Estadística no paramétrica

Regresión no paramétrica I: Métodos basados en kernel

Jorge de la Vega Góngora

Departamento de Estadística,
Instituto Tecnológico Autónomo de México

14 de abril de 2023

Estimadores basados en kernel

Estimador de Gasser-Müller I

- Otro estimador basado en kernel es el estimador de Gasser-Müller (1979), que usa las áreas del kernel como los pesos.
- Este estimador tiene varianza mayor que el estimador de Nadaraya-Watson, pero menor sesgo (para una prueba, ver Fan (1992).
- Se consideran una partición $-\infty = a_0 \leq a_1 \leq a_2 \leq \cdots \leq a_n \leq a_{n+1} = \infty$ y definimos los puntos medios $s_i = (a_i + a_i + 1)/2$. Entonces

$$\hat{m} = \sum_{i=1}^{n} Y_i \int_{s_{i-1}}^{s_i} K\left(\frac{u-x}{h}\right) du$$

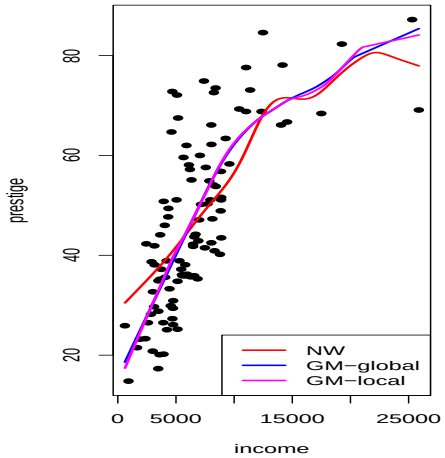
Se toman los valores de la muestra ordenados como puntos de la partición: $a_i = X_{(i)}$. El estimador de Gasser Müller es el que minimiza la función $\sum_{i=1}^{n} w_i(x)(Y_i - \theta)^2$ con los pesos dados por $w_i = \int_{s_{i-1}}^{s_i} K\left(\frac{u-x}{h}\right) du$.

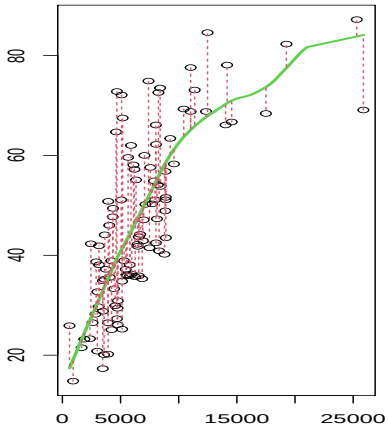
Estimador de Gasser-Müller II

- El paquete `lokern` implementa este estimador en un contexto más amplio. Los fundamentos del paquete se encuentran en el paper: Herrmann (2012): Local bandwidth choice in kernel regression estimation.

```
library(lokern)
par(mfrow=c(1,2))
plot(income, prestige, pch = 16)
# Se agrega un estimador de Nadaraya-Watson para comparación
lines(ksmooth(income, prestige, "normal", bandwidth = 5000), lwd = 2, col = "red")
# es necesario usar is.rand = F para usar G-M
# No es necesario definir la ventana porque la calcula
lines(glkerns(income,prestige, is.rand = F), col = "blue", lwd = 2)
lines(lokerns(income,prestige, is.rand = F), col = "magenta", lwd = 2)
legend("bottomright", legend = c("NW","GM-global","GM-local"),
       col = c("red","blue","magenta"), lty = rep(1,3), lwd = 2)
# también podemos graficar directamente el ajuste (con los residuales)
plot(lokerns(income,prestige, is.rand = F))
```

Estimador de Gasser-Müller III





Estimador de Priestley y Chao (1972)

