Economics 217 - Nonparametric Econometrics

- Topics covered in this lecture
  - Introduction to the nonparametric model
  - The role of bandwidth
  - Choice of smoothing function
  - R commands for nonparametric models
- Much of these notes are inspired by Prof. Bruce Hansen's PhD Econometrics Text.

Linear models to non-parametric models

- What is a non-parametric model?
  - A model that does not assume a strong parametric form of the relationship between independent variables and dependent variables
  - Simple OLS adopts the assumption "linear in parameters". That is, a parametric function that is linear in things we estimate
  - Non-parametric models are occasionally called semi-parametric models, though these can refer to other techniques as well so we will use non-parametric.
- Recall that the linear model can be written as:

$$y_i = \mathbf{x}_i \beta + u_i$$

- In general, the non-parametric model is written as:

$$y_i = s\left(x_{i1}, x_{i2}, \ldots x_{ip}\right) + u_i$$

- The key is choosing the particular form of $s()$, subject to a variety of practical constraints. What are the issues in choosing these functions?

Top-level issues with non-parametric models

- **Issue #1:** Functional Form
  - Ultimately, we *must* choose a form for $s\left(x_{i1}, x_{i2}, ...x_{ip}\right)$. And, there are an infinite number of choices that we have.
- For example, we could simplify:

  $$y_i = s\left(x_{i1}, x_{i2}, ..., x_{ip}\right) + u_i$$

  as

  $$y_i = s_1\left(x_{i1}\right) + s_2\left(x_{i2}\right) + \cdots + s_p\left(x_{ip}\right) + u_i$$

- And still, even under the last form, we'd have to assume something about $s_k()$. But why didn't we use:

  $$y_i = s_1\left(x_{i1}, x_{i2}\right) + s_3\left(x_{i3}\right) + \cdots + s_p\left(x_{ip}\right) + u_i$$

- The choices are (literally) endless.

Top-level issues with non-parametric models

- One option available to the researcher is to choose a parametric function that is ridiculously rich and flexible.

- For example, let's consider the univariate non-parametric model

$$y_i = s(x_i) + u_i$$

- Again, there are a lot of choices for $s()$. One (parametric) choice is the following:

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 \\ &+ \beta_5 x_i^5 + \beta_6 x_i^6 + \beta_7 \log(x_i) + \beta_8 \cos(x_i) + u_i \end{aligned}$$

- Positives for this specification:
  - Can estimate with OLS, get standard errors easily, generate predictions

- Negatives for this specification:
  - Good luck interpreting any coefficients.
  - Functional form is arbitrary.

- We will return to these types of models after we discuss the most basic non-parametric estimation procedures.

Top-level issues with non-parametric models

- A common, very simple and intuitive alternative to a flexible functional form is called "binned estimation".
- Intuitively, we break-up the data into bins, and find the best fit within these bins. Then we aggregate the results and produce a plot describing the relationship between y and x.
  - A popular technique in data science, "k-nearest neighbors", is an extended version of "binned" estimation.
- Formally, this is accomplished through the following equation

$$\widehat{s}(x) = \frac{\sum_i^n \mathbf{1}\left(|x_i - x| < h\right) y_i}{\sum_i^n \mathbf{1}\left(|x_i - x| < h\right)}$$

- In this equations, we have:
  - $\widehat{s}(x)$ The estimate form $s()$ at x
  - h: The bandwidth - the region of $x's$ over which we estimate $s()$ at x
- In this binned approach, at a given x, we take values no more than h above or below x to estimate $\widehat{s}(x)$

Top-level issues with non-parametric models

- This approach can be re-written as a function of a general weighting function.

$$\begin{aligned}\widehat{s}(x) &= \frac{\sum_i^n \mathbf{1}\left(|x_i - x| < h\right) y_i}{\sum_i^n \mathbf{1}\left(|x_i - x| < h\right)} \\ &= \sum_i^n \underbrace{\frac{\mathbf{1}\left(|x_i - x| < h\right)}{\sum_j^n \mathbf{1}\left(|x_j - x| < h\right)} y_i}_{w_i(x)} \\ &= \sum_i^n w_i(x) y_i\end{aligned}$$

- What is the primary issue with estimating this function?

