ARE 213 **Applied Econometrics**

UC Berkeley Department of Agricultural and Resource Economics

SELECTION ON OBSERVABLES DESIGNS:
PART II, NONPARAMETRIC REGRESSION

This set of lecture notes discusses nonparametric regression — roughly speaking, regression models that allow us to relax the linearity assumptions that we have maintained to date. There are two reasons why we might be interested in doing this. First, the treatment of interest may be multi-valued, i.e. D_i may not be binary. If D_i is continuous, then we may wish to estimate $E[Y_i|D_i]$ without imposing linearity or other functional form assumptions. If D_i is randomly assigned, then $E[Y_i|D_i]$ will have a causal interpretation.[1] Alternatively, D_i may be binary, and we may believe that the selection on observables assumption holds, i.e. $Y_i(0) \perp D_i|X_i$. If we want to control for $E[D|X]$, and we don't have reason to believe that $E[D|X]$ is a linear function of X, then we may want to use a nonparametric method to control for X (which is equivalent to estimating $E[D|X]$ nonparametrically). We ignore the latter scenario for the moment and concentrate on the former; we will discuss the latter scenario at the end, however, as a segue into matching and the propensity score.

1 Series Regression

See 2A notes, last page

Suppose that we are interested in estimating $h(X_i) = E[Y_i|X_i]$, where X_i is a scalar. This function may have a causal interpretation (if X is randomly assigned), or it may not — that is not the focus at the moment, even though we are still in the "Selection on Unobservables" section of the course. As before, we define the CEF residual as:

$$\varepsilon_i = Y_i - E[Y_i|X_i]$$

Thus $Y_i = E[Y_i|X_i] + \varepsilon_i$, where ε_i is orthogonal to X_i by construction. If $E[Y_i|X_i]$ is

[1] In the medical literature, this function is sometimes referred to as the dose response function.

a linear function of X_i, then we can estimate $E[Y_i|X_i]$ using a standard linear regression of Y on X. If not, then $X'\hat{\beta}$ gives us the MMSE linear approximation of $E[Y_i|X_i]$. But what if we want something better than the MMSE linear approximation of $E[Y_i|X_i]$? Given that we don't know the functional form of $E[Y_i|X_i]$, we are going to have to approximate it using some method. An obvious candidate is a *series estimator*, analogous to a Taylor Series expansion that you might use to approximate a function $f(x)$ near a point a. To review, we can approximate an infinitely differentiable function $f(x)$ in the neighborhood of a using the following expansion:

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + ...$$

If we choose $a = 0$ (the Maclaurin series), this expansion becomes

$$f(x) \approx f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + ...$$

This suggests that we might estimate $E[Y|X] = h(X)$ using a high degree polynomial of X — the regression coefficients will simply estimate $f(0)$, $\frac{f'(0)}{1!}$, $\frac{f''(0)}{2!}$, etc.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_p x_i^p + \varepsilon_i$$

If the support of x is of limited range, then the approximation should be accurate, but as the support of x becomes wider it will become less accurate. To address this problem, we include "splines" in the regression that allow the function to change as we move along the support of x. The regression spline model is basically a piecewise polynomial and looks like:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_i^j + \sum_{j=p+1}^{l} \beta_j \mathbf{1}(x_i > k_{j-p})(x - k_{j-p})^p + \varepsilon_i$$

In other words, the regression spline contains a normal power series up to degree p (x, x^2, x^3, ..., x^p), and then contains splines that kick in at knots k_1, k_2, ..., k_{l-p-1}. In general,

the knots are evenly spaced over the support of x; the number of knots is often determined by looking at the data. Alternatively, one might place knots at various quantiles of x. There is really no set way to choose the knots — like operating the Digital Conveyer, it is more an art than a science.

The most popular form for regression splines is probably the cubic spline. This is implemented as:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 \mathbf{1}(x_i > k_1)(x_i - k_1)^3 + ... + \beta_l \mathbf{1}(x_i > k_{l-3})(x_i - k_{l-3})^3 + \varepsilon_i$$

The nice thing about series regression is that, if you have any experience with linear regressions, it's very straightforward and easy to implement and understand. It is still focused, however, on estimating the conditional mean, albeit very flexibly. What if you want to estimate an entire distribution, or entire conditional distribution? Or what if you don't want to have to choose where the knots go? Enter kernel density estimation.

2 Kernel Regression

2.1 Kernel Density Estimation

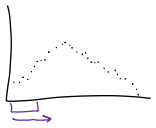
Suppose that we are trying to empirically estimate a density function $f(x)$. Note that at this point we are not talking about a conditional density function that varies over some argument (e.g., $f(y|x)$); we are just talking about estimating the marginal density for a single variable, x. One way to do this is to create a histogram. That is to say, divide the support of x into a bunch of evenly spaced bins, count how many observations fall into each bin, and then create a bar graph where the height of each bar is proportional to the number of observations that fall into the corresponding bin. It turns out that the histogram is a special case of the kernel density estimator.[2]

[2]Specifically, it is equivalent to a kernel density estimator that uses a uniform kernel with bandwidth equal to one-half the histogram bin width, evaluated only at the midpoints of the histogram bins.