- Priestley y Chao (1972) propusieron un estimador de kernel que incorpora la distancia entre puntos observados X_i adyacentes, además de los pesos con kernel K:

$$\hat{m}(x) = \sum_{i=1}^{n} \frac{Y_i(X_{(i)} - X_{(i-1)})}{h} K\left(\frac{x - X_{(i)}}{h}\right)$$

- Intuitivamente, se puede interpretar la introducción de este componente como parte de la estimación de la densidad empírica, $\hat{f}_n(x) = (n(X_{(i)} - X_{(i-1)}))^{-1}$ para $x \in (X_{(i-1)}, X_{(i)}]$.
- Este método está implementado en `R` más como método de apoyo para estimar regresión en frontera en el paquete `npbr`, pero no para estimación en general de suavizadores. Es más común encontrarlo programado en paquetes de `Matlab`.

Polinomios locales I

- Los modelos que se mencionaron anteriormente llevan a cabo un ajuste local constante, es decir, minimizan la suma de cuadrados ponderada $\sum_{i=1}^{n}(Y-i-\theta)w_i$ para diferentes funciones de peso w_i.
- Bajo la consideración de que la funcion $m(x_0)$, donde x_0 es un punto focal, puede ser bien ajustada en una vecindad pequeña de x_0 por un polinomio de orden p

$$m(x_0) \approx \sum_{j=0}^{p} \alpha_j (x_0 - x)^j, \quad \alpha_j = \frac{m^{(j)}(u)}{j!}$$

- En lugar de minimizar la suma de cuadrados ponderada, el estimador de polinomio local minimiza la función de $\alpha = (\alpha_0, \dots, \alpha_p)$:

$$RSS(\alpha) = \sum_{i=1}^{n} \left(Y_i - \sum_{j=0}^{p} \alpha_j (X_j - x)^j \right)^2 K\left(\frac{X_i - h}{h} \right)$$

Podemos simplificar la función anterior con la notación $RSS(\alpha) = \sum_{i=1}^{n} w_i e_i^2 = \sum_{i=1}^{n} w_i (\hat{Y}_i(\alpha) - Y_i)^2$.

Polinomios locales II

- Noten que $\widehat{Y}|X = x_0 = \widehat{\alpha}_0$. Por otra parte, también noten que si $p = 0$, entonces los polinomios son constantes, y estamos en el caso de Nadaraya-Watson.

Representación matricial de los polinomios locales I

- Construimos la matriz

$$\mathbf{X} = \begin{pmatrix} 1 & X_1 - x & (X_1 - x)^2 & \cdots & (X_1 - x)^p \\ 1 & X_2 - x & (X_2 - x)^2 & \cdots & (X_2 - x)^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_n - x & (X_n - x)^2 & \cdots & (X_n - x)^p \end{pmatrix}$$

  junto con la matriz de pesos diaginal $\mathbf{W} = diag(K_k(X_i - x))$ y el vector de respuesta $\mathbf{Y} = (Y_1, \ldots, Y_n)'$.

- Entonces $RSS(\alpha) = (\mathbf{Y} - \mathbf{X}\alpha)'\mathbf{W}(\mathbf{Y} - \mathbf{X}\alpha)$. La solución es el estimador de mínimos cuadrados ponderados:

$$\hat{\alpha} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{Y}$$

  Lo que también muestra que el estimador es una combinación lineal de las observaciones disponibles $\hat{Y} = \mathbf{X}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{Y} = \mathbf{S}\mathbf{Y}$.

Representación matricial de los polinomios locales II

- La matriz $\mathbf{S} = \mathbf{X}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}$ de $n \times n$ se llama la *matriz de suavizamiento*, y tiene propiedades análogas a la *matriz sombrero* $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ que surge en regresión lineal.
- En regresión no paramétrica, los grados de libertad se pueden definir de muchas maneras (no equivalentes entre sí), por analogía a mínimos cuadrados:
  - $gl_1 = tr(\mathbf{S}) = \nu_1$
  - $gl_2 = tr(\mathbf{S}\mathbf{S}')$
  - $gl_3 = tr(2\mathbf{S} - \mathbf{S}\mathbf{S}')$
  - $gl_4 = tr(I - \mathbf{S})$
- En cada paquete que se utilice, se recomienda revisar cuál es la definición de los grados de libertad utilizados, para poder hacer comparaciones de manera consistente.
- La función `locfit` en el paquete del mismo nombre hace el ajuste y nos da información sobre el ajuste.

