

Lecture Note 14: Nonparametric Regression

Suppose that for $i = 1, \dots, n$, (y_i, x_i) are IID from some joint distribution. We assume y_i and x_i are scalar and continuously distributed, and that x_i has compact support in \mathbb{R} . We want to estimate the function

$$m(x) := E[y_i | x_i = x]$$

without making strong parametric assumptions. We will review some standard methods, omitting technical details for the time being.

Series estimator

The basic idea behind series estimators is to approximate the regression function $m(x)$ by a series

$$\begin{aligned} m(x) &\approx \theta_0 g_0(x) + \theta_1 g_1(x) + \dots + \theta_J g_J(x) \\ &= \sum_{j=0}^J \theta_j g_j(x). \end{aligned}$$

Here, the $g_j(\cdot)$ are called *basis functions*, and are chosen in advance, hopefully in such a way that linear combinations of them lead to a wide range of functional forms. We then use some method to estimate the weights θ_j using the observed data.

The simplest example is to take the polynomial functions:

$$\begin{aligned} g_0(x) &= 1 \\ g_1(x) &= x \\ g_2(x) &= x^2 \\ &\vdots \\ g_J(x) &= x^J \end{aligned}$$

We could then estimate the θ_j by least squares:

$$\min_{\theta_0, \dots, \theta_J} \sum_{i=1}^n \left(y_i - \sum_{j=0}^J \theta_j x_i^j \right)^2 = \min_{\theta_0, \dots, \theta_J} \sum_{i=1}^n (y_i - \theta_0 - \theta_1 x_i - \dots - \theta_J x_i^J)^2.$$

This amounts to a polynomial regression of y on x .

A justification for this is the following famous theorem:

Theorem 1 (*Stone-Weierstrass*) Let $m(\cdot)$ be a continuous bounded function with compact domain

$\mathcal{X} \subset \mathbb{R}$. Then, for any $\epsilon > 0$, there exists a polynomial function

$$l(x) = \sum_{j=0}^{\infty} \theta_j x^j,$$

such that

$$\sup_{x \in \mathcal{X}} |m(x) - l(x)| < \epsilon.$$

This says that any continuous function can be approximated arbitrarily well by some polynomial, and suggests that if J is reasonably large, we can approximate $m(x) = E[y_i | x_i = x]$ by a J th order polynomial.

How do we choose J ? Clearly, we cannot have $J \geq n$, because then we will have more parameters $\theta_0, \dots, \theta_J$ to estimate than we have data points. We want to have $(J + 1)$ smaller than n , but if we allow J to grow as $n \rightarrow \infty$, we can still have $\hat{m}(x) \xrightarrow{p} m(x)$ at every x .

Practically speaking, there tends to be a couple of different approaches taken to choosing J . One is to keep adding terms until the last term is no longer statistically significant (using the standard t-statistic).

An approach that seems to work well in practice is v -fold cross validation. The idea is to try to estimate the mean squared error $E[(y_i - \hat{m}(x_i))^2]$, and choose J to minimize the estimated MSE.

Divide the sample into v equally sized groups. For each i , let $\hat{m}_J^{-i}(\cdot)$ be the estimated regression function using observations from the $v - 1$ groups that don't include observation i , and using a J th order polynomial specification. Then form

$$M\hat{S}E(J) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{m}_J^{-i}(x_i))^2.$$

We then choose the order J which minimizes this approximate MSE.

There are other sets of basis functions that can be used, besides the ordinary polynomials. Fourier analysis typically uses sine and cosine-based functions. So-called neural network methods use a different set of basis functions related to the logistic CDF.

Series Logit Estimator

Suppose that y_i is binary instead of being continuous, so that

$$E[y_i | x_i = x] = Pr(y_i = 1 | x_i = x).$$

In principle, we could continue to use the series estimator without modification. However, this has the undesirable property that it can lead to estimates $\hat{m}(x)$ outside the unit interval $[0, 1]$. This is the usual problem with the linear probability model.

