Economics 597: Nonparametric Econometrics

Department of Economics, Finance and Legal Studies
University of Alabama

Spring 2015

Midterm I – Answers

1. (a)

$$\begin{aligned} AMISE\left[\widehat{f}(x)\right] &= \int AMSE\left[\widehat{f}(\psi)\right] d\psi \\ &= \int \frac{h^4}{4}\left[f^{(2)}(\psi)\right]^2 \kappa_2^2(k) + \frac{1}{nh} f(\psi) R(k) d\psi \\ &= \frac{h^4\kappa_2^2(k)}{4}\int \left[f^{(2)}(\psi)\right]^2 d\psi + \frac{R(k)}{nh}\int f(\psi) d\psi \\ &= \frac{h^4\kappa_2^2(k)}{4} R\left(f^{(2)}(\cdot)\right) + \frac{R(k)}{nh} \end{aligned}$$

   (b) $\kappa_2(k)$ is the second moment of the kernel function k. $R\left(f^{(2)}(\cdot)\right)$is defined as roughness of the second derivative of $f(\cdot)$ (our density of interest). h is the bandwidth, $R(k) = \int k(u)^2 du$ is commonly called the 'roughness' or difficulty of the kernel function k. n is the sample size.

   (c) We obtain the optimal bandwidth by differentiating $AMISE$ with respect to h and setting it equal to zero

$$\begin{aligned} \frac{\partial AMISE\left[\widehat{f}(x)\right]}{\partial h} &= \kappa_2^2(k) R\left(f^{(2)}(\cdot)\right) h^3 - \frac{R(k)}{nh^2} = 0 \\ &\Rightarrow \kappa_2^2(k) R\left(f^{(2)}(\cdot)\right) h_{opt}^5 - \frac{R(k)}{n} = 0 \\ &\Rightarrow h_{opt}^5 = \frac{R(k)}{n\kappa_2^2(k) R\left(f^{(2)}(\cdot)\right)} \\ &\Rightarrow h_{opt} = \left[\frac{R(k)}{n\kappa_2^2(k) R\left(f^{(2)}(\cdot)\right)}\right]^{-1/5} \end{aligned}$$

   (d) $R(k)$ shows that less rough kernels lead to smaller optimal bandwidths, as do larger sample sizes (n). The optimal bandwidth is also a function of the second-moment of the kernel function and the unknown roughness of the underlying density.

   (e) To make this bandwidth operational, we must plug in a specific kernel function and then obtain an estimate for the roughness of the density. One popular method is to use a rule-of-thumb type of approach and replace the second derivative with that from a parametric family.

2. To show that we have a second-order degenerate U-statistic, we must show that $H_n(z_i, z_j)$ is symmetric, zero in expectation and has a finite variance. These are given as

$$H_n(z_i, z_j) = k\left(\frac{x_i - x_j}{h}\right) + k\left(\frac{y_i - y_j}{h}\right) - k\left(\frac{x_i - y_j}{h}\right) - k\left(\frac{y_i - x_j}{h}\right)$$

$$
\begin{aligned}
&= k\left(\frac{x_j - x_i}{h}\right) + k\left(\frac{y_j - y_i}{h}\right) - k\left(\frac{x_j - y_i}{h}\right) - k\left(\frac{y_j - x_i}{h}\right) \\
&= H_n(z_j, z_i)
\end{aligned}
$$

assuming we have a symmetric kernel function. Recalling that $f(x) = g(x)$ under the null

