ECON 950 — Winter 2019

Prof. James MacKinnon

3. Linear Methods for Classification

The output variable is now discrete. We want to divide up the space of the input variables into a collection of regions associated with K different predicted outcomes (or groups, or classes).

Let $\boldsymbol{Y}$ be a matrix with K columns of observations on the output variables. Each column contains 0s and 1s, and each row contains a single 1.

For example, if observation 44 belongs to group 3, row 44 of $\boldsymbol{Y}$ will have a 1 in column 3 and 0s in all other columns.

The OLS estimator is

$$\hat{\boldsymbol{B}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y}, \tag{1}$$

where $\hat{\boldsymbol{B}}$ is $(p+1) \times K$ and $\boldsymbol{X}$ has $p+1$ columns, one of them a constant term.

This is a generalization of the **linear probability model**. The latter is used when there are only two outcomes. In that case, we can use just one regression, because the fitted values must sum to 1 over all the equations.

In general, we could get away with estimating $K-1$ regressions and obtaining the fitted values for the K^{th} by using this property.

For any input vector $\boldsymbol{x}$, we can calculate the fitted output $\hat{\boldsymbol{f}}(\boldsymbol{x}) = [1, \boldsymbol{x}^\top]\hat{\boldsymbol{B}}$, which is a K-vector.

Then we classify $\boldsymbol{x}$ as belonging to group (or class, or region) k if k corresponds to the largest element of $\hat{\boldsymbol{f}}(\boldsymbol{x})$.

Unfortunately, as ESL explains, linear regression often performs very badly when $K \geq 3$. The problem is "masking," where middle classes get missed.

ESL works through an example where the largest value of $\boldsymbol{x}^\top\hat{\boldsymbol{B}}_k$ is always for one of the two extreme classes, even though it is easy to see visually that there are three classes which can be separated without error.

This example is shown in ESL-fig4.02.pdf.

3.1. Linear Discriminant Analysis

Suppose that $f_k(\boldsymbol{x})$ is the density of $\boldsymbol{X}$ in class k, where k runs from 1 to K.

Suppose that π_k is the prior probability of class k, where the event $G = k$ means that the class actually is k.

Then, by Bayes' theorem,

$$\Pr(G = k \,|\, \boldsymbol{x}) = \frac{\pi_k f_k(\boldsymbol{x})}{\sum_{\ell=1}^{K} \pi_\ell f_\ell(\boldsymbol{x})}. \tag{2}$$

So we just need to find the $f_k(\boldsymbol{x})$ and combine them with the prior probabilities.

If the density of each class is multivariate normal,

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\bigl(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)\bigr). \tag{3}$$

In the case of **linear discriminant analysis**, we assume that $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$ for all k.

In general, the log of the ratio of the posteriors (that is, the log odds) is

$$\log \frac{\pi_k}{\pi_\ell} + \log \frac{f_k(\boldsymbol{x})}{f_\ell(\boldsymbol{x})}. \tag{4}$$

When the densities are given by (3) with constant $\boldsymbol{\Sigma}$, this reduces to

$$\log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_\ell). \tag{5}$$

Because the two covariance matrices are the same, this simplifies to

$$\log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_\ell) + \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_\ell), \tag{6}$$

which is linear in $\boldsymbol{x}$.

Since this is true for any pair of classes, all boundaries must be hyperplanes.

Where else have we seen a model in which the log of the odds is linear in $\boldsymbol{x}$?

The **linear discriminant function** for class k is

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log \pi_k. \tag{7}$$

We simply classify $\boldsymbol{x}$ as belonging to the class k for which (7) is largest.

Of course, for all the classes, we need to estimate

$$\hat{\pi}_k = N_k/N, \tag{8}$$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{i \in G_k} \boldsymbol{x}_i \tag{9}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{i \in G_k} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^\top. \tag{10}$$

Here G_k is the set of observations that belong to class k.

Since the values of $\delta_k(\boldsymbol{x})$ depend on the $\hat{\pi}_k$, we could change the boundaries of the classes by using different estimates of the π_k.

For example, we could shrink them towards $1/K$:

$$\hat{\pi}_k(\alpha) = \alpha \frac{N_k}{N} + (1-\alpha)\frac{1}{K}. \tag{11}$$

When N is small and the N_k are not too different, this ought to produce better results by accepting more bias in return for less variance.

ESL discusses the relationship between LDA and classification by linear regression. For two classes, there is a close relationship.

3.2. Quadratic Discriminant Analysis

If the $\boldsymbol{\Sigma}_k$ matrices are not equal, then the log odds does not simplify to (6).

In contrast to (7), the discriminant functions are now quadratic in $\boldsymbol{x}$. The **quadratic discriminant functions** are

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + \log \pi_k. \tag{12}$$

Now we have to estimate separate covariance matrices for each class:

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k}\sum_{i\in G_k}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^\top. \tag{13}$$

Note that these matrices are $p \times p$, each with $p(p+1)/2$ parameters to estimate.

Especially when some of the N_k are small, the $\boldsymbol{\Sigma}_k$ may not be estimated very well, which will probably cause the classification procedure to perform much worse on the test data than on the training data.

There is an interesting alternative to QDA that is based on LDA.

Simply augment $\boldsymbol{x}$ by adding all squares and cross-products, and then perform LDA. See ESL-fig4.01.pdf and ESL-fig4.06.pdf. Of course, this also adds a lot of parameters if p is large.

ESL reports that LDA and QDA often work remarkably well, even though the Gaussian assumption (and the equal covariance matrix assumption) is surely not true in the vast majority of cases.

Presumably, the parsimony of LDA causes it to have low variance but high bias. In many cases, it is apparently worth accepting a lot of bias in return for low variance.

Perhaps the data can only support simple decision boundaries such as linear or quadratic ones, and the estimates provided via the Gaussian LDA and QDA models are stable.

Since we need $|\hat{\boldsymbol{\Sigma}}_k|$ and $\hat{\boldsymbol{\Sigma}}_k^{-1}$ rather than just $\hat{\boldsymbol{\Sigma}}$, it is convenient to use the eigen decomposition

$$\hat{\boldsymbol{\Sigma}}_k = \boldsymbol{U}_k \boldsymbol{D}_k \boldsymbol{U}_k^\top. \tag{14}$$

Then

$$\log |\hat{\boldsymbol{\Sigma}}_k| = \sum_{\ell=1}^{p} \log d_{k\ell}, \tag{15}$$

and

$$(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)^\top \hat{\boldsymbol{\Sigma}}_k^{-1} (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k) = \big(\boldsymbol{U}_k^\top (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)\big)^\top \boldsymbol{D}_k^{-1} \big(\boldsymbol{U}_k^\top (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)\big). \tag{16}$$

Getting the eigenvalues and eigenvectors is expensive, but it gives us the determinant and the inverse almost for free.

3.3. Regularized Discriminant Analysis

We can shrink the covariance matrices towards their average:

$$\hat{\boldsymbol{\Sigma}}_k(\alpha) = \alpha \hat{\boldsymbol{\Sigma}}_k + (1 - \alpha) \hat{\boldsymbol{\Sigma}}. \tag{17}$$

Of course, α has to be chosen, perhaps by cross-validation.

Similarly, we could shrink $\hat{\boldsymbol{\Sigma}}$ towards the scalar covariance matrix $\hat{\sigma}^2\mathbf{I}$:

$$\hat{\boldsymbol{\Sigma}}(\gamma) = \gamma\hat{\boldsymbol{\Sigma}} + (1-\gamma)\hat{\sigma}^2\mathbf{I}. \tag{18}$$

Combining (17) and (18), we could use

$$\hat{\boldsymbol{\Sigma}}_k(\alpha,\gamma) = \alpha\hat{\boldsymbol{\Sigma}}_k + (1-\alpha)\big(\gamma\hat{\boldsymbol{\Sigma}} + (1-\gamma)\hat{\sigma}^2\mathbf{I}\big), \tag{19}$$

which has two tuning parameters to specify.

In Section 4.3.3, ESL goes on to discuss reduced-rank LDA, which is closely related to LIML and to Johansen's approach to cointegration.

3.4. Logistic Regression

The log of the odds between any two classes is assumed to be linear in $\boldsymbol{x}$. This implies that

$$\Pr(G=k\,|\,\boldsymbol{x}) = p_k(\boldsymbol{x},\boldsymbol{\theta}) = \frac{\exp(\beta_{k0}+\boldsymbol{x}^\top\boldsymbol{\beta}_k)}{1+\sum_{\ell=1}^{K-1}\exp(\beta_{\ell 0}+\boldsymbol{x}^\top\boldsymbol{\beta}_\ell)}, \tag{20}$$

where $\boldsymbol{\theta}$ contains all of the parameters. There are $(K-1)*(p+1)$ of these.

ESL discusses ML estimation of the logit model ($K=2$). What they have there, and later in Section 4.4.3 on inference, is closely related to the material in Sections 11.2 and 11.3 of ETM.