One possibility is to use a transformation to keep the predicted probabilities between 0 and 1. For example, we can use the inverse-logit transform:

$$\Lambda(z) = \frac{\exp(z)}{1 + \exp(z)},$$

and approximate the conditional probability function as:

$$\begin{aligned} Pr(y_i = 1|x_i = x) &\approx \Lambda(\theta_0 + \theta_1 x + \cdots + \theta_J x^J) \\ &= \frac{\exp(\theta_0 + \theta_1 x + \cdots + \theta_J x^J)}{1 + \exp(\theta_0 + \theta_1 x + \cdots + \theta_J x^J)}. \end{aligned}$$

Notice that Λ is strictly monotone and invertible, so for any function $m(x)$, we can define

$$g(x) := \Lambda^{-1}(m(x)),$$

so that

$$m(x) = \Lambda(g(x)).$$

This suggests that by choosing a sufficiently large J and appropriate coefficients $\theta_0, \dots, \theta_J$, we can approximate any conditional probability function $Pr(y_i = 1|x_i = x)$ very well.

To estimate the $\theta_0, \dots, \theta_J$, one natural way is to use maximum likelihood:

$$(\hat{\theta}_0, \dots, \hat{\theta}_J) = \arg \max_{\theta_0, \dots, \theta_J} \prod_{i=1}^n [\Lambda(\theta_0 + \theta_1 x + \cdots + \theta_J x^J)]^{Y_i} \times [1 - \Lambda(\theta_0 + \theta_1 x + \cdots + \theta_J x^J)]^{1-Y_i}$$

Nearest Neighbor estimator

If x_i were discrete, then a natural estimator for $E[y_i|x_i = x]$ would be

$$\hat{m}(x) = \frac{\sum_i 1(x_i = x) y_i}{\sum_i 1(x_i = x)}.$$

That is, just take the observations with $x_i = x$ and average their outcome values. The k -nearest neighbor estimator is defined as

$$\hat{m}(x) := \frac{1}{k} \sum_{i: x_i \in N_k(x)} y_i,$$

where $N_k(x)$ is set of the k values of x_i that are closest to x . So we find the k observations with x_i values closest to x , and average their outcomes. The idea is that if $m(x)$ is relatively smooth, so it does not change too much as x varies in a small neighborhood, then taking an average over values close to x should give a reasonable approximation.

We can calculate this at a set of different values of x , producing an estimated regression function that is easily plotted.

It's useful to think about what happens for different values of k .

If $k = n$, then we are using all of the observations, and $\hat{m}(x)$ just becomes the sample average of y_i . This produces a perfectly flat estimated function. On the one hand, we are using a lot of data to do the averaging, so the variance is relatively low. But, if $m(x)$ is not actually constant, then our estimator will be biased for many particular values of x .

At the other extreme, we could set $k = 1$. This uses just one observation in the sample average. Since we are only using observations with x_i very close to x , the bias should be relatively small, but at the cost of using very few observations - hence high variance.

How to pick k ? We can use the same v -fold cross validation technique as in the series case. Now, instead of varying the number of terms J , we try different values of k and pick the one that minimizes the CV estimate of MSE.

Kernel estimator

The k -nearest-neighbor method takes averages in “neighborhoods” $N_k(x)$ of a point x , where the neighborhoods are defined in such a way as to contain a fixed number k of observations. An alternative approach is to define a neighborhood as a fixed interval about x . For example, we could take intervals of the form $[x - c, x + c]$, where c is some constant we choose, leading to an estimator:

$$\hat{m}(x) := \frac{\sum_{i=1}^n 1(x_i \in [x - c, x + c]) y_i}{\sum_{i=1}^n 1(x_i \in [x - c, x + c])}.$$

As in the nearest neighbor approach, the intuition is to take local averages of outcomes y_i with x_i close to x .