$$
\begin{aligned}
E\left[H_n(z_i, z_j) | z_i, z_j\right] &= E\left[k\left(\frac{x_i - x_j}{h}\right) + k\left(\frac{y_i - y_j}{h}\right) - k\left(\frac{x_i - y_j}{h}\right) - k\left(\frac{y_i - x_j}{h}\right) | x_i, x_j\right] \\
&= \int f(x_j)\, k\left(\frac{x_i - x_j}{h}\right) dx_j + \int g(y_j)\, k\left(\frac{y_i - y_j}{h}\right) dy_j \\
&\quad - \int f(x_j)\, k\left(\frac{x_i - y_j}{h}\right) dx_j - \int g(y_j)\, k\left(\frac{y_i - x_j}{h}\right) dy_j \\
&= \int f(x_j)\, k\left(\frac{x_i - x_j}{h}\right) dx_j + \int f(y_j)\, k\left(\frac{y_i - y_j}{h}\right) dy_j \\
&\quad - \int f(x_j)\, k\left(\frac{x_i - y_j}{h}\right) dx_j - \int f(y_j)\, k\left(\frac{y_i - x_j}{h}\right) dy_j \\
&= \int f(v_1 h + x_i)\, k\left(\frac{x_i - x_j}{h}\right) dv_1 + \int f(v_2 h + y_i)\, k\left(\frac{y_i - y_j}{h}\right) dv_2 \\
&\quad - \int f(v_3 h + y_i)\, k\left(\frac{x_i - y_j}{h}\right) dv_3 - \int f(v_4 h + x_i)\, k\left(\frac{y_i - x_j}{h}\right) dv_4 \\
&= 0
\end{aligned}
$$

where $v_1 \equiv (x_i - x_j)/h$, $v_2 \equiv (y_i - y_j)/h$, $v_3 \equiv (x_i - y_j)/h$, and $v_4 \equiv (y_i - x_j)/h$. The first and the fourth cancel because under the null that f and g are equivalent, and the range of intergration is the same. The second and third cancel for the same reason. Finally, we want to show that the variance is finite. This is trivial because a second-order kernel $k^2(\psi)$ is assumed to be finite.

3. The code is attempting to find an optimal bandwidth via LSCV. The details are on the next page.

```
 ## LSCV bandwidth selection

 rm(list=ls())                                    ## clearing the memory
set.seed(123456)                                 ## setting the seed for replication purpose

library(stats)                                    ## calling a library in R

 dat <- read.csv("data2000hpr.csv", header=TRUE)   ## reading in the data
 dat <- subset(dat, complete.cases(dat))           ## removing any missing observation
 attach(dat)                                       ## attaching the data

rgdpch00 <- as.matrix(rgdpch00/mean(rgdpch00))    ## calling the (mean deviated data) a matrix
rgdpch00 <- rgdpch00[order(rgdpch00),]            ## ordering the data from smallest to largest

n <- length(rgdpch00)                             ## determining the sample size

KK.store.loo <- matrix(0,nrow=n,ncol=n)           ## creating kernel storage matrix
KK.store.convolution  <- matrix(0,nrow=n,ncol=n)  ## creating a separate kernel storage matrix for the convolution

h <- seq(0.001,0.5,0.001)                         ## creating a sequence of bandwidths for use in the cross-validation
function

lscv.loo <- matrix(0,nrow=length(h),ncol=1)       ## creating a storage matrix to evaluate the cross-validation function at
each h

for (jj in 1:length(h)){                          ## starting a loop over each possible h

 for (j in 1:n){                                 ## starting a loop over the observations

  dx <- (rgdpch00 - rgdpch00[j])/h[jj]           ## creating \psi
  KK <- solve(sqrt(2*pi))*exp(-0.5*dx^2)         ## Gaussian kernel
  KK.convolution <- solve(sqrt(4*pi))*exp(-0.25*dx^2)          ## Gaussian convolution kernel

  KK.store.loo[,j] <- KK                         ## storing the kernel vector
  KK.store.loo[j,j] <- 0                         ## leaving one observation out
  KK.store.convolution[,j] <- KK.convolution     ## storing the convolution kernel vector (not a leave-one-out)

 }
                                                  ## calculating the LSCV value and storing it a storage vector
 lscv.loo[jj] <- solve(n^2*h[jj])*sum(KK.store.convolution) - 2*solve(n*(n-1)*h[jj])*sum(KK.store.loo)

}
```