- **Issue #2:** Weighting and Bandwidth

  - Non-parametric estimates may depend heavily on choice of weighting function $w(x)$

- We will later examine the issues with the weighting function with a few R examples

Nadaraya-Watson Estimator

- Generally, binned-estimation is called either a "local-constant estimator" or the "Nadaraya-Watson" estimator.

$$\widehat{s}(x) \quad = \quad \sum_{i}^{n} w_i(x) y_i$$

- The choice of $w_i(x)$ is crucial. It is helpful to redefine the weighting function as a *Kernel Function*, $k(u)$, where

$$u = \frac{x_i - x}{h}$$

and $k(u)$ has the following properties:

$$\begin{aligned} k(u) &= k(-u) \\ 0 &\leq k(u) < \infty \\ \int_{-\infty}^{\infty} k(u) du &= 1 \\ \int_{-\infty}^{\infty} u^2 k(u) du &< \infty \end{aligned}$$

- That is, $k(u)$ is a bounded pdf and symmetric about zero, with finite variance.

Nadaraya-Watson Estimator

- Choice of $k(u)$ is crucial to any non-parametric study. There are three common choices:

- **Uniform (or "box"):**
  Just as we've described above for the binned estimation

$$k(u) \quad = \quad \frac{1}{2}\mathbf{1}\left(|u| \leq 1\right)$$

- **Epanechnikov:**
  Like uniform, but declining weights in u^2

$$k(u) \quad = \quad \frac{3}{4}\left(1-u^2\right)\mathbf{1}\left(|u| \leq 1\right)$$

- **Gaussian:**
  Weighted as a standard normal distribution

$$k(u) \quad = \quad \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{u^2}{2}\right)$$

R Examples: Nadaraya-Watson and Binned Estimation

- Basic Nadaraya-Watson estimation can be accomplished in R using the `ksmooth` command

- Syntax: ksmooth(x,y, type, bandwidth)
  - x: the running variable
  - y: the outcome variable
  - kernel: type of smoothing ("box" or "normal")
  - bandwidth: exactly as it sounds.

- Evaluate smooth relationship between age and labor force participation

```
plot(ksmooth(subd$age,subd$nilf, "box", bandwidth = 1), col = 1)
lines(ksmooth(subd$age,subd$nilf, "box", bandwidth = 10), col = 2)
lines(ksmooth(subd$age, subd$nilf, "box", bandwidth = 20), col = 3)
lines(ksmooth(subd$age, subd$nilf, "box", bandwidth = 40), col = 4)
```

- With the normal kernel instead of boxed

```
plot(ksmooth(subd$age,subd$nilf, "normal", bandwidth = 1), col = 1)
lines(ksmooth(subd$age,subd$nilf, "normal", bandwidth = 10), col = 2)
lines(ksmooth(subd$age, subd$nilf, "normal", bandwidth = 20), col = 3)
lines(ksmooth(subd$age, subd$nilf, "normal", bandwidth = 40), col = 4)
```

Locally linear regression

- A common alternative to Nadaraya-Watson (NW), though not necessarily better, is the locally linear regression (often called "loess" smoothing)
- Like NW, we produce an estimate for each x
- Unlike NW, we run a full linear regression rather than just estimate an intercept (though we still use an intercept)
- Formally, we solve the following for each x

$$\begin{aligned}
\widehat{s}(x) &= \widehat{\alpha}(x) \\
\textit{where} & \\
\left\{\widehat{\alpha}(x), \widehat{\beta}(x)\right\} &= \underset{\alpha,\beta}{\operatorname{argmin}} \sum_{i}^{n} k\left(\frac{x_i - x}{h}\right)\left(y_i - \alpha - \beta\left(x_i - x\right)\right)^2
\end{aligned}$$

- Then, after we do this, we plot $\widehat{s}(x)$
- When do you think that the Loess regression works better than NW?