We can extend this idea, by weighting the observations by how close they are to x . Let $K(u)$ be a function that is continuous, symmetric about 0 and bounded, and satisfying:

$$\int K(u) du = 1.$$

We call K a kernel. An example is $K(u) = \phi(u)$, the standard normal density function. We weight observations by

$$\frac{K\left(\frac{x_i - x}{h}\right)}{h},$$

where h is called a bandwidth. Note that as $h \rightarrow 0$, only observations with x_i very close to x will receive any weight.

The local averaging estimator then becomes:

$$\hat{m}(x) := \frac{\sum_{i=1}^n K\left(\frac{x_i - x}{h}\right) y_i}{\sum_{i=1}^n K\left(\frac{x_i - x}{h}\right)}.$$

This is called the kernel regression estimator, or the Nadaraya-Watson estimator.

The estimator depends on the choice of kernel K , and on the bandwidth h . The most popular choice for the kernel function is the Epanechnikov kernel:

$$K(u) = 0.75(1 - u^2)1(|u| \leq 1),$$

which has some optimality properties and is easy to calculate. In practice, the Epanechnikov kernel and the normal kernel usually give similar results.

The choice of bandwidth makes a big difference in the resulting estimate. Choosing h small leads the estimator to only use observations very close to x , leading to a “wiggly” function. If we use larger h , the estimated function will be smoother, but since we are averaging observations with quite different x_i values, we expect more bias.

As before, we can use v -fold CV. There are a number of other methods for choosing the bandwidth.

Examples:

The first two graphs use data generated from a linear model. Figure 1 shows the data, the true regression line (marked with “+”), and the fits from a linear, quadratic, and cubic regression.

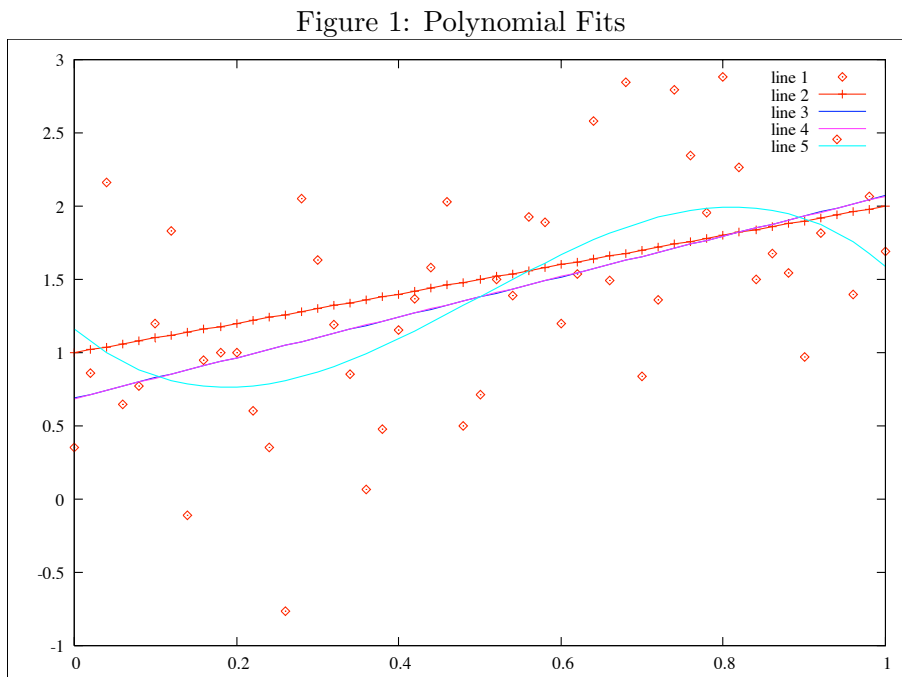
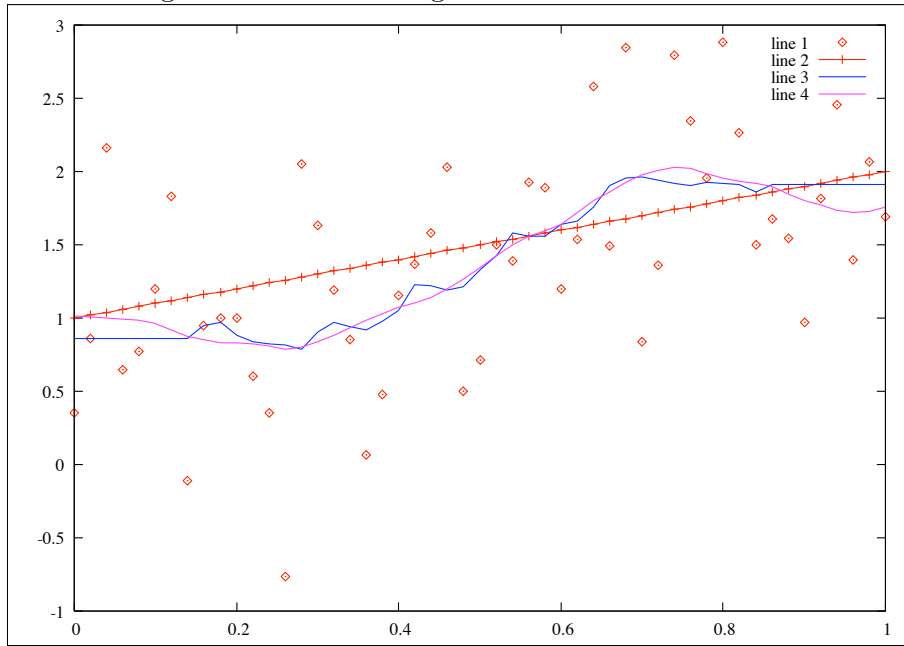