Kernal Estimation

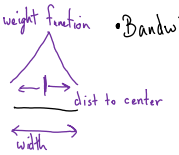
- density $f(x)$
- hist 

unlikely that the true $f(x)$ has these discontinuities if X is a cont. RV

- Instead, have moving windows 

Still have disc.'s, but if increments of bin moving sideways are small enough, the discontinuities are at most ± 1.

- Now, add weights to the obs in each bin, where $\uparrow w_i$ if x_i is close to center and $w_i \downarrow 0$ if x_i is close to edge of bin



- Bandwidth h = width of bin

Kernal density Estimator

- $\hat{f}_h(x) = \frac{1}{N \cdot h} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right)$

x_i = data points
x = point in dist
$K(\cdot)$ = weighting function

Weighting Functions

- Triangle Kernal: (let $h=1$) $K(a) = \chi(|a|<1)\cdot(1-|a|) \Rightarrow \hat{f}_{h=1}(x) = \frac{1}{N}\sum_{i=1}^{N} \chi(|x-x_i|<1)\cdot(1-|x-x_i|)$
- Uniform Kernal: $K(a) = \frac{1}{2}\chi(|a|<1)$
- Epanechnikov Kernal: $K(a) = \frac{3}{4}\chi(|a|<1)(1-a^2)$
- Gaussian Kernal: $K(a) = \frac{1}{\sqrt{2\pi}} e^{-\frac{a^2}{2}}$

Trade-off

- $h \uparrow \Rightarrow \downarrow$ var, $\uparrow$ bias
- $h \downarrow \Rightarrow \uparrow$ var, $\downarrow$ bias of empirical dist?

Choice Criterion

min Integrated Squared Error (ISE)

$$\min_h \text{ISE}(h) = \min_h \int_{\mathbb{R}} \left[\hat{f}_h(x) - f(x)\right] dx$$

$$\text{IMSE} = \text{MISE} = \mathbb{E}\left[\text{ISE}(h)\right] = \int \mathbb{E}\left[\left(\hat{f}_h(x) - f(x)\right)^2\right] dx$$

(function of the data; expectation over X_i (data); integral over function values x in support of X_i)

$$\Rightarrow h^* = \delta\left(\int f''(x)^2 dx\right)^{-0.2} N^{-0.2}, \quad \delta = \left[\frac{\int K(a^2)da}{\int a^2 K(a)da}\right]^{0.2}$$

$f'' \uparrow \Rightarrow h \downarrow$

if true density is choppy, want small $h \Rightarrow$ choppy $\hat{f}_h$

Cross Validation?

- Need to know what the "true" objective function

- have $\text{ISE}(h) = \int\left[\hat{f}_h(x) - f(x)\right]dx$

$$= \int \hat{f}_h^2 - 2\int \hat{f}_h \cdot f + \int f^2$$

(just a const wrt h, so we can ignore this since it will disappear when we $\frac{d}{dh}$)

$2\int \hat{f}_h f$ (leave out est.) $\approx \frac{2}{N}\sum_{i=1}^{N} \hat{f}_{-i,h}(x_i)$ — see bottom of pg 8

$\int g(x)f(x)dx = \mathbb{E}[g(x)] = \overline{g(x)}$

One problem with a histogram is that, unlike most real density functions, it is discontinuous — the value jumps sharply as you move from one bin to the next. Intuitively, we would prefer an estimator that is smooth as you move across different values of x, like most real densities. How might we do this? Well, one way would be to take some window, say of radius 1 unit (i.e., a total of 2 units wide), and move it along the x-axis. As we move it along, we count how many observations fall within the window. For any given point x^*, we define our estimate of $f(x^*)$, $\hat{f}(x^*)$, as the number of observations that fall within the moving window when it is centered at x^* (rescaled so that $\hat{f}(x^*)$ integrates to one at the end). This partially solves the smoothing problem, since $\hat{f}(x^*)$ will never jump by more than one observation for a small enough change in x. However, $\hat{f}(x^*)$ will still be somewhat discontinuous as observations drop into and out of the window. To solve this problem, we could augment our moving window with a moving weighting function — call it a kernel — that places an increasing amount of weight on an observation as it moves closer to the center of the window. For example, consider a function that places zero weight on an observation that is more than one unit from the window's center, 0.01 weight on an observation that is 0.99 units from the window's center, 0.1 weight on an observation that is 0.9 units from the window's center, 0.2 weight on an observation that is 0.8 units from the window's center, and so on, until the observations gets a weight of 1 when it lies directly at the window's center.[3] By utilizing this weighting function, we ensure that a given observation x_i never abruptly drops out of the moving window — it just smoothly "slides" out of the window until its influence disappears entirely at one unit from the center or beyond. This, in essence, is what a kernel density estimator is.

