
# Nadayara-Watson and Linear Model with poly
# Returns: minimized MISE/MSSE and bandwidth/poly associated
##

library(progress)

NW.minimizeMISE = function(
  Kernel = Knorm, h.test = seq(0.01, 0.15, 0.001), M = 100, ni = 100
){
  best = Inf
  best_h = 0

  pb = progress_bar$new(
    format = "
      [NW MISE] Best: :bmise (:bh) | Current: :cmise (:ch)
      [:bar] :current/:total |:percent | :elapsed
    ",
    total = length(h.test)
  )
  Sys.sleep(0.1)
  pb$tick(0)

  for (h in h.test){

    ISE = matrix(NA, M, 1)

    for (i in 1:M){
      boot.X = runif(ni)
      boot.Y = Y(boot.X)
      boot.ISE = integrate(
        function(xi)
          (sapply(xi, function(x) NW.regEst(x, boot.X, boot.Y, h, Kernel)) - m(xi))^2,
        lower=0, upper=1
      )$value
      ISE[i,] = boot.ISE
    }

    MISE = round(mean(ISE), 6)

    if (MISE < best){
      best = MISE
      best_h = h
    }
    pb$tick(tokens=list(bmise=best, bh=best_h, cmise=MISE, ch=h))
```

```r
  }
  sprintf("[NW MISE]: %.6f with %.6f as bandwhidth", best, best_h)
  return (c(best, best_h))
}


NW.minimizeMSSE = function(
  Kernel = Knorm, h.test = seq(0.01, 0.15, 0.001), M = 100, ni = 100
){
  best = Inf
  best_h = 0

  pb = progress_bar$new(
    format = "
      [NW MSSE] Best: :bmsse (:bh) | Current: :cmsse (:ch)
      [:bar] :current/:total |:percent | :elapsed
    ",
    total = length(h.test)
  )
  Sys.sleep(0.1)
  pb$tick(0)
  for (h in h.test){
    SE = matrix(NA, M, ni)
    for (i in 1:M){
      boot.x = seq(0, 1, length = ni)
      boot.X = runif(ni)
      boot.Y = Y(boot.X)
      SE[i,] = (
        sapply(boot.x, function(xi) NW.regEst(xi, boot.X, boot.Y, h, Kernel)) - m(boot.x)
      )^2
    }

    MSE = colMeans(SE)
    MSSE = round(mean(MSE),6)

    if (MSSE < best){
      best = MSSE
      best_h = h
    }
    pb$tick(tokens=list(bmsse=best, bh=best_h, cmsse=MSSE, ch=h))
  }
  sprintf("[NW MSSE]: %.6f with %.6f as bandwhidth", best, best_h)
  return (c(best, best_h))
}

# -----
# Haven't found a way to compute MISE with LM
# But as MSSE approx. MISE, this is fine.

LM.minimizeMSSE = function(
    poly.test = seq(5, 20, 0.01), M = 200, ni = 100, raw=F)
{
  best = Inf
  best_poly = 0

  pb = progress_bar$new(
    format = "
      [LM MSSE] Best: :bmsse (:bp) | Current: :cmsse (:cp)
      [:bar] :current/:total |:percent | :elapsed
    ",
    total = length(poly.test)
  )
  Sys.sleep(0.1)
```

```
  pb$tick(0)
  for (p in poly.test){
    SE = matrix(NA, M, ni)
    for (i in 1:M){
      boot.X = runif(ni)
      boot.Y = Y(boot.X)
      SE[i,] = (lm(boot.Y ~ poly(boot.X, degree = p, raw=raw))$fitted - m(boot.X))^2
    }

    MSE = colMeans(SE)
    MSSE = round(mean(MSE),6)

    if (MSSE < best){
      best = MSSE
      best_poly = p
    }
    pb$tick(tokens=list(bmsse=best, bp=best_poly, cmsse=MSSE, cp=p))
  }
  sprintf("[LM MSSE]: %.6f with %f as poly", best, best_poly)
  return (c(best, best_poly))
}
```