They do not discuss estimation of the multinomial logit (multilogit) model except in Exercise 4.4.

For binary logit, the probability of class 1 is

$$p(\boldsymbol{x},\boldsymbol{\theta}) = \frac{\exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})}{1+\exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})} = \frac{1}{1+\exp(-\beta_0 - \boldsymbol{x}_i^\top \boldsymbol{\beta})}. \tag{21}$$

Thus the contributions to the loglikelihood are

$$\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} - \log\bigl(1+\exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\bigr) \quad \text{if} \quad y_i = 1 \tag{22}$$

and

$$-\log\bigl(1+\exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\bigr) \quad \text{if} \quad y_i = 0. \tag{23}$$

The sum of these contributions over all observations is the loglikelihood function:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{N} \Big(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) - \log\big(1 + \exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\big) \Big). \tag{24}$$

To maximize $\ell(\boldsymbol{\beta})$, we differentiate with respect to β_0 and each element of $\boldsymbol{\beta}$ and set the derivatives to 0.

The first-order condition for β_0 is interesting:

$$\sum_{i=1}^{N} y_i - \sum_{i=1}^{N} \frac{\exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})}{1 + \exp(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})} = 0. \tag{25}$$

So the sum of the y_i must be equal to the sum of the probabilities that $y = 1$. Thus, at the ML estimates, the expected number of 1s must equal the actual number.

This is similar to the condition for OLS that the mean of the regressand must equal the mean of the fitted values.

The loglikelihood (24) can be maximized by a quasi-Newton method that is equivalent to iteratively reweighted least squares. See ESL, p. 121 or ETM pp. 455-456.

3.5. Regularized Logistic Regression

Of course, we can penalize the loglikelihood function for (multinomial) logit, using either an L_1 or L_2 penalty, or perhaps both in the fashion of the elastic-net penalty.

ESL only discusses the L_1 (lasso) case.

For the logit/lasso case, instead of maximizing

$$\sum_{i=1}^{N}\Big(y_i(\beta_0+\boldsymbol{x}_i^\top\boldsymbol{\beta})-\log\big(1+\exp(\beta_0+\boldsymbol{x}_i^\top\boldsymbol{\beta})\big)\Big), \tag{26}$$

we would maximize

$$\frac{1}{N}\sum_{i=1}^{N}\Big(y_i(\beta_0+\boldsymbol{x}_i^\top\boldsymbol{\beta})-\log\big(1+\exp(\beta_0+\boldsymbol{x}_i^\top\boldsymbol{\beta})\big)\Big)-\lambda\sum_{j=1}^{p}|\beta_j|. \tag{27}$$

The factor of $1/N$ is there to make λ not depend on N. Maximizing (27) apparently requires nonlinear programming, but specialized procedures based on coordinate descent are much faster.

For logit/ridge, we would replace the penalty term in (27) by

$$-\lambda \sum_{j=1}^{p} \beta_j^2 = -\lambda ||\boldsymbol{\beta}||^2 = -\lambda \boldsymbol{\beta}^\top \boldsymbol{\beta}. \tag{28}$$

I found a paper (le Cessie and van Houwelingen, JRSS C, 1992, 1366 cites) that discusses this in detail. Estimation can be done using quasi-Newton methods, and cross-validation can be used to choose λ.

Of course, we need to standardize the predictors before we use any sort of penalty.

Although statisticians and econometricians typically use 0 and 1 as the values of y_i for classification problems with two classes, the machine learning community typically uses -1 and $+1$.

With this convention, the negative of the loglikelihood (24) is replaced by

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{N} \log\left(1 + \exp(-y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\right). \tag{29}$$

Define $f(\boldsymbol{x}_i)$ as $\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}$. Then $y_i f(\boldsymbol{x}_i)$ is called the **margin**. When the margin is positive/negative, the classification is correct/ incorrect.

3.6. Logistic Regression and LDA

Recall from (6) that the log of the odds between classes k and K for LDA is

$$\begin{aligned} &\log \frac{\pi_k}{\pi_K} - \frac{1}{2}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_K)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_K) + \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_K) \\ &= \alpha_{k0} + \boldsymbol{x}^\top \boldsymbol{\alpha}_k. \end{aligned} \tag{30}$$

For logistic regression, the log of the same odds is

$$\beta_{k0} + \boldsymbol{x}^\top \boldsymbol{\beta}_k. \tag{31}$$

Except for the names of the coefficients, (30) and (31) look identical!

However, the coefficients are estimated differently. When we estimate LDA and logit models, we maximize different things.

The joint density of $\boldsymbol{x}$ and G is

$$\Pr(\boldsymbol{x}, G = k) = \Pr(\boldsymbol{x}) \Pr(G = k|\boldsymbol{x}). \tag{32}$$

LDA maximizes the loglikelihood function

$$\sum_{i=1}^{N} \sum_{k=1}^{K} \log \Pr(\boldsymbol{x}_i, G = k) = \sum_{i=1}^{N} \sum_{k=1}^{K} \big(\log \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) + \log \pi_k\big), \tag{33}$$

where $\phi(\cdot)$ is the multivariate normal density. This is based on the joint density (32) of G and $\boldsymbol{x}$.

Implicitly, the joint density that appears in (33) depends on the marginal density

$$\Pr(\boldsymbol{X}) = \sum_{k=1}^{K} \pi_k \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}), \tag{34}$$

which we obtain by summing over all the classes.

In contrast, multinomial logit maximizes the loglikelihood function

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\log \Pr(G=k|\boldsymbol{x}_i), \tag{35}$$

which is based on the probability of $G = k$ conditional on $\boldsymbol{x}$. It does not depend on the marginal density (34).

Because LDA uses more information, it should be more efficient than multilogit. ESL cites a paper by Efron that says the efficiency loss is about 30%.

The loss could be much greater if we have lots of unlabelled observations. They can be used to estimate $\boldsymbol{\Sigma}$ even though we don't know what class they belong to.

However, the additional information comes from the assumption that the $\boldsymbol{x}$ are multivariate normal, which is a very strong assumption. Since (35) conditions on the $\boldsymbol{x}_i$, it makes no assumption about how they are generated.

Moreover, the assumptions of LDA make no sense if any of the regressors is categorical. Thus multilogit is safer and more widely applicable.

4. Kernel Density Estimation

Two useful books are Li and Racine (2007) and Henderson and Parmeter (2015).

The simplest way to estimate a CDF graphically is to use the **empirical distribution function**, or **EDF**.

Suppose we have a sample x_i, $i = 1, \ldots, n$, of realizations of a random variable X. Then the EDF at any point x is

$$\hat{F}(x) \equiv \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(x_i \leq x), \tag{36}$$

where $\mathbb{I}(\cdot)$ is the **indicator function**. This is not a smooth function of x.

The traditional way to estimate a probability density function graphically is to form a **histogram**. This would be the right thing to do if the data were discrete.

The interval containing the x_i is partitioned into a set of subintervals by a set of points z_j, $j = 1, \ldots, m$, with $z_j < z_{j+1}$ for all j, where typically $m \ll n$.

Like the EDF, the histogram is a locally constant function with discontinuities. Unlike the EDF, the histogram is discontinuous at the z_j, not the x_i.

Let j be such that $z_j \leq x < z_{j+1}$ for some x. Then the histogram is just the following estimate of the density function at x:

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbb{I}(z_j \leq x_i < z_{j+1})}{z_{j+1} - z_j}. \tag{37}$$

The value of the histogram at x is the proportion of the sample points contained in the same bin as x, divided by the length of the bin.

A histogram is extremely dependent on the choice of the partitioning points z_j.

With just two z_j, the histogram would look like a uniform distribution with lower limit z_1 and upper limit z_2.

With a great many z_j, many bins would be empty. The remaining bins would contain spikes, because $z_{j+1} - z_j$ would tend to 0 as the partition became finer.

We want neither too few nor too many bins. To prove anything about asymptotic validity, we would need a rule for increasing the number of bins as $n \to \infty$.

4.1. Kernel estimation of distribution functions

The discontinuous indicator function $\mathbb{I}(x_i \leq x)$ in (36) can be interpreted as the CDF of a degenerate random variable which puts all its probability mass on x_i.

The EDF can be thought of as the unweighted average of these CDFs.

We can obtain a smooth estimator of the CDF by replacing the discontinuous function $\mathbb{I}(x \geq x_i)$ in (36) by a continuous CDF that has support in an interval containing x_i. This will give us a weighted average.

Let $K(z)$ be any continuous CDF corresponding to a distribution with mean 0. This function is called a **cumulative kernel**. It usually corresponds to a distribution with a density that is symmetric around the origin, such as the standard normal.

In order to be able to control the degree of smoothness of the estimate, we set the variance of the distribution characterized by $K(z)$ to 1 and introduce the **bandwidth parameter** h as a scaling parameter.

