IECD 2C2023: Entrega 1

Gonzalo Barrera, Ana Bianco, Pedro Cosatto

2023-09-06

Estimación No Paramétrica de la densidad: elección de la ventana

Fecha de Entrega: Viernes 15 de Septimbre 23:59 hs

Este TP se relizará en grupos de dos personas. Se debe entregar un archivo tipo .Rmd con lo comandos y un informe (que puede ser el mismo .Rmd renderizado a PDF o HTML) con los resultados. Los dos archivos que se entreguen deben contener en el nombre los apellidos de los dos integrantes del grupo.

(90 ptos.) Selección de ventana h por convalidación cruzada "deje-uno-afuera"

El objetivo de este ejercicio es implementar una función que seleccione la ventana h del estimador de densidad por núcleo gaussiano univariado, maximizando una pseudo–verosimilutd computada sacando de los datos una observación a la vez (leave–one–out cross–validation). Recordemos, que si observamos los datos $x_1, \ldots, x_n$, la ventana de log-verosimiltud por convalidación cruzada la calculamos como

$$h^*_{CV} = \arg\max_{h \in H} \left(\frac{1}{n} \sum_{i=1}^{n} \log \hat{f}_h^{(-i)}(x_i) \right) \text{ , donde} \hat{f}_h^{(-i)}(x_i) = \frac{1}{(n-1)h} \sum_{j \neq i} K\left(\frac{x_i - x_j}{h} \right)$$

1. (5 ptos.) Cree una función `bw.loocv` (en el estilo de `bw.nrd, bw.ucv` et al), que dependa de dos parámetros:
   - datos `x` que provengan de una muestra i.i.d. de una variable aleatoria univariada X,
   - `grilla.h`, una secuencia de valores de ventana a probar, opcionalmente vacía.
2. (20 ptos.) Para el caso en que la función `bw.loocv` se llama sin valor del parámetro `grilla.h`, asígnele una secuencia de 100 elementos, que cubra un rango de dos órdenes de magnitud alrededor del estimador de Silverman (ver `bw.nrd0`)
3. (20 ptos.) Escriba un for-loop donde, para cada valor de `grilla.h`,
   a. Para cada índice `i` entre 1 y `length(x)`, genere $\log \hat{f}_h^{(-i)}$, evalúelo en x_i y guarde el resultado.
   b. Compute el promedio de los valores en (a), que llamaremos $\hat{l}_{h,CV}$.
4. (15 ptos.) Devuelva una lista con tres elementos:
   - `h.CV`, el valor de `grilla.h` que maximiza la pseudo-logverosimilitud en `x`,
   - la retícula `grilla.h` donde maximiza la pseudo-logverosimilitud,
   - `loglikes`, los promedio de las log-verosimilitudes calculados en (3) para cada elemento de `grilla.h`.
5. Cargue la muestra provista en `entrega1.txt` (o directamente con `X <- muestra(1234)`). Trabajaremos con una grilla de ventanas entre 0.1 y 1 con paso 0.01 y usando `bw.loocv`, genere los siguientes dos gráficos:
   a. (15 ptos.) Un gráfico de los valores de log-verosimilitud promedio para cada valor de `grilla.h`, incluyendo líneas de referencia para el `h.CV` de `bw.loocv`, junto con los h sugeridos por Silverman (1986), Sheather & Jones (1991) y VC insesgada de R (`bw.nrd0, bw.SJ, bw.ucv`, respectivamente).
   b. (15 ptos.) Un gráfico de densidad de `x`, superponiendo las $\hat{f}_{h^*_M}$, donde $M \in$ {loocv, nrd0, SJ, ucv}

(0 ptos.) BONUS: Regla de un error estándar

Al estimar la log-verosimilitud por validación cruzada $\hat{l}_{h,CV}$, obtenemos un *conjunto* de estimadores de l, que a su vez se puede usar para aproximar la distribución del estimador. Así, si varios valores de `h` dan aproximadamente la misma $\hat{l}_{h,CV}$, preferiremos aquél h que resulte en la estimación más "parsimoniosa" o suave. Para el estimador de densidad por núcleos, esto puede interpretarse como el del h más grande, que resulta en una densidad estimada más suave.

Exploraremos el siguiente método de selección.

