LECTURE NOTE 12: NONPARAMETRIC METHODS (NOT ON FINAL EXAM)

1 Introduction

Most of the statistics and econometrics you have learned until now has been *parametric*, in the sense that the researcher imposes a model structure (e.g., a linear function linking X and Y) first and then estimates the parameters of that structure. This lecture introduces a number of *nonparametric* methods that allow the data to determine the model's structure. We focus on two tasks: representing the distribution of a single variable and representing the bivariate relationship between two variables. Because Lecture Note 11 discussed methods closely related to the second task, we will start there.

2 Nonparametric Estimation of the Regression Function

Lecture Note 2 focused on estimating a linear relationship between X and Y:

$$Y = \beta_0 + \beta_1 X + U$$

This linear function is sometimes interpreted as an approximation of $E[Y|X]$, the conditional expectation of Y given X. In practice, the conditional expectation may be non-linear, so we might write:

$$E[Y|X] = g(X)$$

where the flexible function $g(X)$ is called the *regression function*. We then seek to flexibly estimate:

$$Y = g(X) + U$$

We discuss two approaches to this estimation problem.

2.1 Polynomial Regression

The first approach, polynomial regression, should be familiar. It involves adding higher-order polynomial terms to our original linear regression:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_K X^K + U$$

One can prove that as K grows to infinity, the polynomial becomes arbitrarily close to $g(X)$, the true regression function. That is to say, as K grows, the bias of the estimator shrinks. Of course, the variance of the estimator increases as we add more polynomial terms, so we once again have a tradeoff between bias and variance.

One unappealing aspect of polynomial regression is that the estimate of $g(x)$ for some value x is jointly determined by all X_i, even those very far away from x. Intuitively, we might want to limit $\hat{g}(x)$ to be based only on X_i's close to x.

2.2 Local Regression

Local regression achieves this goal by setting a grid of x_k values and then estimating the regression function at each x_k, $g(x_k)$, only using observations close to x_k. We will study a local regression estimator called *local linear regression*. This estimator involves running a linear regression using observations near x_0 to predict $\hat{g}(x_0)$, then doing the same for x_1, x_2, and so forth. Finally, all of the estimates $\hat{g}(x_0), \hat{g}(x_1), \hat{g}(x_2), \cdots$ are plotted in a graph, usually connected by interpolation.

Local linear regression requires two choices by the researcher. The first choice is the *bandwidth* (h), which determines how far away from x_k we are willing to go to estimate $g(x_k)$. A smaller bandwidth will lead to less bias and more variance. A local linear regression with a larger bandwidth will look smoother but will pick up fewer nuances in the shape of the regression function. Researchers have developed a number of approaches for choosing the best bandwidth, but in practice, these mainly serve as a starting point. Nonparametric estimation typically involves experimentation with a wide range of bandwidths.

The second choice is the *kernel function*, which determines how we weight observations within the bandwidth. Often we wish to put more weight on observations closer to the grid point x_k. We write the kernel function as $K(z)$, where $z = \frac{X_i - x_k}{h}$. Note that $z = 0$ when X_i exactly equals the grid point x_k, and $z = -1$ or 1 at the boundary of the bandwidth. The kernel is defined to be a positive function ($K(z) \geq 0$) with mean zero ($\int zK(z)dz = 0$) and unit mass ($\int K(z)dz = 1$). A simple example is the *uniform (or rectangular) kernel*, which assigns equal weight to all obsevations within the bandwidth. In Lecture Note 11, we used the uniform kernel to estimate the regression function in a regression discontinuity design. Another simple option is the *triangular kernel*, which puts more weight on observations closer to x_k, such that the weight declines

linearly with distance within the bandwidth. Outside the regression discontinuity setting, a more common choice is the *Epanechnikov kernel*, which puts more weight on observations closer to x_k, such that the weight declines *quadratically* with distance within the bandwidth. In practice, the choice of kernel tends to be much less important than the choice of bandwidth.

After choosing a kernel $K(\cdot)$ and a bandwidth h, we solve the weighted least squares problem:

$$\min_{b_0,b_1} K\left(\frac{X_i - x_k}{h}\right)(Y_i - b_0 - b_1 X_i)^2$$

The solution is for the slope b_1 is the same as in weighted least squares:

$$\hat{\beta}_1 = \frac{\sum_i w_i(Y_i - \bar{Y})(X_i - \bar{X})}{\sum_i w_i(X_i - \bar{X})^2}$$

where $\bar{Y} = \sum_{i=1}^N w_i Y_i$, $\bar{X} = \sum_{i=1}^N w_i X_i$, and the weights are:

$$w_i = \frac{K\left(\frac{X_i - x_0}{h}\right)}{\sum_{j=1}^N K\left(\frac{X_j - x_0}{h}\right)}$$