R Examples: Loess Estimator

- Again, let’s evaluate the relationship between age and labor force participation.
- The "loess" function is one way to execute local-linear estimation in R.
- "loess" allows for both first and second degree polynomial smoothing.

```
fit.lm<-lm(subd$nilf~subd$age)
fit.loess1<-loess(subd$nilf~subd$age,span=1, degree=1)
fit.loess2<-loess(subd$nilf~subd$age,span=1, degree=2)
```

- And now we plot it.

```
plot(subd$age,predict(fit.lm),type="l",lwd=2,ylim=c(0,1))
lines(subd$age,predict(fit.loess1),col=1,lty=2)
lines(subd$age,predict(fit.loess2),col=4)
```

- Now let’s evaluate the role of "span" which is the commands bandwidth control

```
fit.loess1<-loess(subd$nilf subd$age,span=1, degree=1)
fit.loess2<-loess(subd$nilf~subd$age,span=10, degree=1)
fit.loess3<-loess(subd$nilf~subd$age,span=.1, degree=1)
```

- Again, we plot:

```
plot(subd$age,predict(fit.lm),type="l",lwd=2,ylim=c(0,1))
lines(subd$age,predict(fit.loess1),col=1,lty=2)
lines(subd$age,predict(fit.loess2),col=3)
lines(subd$age,predict(fit.loess3),col=4)
```

Optimal Bandwidth Selection

- How do we choose optimal bandwidth?
- The tradeoffs are fairly straightforward.
    - Large h: reduces variance but increases bias and oversmoothing
    - Small h: reduces bias but increases noise
- Need a technique to systematically balance these objectives.
- **Cross Validation** is the general technique that is used for choosing bandwidth
- **Leave-one-out** bandwidth selection is a type of cross-validation, the standard approach
    - The technique itself drops an observation, generates the model, and predicts the outcome for the dropped observation using the model.
    - We choose the bandwidth that minimizes any out-of-sample prediction errors.

Optimal Bandwidth Selection

- Cross-Validation procedure
  1. Choose h
  2. Estimate $\widehat{s}(x)$ without observation i. Label this estimate $\widehat{s}_{-i}(x,h)$
  3. Calculate prediction error for i: $\tilde{e}_i = y_i - \widehat{s}_{-i}(x,h)$
  4. Repeat for all i
  5. Calculate $CV(h) = \sum_i^n \tilde{e}_i^2$
  6. Repeat for all other h.
- Choose h that minimizes $CV(h)$
- This technique could obviously take a while. For example, with a dataset of 1000 observations and 100 choices of bandwidth, 100,000 regressions are run in total.

R: Optimal Bandwidth Selection

- To demonstrate leave-one-out, let’s first create some fake data

```
x<-seq(-10,10,length=1000)
y<-sin(x)+rnorm(1000,0,1)
```

- Then, let’s have a look at a few loess plots and talk about what we see.

```
fit.loess1<-loess(y~x, family="gaussian",span=1, degree=1)
fit.loess2<-loess(y~x, family="gaussian",span=.05, degree=1)
plot(x,predict(fit.loess1),type="l",lwd=2,ylim=c(-2,2))
lines(x,predict(fit.loess2),col=1,lty=2)
```

- For the leave-one-out estimator, let’s create a fake data frame to use

```
small<-data.frame(y,x)
```

R: Optimal Bandwidth Selection

- Again, the basic process for leave-one-out is the following
  - For each h, iterate through each observation i
    - Drop i, estimate the model with the rest
    - Use model to predict i
    - Calculate squared error of prediction ⇒ save
  - Choose h that minimizes sum of squared residuals of the out of sample predictions
- Precisely:

```
for(h in 1:20){
  for(i in 1:nrow(small)){
    smalldrop<-small[i,]
    smallkeep<-small[-i,]
    fit<-loess(y~x,smallkeep, family="gaussian",span=(h/20), degree=1)
    dropfit<-predict(fit,smalldrop,se=FALSE)
    sqrerr<-(smalldrop$y-as.numeric(dropfit))^2
    if(i*h==1){results<-data.frame(h,i,sqrerr)}
    if(i*h>1){results<-rbind(results,data.frame(h,i,sqrerr))}
  }
}
```

- Use tapply (or some other function) to find the minimizing h

```
tapply(results$sqrerr,results$h,FUN=sum,na.rm=TRUE)
```

Series estimation

- Series estimation involves using a flexible polynomial to estimate an unknown function.
- Though earlier I mentioned that the choice of polynomial is arbitrary, there is a science behind it.
- **Stone-Weierstrass Theorem** (1885, 1937, 1948)
  - Any continuous function can be well approximated by a polynomial of a sufficiently high order.
- How do we choose such a function?
- Two main considerations:
  - Do we interact variables of interest?
  - What order polynomial should we use?