Representación matricial de los polinomios locales III

```
library(locfit)
```

```
locfit 1.5-9.7 2023-01-02
```

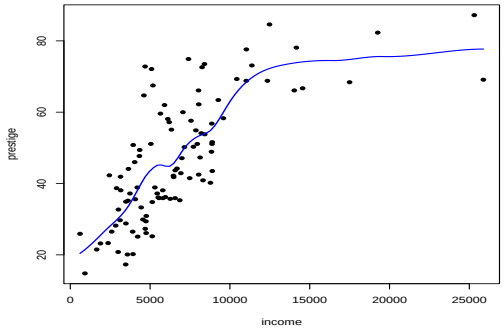
```
# nn = parámetro para determinar el alcance de los vecinos más cercanos
# deg = grado del polinomio a ajustar
(m1 <- locfit(prestige ~ lp(income, nn=0.5, deg=3)))
```

```
Call:
locfit(formula = prestige ~ lp(income, nn = 0.5, deg = 3))

Number of observations:          102
Family:  Gaussian
Fitted Degrees of freedom:       9.598
Residual scale:                  11.3
```

```
plot(income, prestige, pch = 16)
lines(m1, lwd = 2, col = "blue")
```

Representación matricial de los polinomios locales IV



- Los dos puntos claves de la regresión de polinomios locales son:
  1. Seleccionar p: Se recomienda usar un orden impar: si p es impar entonces el polinomio tiene la misma varianza que el polinomio de orden $p+1$, pero con menor sesgo.
  2. Seleccionar h o nn: La selección se puede realizar a través de validación cruzada, como se verá a continuación.

Selección del parámetro nn I

- El objetivo de seleccionar la ventana h (que en locfit es `nn`) es minimizar el error cuadrático medio en el valor focal x_0:

$$MSE(\hat{Y}|x_0) = \mathsf{E}(\hat{Y}|x_0 - \mu|x_0)^2 = Var(\hat{Y}|x_0) + sesgo^2(\hat{Y}|x_0)$$

- Para seleccionar h, se puede hacer,
  1. ya sea analizndo el comportamiento de los residuales $e_i = \hat{Y}_i - Y_i$, en términos de h, $RSS(h)$, o bien
  2. Utilizando validación cruzada: se ajusta un modelo dejando una observación fuera. Se elige la h que minimice

$$CV(h) = \frac{\sum_{i=1}^{n}(\hat{Y}_{-i}(h) - Y_i)^2}{n}$$

También se puede usar la función de variación cruzada generalizada,

$$GCV(h) = \frac{n}{(n-\nu_1)}\sum_{i=1}^{n}(\hat{Y}_{-i}(h) - Y_i)^2$$

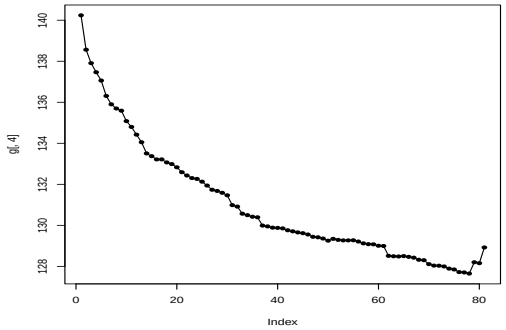
donde $\nu_1 = tr(\mathbf{S})$ con **S** la matriz de suavizamiento que definimos antes.

