More about Linear Models

OLS tends to break down when p is not reasonably small relative to n.

As p increases, for n fixed, the coefficient estimates often become very noisy, leading to large prediction errors.

How much correlation there is among the columns of $\boldsymbol{X}$ matters.

When $\boldsymbol{X}$ is sufficiently **ill-conditioned**, a least squares algorithm may simply fail. It either yields no answer, or NaNs.

The **condition number** of a real symmetric matrix, say $\boldsymbol{X}^\top \boldsymbol{X}$, is

$$\kappa(\boldsymbol{X}^\top \boldsymbol{X}) = \frac{\lambda_{\max}(\boldsymbol{X}^\top \boldsymbol{X})}{\lambda_{\min}(\boldsymbol{X}^\top \boldsymbol{X})}, \tag{1}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of $\boldsymbol{X}^\top \boldsymbol{X}$.

Ideally κ should not be too large. You are probably safe if $\kappa < 10^6$.

Two things that can make κ large are regressors that vary a lot in magnitude and regressors that are highly correlated.

- Rescaling regressors that are very small or very large can help a great deal.
- If we, say, divide a regressor by 10^6 (going from dollars to millions of dollars), then the corresponding coefficient is multiplied by 10^6.

When $p > n$, least squares totally breaks down.

Even when $p << n$, we may encounter numerical problems, and the estimates may be very noisy, with large standard errors.

One approach is to reduce p by using **forward selection**, **backward selection**, or some other **model selection** procedure.

Any such procedure requires that we choose among models with different numbers of regressors.

This can be done in a number of ways, including sequential testing and cross-validation.

Criterion Functions

One approach to choosing the regressors to be included is to minimize a **criterion function** that penalizes the number of regressors.

The two best known are probably the **AIC** (**Akaike Information Criterion**) and the **BIC** (**Bayesian Information Criterion**).

Also very well known is the $\boldsymbol{C_p}$, or **Mallows** $\boldsymbol{C_p}$, criterion.

Note that the inventor of this criterion was the late Colin Lingwood Mallows, not a nonexistent person named Mallow, as ISLR/ISLP implicitly claims!

If d is the number of included regressors, the formula for C_p is

$$C_p = \frac{1}{n}\left(\text{SSR} + 2d\hat{\sigma}^2\right). \tag{2}$$

The factor of $1/n$ is irrelevant. What matters is that C_p adds a penalty of $2d\hat{\sigma}^2$ to SSR.

Typically, $\hat{\sigma}^2$ is estimated from the most general model, although this may not be a good idea if d/n is large. There are other possibilities.

In the linear regression case, it is best to use the unbiased estimator s^2 for the most general model as $\hat{\sigma}^2$ in (2).

Note that $\hat{\sigma}^2$ should (normally) stay the same as we make the model more or less parsimonious and thus vary d.

The **Akaike information criterion**, or **AIC**, is defined for all models estimated by maximum likelihood.

For linear regression models with Gaussian errors, it is

$$\mathrm{AIC} = \frac{1}{n\hat{\sigma}^2}(\mathrm{SSR} + 2d\,\hat{\sigma}^2), \tag{3}$$

which is just C_p divided by $\hat{\sigma}^2$.

For models estimated by maximum likelihood, the AIC is

$$\text{AIC} = 2\log L(\boldsymbol{\theta} \,|\, \boldsymbol{y}) - 2d. \tag{4}$$

Here we are implicitly using a different estimate of σ^2 for every value of d, and we maximize instead of minimize.

For linear models, it is probably better to use (3).

The penalties for C_p and AIC do not increase with n. This means that they may not choose the correct model asymptotically.

If we want a criterion function to have that property, then we need to make the penalty increase at the right rate with n.

The most popular criterion function that has this property is the **Bayesian information criterion**, or **BIC**.

It is sometimes called the **Schwarz information criterion**.

For regression models,

$$\mathrm{BIC} = \frac{1}{n\hat{\sigma}^2}\big(\mathrm{SSR} + \log(n)d\hat{\sigma}^2\big). \tag{5}$$

Thus BIC replaces $2d\hat{\sigma}^2$ in (3) by $\log(n)d\hat{\sigma}^2$.

For $n > 7$, $\log(n) > 2$. Thus the BIC penalty is greater than the AIC (and C_p) penalty for almost all sample sizes.

For models estimated by maximum likelihood, BIC is

$$\mathrm{BIC} = 2\log L(\boldsymbol{\theta}\,|\,\boldsymbol{y}) - d\log(n). \tag{6}$$

This is similar to (4), and once again we maximize it.

(5) and (6) would be equivalent if we used the ML estimate of σ^2 corresponding to each d in the former. But this is not recommended.