Referencia: ESL, sección 7.10.1

6. Reescriba la función `bw.loocv`, de manera que
   a. tome como argumento extra el número de errores estándares `n.se`, y a partir de él
   b. compute y devuelva `h.CV` y `loglikes` $\hat{l}_{h,CV}$ justo como antes, pero además
      - `loglikes.se`, el error estándar de las log-verosimilitudes computadas por convalidación cruzada LOO, que notaremos SE, y
      - `h.nSE` definido como el máximo `h` dentro de `grilla.h` tal que su promedio de `loglike` sea por lo menos $\hat{l}_{h.CV} - \text{n.se} \times SE(\hat{l}_{h.CV})/\sqrt{n}$.
7. Usando el retorno de esta nueva `bw.loocv`, grafique $\hat{l}_{h.CV}$ para la `grilla.h`, junto con líneas de error de n.se errores estándares, y las ventanas h correspondientes a n.se $\in \{0, 1, 2\}$.
   - ¿Qué efecto tienen estas ventanas en la estimación?

Referencia: notas de Tibshirani

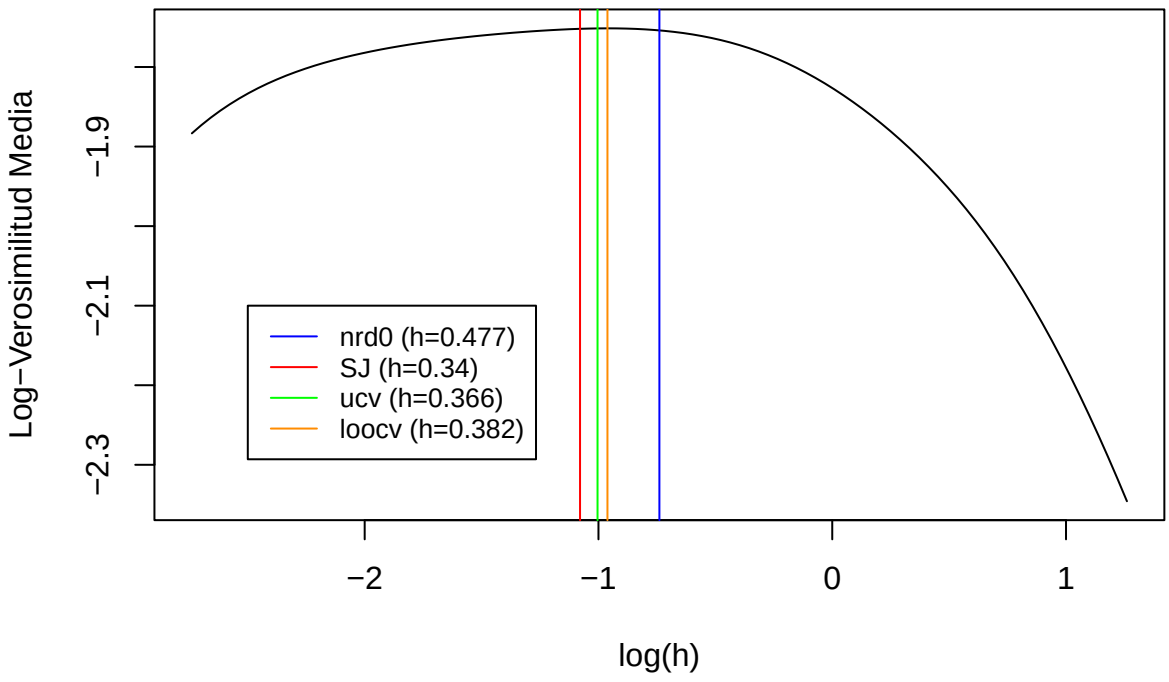
Abajo damos un bosquejo de la función a programar y el método generador de los datos como base. Esperamos les resulte de utilidad.

```
hacer_kde <- function(x, h=1) {
  kde <- function(t) {
    ret <- vector("numeric", length(t))
    for (i in seq_along(t)) {
        ret[i] <- (1 / h) * mean(dnorm((t[i] - x) / h))
    }
    return(ret)
  }
}



bw.loocv <- function(x, grilla.h=NA, n.se=1) {
  if (any(is.na(grilla.h))) {
    grilla.h <- bw.nrd0(x) * exp(seq(-2, 2, length.out=100))
  }
  n.h <- length(grilla.h)
  n.x <- length(x)

  loglikes <- vector("numeric", n.h)
  loglikes.se <- vector("numeric", n.h)
  for(j in 1:n.h) {
    h_j <- grilla.h[j]
    lls = vector("numeric", n.x)  # l(log)l(ikelihood)s de cada estimador LOOCV
    for (i in 1:n.x) {
      loo_kde <- hacer_kde(x[-i], h_j)
      lls[i] = log(loo_kde(x[i]))
    }
    loglikes[j] <- mean(lls)
    loglikes.se[j] <- sd(lls)
```