Selección de nn: Ejemplo I

- En el siguiente ejemplo calculamos vía validación cruzada el valor óptimo de la constante de suavizamiento.
- La función `gcv` en la columna 4 nos regresa la validación cruzada generalizada. Podemos ver que el valor óptimo para el parámetro de suavizamiento de acuerdo al criterio del gcv está alrededor de 0.97 en este ejemplo.

```
alfa <- seq(0.20,1,by=0.01)
n1 <- length(alfa)
g <- matrix(nrow=n1, ncol = 4)
for (k in 1:n1){
  g[k,] <- gcv(prestige ~ lp(income, nn=alfa[k], deg=1))
}
plot(g[,4],type="o",pch=16)
```

Selección de nn: Ejemplo II




```
tail(cbind(alfa,g[,4]))
```

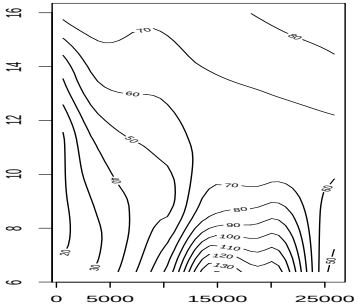
```
      alfa
[76,] 0.95 127.7222
[77,] 0.96 127.7049
[78,] 0.97 127.6503
[79,] 0.98 128.2009
[80,] 0.99 128.1568
[81,] 1.00 128.9225
```

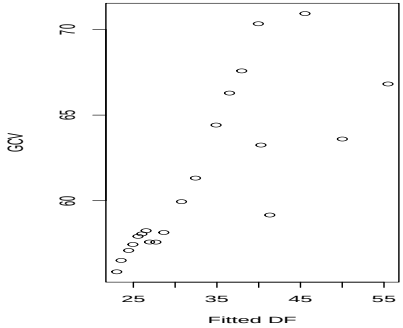
Ajuste con dos variables I

- Podemos hacer también un ajuste en dos variables. Agregaremos la variable de educación que forma parte del mismo conjunto de datos

```
par(mfrow=c(1,2))
m2 <- locfit(prestige ~lp(income,education,scale = T, nn=0.5),
             data=datos)
plot(m2)
gcvplot(prestige ~ income + education, data = datos, alpha = seq(0.1, 0.3, by = 0.01))
```

Ajuste con dos variables II





Ajuste con dos variables III

```
r <- summary(gcvplot(prestige ~ income + education, data = datos, alpha = seq(0.1, 0.3, by = 0.01)))
tail(r)
```

```
           df      GCV
[16,] 26.02679 58.06147
[17,] 25.55651 57.92043
[18,] 24.94333 57.44564
[19,] 24.43838 57.09504
[20,] 23.53892 56.51250
[21,] 23.01038 55.84793
```

```
#valor óptimo
seq(0.1,0.3,by=0.01)[which(min(r[,2]) == r[,2], arr.ind = T)]
```

```
[1] 0.3
```

Inferencia estadística para modelos de polinomios locales I

- Hemos visto que los valores ajustados son combinaciones lineales de las respuestas observadas:

$$\hat{Y}_i = \sum_{j=1}^{n} S_{ij} Y_j \text{ o } \hat{Y} = \mathbf{SY}$$

con diferentes maneras de definir grados de libertad.

- Los grados de libertad del residual serán $gl_{res} = n - gl_i$ y la varianza estimada del error es:

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} e_i^2}{gl_{res}}$$

En el modelo previamente ajustado, tenemos que $gl_1 = 16.752$, $gl_{res} = 102 - 16.752 = 85.248$ y $\hat{\sigma} = 7.13$.

- La varianza estimada del valor ajustado $\hat{Y}_i$ en $x = X_i$ es $\hat{Var}(\hat{Y}_i) = \hat{\sigma}^2 \sum_{j=1}^{n} S_{ij}$ y un intervalo de confianza del $(1-\alpha) \times 100$ para $\mu|x_0$ está dado por:

$$\hat{Y}_i \pm 2\sqrt{\hat{Var}(\hat{Y}_i)}$$

Inferencia estadística para modelos de polinomios locales II

- La banda de confianza se construye a partir de los valores puntuales.