Series estimation

- Two techniques:
  - Approximation by **series**
  - Approximation by **spline**
- In the former, we essentially choose a flexible polynomial, including all powers of variables and cross-products of variables and their powers.
- Two defining features of approximation by series
  - p number of variables:
  - k order of the polynomial.
- A simple series regression, $p = 2$ and $k = 1$, is the following:

$$s(x) = \beta_0 + \beta_1 x_1 + \beta_1 x_2 + \beta_{12} x_1 x_2$$

- Assuming $p = 2$ and $k = 2$ we get:

$$\begin{aligned} s(x) &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 \\ &+ \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{122} x_1 x_2^2 + \beta_{112} x_1^2 x_2 + \beta_{1122} x_1^2 x_2^2 \end{aligned}$$

- Just by going from $k = 1$ to $k = 2$, dimension more than doubled

Series estimation

- In general, series estimation has a dimension $K = (1+k)^p$
  - This can obviously get pretty big depending on the dataset and desire for a smooth fit
- There is also a downside to a polynomial fit of this type: **Runge's Phenomenon**
  - Polynomials can be very bad at interpolation.
  - In other words, they might do well predicting the actual data, but very poorly when generating out-of-sample predictions.
- To study this, let's plot the function

$$s(x) = \frac{1}{1+x^2}$$

And try to estimate it with a polynomial.

R Example: Runge's phenomenon

- Try this with linear regression, and 10th order polynomial
- First, let's create some fake data

```
x<-seq(-10,10,by=1)
y<-1/(1+x^2)
x2<-x^2
x3<-x^3
x4<-x^4
x5<-x^5
x6<-x^6
x7<-x^7
x8<-x^8
x9<-x^9
x10<-x^10
```

- Then let's plot:

```
plot(y~x,ylim=c(-0.25,1))
lines(predict(lm(y~x))~x,col=1,lwd=2)
lines(predict(lm(y~x+x2))~x,col=2,lwd=2)
lines(predict(lm(y~x+x2+x3+x4+x5+x6+x7+x8+x9+x10))~x,col=3,lwd=2)
```

R Example: Runge's phenomenon

- Next, let's generate a new dataset, and evaluate out-of-sample predictions

```
xnew<-data.frame(x=seq(-5,5,by=0.01))
xnew$x2<-xnew$x^2
xnew$x3<-xnew$x^3
xnew$x4<-xnew$x^4
xnew$x5<-xnew$x^5
xnew$x6<-xnew$x^6
xnew$x7<-xnew$x^7
xnew$x8<-xnew$x^8
xnew$x9<-xnew$x^9
xnew$x10<-xnew$x^10
ynew<-1/(1+xnew$x^2)
```

- Then let's plot:

```
plot(ynew~xnew$x,ylim=c(-0.25,1),cex=0.25)
lines(predict(lm(y~x+x2+x3+x4+x5+x6+x7+x8+x9+x10),xnew)~xnew$x,col=4,lwd=2)
```

- The original fit was created using 11 data points evenly spaced between -5 and 5. How did we do away from these points?

Spline estimation

- Spline estimation is an alternative to series estimation which is also based on polynomials, but allows for the polynomial to "evolve" with the value of the dependent variable.

- To develop a spline model, suppose that $s(x)$ is univariate, and that $x \in \left(\underline{x}, \overline{x}\right)$

- Further, suppose that we have chosen N "knots" $\{t_1, t_2, ..., t_N\} \in \left(\underline{x}, \overline{x}\right)$.
  - These knots split up the relevant range of x, and as you will see, are crucial to the estimation of spline functions.

- With these knots, a spline function is defined by the following:

$$s(x) = \sum_{j=0}^{k} \beta_k x^k + \sum_{z=1}^{N} \gamma_z \left(x - t_z\right)^k \mathbf{1}\left(x \geq t_z\right)$$

- Characteristics of the spline function
  - Conitinuous derivatives up to $k-1$
  - In practice k is usually 3 to have continuous second derivatives.