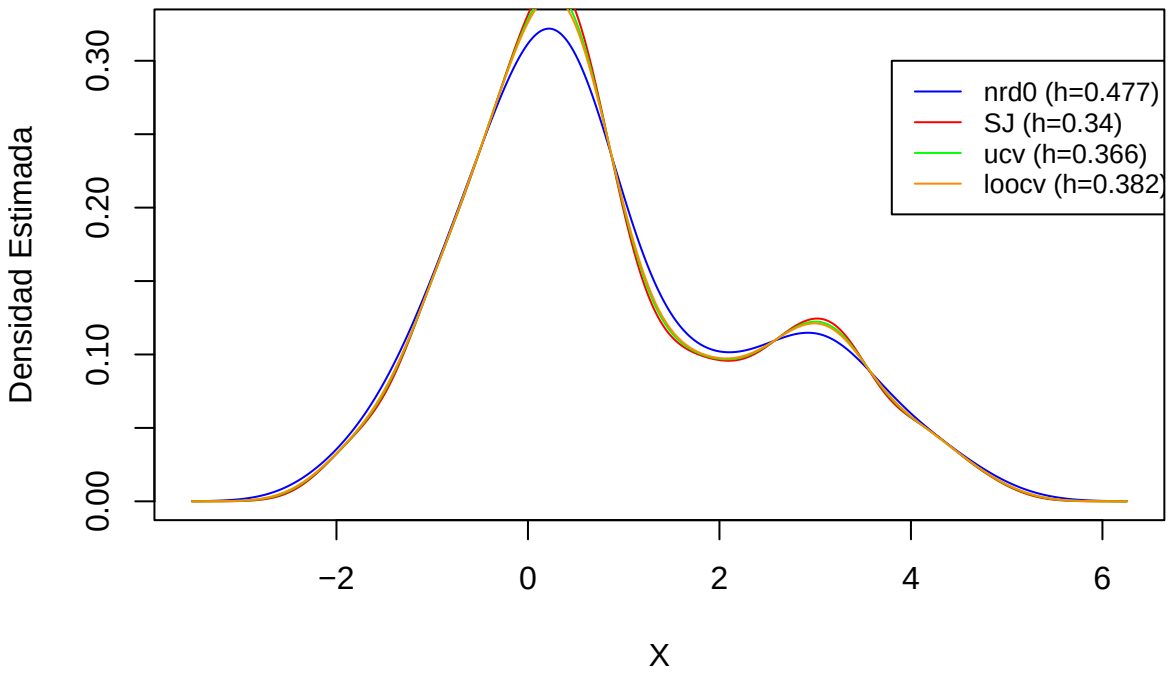
```
  }
  j.opt=which.max(loglikes)
  h.CV=grilla.h[j.opt]
  h.nSE=max(
    grilla.h[
      loglikes >= loglikes[j.opt] - n.se * loglikes.se[j.opt] / sqrt(n.x)
    ]
  )
  return(
    list(
      grilla.h=grilla.h,
      loglikes=loglikes,
      loglikes.se=loglikes.se,
      h.CV=h.CV,
      h.nSE=h.nSE,
      n.se=n.se,
      j.opt=j.opt
    )
  )
}

muestra <- function(seed=1234, n=200) {
  set.seed(seed)
  binoms <- rbinom(n, size=1, p=0.75)
  return (
    binoms * rnorm(n, mean=0, sd=1)
    + (1 - binoms) * rnorm(n, mean=3.25, sd=sqrt(0.5))
  )
}
write.table(muestra(1234), "entrega1.txt", row.names=FALSE, col.names=FALSE)
```

```
x <- sort(muestra()) # Por practicidad para graficar, no cambia el resultado.
ret <- bw.loocv(x)
hs <- c(nrd0=bw.nrd0(x), SJ=bw.SJ(x), ucv=bw.ucv(x), loocv=ret$h.CV)
colores <- c(nrd0="blue", SJ="red", ucv="green", loocv="darkorange")
```