Figure 2 shows k-nearest neighbor and Epanechnikov kernel regression estimates.

Figure 2: K-nearest neighbor and Kernel Estimates



Figures 3 and 4 repeat the exercise, but the data are generated from the following model:

$$y = \cos(8 \cdot x) + \epsilon,$$

where ϵ is normally distributed.

Figure 3: Polynomial Fits

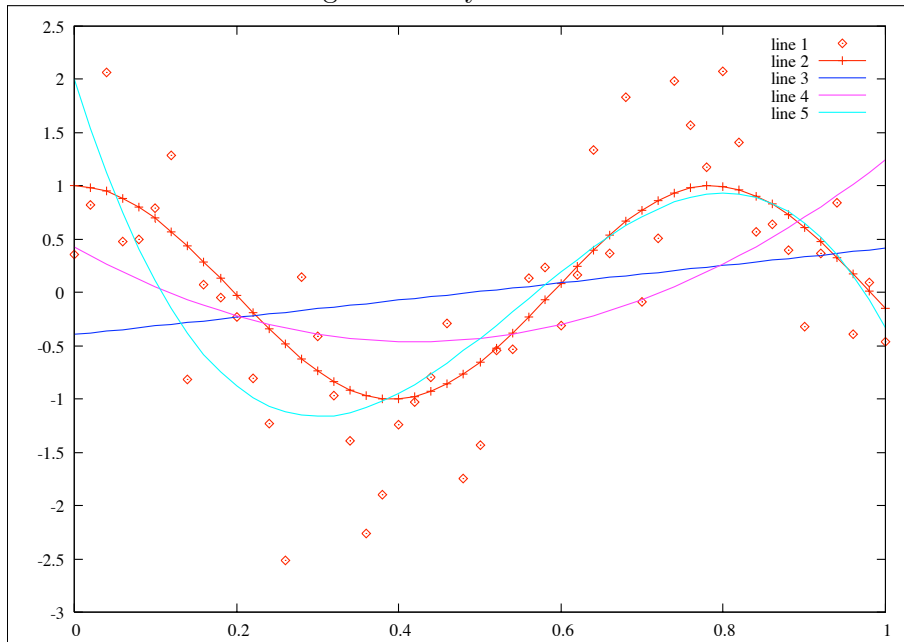


Figure 4: K-nearest neighbor and Kernel estimates

