Scaling

Some methods, including KNN, require that all the explanatory variables be scaled similarly.

Using Euclidean distance to measure how close each $\boldsymbol{x}_i$ is to $\boldsymbol{x}_0$ makes sense only if the variables have all been scaled.

Except for binary inputs, it is customary to scale all inputs to have mean 0 and variance 1.

This is easy enough if there is just one sample. But how do we handle training samples, validation samples, and test samples?

ISLR/ISLP takes the entire sample, performs the rescaling, and then divides the sample into training, validation (maybe), and test samples.

For each explanatory variable, say x_j, we estimate the sample mean $\bar{x}_j$ and sample standard deviation $\bar{s}_j$ for the entire sample.

We then rescale all observations as

$$x_{ji}^s = \frac{x_{ji} - \bar{x}_j}{\bar{s}_j}. \tag{1}$$

For the training and test data combined, the rescaled x_{ji}^s will have mean precisely 0 and variance precisely 1.

But this is not so for either the training or test data by themselves. The rescaled means and variances might differ noticeably from 0 and 1 if either dataset is small.

This might also happen if the dataset was not divided at random, which might provide a valuable warning.

With this sort of scaling, the test sample, and the validation sample (if any), are not independent of the training sample.

If the test data become available after the training and/or validation data, we cannot very well rescale all the data at once.

Another possibility would be to rescale the training and test data separately. Thus each set of data would have mean 0 and variance 1.

- But this would have perverse consequences if the two datasets were very different.
- The same values of x_i would map to different values of x_i^s, perhaps very different values.

When making predictions about the outcomes for x_0, we need to rescale x_0 in exactly the same way as we rescaled the training data.

Thus, if we estimate a model and subsequently use it to predict the outcome for x_0, we should first rescale x_0 using the mean and variance obtained originally.

The `scale()` function in R is an easy way to rescale one or more variables, but we need to have all the data.

Also, we need to keep binary variables separate from the quantitative ones that are being rescaled.

Bootstrap Methods

For OLS (and NLS, and ML, and GMM), it is fairly easy (at least in theory) to obtain standard errors. We need to estimate the covariance matrix of the vector of parameter estimates, say $\hat{\boldsymbol{\theta}}$.

Asymptotic theory often provides an expression for $\text{Var}(\hat{\boldsymbol{\theta}})$, which depends on $\boldsymbol{\theta}$ itself and on other things computed from the data.

We evaluate $\text{Var}(\hat{\boldsymbol{\theta}})$ at the appropriate values to obtain $\widehat{\text{Var}}(\hat{\boldsymbol{\theta}})$. The standard errors are the square roots of the elements on the principal diagonal of $\widehat{\text{Var}}(\hat{\boldsymbol{\theta}})$.

For OLS, $\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$, and $\widehat{\text{Var}}(\hat{\boldsymbol{\beta}}) = s^2(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$.

We can then use $\widehat{\text{Var}}(\hat{\boldsymbol{\theta}})$ to obtain standard errors for predictions, and hence **prediction intervals**, often by using the **delta method**.

But it is not so easy to obtain prediction intervals for estimates based on non-parametric procedures.

Bootstrap methods can be used whenever analytic methods do not exist, are too difficult to use, or do not work well.

Even for parametric models, $\widehat{\mathrm{Var}}(\hat{\boldsymbol{\theta}})$ may not work well (especially with the delta method), and prediction intervals may perform badly.

In such cases, bootstrap methods can yield more reliable confidence intervals, prediction intervals, and hypothesis tests.

The residual bootstrap, wild bootstrap, and wild cluster bootstrap often work very well, but they only apply to regression models.

The **pairs bootstrap**, or **resampling bootstrap**, does not make this assumption, but it does assume that observations are independent.

- Form all the $(\boldsymbol{X}_i, y_i)$ into a matrix $\boldsymbol{Z}$ with typical row $\boldsymbol{Z}_i$. In R, it would be natural to store them in a data frame.
- Then draw each of B bootstrap datasets $\boldsymbol{Z}^{*b}$, for $b = 1, \ldots, B$, by resampling from the $\boldsymbol{Z}_i$.
- This means filling each row of a bootstrap sample $\boldsymbol{Z}^{*b}$ by picking each of the $\boldsymbol{Z}_i$ with probability $1/n$.

When we chose validation samples, we sampled the z_i **without replacement**. Once a row of $\boldsymbol{Z}$ went into a particular validation sample, it could never go in again.

But for the pairs bootstrap, we have to resample **with replacement**. A given $\boldsymbol{Z}_i$ may appear any number of times between 0 and n in a given bootstrap sample $\boldsymbol{Z}^{*b}$.

Of course, the probability that it appears n times is tiny!