This gives the **kernel CDF estimator**

$$\hat{F}_h(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(\frac{x_i - x}{h}\right). \tag{38}$$

This estimator depends on the cumulative kernel $K(\cdot)$ and the bandwidth h.

As $h \to 0$, a typical term of the summation on the right-hand side of (38) tends to $\mathbb{I}(x_i \geq x) = \mathbb{I}(x \leq x_i)$, and so $\hat{F}_h(x)$ tends to the EDF $\hat{F}(x)$ as $h \to 0$.

At the other extreme, as h becomes large, a typical term of the summation tends to the constant value $K(0)$, which makes $\hat{F}_h(x)$ very much too smooth.

In the usual case in which $K(z)$ corresponds to a symmetric distribution, $\hat{F}_h(x)$ tends to 0.5 as $h \to \infty$.

It has been shown that $h = 1.587 s n^{-1/3}$ is optimal for CDF estimation, where s is the standard deviation of the x_i.

Here "optimal" means that we minimize the **asymptotic mean integrated squared error**, or **AMISE**; see below.

4.2. Kernel estimation of density functions

For density estimation, we can choose $K(z)$ to be not only continuous but also differentiable. Then we define the **kernel function** $k(z)$, often simply called the **kernel**, as $K'(z)$.

If we differentiate equation (38) with respect to x, we obtain the **kernel density estimator**

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} k\Big(\frac{x_i - x}{h}\Big). \tag{39}$$

Notice that we divide by nh rather than just n.

Like the kernel CDF estimator (38), the kernel density estimator (39) depends on the choice of kernel $k(\cdot)$ and the bandwidth h.

One very popular choice for $k(\cdot)$ is the **Gaussian kernel**, which is just the standard normal density $\phi(\cdot)$. It gives a positive (although perhaps very small) weight to every point in the sample.

Another commonly used kernel, which has certain optimality properties, is the **Epanechnikov kernel**,

$$k_1(z) = \frac{3(1 - z^2/5)}{4\sqrt{5}} \quad \text{for } |z| < \sqrt{5}, \quad 0 \text{ otherwise.} \tag{40}$$

This kernel gives a positive weight only to points for which $|(x_i - x)|/h < \sqrt{5}$.

Yet another popular kernel is the **biweight kernel**:

$$k_2(z) = \frac{15}{16}(1 - z^2)^2 \mathbb{I}(|z| \leq 1). \tag{41}$$

This is quite similar to the Epanechnikov kernel, but it squares the argument and involves different constants.

Three properties shared by all these kernels, and other **second-order kernels**, are

$$\kappa_0(k) \equiv \int_{-\infty}^{\infty} k(z)dz = 1, \tag{42}$$

$$\kappa_1(k) \equiv \int_{-\infty}^{\infty} z\,k(z)dz = 0, \tag{43}$$

and

$$\kappa_2(k) \equiv \int_{-\infty}^{\infty} z^2 k(z)dz < \infty. \tag{44}$$

The first property is shared by all PDFs.

The second property is that the kernel has first moment zero. It is satisfied by any kernel that is symmetric about zero.

The third property is that the kernel have finite variance. It is essential for estimates based on the kernel k to have finite bias.

The big difference between the Epanechnikov and Gaussian kernels is that the former is 0 for $|z| > \sqrt{5}$, while the latter is always positive.

It can be shown that, to highest order, the bias of the kernel estimator is

$$\mathrm{E}\big(\hat{f}_h(x) - f(x)\big) \cong \frac{h^2}{2} f''(x)\kappa_2(k), \tag{45}$$

where $f''(x)$ is the second derivative of the density $f(x)$. Recall from (44) that $\kappa_2(k)$ is the second moment of the kernel k.

Notice that the bias does not depend directly on the sample size. It only depends on n through h, which should become smaller as n increases.

Since bias is proportional to h^2, it may seem that we should make h very small. But that turns out to be desirable only when n is very large.

The shape of the density matters. If the slope of the density is constant, then $f''(x) = 0$, and there is no bias.

It can also be shown that, to highest order, the variance of $\hat{f}_h(x)$ is

$$\mathrm{E}\big(\hat{f}_h(x) - f(x)\big)^2 \cong \frac{1}{nh} f(x) R(k), \tag{46}$$

where

$$R(k) = \int k^2(z) dz \tag{47}$$

measures the "difficulty" of the kernel.

Note that the variance depends inversely on both the sample size and the bandwidth.

It makes sense that the variance goes up as h goes down, because fewer observations are averaged to give us the estimate for any x.

In choosing h, there is a tradeoff between bias and variance. A larger h increases bias but reduces variance.

Making h larger is like making k larger in kNN estimation.

The **asymptotic mean squared error**, or **AMSE**, is

$$\begin{aligned} \operatorname{AMSE}\bigl(\hat{f}_h(x)\bigr) &= \operatorname{Bias}^2\bigl(\hat{f}_h(x)\bigr) + \operatorname{Var}\bigl(\hat{f}_h(x)\bigr) \\ &\cong \tfrac{1}{4}\kappa_2^2(k)\bigl(f''(x)\bigr)^2 h^4 + (nh)^{-1} f(x) R(k). \end{aligned} \tag{48}$$

If we held h fixed as $n \to \infty$, the first term (bias squared) would stay constant, and the second term (variance) would go to zero.

Thus we want to make h smaller as n increases. But we need to ensure that $nh \to \infty$ as $n \to \infty$ to make the second term go away.

The AMSE depends on x, so it will be different in different parts of the distribution.

There is no law requiring h to be the same for all x, although using more than one value risks causing visible artifacts where h changes.

If our objective is simply to draw a picture that looks nice and accurately portrays the true distribution, we may well want to use more than one value of h, but we will have to smooth out the artifacts.

To get an overall result, it is common to consider the **asymptotic mean integrated squared error**, or **AMISE**:

$$\begin{aligned} \text{AMISE}\big(\hat{f}_h(x)\big) &= \int_{-\infty}^{\infty} \text{AMSE}\big(\hat{f}_h(z)\big)dz \\ &\cong \tfrac{1}{4}h^4\kappa_2^2(k)R(f'') + \frac{R(k)}{nh}, \end{aligned} \tag{49}$$

where $R(f'')$ measures the "roughness" of $f(x)$. Note that $R(f'')$ should not be confused with $R(k)$!

Larger values of $R(f'')$ imply that the density is harder to estimate.

AMISE involves the same tradeoff between bias and variance as AMSE, but it does not depend on x because we have integrated it out.

The Epanechnikov kernel is optimal, in the sense that it minimizes AMISE. The efficiency of some other kernel, say $k_g(\cdot)$, relative to $k_1(\cdot)$ is

$$\frac{R(k_g)\kappa_2(k_g)^{1/2}}{R(k_1)}. \tag{50}$$

The quantity $\kappa_2(k_1)$ does not appear here, because $\kappa_2(k_1) = 1$. The loss in efficiency relative to Epanechnikov is roughly 0.61% for biweight and 5.13% for Gaussian.

4.3. Bandwidth selection

The choice of bandwidth is far more important than the choice of kernel.

The optimal bandwidth for minimizing AMSE is

$$h_{\text{opt}} = n^{-1/5} \left(\frac{f(x) R(k)}{\kappa_2^2(k) \big(f''(x)\big)^2} \right)^{1/5}. \tag{51}$$

and the optimal bandwidth for minimizing AMISE is

$$h_{\text{opt}} = n^{-1/5} \left(\frac{R(k)}{\kappa_2^2(k) R(f'')} \right)^{1/5}. \tag{52}$$

These results make it clear that h should get smaller as n gets larger, but quite

slowly. For example,

$$\begin{aligned} n = 10 &\longrightarrow h \propto 0.63096 \\ n = 100 &\longrightarrow h \propto 0.39811 \\ n = 1000 &\longrightarrow h \propto 0.25119 \\ n = 10{,}000 &\longrightarrow h \propto 0.15849 \\ n = 100{,}000 &\longrightarrow h \propto 0.10000 \end{aligned}$$

Here n increases by a factor of 10,000, and h shrinks by a factor of just 6.3.

For any density and any kernel, one can figure out $R(k)$, $\kappa_2(k)$, and $R(f'')$. In the case of the Gaussian kernel and a normal density, these are

$$R(k) = (2\sqrt{\pi})^{-1}, \quad \kappa_2(k) = 1, \quad \text{and} \quad R(f'') = \frac{2}{8\sqrt{\pi}\sigma^5}. \tag{53}$$

Substituting these into expression (52) for h_{opt}, we find that

$$h_{\text{opt}} = \left(\frac{8\sqrt{\pi}\sigma^5}{6\sqrt{\pi}}\right)^{1/5} n^{-1/5} = \left(\frac{4}{3}\right)^{1/5} \sigma n^{-1/5} = 1.059\sigma n^{-1/5}. \tag{54}$$

This leads to **Silverman's rule-of-thumb bandwidth**:

$$h_{\text{rot}} = 1.059 s n^{-1/5}, \tag{55}$$

where s is the sample standard deviation.