Formally, the kernel density estimator is defined as:

$$\hat{f}_h(x) = \frac{1}{Nh} \sum_{i=1}^{N} K(\frac{x - x_i}{h})$$

In this definition, N is the size of the data set (each observation is indexed by i), $K(.)$ is the "kernel density function," and h is the "kernel bandwidth." How does this estimator

[3]This particular weighting function is referred to as the triangle kernel.

work? Basically, it works as I described above, but in the context of the formal definition it works as follows. Suppose that $h = 1$ and $K(u) = \mathbf{1}(|u| < 1)(1 - |u|)$, i.e. the triangle kernel that I described above in which the weight is a linear function of the distance (until a certain point). At a given point x^* we have

$$\hat{f}_1(x^*) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(|x^* - x_i| < 1)(1 - |x^* - x_i|)$$

An alternative way for writing this that might be easier to digest is

$$\hat{f}_1(x^*) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(d(x^*, x_i) < 1)(1 - d(x^*, x_i))$$

where $d(.)$ is the Euclidean distance metric. So, at any given point x^*, we sum over *every* observation x_i in the data set (as you might guess, this is rather computationally intensive). If x_i falls more than one unit of distance away from x^*, however, it receives a weight of zero and has no input into the sum. If x_i falls less than one unit away from x^*, then it receives a positive weight, and the value of that weight increases the closer that x_i is to x^*. In this particular case, the weight increases linearly as the distance gets closer, but that is not the case for alternative kernels. In general, however, the weight is weakly increasing as the distance gets closer. Computing this summation at all possible values of x^* maps out the entire function $\hat{f}_h(x)$.

The preceding discussion ignored the contributions of the bandwidth, h, and the kernel density function, $K(.)$, by choosing $h = 1$ and $K(.)$ as the triangle kernel. What alternative values might we use for these parameters? The kernel is probably the *less* important of the two choices. Kernels are defined to integrate to 1 — besides the triangle kernel, other common kernels include:

Uniform: $\frac{1}{2} \cdot \mathbf{1}(|u| < 1)$

Epanechnikov: $\frac{3}{4} \cdot \mathbf{1}(|u| < 1)(1 - u^2)$

Gaussian: $\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$

Note that the uniform kernel "abruptly" drops observations (i.e., they have a weight of 0.5 as along as they're within one unit of distance, and then a weight of 0 as soon as they move beyond one unit), providing less smoothing. The Gaussian kernel, in contrast, never drops out an observation (i.e., *all* the observations have an effect on $\hat{f}_h(x^*)$ at every point x^*, although observations that are far from x^* have a trivial effect), providing more smoothing.[4]

In most applications, the choice of bandwidth, h, is more important than the choice of kernel. For any given kernel, increasing the bandwidth extends the "reach" of the kernel. Take, for example, the uniform kernel, $\frac{1}{2} \cdot \mathbf{1}(|\frac{x-x_i}{h}| < 1)$. When $h = 1$, then any observations within a radius of one unit of x will affect the kernel density estimator at x, and all other observations will not. When $h = 2$, however, then observations within a radius of two units of x will affect the kernel density estimator at x. For any value of h, observations within a radius of h units of x will affect the kernel density estimator at x. The same thing holds for the triangle and Epanechnikov kernels. The Gaussian kernel gives positive weight to all observations at any point x. Nevertheless, increasing h is equivalent to increasing the variance in the Gaussian kernel — this makes the kernel flatter and redistributes weight away from observations close to x towards observations that are further from x (of course, closer observations still receive more weight in absolute terms, but relatively speaking their contribution diminishes).

Increasing h, and thus increasing the reach of the kernel, has the effect of increasing the amount of smoothing in the kernel density estimator. Intuitively, if the kernel bandwidth is miniscule, then at any given point $\hat{f}_h(x)$ will be influenced by just one or two (or zero) observations, so it will vary tremendously as observations move into and out of the kernel's reach. As you increase the bandwidth, however, $\hat{f}_h(x)$ becomes influenced by more and more observations at any given point, and so the addition or deletion of a single observation as you change x has little effect on $\hat{f}_h(x)$. In other words, it looks smooth. So more bandwidth means more smoothing, less bandwidth means less smoothing.

[4] All of these kernels are known as "second order" kernels, which is to say that their first non-zero moment is the second moment (any good kernel has its first moment, i.e. its mean, equal to zero). Higher order kernels exist, but they are not commonly used.