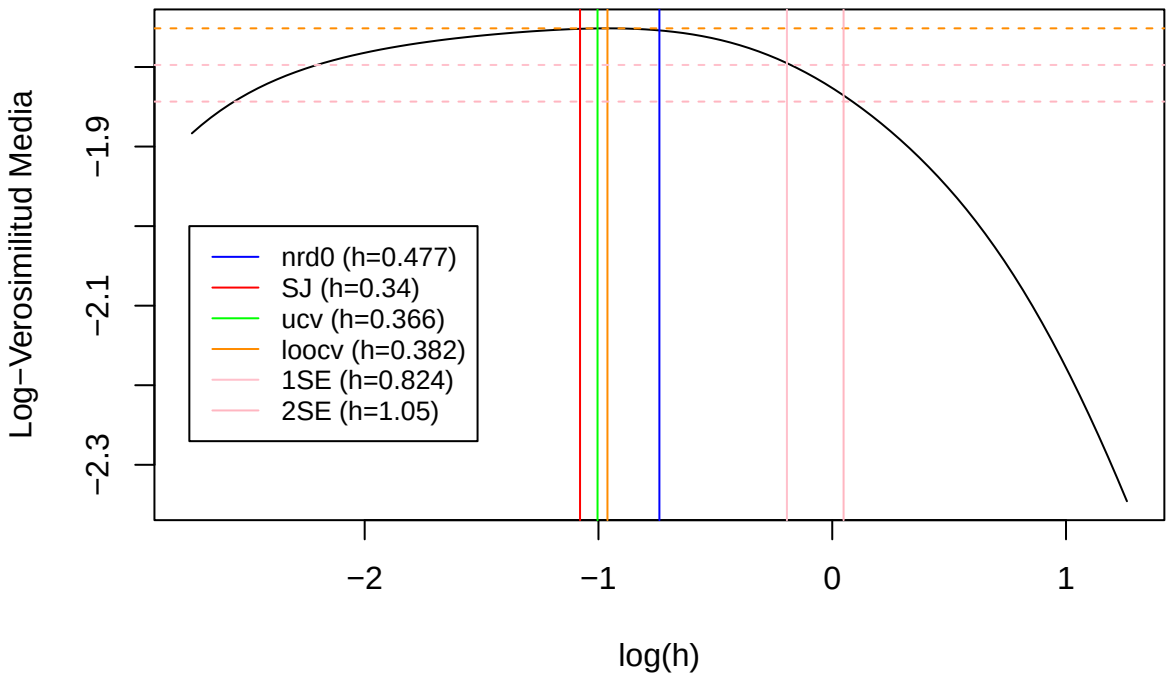
```
plot(log(ret$grilla.h), ret$loglikes, type="l", ylab="Log-Verosimilitud Media", xlab="log(h)")
for (name in names(hs)) {
  abline(v=log(hs[[name]]), col=colores[name])
}
etiquetas <- function(hs) { paste0(names(hs), " (h=", round(hs, 3), ")")}
legend(-2.5, -2.1, legend=etiquetas(hs), col=colores, lty=1, cex=0.8)
```