But the probability that it appears 0 times is not small:

$$\Pr(\boldsymbol{Z}_i \notin \boldsymbol{Z}^{*b}) = \left(1 - \frac{1}{n}\right)^n. \tag{2}$$

As $n \to \infty$, expression (2) tends to $e^{-1} \approx 0.36788$. This is so because $\log(1 - \frac{1}{n}) \to -1/n$ as $n \to \infty$.

Thus the probability that bootstrap sample b contains the data for observation i is roughly $0.632 = 1 - 0.368$.

When the data are not independent but do fall into disjoint clusters, we can easily modify the pairs bootstrap.

Suppose that $[\boldsymbol{y}_g \;\; \boldsymbol{X}_g]$, for $g = 1, \ldots, G$, contains the data for cluster g.

Then each bootstrap sample looks like

$$\begin{bmatrix} \boldsymbol{y}_1^* & \boldsymbol{X}_1^* \\ \boldsymbol{y}_2^* & \boldsymbol{X}_2^* \\ \vdots & \\ \boldsymbol{y}_G^* & \boldsymbol{X}_G^* \end{bmatrix}, \tag{3}$$

where each of the $[\boldsymbol{y}_g^* \;\; \boldsymbol{X}_g^*]$ pairs is drawn at random, with replacement, from the original $[\boldsymbol{y}_g \;\; \boldsymbol{X}_g]$ pairs.

One problem with the **pairs cluster bootstrap** is that, unless the clusters are all the same size, the bootstrap samples will have different numbers of observations.

But using the ordinary pairs bootstrap when there is clustering can yield seriously invalid inferences.

Bootstrap samples can be used for many things. One of the oldest and simplest is to calculate **bootstrap standard errors**.

1. Generate B bootstrap samples, $\boldsymbol{Z}^{*b}$, for $b = 1, \ldots, B$.
2. For each bootstrap sample, compute an estimate $\hat{\theta}_b^*$ in the same way as $\hat{\theta}$ was computed from the original sample.
3. Calculate $\bar{\theta}^*$, the sample mean of the $\hat{\theta}_b^*$, and

$$\widehat{\mathrm{Var}}^*(\hat{\theta}_b^*) = \frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}_b^* - \bar{\theta}^*)^2. \tag{4}$$

The bootstrap standard error $\mathrm{se}^*(\hat{\theta})$ is the square root of $\widehat{\mathrm{Var}}^*(\hat{\theta}_b^*)$.

When $\boldsymbol{\theta}$ is a vector of parameters, the analog of (4) is

$$\widehat{\mathrm{Var}}^*(\hat{\boldsymbol{\theta}}_b^*) = \frac{1}{B-1}\sum_{b=1}^{B}(\hat{\boldsymbol{\theta}}_b^* - \bar{\boldsymbol{\theta}}^*)(\hat{\boldsymbol{\theta}}_b^* - \bar{\boldsymbol{\theta}}^*)^\top. \tag{5}$$

We can estimate the bias of $\hat{\theta}$ as $\bar{\theta}^* - \hat{\theta}$.

This leads to the **bias-corrected** estimate

$$\hat{\theta}_{\text{bc}} = \hat{\theta} - (\bar{\theta}^* - \hat{\theta}) = 2\hat{\theta} - \bar{\theta}^*. \tag{6}$$

Bootstrap bias correction does not seem to be used much in the context of machine learning.

However, it can be very useful for time-series models such as vector autoregressions.

Note that we might expect $\hat{\theta}_{\text{bc}}$ to have a larger variance than $\hat{\theta}$ because of the factor of 2 in (6).

However, the covariance with $\bar{\theta}$ also matters. There are cases in which a bias-corrected estimate can have less variance than the original.

See MacKinnon and Smith (1998).

Unfortunately, the fact that the resampling bootstrap omits about 36.8% of the observations from each bootstrap sample can make bootstrap estimates very misleading.

- For many procedures, like KNN and smoothing splines, how close the neighbouring points are to x_0 has a big impact on bias.
- If we choose the tuning parameter via cross-validation, this will also affect variance, because CV will suggest too much flexibility.
- Thus it is not always safe to estimate the bias, variance, or MSE of such procedures using the resampling bootstrap.

Bootstrap methods, like the wild bootstrap, that hold $\boldsymbol{X}$ fixed across bootstrap samples do not encounter these problems.

But the wild bootstrap is normally used with (linear) regression models. Can it also be used with methods like KNN?

The resampling bootstrap plays a key role in **bootstrap aggregation**, or **bagging**, and in **random forests**.

Prediction Intervals

In the regression case, we may want to quantify how accurate our predictions are by forming a **prediction interval**.

This is like a confidence interval, but for a prediction not a parameter.