Smoother seems like it should be better (just watch some beer commercials), and isn't that the purpose of using the kernel density estimator to begin with? Why not just set h to be some very large number? The answer is that there is a tradeoff between reducing variance (smoothing) and increasing bias.[5] Increasing h increases the influence of observations that are far away from x. This introduces bias since, in general, the true density $f(x)$ is not constant — you are performing an extrapolation when you allow observations that are far from x to influence $\hat{f}_h(x)$ at x. In an extreme case, if you set h to be an enormous number, you should compute $\hat{f}_h(x)$ to basically be flat along the support of x.

How do we choose an "optimal" value for h? Like choosing the number of knots with regression splines, or operating the Digital Conveyer, choosing the value of h is more an art than a science. The preferred method is to simply eyeball it — choose h so that the density is reasonably smooth but still has shape. Alternatively, if you'd really like a formula to do the work for you, you can choose h such that it minimizes the integrated mean squared error (IMSE). The integrated squared error (ISE) is:

$$ISE(h) = \int (\hat{f}_h(x) - f(x))^2 dx$$

The IMSE is equal to the mean integrated squared error (MISE):

$$E[ISE(h)] = \int E[(\hat{f}_h(x) - f(x))^2] dx$$

Minimizing this quantity with respect to h yields:

$$h^* = \delta(\int f''(x)^2 dx)^{-0.2} N^{-0.2}$$

δ varies according to the kernel; $\delta = (\frac{\int K(u)^2 du}{(\int u^2 K(u) du)^2})^{0.2}$. Note that h^* depends on the square of the second derivative of $f(x)$ (the true density) — if $f(x)$ is highly variable, then $(\int f''(x)^2 dx)$ will be large, so h^* will be smaller. Intuitively, if $f(x)$ fluctuates a lot, then you

[5] This is seemingly the tradeoff that we always face in econometrics.

want a smaller bandwidth because you don't want to extrapolate much from observations that are far from x. Or, to put it more basically, you're not going to approximate a (true) choppy density with an (estimated) smooth density.

Of course, using this formula for h^* presupposes that we know $f(x)$, in which case, why are we trying to estimate it? There are two solutions to this problem. First, we can simply assume that $f(x)$ is close to some known distribution, say, the normal density.[6] In that case, $h^* = 1.364 \cdot \delta \cdot N^{-0.2} \cdot s$, where s is the sample standard deviation of x. In practice, it is common to use Silverman's plug-in estimate of the optimal bandwidth:

$$h^* = 1.364 \cdot \delta \cdot N^{-0.2} \cdot \min(s, iqr/1.349)$$

assumes normal density
→ could use this as starting value for an h search
→ still want to show end results are robust to choices of h.

where iqr is the sample interquartile range (i.e., the distance between the 0.25 quantile and the 0.75 quantile), in order to minimize the effects of extreme outliers.

Alternatively, we might think that we should look to the data to get a guess at $f(x)$ when we are choosing h. This method for choosing h is known as *cross-validation*. We know that the ISE is

$$ISE(h) = \int (\hat{f}_h(x) - f(x))^2 dx = \int \hat{f}_h(x)^2 dx - 2\int \hat{f}_h(x) f(x) dx + \int f(x)^2 dx$$

The last term does not depend on h, so we can omit it from the analysis. We thus have

$$ISE(h) = \int (\hat{f}_h(x) - f(x))^2 dx = \int \hat{f}_h(x)^2 dx - 2\int \hat{f}_h(x) f(x) dx + C$$

Our estimate of the quantity above turns out to be:

$$CV(h) = \frac{1}{N^2 h} \sum_i \sum_j \int K(\frac{x_i - x_j}{h} - t) K(t) dt - \frac{2}{N} \sum_{i=1}^{N} \hat{f}_{-i,h}(x_i)$$

[6] This seems a little stupid because you're presupposing a lot about the thing you're trying to estimate. Hence we won't concentrate on how h^* is derived.

where $\hat{f}_{-i,h}(x_i)$ is the kernel density estimate of $f(x)$ when omitting observation i from the data set. Note that $CV(h)$ no longer depends on knowing $f(x)$; the CV method thus works by choosing h such that we minimize $CV(h)$. Intuitively, the fact that h appears in the denominator of $\frac{1}{N^2h}$ puts upward pressure on h. The fact that it appears in $\int K(\frac{x_i-x_j}{h} - t)K(t)dt$ puts downward pressure on h. To see this, write the term as $\int K(a-t)K(t)dt = E[K(a-t)]$, where we think of t as a random variable with density $K(t)$. As h gets large, a approaches 0, and $E[K(a-t)]$ goes to $E[K(t)]$. This makes $E[K(a-t)]$ large because, by definition, $K(a-t)$ takes its maximum value at the points at which $K(t)$ has the most probability mass. So a large h makes $\int K(\frac{x_i-x_j}{h} - t)K(t)dt$ large but $\frac{1}{N^2h}$ small, and there's a balancing act.