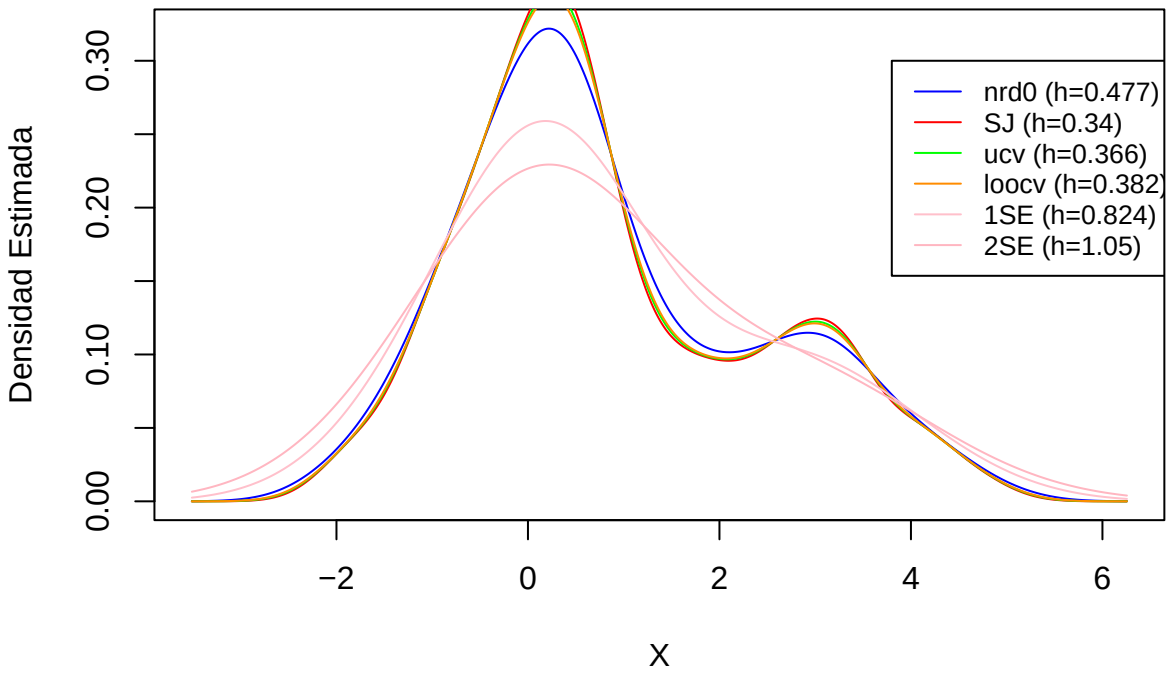

```r
# 5.a
su_kde <- density(x)
sop = su_kde$x # density calcula un soporte conveniente para el kde
mi_kde <- hacer_kde(x, h=bw.nrd0(x))
plot(sop, su_kde$y, type="n", xlab="X", ylab="Densidad Estimada",)
for (name in names(hs)) {
  lines(sop, hacer_kde(x, hs[[name]])(sop), type="l", col=colores[name])
}
legend(3.8, 0.3, legend=etiquetas(hs), col=colores, lty=1, cex=0.8)
```




```
ll <- ret$loglikes[ret$j.opt]
ll.se <- ret$loglikes.se[ret$j.opt]
n.x <- length(x)
lls <- c(
  "loocv"=ll,
  "1SE"= ll - ll.se / sqrt(n.x),
  "2SE"= ll - 2 * ll.se / sqrt(n.x)
)
hs["1SE"] <- ret$h.nSE
hs["2SE"] <- bw.loocv(x, n.se=2)$h.nSE
plot(log(ret$grilla.h), ret$loglikes, type="l", ylab="Log-Verosimilitud Media", xlab="log(h)")
colores["1SE"] <- "pink"
colores["2SE"] <- "lightpink"
for (name in names(hs)) {
  abline(h=lls[name], col=colores[name], lty=2)
  abline(v=log(hs[name]), col=colores[name], lty=1)
}
legend(x=-2.75, y=-2, legend=etiquetas(hs), col=colores, lty=1, cex=0.8)
```




```r
# Repito 5.a para todos los estimadores
su_kde <- density(x)
sop = su_kde$x # density calcula un soporte conveniente para el kde
mi_kde <- hacer_kde(x, h=bw.nrd0(x))
plot(sop, su_kde$y, type="n", xlab="X", ylab="Densidad Estimada")
for (name in names(hs)) {
  lines(sop, hacer_kde(x, hs[[name]])(sop), type="l", col=colores[name])
}
legend(3.8, 0.3, legend=etiquetas(hs), col=colores, lty=1, cex=0.8)
```


Densidad Estimada
0.00
0.10
0.20
0.30
−2
0
2
4
6
X
nrd0 (h=0.477)
SJ (h=0.34)
ucv (h=0.366)
loocv (h=0.382)
1SE (h=0.824)
2SE (h=1.05)