When a distribution has heavy tails, s is not a very good measure of dispersion.

A more robust measure is the inter-quartile range. For a normal distribution, $\sigma = \text{IQR}/1.349$. Thus we can either replace s in the rule-of-thumb bandwidth by IQR/1.349, or (to avoid over-smoothing) replace it by $\min(s, \text{IQR}/1.349)$.

For the Epanechnikov kernel, the constant that corresponds to 1.059 is 1.049. It is a little bit smaller because of the greater efficiency of estimation based on the Epanechnikov kernel.

Of course, any rule-of-thumb bandwidth may fail badly if the distribution being estimated differs a lot from the normal. There are likely to be severe problems if the distribution is multi-modal.

We evidently need a way to estimate $R(f'')$.

H&P (2015) discuss various plug-in methods. All of these essentially require that we obtain a kernel estimate of $R(f'')$, which is then used to estimate the optimal bandwidth.

But estimating $R(f'')$ requires a bandwidth parameter, and estimating it requires another bandwidth parameter, and so on!

4.4. Some numerical examples

Figure 1 shows kernel density estimates for CRVE t statistics based on 5000 replications using a Gaussian kernel.

We would expect these t statistics to be more or less symmetrically distributed, but with variance greater than 1 and quite possibly kurtosis greater than $3\sigma^4$.

There are three estimated densities, using

$$\begin{aligned} h_{\text{rot}} &= 1.059 n^{-1/5} \\ h_{\text{big}} &= 1.5 \times 1.059 n^{-1/5} \\ h_{\text{small}} &= 0.5 \times 1.059 n^{-1/5} \end{aligned}$$

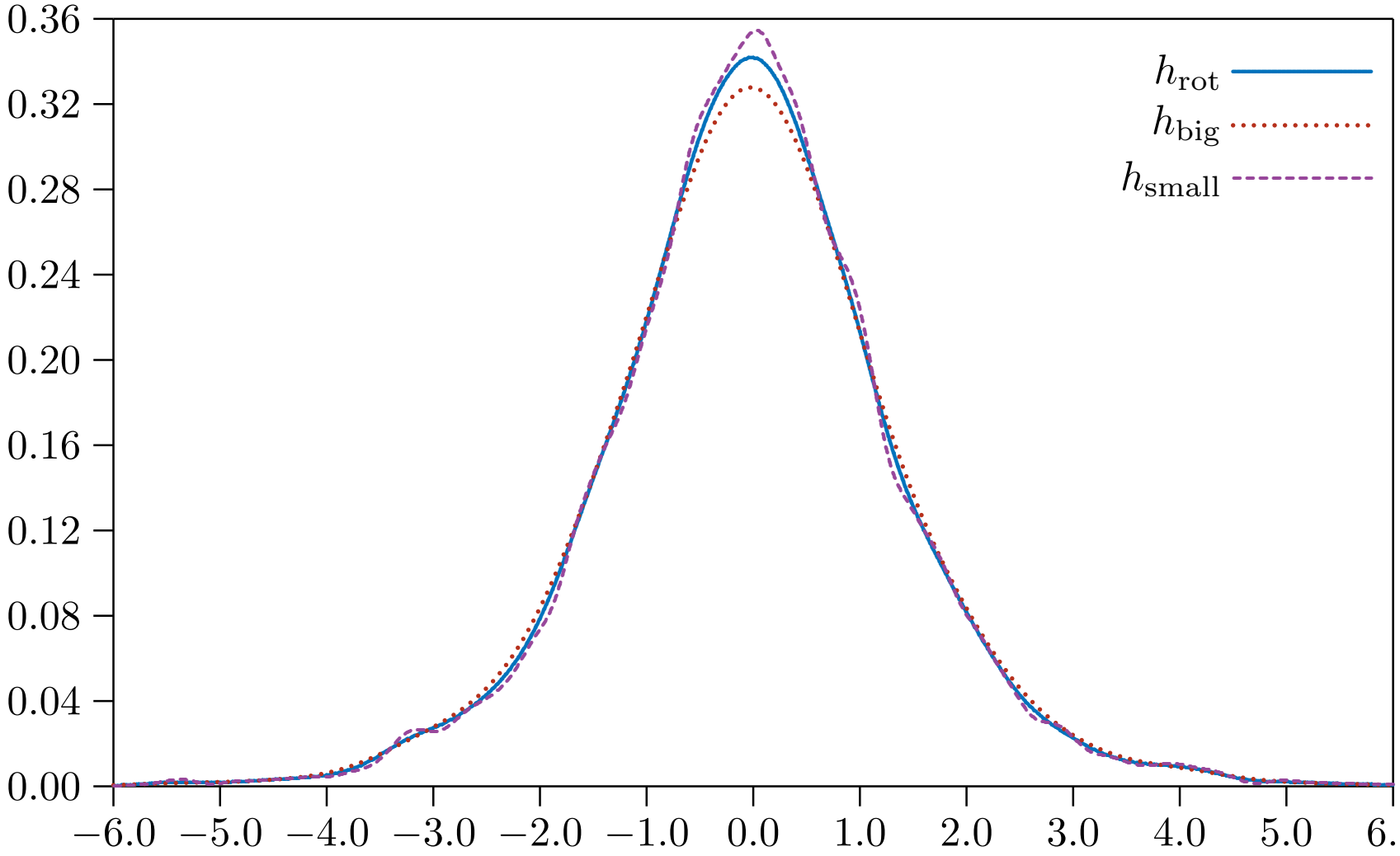

Figure 1. Kernel density estimates for CRVE t statistics

The figure is plotted for 601 values of t evenly spaced between -6.0 and 6.0.

It is clear from the figure that h_{small} is too small, because there are lots of wiggles.

It is not so obvious that h_{big} is too big.

All three estimates give us a pretty good idea of what the density looks like. h_{big} gives the nicest picture, but perhaps the peak is too low.

Figure 2 is similar, but it graphs the density of 4999 bootstrap (t^*) statistics. It seems even more obvious that h_{small} is too small.

In this case, we can generate as many observations as we like. With n sufficiently large, we should get essentially the same figure for any sensible value of h.

With real data, on the other hand, n may be too small to yield reliable estimates.

We can often obtain a density that looks nice by making h too large, but it may deviate a lot from the truth, especially in the tails and near the peak.

If multi-modal densities are possible and interesting, we do not want to make the bandwidth so large that the second mode disappears.