In practice, cross-validation is computationally burdensome on any data set of moderate to large size. Estimating a kernel density on a data set of approximately 10,000 observations takes several seconds for me using *Stata/MP*. Doing this for hundreds or thousands of different values of h is clearly a time-consuming task.[7]

In most situations it is thus best to just eyeball the choice of h — perhaps start with the Silverman plug-in estimate and then see what happens as you change h in either direction. Regardless of what method you use to choose h, it is important to explore the sensitivity of your kernel density estimate to changes in the bandwidth — for example, what happens if you double h or halve h? Any critical reader is going to ask you whether your estimate is sensitive to the choice of bandwidth.

2.2 Kernel Regression

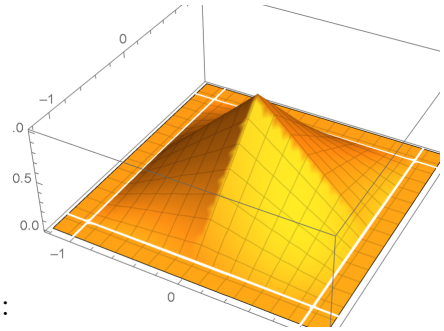
All of our discussion so far has centered on estimating a univariate density. In mean-land (where all we focus on is means), this is equivalent to focusing on estimating $E[Y]$. But what if we want to estimate a multivariate density or a conditional density? The latter is equivalent to estimating $E[Y|X]$ in mean-land. We will get to that in a minute, but first

[7]Note that the number of locations at which $\hat{f}_{-i,h}(x_i)$ will need to be calculated is heavily dependent on the size of the data set.

consider the case in which we want to estimate a bivariate density, $f(x, y)$. The multivariate kernel density estimator is a simple extension of the univariate kernel density estimator. In the bivariate case we have:

$$\hat{f}_h(x,y) = \frac{1}{Nh^2}\sum_{i=1}^{N} K(\frac{x-x_i}{h}, \frac{y-y_i}{h})$$

Of course, we now need a bivariate kernel. Multivariate kernels are generally just the product of several univariate kernels. For example, bivariate versions of the uniform, triangle, and Epanechnikov kernels are:



Uniform: $\frac{1}{4} \cdot \mathbf{1}(|u_1| < 1) \cdot \mathbf{1}(|u_2| < 1)$

Triangle: $((1 - |u_1|) \cdot \mathbf{1}(|u_1| < 1)) \cdot ((1 - |u_2|) \cdot \mathbf{1}(|u_2| < 1))$

Epanechnikov: $\frac{9}{16} \cdot (\mathbf{1}(|u_1| < 1)(1 - u_1^2)) \cdot (\mathbf{1}(|u_2| < 1)(1 - u_2^2))$

So a multivariate kernel density estimator using the triangle kernel is:

$$\hat{f}_h(x,y) = \frac{1}{Nh^2}\sum_{i=1}^{N}(1 - |\frac{x-x_i}{h}|) \cdot \mathbf{1}(|\frac{x-x_i}{h}| < 1)) \cdot ((1 - |\frac{y-y_i}{h}|) \cdot \mathbf{1}(|\frac{y-y_i}{h}| < 1)$$

This is just a natural extension of the univariate kernel density estimator — the same intuition still holds. It's possible (and often desirable) to choose different bandwidths for different variables; the h^2 term becomes $h_1 h_2$, the h associated with x becomes h_1, and the h associated with y becomes h_2. There is, however, a dark side to this increase in dimensionality, but we'll worry about that later.

Although estimating bivariate (or multivariate) densities is of some interest, our real focus lies in estimating conditional densities, $f(y|x)$, and conditional expectations, $E[y|x]$. The conditional density $f(y|x)$ equals $\frac{f(x,y)}{f(x)}$. Thus we can estimate $f(y|x)$ as

$$\hat{f}_h(y|x) = \frac{\hat{f}_h(x,y)}{\hat{f}_h(x)} = \frac{1}{h}\sum_{i=1}^{N} K(\frac{x-x_i}{h}, \frac{y-y_i}{h}) / \sum_{i=1}^{N} K(\frac{x-x_i}{h})$$

We can then use this conditional density to nonparametrically estimate the conditional expectation of y given x, $E[y|x]$. We know that $E[y|x] = \int yf(y|x)dy$. Thus a nonparametric estimator for $E[y|x]$ is:

$$\int y\hat{f}_h(y|x)dy = \int y\frac{\sum_{i=1}^{N} K(\frac{x-x_i}{h}, \frac{y-y_i}{h})}{h\sum_{i=1}^{N} K(\frac{x-x_i}{h})}dy$$