Ejemplo. [Datos para inferencia con `locfit`]

```
(prediccion <- predict(m2,newdata = data.frame(income = c(2000,1000),
                                               education = c(10,20)), se.fit = T))

$fit
[1]  27.62854 121.63781

$se.fit
[1]  2.614976 92.132154

$residual.scale
      rv
7.133913
```

```
prediccion$fit[1] + c(-2,2)*prediccion$se.fit[1]
```

```
[1] 22.39858 32.85849
```

```
prediccion$fit[2] + c(-2,2)*prediccion$se.fit[2]
```

```
[1] -62.6265 305.9021
```

□

Inferencia estadística para modelos de polinomios locales III

- Se puede usar una prueba F para pruebas de hipótesis en modelos anidados (asumiendo que el modelo chico (ch) está anidado con el grande (gd)).

$$F = \frac{\left(\frac{RSS_{ch} - RSS_{gd}}{|gl_{ch} - gl_{gd}|}\right)}{\hat{\sigma}^2_{gd}} \sim F_{(|gl_{ch} - gl_{gd}|, gl_{gd})}$$

Ejemplo. [Prueba de hipótesis]

Por ejemplo, para el modelo dimensiona m_2 como el modelo grande y m_1 como el modelo chico:

```
m1 <- locfit(prestige ~lp(income,scale = T, nn=0.5), data=datos)
m2 <- locfit(prestige ~lp(income,education,scale = T, nn=0.5), data=datos)
df1 <- summary(m1)$dp[7]
df2 <- summary(m2)$dp[7]
(FF <- as.numeric(((df1*rv(m1) - df2*rv(m2))/(abs(df1-df2)))/rv(m1)))
```

```
[1] 0.1289399
```

```
pf(FF,abs(df1-df2),df1,lower.tail=F)
```

```
[1] 0.9968595
```

□

Modelos de vecinos cercanos

Vecinos cercanos

- En realidad el procedimiento de vecinos cercanos no es en sí un método nuevo, es simplemente otra forma de especificar el ancho de banda, considerando el porcentaje de puntos que son vecinos cercanos al punto de referencia.
- Este valor es especificado como un valor entre 0 y 1, que indica la proporción de puntos que se quieren considerar alrededor del punto base. En algunas funciones, a este número le llaman el `span`.
- Dado que se tiene que cumplir una proporción fija de observaciones alrededor del punto de referencia, este caso genera bins de longitud variable, por lo que correspondería a tener varios anchos de banda h_i para cada bin.
- El ejemplo más característico de la aplicación de este método es el estimador LOESS, pero puede ser aplicado en cualquier contexto de estimadores donde se tengan que formar bins.

LOESS I

- Cleveland (1979) introdujo una técnica llamada `LOWESS`, que significa *LOcally WEighted Scatterplot Smoothing*. Otro modelo derivado se llama `loess` que significa *LOcal regrESSion*.
- Considerando el contexto de una regresión lineal múltiple, se tiene un conjunto de regresores $\mathbf{x}_i = (X_{i1}, X_{i2}, \dots, X_{ip})$ para predecir Y_i. La adyacencia de los regresores se define como la distancia $d(\mathbf{x}, \mathbf{x}^*)$ con alguna definición de distancia.
- Cada punto $\mathbf{x}_j$ influencia la regresión en $\mathbf{c}^*$ de acuerdo a su distancia a aquel punto. La función `loess` se usa una función tricubo:

$$w_i = \left(1 - \left(\frac{d_i}{d_q}\right)^3\right)^3 I(|d_i/d_q| \leq 1)$$

donde sólo q de n puntos cercanos a $\mathbf{x}_i$ se consideran en la vecindad de $\mathbf{x}_i$ y