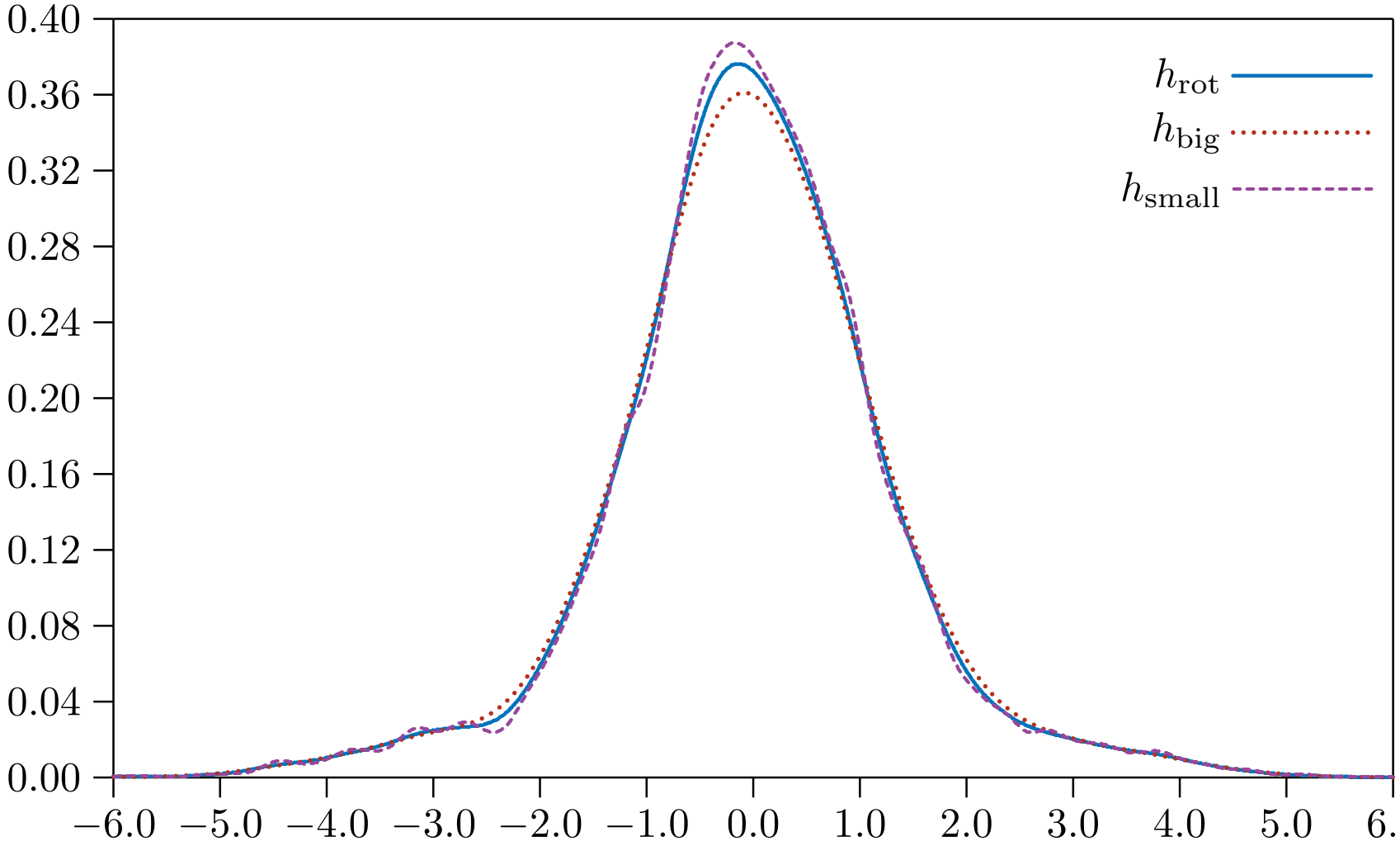

Figure 2. Kernel density estimates for bootstrap CRVE t^* statistics

4.5. Cross-Validation

A widely-used approach for choosing h is **cross-validation**.

The goal is to minimize the difference between the kernel estimate and an estimate of the density.

Define **integrated squared error**, or **ISE**, as

$$\begin{aligned} \text{ISE}(\hat{f}, f) &= \int \left(\hat{f}(x) - f(x)\right)^2 dx \\ &= \int \hat{f}^2(x)dx - 2\int \hat{f}(x)f(x)dx + \int f^2(x)dx. \end{aligned} \tag{56}$$

Since the last term here does not depend on $\hat{f}$, we can ignore it and just minimize

$$\text{ISE}^\dagger(\hat{f}, f) = \int \hat{f}^2(x)dx - 2\int \hat{f}(x)f(x)dx. \tag{57}$$

But how do we estimate $f(x)$?

Consider the **leave-one-out estimator**

$$\hat{f}_{-i}(x) = \frac{1}{h(n-1)} \sum_{j \neq i}^{n} k\Big(\frac{x_j - x}{h}\Big). \tag{58}$$

This is just the kernel estimator of $f(x)$ using every observation except the i^{th}. It is normally computed at the point $x = x_i$, so as to get an estimate of $f(x_i)$ that does not depend on x_i.

If we average the leave-one-out estimators over all i, we get

$$\bar{f}_{-i}(x) = \frac{1}{n} \sum_{j=1}^{n} \hat{f}_{-j}(x). \tag{59}$$

This is a way to estimate $f(x)$.

The **least squares cross-validation**, or **LSCV**, function can be written as

$$\text{LSCV}(h) = \frac{1}{hn^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \bar{k}\Big(\frac{x_i - x_j}{h}\Big) - \frac{2}{h(n^2-n)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k\Big(\frac{x_i - x_j}{h}\Big), \tag{60}$$

where $\bar{k}(\cdot)$ is a **convolution kernel**. If the original kernel is Gaussian, then

$$k(z) = \frac{1}{\sqrt{2\pi}} \exp(-z^2/2) \quad \text{and} \quad \bar{k}(z) = \frac{1}{\sqrt{4\pi}} \exp(-z^2/4). \tag{61}$$

Note that we can calculate both kernels with just one exponentiation, and we can calculate LSCV(h) with one double summation.

The value of $k\big((x_i - x_i)/h\big)$ is just a constant. For the Gaussian kernel, it is $1/\sqrt{2\pi}$. So we don't actually need to check for $i = j$ when doing the double summation.

We can find the optimal h by grid search, plotting LSCV(h) against h. It will often be smaller than the rule-of-thumb bandwidth.

We might, for example, search over

$$\tfrac{1}{2} h_{\text{rot}} \leq h \leq 1.5 h_{\text{rot}}, \tag{62}$$

or some other (probably shorter) range based on experience.

H&P report that, for small samples, LSCV bandwidths often seem to be too small, leading to estimated densities that have several modes.

Of course, if we are just using kernel estimation to draw a picture, we can try several values of h around h_{rot} and see what the estimated densities look like.

Leave-one-out cross-validation is not the only type. Another possibility is m-fold cross-validation, where $m = 5$ and $m = 10$ are popular choices.

In some cases, such as this one, leave-one-out cross-validation is cheaper than m-fold cross-validation. In others, it is more expensive.

4.6. Kernel regression

The simplest approach to nonparametric regression is **kernel regression**.

Suppose that two random variables Y and X are jointly distributed, and we wish to estimate the conditional expectation $\mu(x) \equiv \mathrm{E}(Y \mid x)$ as a function of x, using a sample of paired observations (y_i, x_i) for $i = 1, \ldots, n$.

For given x, consider the function $G(x)$ defined as

$$G(x) = \mathrm{E}\big(Y \mathbb{I}(X \leq x)\big) = \int_{-\infty}^{x} \int_{-\infty}^{\infty} y\, f(y, z)\, dy\, dz, \tag{63}$$

where $f(y, x)$ is the joint density of Y and X. Let $g(x) \equiv G'(x)$ denote the first derivative of $G(X)$.

A natural unbiased estimator of $G(x)$ is $\frac{1}{n}\sum_{t=1}^{n} y_i \mathbb{I}(x_i \leq x)$.

But this estimator, like the EDF, is discontinuous. We need to replace the indicator function by something smoother if we are to estimate the derivative of $G(x)$.

The simplest approach is to replace $\mathbb{I}(x_i \leq x)$ by a cumulative kernel. Thus we obtain the biased but smooth estimator

$$\hat{G}_h(x) = \frac{1}{n}\sum_{i=1}^{n} y_i K\Big(\frac{x - x_i}{h}\Big), \tag{64}$$

where K is a cumulative kernel (that is, the CDF of a distribution with mean 0 and variance 1), and h is a bandwidth parameter.

In order to obtain a kernel regression, we need to find the derivative of (64), say $\hat{g}_h(x)$, and estimate the marginal density of X.

This yields the **Nadaraya-Watson**, or **locally constant**, estimator

$$\hat{\mu}_h(x) = \frac{\sum_{i=1}^n y_i k_i}{\sum_{i=1}^n k_i}, \quad k_i(x) \equiv k\Big(\frac{x - x_i}{h}\Big), \tag{65}$$

where $k \equiv K'$ is a kernel function.

The numerator of (65) is a weighted average of the values of y_i in the neighborhood of x, and the denominator is a kernel estimate of the density of X at the point x.

The Nadaraya-Watson estimator is the solution to the estimating equation

$$\sum_{i=1}^n k_i(x)\big(y_i - \hat{\mu}_h(x)\big) = 0. \tag{66}$$

This is the empirical counterpart of a weighted average of the elementary zero functions $y_i - \mu(x)$.

But these elementary zero functions do not have mean 0, because the conditional expectation of y_i is not $\mu(x)$ but $\mu(x_i)$.

This causes bias. The correct, but infeasible, zero function would be $y_i - \mu(x_i)$.

A better approximation to the correct zero function is given by the two-term Taylor expansion $\mu(x) + \mu'(x)(x_i - x)$, in which both $\mu(x)$ and $\mu'(x)$ are unknown.

Both of these unknowns can be estimated simultaneously by solving the estimating equations

$$\sum_{i=1}^{n} k_i(x)\big(y_i - \mu(x) - \mu'(x)(x_i - x)\big) = 0 \tag{67}$$

and

$$\sum_{i=1}^{n} k_i(x)(x_i - x)\big(y_i - \mu(x) - \mu'(x)(x_i - x)\big) = 0. \tag{68}$$

The simplest way to solve these equations is to run the linear regression

$$k_i^{1/2}(x)\, y_i = \mu(x) k_i^{1/2}(x) + \mu'(x)(x_i - x) k_i^{1/2}(x) + \text{residual}, \tag{69}$$

so as to obtain the **locally linear estimator** of $\mu(x)$, which is just the first estimated coefficient, say $\hat{\mu}_h^{\mathrm{LL}}$. Regression (69) is called a **local(ly) linear regression**.

We must run regression (69) for every value of x at which we wish to evaluate $\mu(x)$. How many observations it involves depends on the kernel and the bandwidth.

For a Gaussian kernel, regression (69) always has n observations. For an Epanechnikov kernel, it typically has a smaller (perhaps much smaller) number.