Focus for the moment on the numerator of the fraction above — the denominator is constant with respect to y, so we can pull it out of the integral and set it aside for now. Getting rid of the denominator, we have

$$\int y\sum_{i=1}^{N} K(\frac{x-x_i}{h}, \frac{y-y_i}{h})dy = \sum_{i=1}^{N} K(\frac{x-x_i}{h})\int yK(\frac{y-y_i}{h})dy$$

Time to integrate by substitution. Let $u = (y - y_i)/h$. We get:

$$\sum_{i=1}^{N} K(\frac{x-x_i}{h})\int K(u)(uh + y_i)hdu$$

So we have

$$\sum_{i=1}^{N} K(\frac{x-x_i}{h})\int K(u)(uh^2 + hy_i)du = \sum_{i=1}^{N} K(\frac{x-x_i}{h})(h^2\int uK(u)du + hy_i\int K(u)du)$$

But we know that the first moment of a random variable U distributed with kernel $K(u)$ equals 0 (i.e., $\int uK(u)du = 0$) and that any kernel $K(u)$ must integrate to 1 (i.e., $\int K(u)du = 1$). We therefore have

$$\sum_{i=1}^{N} K(\frac{x-x_i}{h})(h^2\int uK(u)du + hy_i\int K(u)du) = h\sum_{i=1}^{N} K(\frac{x-x_i}{h})y_i$$

Picking up the denominator that we set aside earlier, we finally have:

$$\int y\hat{f}_h(y|x)dy = \frac{\sum_{i=1}^{N} K(\frac{x-x_i}{h})y_i}{\sum_{i=1}^{N} K(\frac{x-x_i}{h})}$$

Pretty neat, and much simpler, huh? This procedure is called *kernel regression*, for reasons that should be obvious. For the intuition of what this is doing, note that we could write it as $E[y|x] = \frac{1}{W}\sum_{i=1}^{N} w(x, x_i) \cdot y_i$, where $W = \sum_i w(x, x_i)$. In other words, this just a weighted mean of the y_i. How are the weights determined? The weights are determined by the kernel function, $K(\frac{x-x_i}{h})$. Thus, for a given value x^*, $\hat{E}[y|x^*]$ is a weighted sum of the y_i's, where the weights are decreasing for observations (y_i, x_i) that lie far from x^*. In other words, $\hat{E}[y|x^*]$ is essentially the average value of y_i for observations in which x_i is close to x^*. Again, the bandwidth determines how much extrapolation occurs — if h is high, then observations for which x_i is far from x^* can still influence $\hat{E}[y|x^*]$. This will make $\hat{E}[y|x^*]$ smoother at the cost of potentially increasing bias.

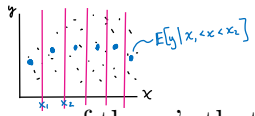
Kernal Reg. $= \overline{y}_h(x^*)$

Multivariate kernel regression is a simple extension of bivariate kernel regression. If we have two righthand side variables, x_1 and x_2, then we can nonparametrically estimate $\hat{E}[y|x_1, x_2]$ as

$$\int y\hat{f}_h(y|x_1, x_2)dy = \frac{\sum_{i=1}^{N} K(\frac{x_1-x_{1i}}{h})K(\frac{x_2-x_{2i}}{h})y_i}{\sum_{i=1}^{N} K(\frac{x_1-x_{1i}}{h})K(\frac{x_2-x_{2i}}{h})}$$

2.3 Lowess Regression

→ Just do a bin scatter plot too to check results — checking relationships between variables



• Trying to summarize E[y|x]
• Units of x vs. quantiles of x

local lin. Reg. $\hat{y}_h(x^*)$

Kernel regression takes a weighted mean of the y_i's that are "near" x when estimating $E[y|x]$. In this sense, it is a "local constant estimator" — $E[y|x]$ is assumed to be constant within a neighborhood near x (of course, for a small enough neighborhood — i.e., a small enough bandwidth — this should be true, but that doesn't mean that we have enough data to work with such a bandwidth). What if we instead used a "local regression estimator?" That is to say, what if we ran a weighted regression on the observations that are "near" x when estimating $E[y|x]$? This is the basic idea underlying Lowess regression — it's basically a kernel regression regression.[8]

At first glance, running a local regression may seem somewhat redundant. After all,

[8] You will be forgiven if, at this point, you start having visions of the episode of *Da Ali G Show* featuring "Boutros Boutros Boutros-Ghali."

a regression line always passes through $(\bar{x}, \bar{y})$, so if the data used to estimate $\hat{E}[y|x^*]$ are centered at x^*, then $\hat{E}[y|x^*]$ will equal the average of the y_i's in the neighborhood of x^* regardless of whether we take a mean of y or run a regression of y on x^* and use $\hat{\beta}x^*$. There are two reasons, however, that the local regression estimator can give different estimates than the local constant estimator. First, the data used to estimate the regression are not necessarily centered at x^* — they can easily be skewed one direction or the other. Second, we can (and often do) include higher order terms in the regression (e.g., a quadratic or cubic in x).