$$d_q = \underset{w \in V(x_i)}{\text{máx}} \{d(\mathbf{x}_i, \mathbf{x}_w)\}$$

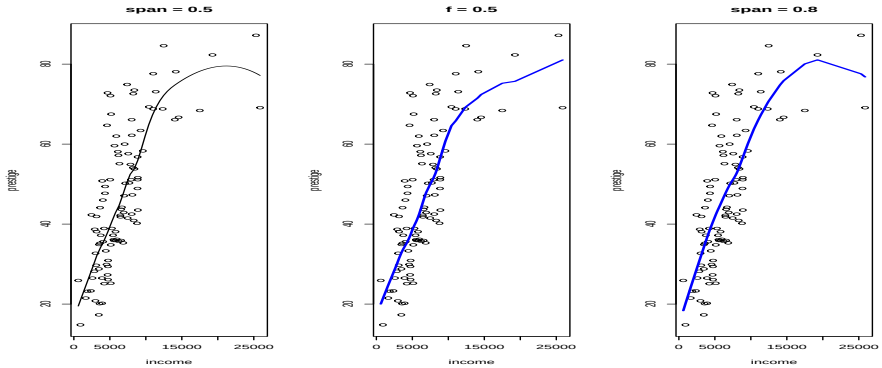
LOESS II

- Si q es grande, la curva `loess` será suave pero menos sensible a outliers. Conforme q se hace pequeña, el ajuste se parece más a una interpolación de los datos y la curva tiene muchos quiebres. Usualmente q se elige de tal manera que $0.10 \leq q/n \leq 0.25$. Usualemente se considera para construir el `loess` con un modelo lineal o cuadrático.
- Usualmente el `loess` sólo depende de un parámetro $\alpha = q/n$ y un polinomio local de primer o segundo orden. Pero requiere de muchos datos para que el ajuste trabaje bien.

Ejemplo. [Ajuste de loess]

```
par(mfrow = c(1,3))
scatter.smooth(income, prestige, span = 0.5, degree = 2, main  = "span = 0.5") #opción 1 loess
plot(income, prestige, main = "f = 0.5")
lines(lowess(income, prestige, f = 0.50), col = "blue", lwd = 2) #opción 2 lowess
plot(income, prestige, main = "span = 0.8")
m3 <- loess(prestige ~ income, span = 0.80)
fit3 <- predict(m3)
lines(income[order(income)], fit3[order(income)], col = "blue", lwd = 2) #opción 3 loess
```

LOESS III


span = 0.5
f = 0.5
span = 0.8
prestige
income
0
5000
15000
25000
20
40
60
80

loess para superficies I

```
library(RColorBrewer)
(m4 <- loess(prestige ~ income + education, span=0.4))

Call:
loess(formula = prestige ~ income + education, span = 0.4)

Number of Observations: 102
Equivalent Number of Parameters: 18
Residual Standard Error: 7.237

# aproximación de R^2
cor(prestige,m4$fitted)^2

[1] 0.8658069
```

Ahora creamos un grid para hacer nuestro ajuste

loess para superficies II

```r
grid.inc <- seq(min(income),max(income),length.out=50)
grid.edu <- seq(min(education),max(education),length.out=50)
grid.mar <- list(income = grid.inc, education = grid.edu)

# obten valores interpolados
prestige.ip <- predict(m4, expand.grid(grid.mar))
prestige.z <- matrix(prestige.ip, length(grid.inc),length(grid.edu))

# grafica los valores interpolados
nclr <- 8
plotclr <- brewer.pal(nclr, "PuOr")
plotclr <- plotclr[nclr:1] # ordena colores
image(grid.inc, grid.edu, prestige.z, col=plotclr)
contour(grid.inc, grid.edu, prestige.z, add=TRUE)
points(income, education)
```

loess para superficies III