Observations near x get large weights, and observations far away get small weights. With kernels such as Epanechnikov, the latter actually get zero weights.

We could add additional terms, such as

$$\mu''(x)k_i^{1/2}(x)(x_i - x)^2, \tag{70}$$

to regression (69). This would give us a **locally quadratic estimator**. More generally, we would have a **locally polynomial estimator**.

It can be shown that, to second order, the bias of the locally constant estimator is

$$\begin{aligned} &\frac{\kappa_2(k)}{2f(x)}h^2\big(2\mu'(x)f'(x) + \mu''(x)f(x)\big) \\ &\quad = \frac{1}{2}\kappa_2(k)h^2\mu''(x) + \kappa_2(k)h^2\mu'(x)\frac{f'(x)}{f(x)}, \end{aligned} \tag{71}$$

where $\mu'(x)$ and $\mu''(x)$ denote the first and second derivatives of the conditional mean function evaluated at x.

The first term depends on the second derivative of $\mu(x)$, and the second term depends on the first derivative. So there will be no bias if $\mu(x)$ is a horizontal line.

Recall that $f(x)$ is the density of the x_i at x, and $f'(x)$ is its first derivative. Also, recall from (44) that $\kappa_2(k)$ is the second moment of the kernel k.

Thus bias will be larger for kernels with larger variance and when the density of x is changing more rapidly.

Similarly, to second order, the bias of the locally linear estimator is

$$\text{Bias}(\hat{\mu}_h^{\text{LL}}) = \frac{1}{2}\kappa_2(k)h^2\mu''(x). \tag{72}$$

The bias of the LL estimator, expression (72), is equal to the first term of the bias of the LC estimator in the second line of (71).

The second term in (71) depends on $\mu'(x)$, but the common term depends only on $\mu''(x)$. So, as we might expect, the LL estimator is unbiased if $\mu(x)$ is linear.

To highest order, the variance of both estimators is the same:

$$\mathrm{Var}(\hat{\mu}_h^{\mathrm{LC}}) = \mathrm{Var}(\hat{\mu}_h^{\mathrm{LL}}) \cong \frac{\sigma^2 R(k)}{nhf(x)}, \tag{73}$$

where σ^2 is the variance of the regression disturbances. Unlike the bias, this does not depend on the regression function we are estimating.

It is possible to find an optimal bandwidth by minimizing AMISE, but the result depends on:

- the sample size, with a factor of $n^{-1/5}$;
- the density of x, the regressor;
- the variance of the disturbances;
- the shape of the regression function; and
- the kernel.

In practice, people generally do not attempt to estimate the optimal bandwidth.

4.7. Least squares cross-validation

For the locally constant estimator, we minimize

$$\text{LSCV}(h) = \sum_{i=1}^{n} \big(y_i - \hat{\mu}_{-i}(x_i)\big)^2, \tag{74}$$

where

$$\hat{\mu}_{-i}(x_i) = \frac{\sum_{j\neq i}^{n} y_j k_h(x_j, x_i)}{\sum_{j\neq i}^{n} k_h(x_j, x_i)}. \tag{75}$$

For each i, this is just the leave-one-out LC estimator of $\text{E}(y_i)$. It is the kernel-weighted average of all the y_j for $j \neq i$.

Weights are large when x_j is close to x_i, and either very small (Gaussian kernel) or zero (Epanechnikov and many other kernels) when x_j is far from x_i.

The leave-one-out estimator is precisely what we would use to forecast y_i if we observed x_i but not y_i, under the assumption that the regression function is locally constant.

In the locally linear case, (75) would be replaced by the constant term for the leave-one-out kernel-weighted regression of y_i on a constant and x_i.

Although (74) and (75) involve a double summation, the cross-validation function can be rewritten as

$$\sum_{i=1}^{n} \big(y_i - \hat{\mu}(x_i)\big)^2 \pi_h(x_i), \tag{76}$$

where, in the LC case,

$$\pi_h(x_i) = \left(\frac{\sum_{j=1}^{n} k_h(x_j, x_i)}{\sum_{j \neq i}^{n} k_h(x_j, x_i)} \right)^2. \tag{77}$$

Presumably we can do something similar for the LL estimator.

Once we have calculated $\pi_h(x_i)$, we can calculate LSCV(h) via a single summation. This might be very useful if we were bootstrapping and the x_i were fixed across bootstrap samples.

4.8. A numerical example

I generated 400 observations from an artificial DGP that is linear for x_i below a certain value and quite nonlinear beyond that point.

I used an Epanechnikov kernel, with either a default bandwidth of $h = sn^{-1/5}$ or a value of h chosen by cross-validation.

The LC estimates totally miss the rightmost data points, and also perform poorly for the leftmost ones.

The reason is obvious: For the rightmost points, LC is taking a weighted average of points that are (almost) all to the left of them.

To avoid this, LSCV makes h quite small, which causes the fitted values to wiggle.

LL estimates are much more plausible than LC ones. h chosen by cross-validation is smaller than baseline value, but not much difference between two sets of estimates.

Values of the cross-validation function:

LC: baseline h (0.9803): 2.8525 optimal h (0.2498): 2.4960
LL: baseline h (0.9803): 2.4947 optimal h (0.6920): 2.4693

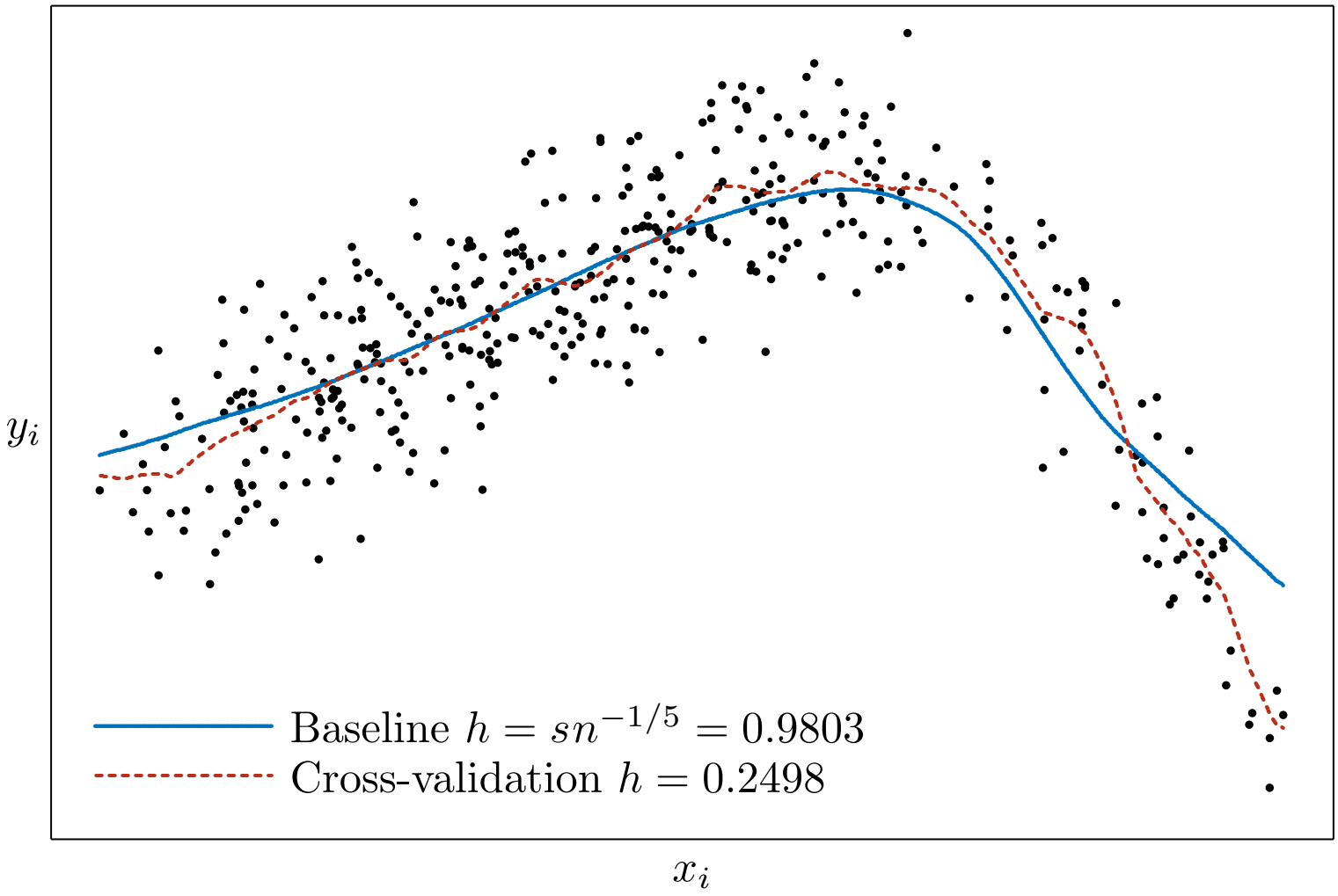

Figure 3. Locally constant kernel regression using simulated data

It is not a coincidence that the optimal bandwidth is much larger for LL regression than for LC regression.