The algorithm for running a Lowess regression, introduced in Cleveland (1979), is as follows:

(1) For each data point x_i in the data set, run a weighted regression of y_j on x_j and x_j^2 (or higher order terms of x), where the weight for each observation j in the regression is generated by a tricubic kernel: $K(x_i, x_j) = \mathbf{1}(\frac{|x_i - x_j|}{h_i} < 1)(1 - (\frac{|x_i - x_j|}{h_i})^3)^3$. Note that for each data point we are running a regression potentially containing the entire data set. The kernel bandwidth, h_i, is the distance to the rth nearest x_j, i.e. the bandwidth varies by x_i so that the regression window always contains r observations. For each point x_i, $\hat{E}[y_i|x_i] = \hat{\beta}_{0i} + \hat{\beta}_{1i}x_i + \hat{\beta}_{2i}x_i^2$, where the coefficient estimates come from the local regression estimated in the neighborhood of x_i.[9]

(2) Let $\hat{\varepsilon}_i = y_i - \hat{E}[y_i|x_i]$. Define weights $\delta_j = \mathbf{1}(|\frac{\hat{\varepsilon}_j}{6s}| < 1)(1 - (\frac{\hat{\varepsilon}_j}{6s})^2)^2$, where s is the median of $|\hat{\varepsilon}_i|$.

(3) Generate a new set of $\hat{E}[y_i|x_i]$ for each observation i by fitting a weighted regression of the same order as in step (1) using weights $\delta_j K(x_i, x_j)$ (note that i indexes the observation at which the regression is begin fit, and j indexes all the observations in that regression).

(4) Repeat steps (2) and (3) t times. For the final pass, compute $\hat{E}[y|x]$ for all values of x (like with kernel regression) rather than just the values x_i that appear in the data set.

[9]Note the slight departure from the kernel regression estimates in that we are only estimating the regression at actual points x_i that occur in the data set, not any arbitrary x. This is because we are going to generate residuals in the next step that will be used to downweight outliers.

These are your Lowess regression fitted values.

Lowess regression is a fairly popular way of data smoothing or nonparametrically estimating conditional expectations. It is quite similar to kernel regression but with three key differences. First, it is a locally weighted regression of y_i's near x, rather than a locally weighted mean of y_i's near x. Second, it uses a variable size bandwidth (h_i) that increases in places where the x_i's are sparse and decreases in places where the x_i's dense. Finally, it uses a multiple pass procedure that downweights outliers when it runs the local regressions (hence the δ_j terms).

3 The Curse of Dimensionality

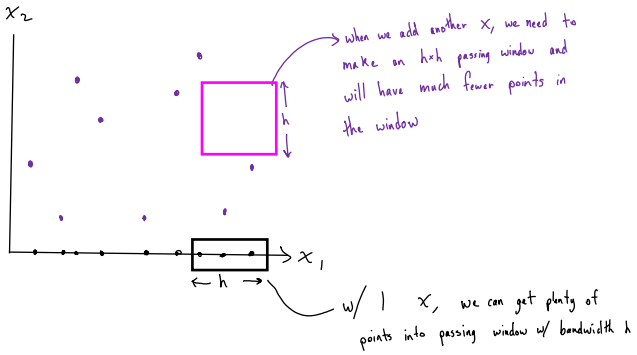
The nonparametric methods discussed above — series regression, kernel density estimation, kernel regression, and Lowess regression — generally work well when estimating a univariate density or the expectation of Y conditional on a single X. If we are interested in recovering causal estimates, however, then we rarely have just a single explanatory variable (unless the treatment is randomly assigned). Instead, we often assume that the treatment assignment, D_i, is independent of potential outcomes, $Y_i(0)$, conditional on an entire set of covariates, X_i. We demonstrated in the previous notes that, if this "selection on observables" assumption holds, and we have a constant treatment effect, then it is sufficient to simply control for $E[D_i|X_i]$ when regressing Y_i on D_i. We rarely know the functional form of $E[D_i|X_i]$, but using the nonparametric procedures presented above we can, in principle, estimate $E[D_i|X_i]$ without any functional form assumptions.

In practice, however, we are plagued by the "Curse of Dimensionality." If we want to estimate $E[D|X]$ via kernel regression, then need to essentially estimate a multivariate density of dimension r, where r is the number of variables in X. However, it turns out that the sparsity of the data grows exponentially with r — the more dimensions you have, the less and less data you have to estimate $E[D|X^*]$ at any given point X^*.[10] This fact is known

[10]The same thing holds with series regressions. Essentially, as you add additional variables to X, you

Suppose we want to est. $\mathbb{E}[D|X]$.