To start with, consider the linear regression model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{u}, \quad \mathrm{E}(\boldsymbol{u}\boldsymbol{u}^\top) = \sigma^2 \mathbf{I}. \tag{7}$$

Suppose we estimate this using n training observations and attempt to forecast y_0 associated with observed $\boldsymbol{X}_0$.

The forecast is $\boldsymbol{X}_0\hat{\boldsymbol{\beta}}$. The **prediction error**, or **forecast error**, has mean zero, and variance

$$\begin{aligned} \mathrm{E}(y_0 - \boldsymbol{X}_0\hat{\boldsymbol{\beta}})^2 &= \mathrm{E}(\boldsymbol{X}_0\boldsymbol{\beta} + u_0 - \boldsymbol{X}_0\hat{\boldsymbol{\beta}})^2 && (8) \\ &= \mathrm{E}(u_0^2) + \mathrm{E}(\boldsymbol{X}_0\boldsymbol{\beta} - \boldsymbol{X}_0\hat{\boldsymbol{\beta}})^2 && (9) \\ &= \sigma_0^2 + \mathrm{Var}(\boldsymbol{X}_0\hat{\boldsymbol{\beta}}). && (10) \end{aligned}$$

There are two sources of error in (10). The first is the irreducible error σ_0^2, and the second is the estimation error in $\hat{\boldsymbol{\beta}}$.

Realistically, there is also a third source of error, namely, that the true expectation of y_0 is not $\boldsymbol{X}_0\boldsymbol{\beta}$.

The easiest way to estimate σ_0^2 is to use s^2, the standard error of the regression for the training sample.

For a correctly specified model, s^2 deals with the fact that residuals are too small by dividing the SSR by $n - p - 1$.

But this only works for the linear model estimated by OLS, and it assumes correct specification.

For nonlinear models, and linear models that use regularization, estimating σ^2 from the training sample can be tricky.

We can do better if we have a validation sample; see below.

The second term in (10) can be estimated by

$$\boldsymbol{X}_0^\top \widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}})\boldsymbol{X}_0, \tag{11}$$

where $\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}})$ is any valid covariance matrix estimate.

It might be

$$\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}}) = s^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1} \quad \text{or} \tag{12}$$

$$\widehat{\mathrm{Var}}_{\mathrm{h}}(\hat{\boldsymbol{\beta}}) = \frac{n}{n-p-1}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\left(\sum_{i=1}^{n}\hat{u}_i^2\boldsymbol{X}_i^\top\boldsymbol{X}_i\right)(\boldsymbol{X}^\top\boldsymbol{X})^{-1}. \tag{13}$$

So our estimate of squared prediction error is

$$\mathrm{se}^2_{\mathrm{pred}} = s^2 + \boldsymbol{X}_0^\top \widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}})\boldsymbol{X}_0. \tag{14}$$

The 95% prediction interval is then

$$[\hat{y}_0 - 1.96\mathrm{se}_{\mathrm{pred}},\ \ \hat{y}_0 + 1.96\mathrm{se}_{\mathrm{pred}}]. \tag{15}$$

The prediction interval in (15) is based entirely on the training data.

It is important to note that $\boldsymbol{X}_0^\top \widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}})\boldsymbol{X}_0$ may be small even when some elements of $\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}})$ are very large.

If there are large variance terms but also large and negative covariance terms, then weighted averages of the elements of $\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}})$ can be small.

If we have a validation sample with n_v observations, we can estimate the prediction error directly as

$$s^2_{\text{val}} = \frac{1}{n_v}\sum_{i=1}^{n_v}(y_i - \boldsymbol{X}_i\hat{\boldsymbol{\beta}})^2. \tag{16}$$

Here s^2_{val} is really an estimate of the sum of three things: σ_0^2, plus the variance of the forecast, plus the squared misspecification error.

When there is misspecification, s^2_{val} may greatly exceed s^2_{pred} in (14).

R can compute predictions and prediction intervals for various models using `predict()`, `predict.nls()`, `predict.glm()`, and so on.

If we use bootstrap methods to evaluate prediction intervals, we need to be careful.

Whatever method we use has to allow for both the irreducible error and the estimation errors.

Even for KNN and smoothing splines, estimation errors arise from the randomness of the training sample.

Recall that for KNN the prediction for $\boldsymbol{x}_0$ is $\frac{1}{K}\sum_{i\in\mathcal{N}_0} y_i$. This is random for two reasons:

- The K nearest neighbours would usually be different if we had a different training sample.
- The values of y_i associated with the $\boldsymbol{x}_i \in \mathcal{N}_0$ would be different even if the $\boldsymbol{x}_i$ were the same.

The resampling bootstrap does not do a great job of accounting for these two sources of randomness.