The choice of h involves a tradeoff between bias and variance. We saw in (73) that the variance of the two estimators is similar and declines with h.

We also saw that the bias of LC, in (71), is larger than the bias of LL, in (72). These both increase with h.

Therefore, the tradeoff favours making h larger for LL than for LC.

We can afford to make the bias term larger by making h larger when the bias term is smaller to begin with.

4.9. More on Kernels

We previously defined the Epanechnikov kernel as

$$k_1(z) = \frac{3(1 - z^2/5)}{4\sqrt{5}} \quad \text{for } |z| < \sqrt{5}, \;\; 0 \;\; \text{otherwise.} \tag{78}$$

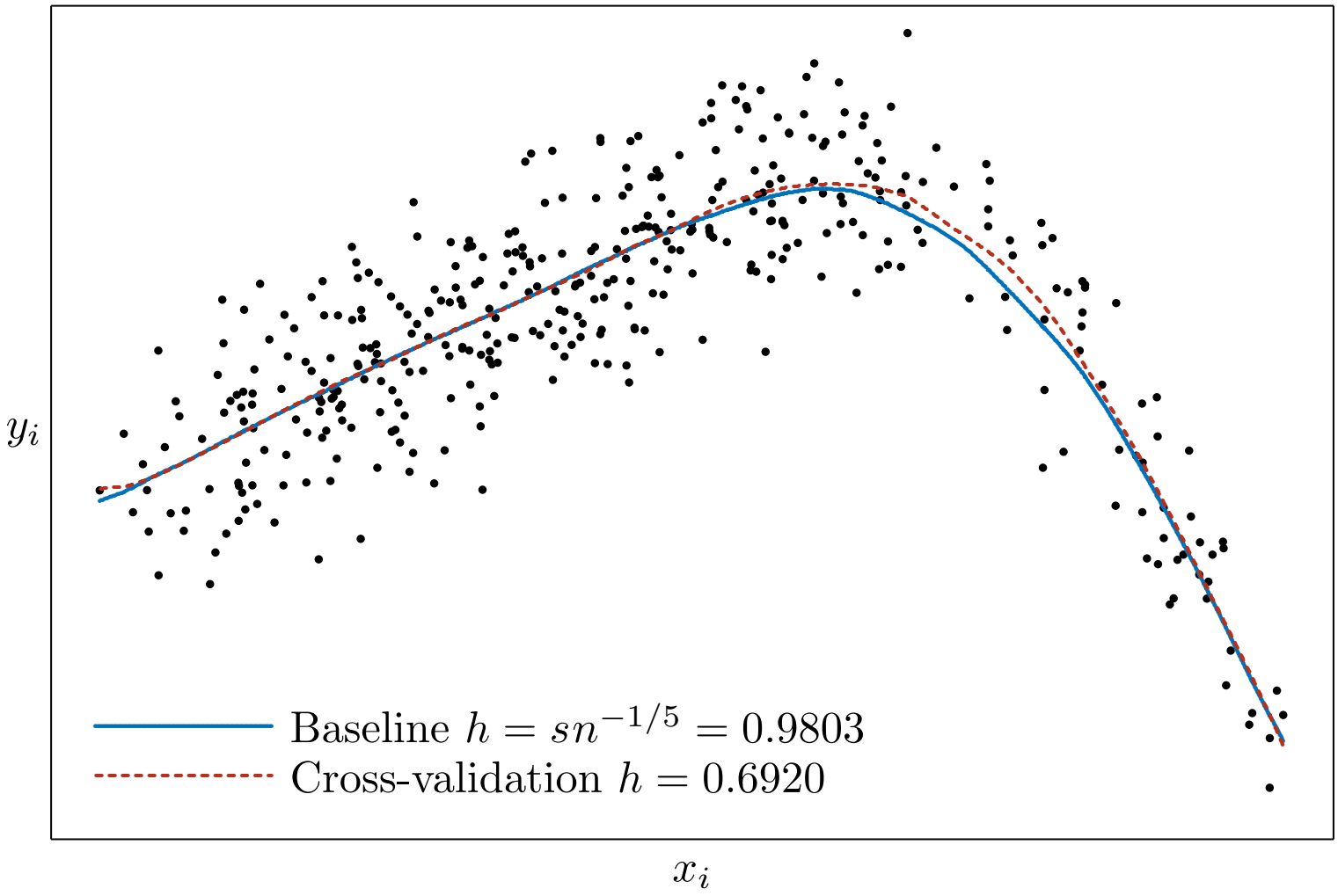

Figure 4. Locally linear kernel regression using simulated data

ESL define it as

$$D(t) = \frac{3}{4}(1 - t^2) \quad \text{for } |t| < 1, \;\; 0 \;\; \text{otherwise.} \tag{79}$$

Thus $t = z/\sqrt{5}$ and $k_1(z) = D(t)/\sqrt{5}$.

We can obviously choose bandwidths so that kernel regression based on (78) and (79) are identical.

ESL define

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right), \tag{80}$$

whereas we used

$$k_1\left(\frac{x - x_0}{h}\right). \tag{81}$$

It seems odd that ESL use absolute values in (80) when the argument of $D(t)$ is squared.

A kernel that looks a lot like the Epanechnikov kernel but is differentiable is the **tri-cube kernel**

$$D(x) = \left(1 + |t^3|\right)^3 \quad \text{for } |t| < 1, \;\; 0 \;\; \text{otherwise.} \tag{82}$$

4.10. Local Regression in Higher Dimensions

It is easy to generalize Nadaraya-Watson kernel regression and locally linear or quadratic regression to more than one dimension, although it may not be a good idea for $p > 2$.

For example, we might have

$$\boldsymbol{b}(x) = [1 \;\; x_1 \;\; x_2]^\top, \tag{83}$$

for locally linear regression in two dimensions, or

$$\boldsymbol{b}(x) = [1 \;\; x_1 \;\; x_2 \;\; x_1^2 \;\; x_2^2 \;\; x_1 x_2]^\top, \tag{84}$$

for locally quadratic regression in two dimensions. We minimize

$$\sum_{i=1}^{N} K_\lambda(\boldsymbol{x}_0, \boldsymbol{x}_i)\big(y_i - \boldsymbol{b}^\top(\boldsymbol{x}_i)\boldsymbol{\beta}(\boldsymbol{x}_0)\big)^2 \tag{85}$$

and obtain the fitted values $\hat{f}(\boldsymbol{x}_0) = \boldsymbol{b}^\top(\boldsymbol{x}_0)\hat{\boldsymbol{\beta}}(\boldsymbol{x}_0)$.

The kernel is usually a radial Epanechnikov or tri-cube:

$$K_\lambda(\boldsymbol{x}_0, \boldsymbol{x}) = D\left(\frac{||\boldsymbol{x} - \boldsymbol{x}_0||}{\lambda}\right), \tag{86}$$

where the predictors should usually be standardized.

It is impossible to maintain both low bias and low variance, unless the sample has a great many points near every interesting value of $\boldsymbol{x}_0$. This is extremely difficult to achieve unless N is very large.

For bias to be small, we need all the points that get much weight to be near $\boldsymbol{x}_0$, which implies that λ must be small.

For the variance to be small, we need there to be a lot of points that are near $\boldsymbol{x}_0$, which implies that λ must be large.

The only way that λ can be small enough for low bias and large enough for low variance is if N increases exponentially in p.

4.11. Structured Local Regression Models

Unless p is very small, we need to impose structure on the model.

One approach is to use a **structured kernel** such as

$$K_{\lambda \boldsymbol{A}}(\boldsymbol{x}_0, \boldsymbol{x}) = D\left(\frac{(\boldsymbol{x} - \boldsymbol{x}_0)^\top \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}_0)}{\lambda}\right), \tag{87}$$

where $\boldsymbol{A}$ is a positive definite matrix that gives more or less weight to certain directions.

ESL prefer to use **structured regression functions** such as

$$f(\boldsymbol{x}) = \alpha + \sum_{j=1}^{p} g_j(x_j) + \sum_{k<\ell} g_{k\ell}(x_k, x_\ell) + \ldots \tag{88}$$

These are generalizations of the partially linear regression model. Typically, there cannot be too many higher-order terms. For **additive models**, there are just the p functions $g_j(x_j)$.

Instead of a nonlinear function for one variable and a linear model for all the others, (88) has many one-dimensional and two-dimensional nonlinear models to estimate.

This can be done iteratively. Consider the additive case. If we centre the data and all the $g_j(x_j)$ except $g_k(x_k)$ are assumed known, we can estimate an additive model by repeatedly running the local regression

$$y - \sum_{j \neq k} g_j(x_j) = g_k(x_k) + \text{resid.} \tag{89}$$

We cycle through j from 1 to p until convergence. We may have to estimate a lot of local regressions, but each one is just one-dimensional.