- Sparcity grows exponentially with k (# of X_s)
- Some use semi parametric methods to deal w/ dimensionality



Matching Estimators

→ Want to operationalize the Conditional Independence Assumption

CIA: $Y_i(1), Y_i(0) \perp\!\!\!\perp D_i | X_i$

→ When matching, we can compare obs w/ same (similar) X_i but different D_i

N_T treated, N_c control units

N_T sets of weights w/ N_c weights per set

$w_i(j)$ $\quad i \in \{1,\dots,N_T\}$, $j \in \{1,\dots,N_c\}$, $\sum_j w_i(j) = 1$

$$\hat{\tau}_{M(TOT)} = \frac{1}{N_T} \sum_{i \text{ s.t. } D_i=1} \left[y_i - \sum_{j \text{ s.t. } D_j=0} w_i(j) \cdot y_j \right]$$

- could set $w_i(j) = \frac{1}{N}$ $\forall j$ ⇒ unadjusted difference in means
- want ↑ $w_i(j)$ for obs i,j that are close

Exact Matching

- fully saturated bins of discrete X's (both C & T)
- $w_i(j) = 1$ if $X_i = X_j$, 0 o.w. if 1-1 ratio of C:T
- $w_i(j) = \frac{1}{N(x)}$ / # of obs in bin (x), 0 o.w.
  → need to ensure $\sum_j w_i(j) = 1$

Nearest Neighbor Matching

$w_i(j) = 1$ if i is j's NN, 0 o.w.

→ need distance metric

Euclidean: $(X_i - X_j)'(X_i - X_j)$

Mahalanobis: $(X_i - X_j)' \Sigma_x^{-1} (X_i - X_j)$, $\Sigma_x = \text{cov}(X)$

- good for roughly equal # of C, T
- When we have more of either and don't want to throw away data by only matching 1-1, ...

Kernal Matching

$$w_i(j) = \frac{K\left(\frac{x_i - x_j}{h}\right)}{\sum_k K\left(\frac{x_i - x_k}{h}\right)} \longrightarrow \frac{K\left(\frac{1}{h} d(x_i, x_j)\right)}{\sum_k K\left(\frac{1}{h} d(x_i, x_k)\right)} \quad \text{for metric } d(\cdot,\cdot)$$

- defining distance between unordered discrete X's (like race)
  - → exact match on unordered discrete OR if we have separate indicators for each factor
    $0\,0\,0\,1 = x_1$ vs $0\,1\,0\,0 = x_2$ vs $0\,0\,1\,0 = x_3$ $\Rightarrow d(x_1,x_2) \stackrel{?}{=} d(x_2,x_3) \stackrel{?}{=} d(x_3,x_1)$ what d would give us this?
  - → ordered discrete? Not cardinal

Propensity Score Methods

Strongly ignorable treatment selection: $Y_i(0), Y_i(1) \perp D_i \mid X_i$ (1)

$0 < P(D_i = 1 \mid X_i) < 1$ (2)

Propensity Score: $p(x) \equiv P(D_i = 1 \mid X_i = x) = \mathbb{E}[D_i \mid X_i = x]$

WTS: $p(x) = P(D_i = 1 \mid p(x)) = P(D_i = 1 \mid Y_i(0), Y_i(1), p(x))$

$$P(D_i = 1 \mid Y_i(0), Y_i(1), p(x)) = \mathbb{E}[D_i \mid Y_i(0), Y_i(1), p(x)]$$

$$= \mathbb{E}\left\{\mathbb{E}[D_i \mid \cancel{Y_i(0)}, \cancel{Y_i(1)}, \cancel{p(x)}, X_i = x] \,\middle|\, Y_i(0), Y_i(1), p(x)\right\}$$

since $D_i \perp\!\!\!\perp Y_i(0), Y_i(1) \mid X_i$; since $p(x)$ is fixed given $X_i = x$

$$= \mathbb{E}\left\{\mathbb{E}[D_i \mid X_i = x] \,\middle|\, Y_i(0), Y_i(1), X_i = x\right\}$$

$$= \mathbb{E}\left\{p(x) \mid Y_i(0), Y_i(1), X_i = x\right\}$$

$$= p(x)$$

as the widely feared "Curse of Dimensionality."

It is thus generally infeasible to apply fully nonparametric methods when X contains more than a couple variables. Although you can sometimes reduce the dimensionality problems by making various parametric assumptions (e.g., assuming an additive structure between the different elements of X when using series estimators), you can never truly defeat the Curse of Dimensionality. It is, after all, a curse.

4 Additional References

Cleveland, William. "Robust Locally Weighted Regression and Smoothing Scatterplots." *Journal of the American Statistical Association*, 1979, 74, 829-836.

have to include increasingly large numbers of interactions between all the powers and splines of the various x variables, and the whole thing starts to blow up.