Spline estimation

$$s(x) = \sum_{j=0}^{k} \beta_k x^k + \sum_{z=1}^{N} \gamma_z (x - t_z)^k \mathbf{1}(x \geq t_z)$$

- There are two critical parts to the spline function:
  - $\sum_{j=0}^{k} \beta_k x^k$ is the basic polynomial
  - $\sum_{z=1}^{N} \gamma_k (x - t_z)^k \mathbf{1}(x \geq t_z)$ at the maximum degree
- Critical issues moving forward is choosing t_z's. Cross-validation is the technique that is typically used for this
- However, must choose either flexibility (location of t_z's), or depth of changes to the polynomial (number of t_z's).
  - If you limit yourself to very small number of t_z's, can grid search over a few t_z's.
  - If you want to possibly have a lot of flexibility in the spline, evenly space

R Example: Spline estimation

- Let's do another example with $s(x) = \frac{1}{1+x^2}$. We'll compare the 10th-order polynomial with a third-degree spline with four knots at -3, -1, 1, and 3.
  - Since the data lie between -5 and 5, the knots are evenly spaced
- To construct the spline, let's first write out the equation:

$$\begin{aligned} s(x) \quad &= \quad \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \gamma_1 (x - (-3)))^3\, \mathbf{1}(x \geq -3) + \gamma_2 (x - (-1))^3\, \mathbf{1}(x \geq -1) \\ &\quad + \gamma_3 (x-1)^3\, \mathbf{1}(x \geq 1) + \gamma_4 (x-3)^3\, \mathbf{1}(x \geq 3) + u \end{aligned}$$

- To code in R, let's create the knots at the original and new data

```
k1<-ifelse(x>(-3),(x-(-3))^3,0)
k2<-ifelse(x>(-1),(x-(-1))^3,0)
k3<-ifelse(x>(1),(x-1)^3,0)
k4<-ifelse(x>(3),(x-(3))^3,0)
xnew$k1<-ifelse(xnew$x>(-3),(xnew$x-(-3))^3,0)
xnew$k2<-ifelse(xnew$x>(-1),(xnew$x-(-1))^3,0)
xnew$k3<-ifelse(xnew$x>(1),(xnew$x-1)^3,0)
xnew$k4<-ifelse(xnew$x>(3),(xnew$x-(3))^3,0)
```

- Then plot

```
plot(ynew~xnew$x,ylim=c(-0.25,1))
lines(predict(lm(y~x+x2+x3+x4+x5+x6+x7+x8+x9+x10),xnew)~xnew$x,col=4,lwd=2)
lines(predict(lm(y~x+x2+x3+k1+k2+k3+k4),xnew)~xnew$x,col=1,lwd=2)
```

R Example: GAM package in R

- There are a number of non-parametric econometrics packages in R
  - library "gam" is the easiest to use
  - library "mvcv" has more bells and whistles - check it out on your own as you wish.
- We will estimate (again) labor force participation with gam, as a function of age:

```
gamresults<-gam(nilf ~s(age,4), data=subd)
summary(gamresults)
plot(gamresults,se=TRUE,rug=FALSE,terms="s")
```

- In the first line, "s(age,4)" specifies a smooth function of the variable "age" with a smoothing parameter of 4.
  - This smoothing parameter goes into a complicated procedure called "backfitting", but the entire procedure is based on third-order splines.
- "s(age,1)" would yield a linear regression.
- The dependent variable is always demeaned to zero before estimation. So, $\mathbb{E}\left(s(age,)\right) = 0$. This is useful for inference.

R Example: GAM package in R

- Now we add-in education, which is a factor variable.

```
gamresults<-gam(nilf ~s(age,4)+educ, data=subd)
summary(gamresults)
par(mfrow=c(1,2))
plot(gamresults,se=TRUE,rug=FALSE,terms="s")
abline(v=0)
abline(h=0)
plot(gamresults,se=TRUE,rug=FALSE,terms="educ")
abline(v=0)
abline(h=0)
```

- The use of the function "abline" places a horizontal and vertical intercept at zero, with the former being the benchmark for being different from the sample average.
  - That is, if the two standard deviation confidence bands do not include zero, we reject zero as a hypothesized value at that point.
  - Since $\mathbb{E}\left(s(age,)\right) = 0$, we conclude that the estimate at that point is significantly different from the sample average.
- GLM restrictions (eg. families, links) can be used with gam.