Another type of model is the **varying coefficient model**. Let z denote x_p and define $q \equiv p - 1$. Then consider the model

$$f(\boldsymbol{x}) = \beta_0(z) + \beta_1(z)x_1 + \ldots + \beta_q(z)x_q, \tag{90}$$

where there are now p nonlinear functions to estimate. Conditional on them, we simply have a linear regression model.

It can be fitted by locally weighted least squares. We minimize

$$\sum_{i=1}^{N} K_\lambda(z_0, z_i)\big(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}(z_0)\big)^2 \tag{91}$$

with respect to the vector $\boldsymbol{\beta}(z_0)$ for each value of z_0.

4.12. Local Likelihood

Any parametric model can be converted to a local one by using weights that vary across observations according to the value of $\boldsymbol{x}$.

In particular, it is easy to turn globally linear models into locally linear ones.

Suppose the model has a loglikelihood function

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{N} \ell(y_i, \boldsymbol{x}_i^\top \boldsymbol{\beta}). \tag{92}$$

An obvious example is a logit or probit model. Then we can estimate a locally linear version by maximizing

$$\ell\big(\boldsymbol{\beta}(\boldsymbol{x}_0)\big) = \sum_{i=1}^{N} K_\lambda(\boldsymbol{x}_0, \boldsymbol{x}_i)\ell\big(y_i, \boldsymbol{x}_i^\top \boldsymbol{\beta}(\boldsymbol{x}_0)\big). \tag{93}$$

Here we weight contributions to the loglikelihood instead of squared residuals. We could also estimate a model with varying coefficients by maximizing

$$\ell\big(\boldsymbol{\theta}(z_0)\big) = \sum_{i=1}^{N} K_\lambda(z_0, z_i)\ell\big(y_i, \boldsymbol{x}_i^\top \boldsymbol{\theta}(z_0)\big) \tag{94}$$

with respect to the vector $\boldsymbol{\theta}(z_0)$ for each value of z_0; compare (91).

Consider the multiple logit model with J responses, where

$$\Pr(G = j \mid \boldsymbol{x}) = \frac{\exp(\beta_{j0} + \boldsymbol{x}^\top \boldsymbol{\beta}_j)}{1 + \sum_{k=1}^{J-1} \exp(\beta_{k0} + \boldsymbol{x}^\top \boldsymbol{\beta}_k)}, \tag{95}$$

where $\beta_{J0} = 0$ and $\boldsymbol{\beta}_J = \mathbf{0}$.

The local loglikelihood for this model is

$$\sum_{i=1}^{N} K_\lambda(\boldsymbol{x}_0, \boldsymbol{x}_i)\Bigg(\beta_{g_i 0}(\boldsymbol{x}_0) + (\boldsymbol{x}_i - \boldsymbol{x}_0)^\top \boldsymbol{\beta}_{g_i}(\boldsymbol{x}_0) - \log\Bigg(1 + \sum_{k=1}^{J-1} \exp\left(\beta_{k0}(\boldsymbol{x}_0) + (\boldsymbol{x}_i - \boldsymbol{x}_0)^\top \boldsymbol{\beta}_{g_i}(\boldsymbol{x}_0)\right)\Bigg)\Bigg). \tag{96}$$

Because the regressions are centred at $\boldsymbol{x}_0$, the posterior probabilities at $\boldsymbol{x}_0$ are simply

$$\widehat{\Pr}(G = j \,|\, \boldsymbol{x}_0) = \frac{\exp\big(\hat{\beta}_{j0}(\boldsymbol{x}_0)\big)}{1 + \sum_{k=1}^{J-1} \exp\big(\hat{\beta}_{k0}(\boldsymbol{x}_0)\big)}. \tag{97}$$

They do not depend on the vectors $\hat{\boldsymbol{\beta}}_j(\boldsymbol{x}_0)$.

Since $\widehat{\Pr}(G = j \,|\, \boldsymbol{x}_0)$ just depends on $\boldsymbol{x}_0$ and the coefficients $\hat{\beta}_{j0}$, $j = 1, J-1$, we can calculate its standard error using the delta method.

This model can be used for classification in low dimensions.

4.13. Kernel Estimation and Classification

This subsection is based on Sections 6.6 and 6.8 of ESL.

In earlier parts of this section, I used the notation

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} k\left(\frac{x_i - x}{h}\right) \tag{98}$$

for a kernel density estimator using the kernel $k(\cdot)$ with bandwidth h. For the Gaussian kernel, ESL prefers

$$\hat{f}_X(x) = \frac{1}{N} \sum_{i=1}^{N} \phi_\lambda(x_i - x), \tag{99}$$

where $\phi_\lambda(\cdot)$ is the normal density with mean 0 and variance λ^2.

When $k(\cdot)$ is the standard normal, $h = \lambda$, $n = N$, and (98) reduces to (99). The factor of $1/\lambda$ does not explicitly appear in (99) because it is part of ϕ_λ.

In the multivariate case, (99) generalizes to

$$\hat{f}_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{(2\lambda^2\pi)^{p/2}} \sum_{i=1}^{N} \exp\Big(-\frac{1}{2}\big(||\boldsymbol{x}_i - \boldsymbol{x}||^2/\lambda\Big). \tag{100}$$

Of course, the **curse of dimensionality** implies that we are likely to get very poor estimates when p is large and N is not extremely large.

We can use estimates like (100) for classification.

Suppose there are K classes. Then

$$\widehat{\Pr}(G = k \,|\, \boldsymbol{x}) = \frac{\hat{\pi}_k \hat{f}_k(\boldsymbol{x})}{\sum_{\ell=1}^{K} \hat{\pi}_\ell \hat{f}_\ell(\boldsymbol{x})}. \tag{101}$$

This is just the empirical counterpart of (3). In most cases, we use the sample proportions to estimate the $\hat{\pi}_k$, as in (8).

ESL note that, when classification is the goal, it is only points near the boundaries that we are interested in. We want to estimate the posterior probabilities accurately in those regions.

The **naive Bayes** estimator makes the extremely strong assumption that the joint density of the components of $\boldsymbol{x}$ is the product of the marginal densities:

$$f_k(\boldsymbol{x}) = \prod_{j=1}^{p} f_{kj}(x_j), \quad k = 1, \ldots, K. \tag{102}$$

For any $\boldsymbol{x}$ that interests us, we simply obtain p kernel density estimates, $\hat{f}_{kj}$, and use their product to estimate $\hat{f}_k(\boldsymbol{x})$.

If unrestricted kernel density estimates based on (100) are too noisy and the naive Bayes estimator is too biased, another possibility is to estimate **normal mixture models**.

In the multivariate case, a normal mixture has the form

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \phi(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{103}$$

where the **mixing proportions** α_m sum to unity.

We can estimate a model like (103) for each class, and these lead to flexible models for $\Pr(G \mid \boldsymbol{x})$.

As previously discussed, we can also use kNN methods for classification. Simply classify $\boldsymbol{x}$ as belonging to whatever class is most common among the k nearest neighbours to $\boldsymbol{x}$.

4.14. The Curse of Dimensionality

The performance of kernel estimation (and smoothing methods in general, including kNN) rapidly deteriorates as p increases.

The fundamental problem is that the volume of a mathematical space increases exponentially with the number of dimensions.

The unit interval is a 1-dimensional hypercube. If we take 100 evenly-spaced points on a grid, the distance between them will be only $10^{-2} = 0.01$.

Any point we pick at random will be "close" to several of the points on the grid.

Suppose there are two dimensions instead of one. To get the same distance between points on the grid (in the direction of each axis), we need $100^2 = 10,000$ points.

Moreover, the average Euclidean distance between a point and its nearest neighbours will be greater than 0.01, so that a randomly chosen point in the hypercube (a square in this case) will be further away from the nearest point on the grid.

For three dimensions, we need $100^3 = 1,000,000$ points to achieve a distance of 0.01 in the direction of each axis, and the average Euclidean distance between a point and its nearest neighbours will be even greater.

Thus, unless the sample size increases very rapidly as p increases, the data become much "sparser" as p increases. Any estimator based on local averages will have fewer and fewer nearby points to average over as p increases.

When predictors take on discrete values, we get the same sort of problem. Suppose that each predictor is binary. Then the number of possible sets of values is 2^p.

For large p, this will be much larger than N, so many possible sets of predictor values will never appear in any given sample.

A machine-learning algorithm that does not impose strong assumptions about functional form cannot make reliable predictions about events that are not "close" to ones observed in the sample.

This is one reason why high-dimensional methods, such as lasso, ridge, and elastic net, do make strong assumptions about functional form.

We can handle problems with large p, or we can allow the relationship between inputs and outputs to be very general, but we cannot do both.