The solution for the intercept b_0 is:

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

And the estimated regression function at x_k is:

$$\hat{g}(x_k) = \hat{\beta}_0 + \hat{\beta}_1 x_k$$

This procedure is carried out for every grid point x_k and then plotted in a graph. The basic idea is that we run a separate 'mini-regression' around every grid point and then connect the predicted values. In Stata, `lpoly` implements local polynomial regression. For local linear regression, set the `degree` option to 1, and use `bw` to control the bandwidth. It uses the *Epanechnikov kernel* by default, but you can change it using the `kernel` option. In R, you can use `locpoly()` from the `KernSmooth` package, with `degree` set to 1, and use `bandwidth` to control the bandwidth. It defaults to the *Gaussian kernel*, which you can modify with the `kernel` option.

3 Nonparametric Estimation of the Density

Often we are interested in representing a distribution of a single variable beyond just estimating its mean and standard deviation. The familiar histogram is one approach to estimating the density, but we can extend

the kernel methods from above for a more flexible approach.

3.1 Histogram

You have probably already encountered the histogram. Suppose X is a continuous variable that takes on values in the range $[a, b]$. Divide this range into equal intervals $k = 1, \cdots, K$, so that the size of each interval is $\frac{b-a}{K}$. Some histograms plot the number of observations in the interval, N_k. Others plot the share of observations in the interval, $\frac{N_K}{N}$. Yet others rescale this share to estimate the density of X, $f_X(x)$. Recall that a probability density function must integrate to 1. To ensure this property, we multiply the share by $\frac{K}{b-a}$. Then we get an estimate of the density at any point x in interval k:

$$\hat{f}_X(x) = \frac{N_k}{N} \times \frac{K}{b-a}$$

It can be shown that as K increases, the bias of the histogram estimator shrinks, and the variance grows. That is to say, shorter intervals lead to noisier but more accurate estimates.

In Stata, you can implement this estimator using `hist`. In base R, you can use `hist()`, with `freq = FALSE` if you want the density rather than the frequency N_k. In R using `tidyverse`, you can use `ggplot(df, aes(x = varname)) + geom_hist()` for the frequency or `ggplot(df, aes(x = varname, y = after_stat(density))) + geom_hist()` for the density.

3.2 Kernel Density

One disadvantage of the histogram is that it may assign very different density estimates to values of X that are very close to each other but lie on different sides of the interval boundary. Another disadvantage is that estimates at the boundary tend to be more biased than estimates in the middle. We can address these concerns with the *centered histogram*, which estimates the density at any point as if that point is at the middle of the interval. We choose an interval width $2h$ around each point x_k and then calculate:

$$\hat{f}_X(x_k) = \sum_{i=1}^{N} \frac{1\left[x_k - h < X_i < x_k + h\right]}{2hN}$$

where $1[\cdot]$ is an indicator function that equals 1 if x_i is within h of x_k and equals 0 otherwise.

In fact, the centered histogram is a specific case of an estimator called the *kernel density estimator*. Here, every observation in the interval from $x_k - h$ to $x_k + h$ is given equal weight; it uses the *uniform (or rectangular)*

kernel described above. We can rewrite the formula in the following way:

$$\hat{f}_X(x_k) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{X_i - x_k}{h}\right)$$

where $K\left(\frac{X_i - x_k}{h}\right) = \frac{1}{2} \times 1\left[-1 < \frac{X_i - x_k}{h} < 1\right]$ is the *uniform kernel*. As in Section 2, we have many kernel options other than the uniform kernel, and again a common choice is the *Epanechnikov kernel*.

Given the choice of kernel, each observation i is assigned a weight between 0 and 1 based on the location of X_i relative to x_k and the chosen value of the bandwidth h. These weights are then summed across observations and divided by Nh to form the estimate $\hat{f}_X(x_k)$. This procedure, called *kernel density estimation*, is carried out over a grid of equidistant values x_k and then plotted in a graph. Exactly the same bandwidth tradeoffs arise as earlier in the lecture note: a smaller bandwidth will lead to less bias and more variance.

In Stata, the `kdensity` function implements kernel density estimation. By default, it uses the Epanechnikov kernel and sets the bandwidth based on a forumla for the bandwidth that would be optimal if the data were normally distributed. But one can modify these defaults with the `kernel()` and `bwidth()` options. In R, you can use `ggplot(df, aes(x = varname)) + geom_density()`. You can modify the kernel and bandwidth with the `kernel` and `bw` arguments. The default in `ggplot` is the *Gaussian kernel*, which uses the formula for the normal distribution.