We can also use the **adjusted $\boldsymbol{R}^2$**, or $\bar{R}^2$, as a criterion function.

$$\bar{R}^2 = 1 - \frac{n-1}{n-d-1}\frac{\text{SSR}}{\text{TSS}}. \tag{7}$$

Since TSS does not change with d, maximizing $\bar{R}^2$ is equivalent to minimizing

$$\frac{\text{SSR}}{n-d-1}. \tag{8}$$

There is less theoretical justification for minimizing (8) than for minimizing C_p, or maximizing AIC or BIC.

If we want a parsimonious model that will never over-fit asymptotically, it is good to use BIC.

If the sample is not very large and we care more about potentially omitting variables that belong than including ones that do not, we can use AIC or, equivalently, C_p.

ISLR/ISLP illustrates these methods using (simulated) data on credit card defaults, with $n = 10000$.

They use best-subset selection (feasible because $p = 11$), along with forward stepwise and backward stepwise selection.

All methods agree that `rating`, `income`, and `student` are the top three predictors.

However, when `limit` is added, best-subset drops `rating` in favour of `cards`, the number of credit cards that a person holds.

Figure 6.02 shows results for C_p ($d_{\text{opt}} = 6$), BIC ($d_{\text{opt}} = 4$), and $\bar{R}^2$ ($d_{\text{opt}} = 7$). So they give different answers.

Figure 6.03 repeats the BIC results and adds validation set (7500 training, 2500 validation) and 10-fold cross-validation results.

Both validation methods choose $d_{\text{opt}} = 6$.

They did not try multiple validation sets or repeat the 10-fold cross-validation with different random folds.

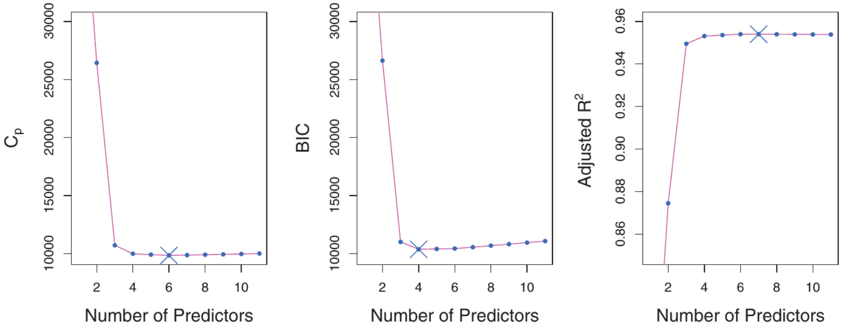


FIGURE 6.2. *C_p, BIC, and adjusted R^2 are shown for the best models of each size for the* Credit *data set (the lower frontier in Figure 6.1). C_p and BIC are estimates of test MSE. In the middle plot we see that the BIC estimate of test error shows an increase after four variables are selected. The other two plots are rather flat after four variables are included.*

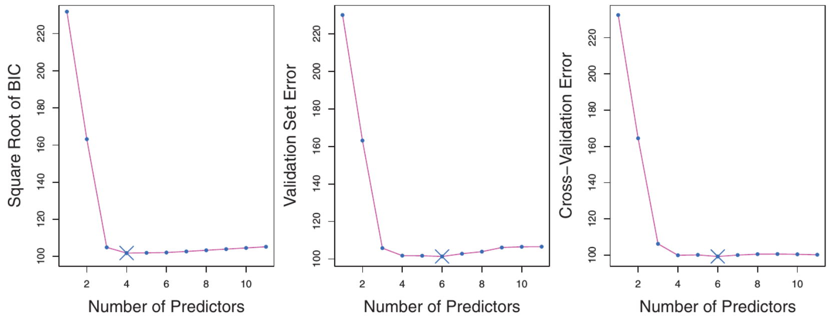


FIGURE 6.3. *For the* `Credit` *data set, three quantities are displayed for the best model containing d predictors, for d ranging from* 1 *to* 11*. The overall* best *model, based on each of these quantities, is shown as a blue cross.* Left: *Square root of BIC.* Center: *Validation set errors.* Right: *Cross-validation errors.*

In this example, and many others, several values of d yield very similar values of whatever criterion function(s) is (are) used.

One rule of thumb is to back off slightly (i.e. reduce d) by using the **one standard-error rule**.

- The idea is to calculate the standard error of whatever criterion function is being used for each d.
- Instead of using d_{opt}, try $d_{\text{opt}} - 1$, then $d_{\text{opt}} - 2$, and so on, stopping with the value of d that yields a criterion function less than one standard error below the optimal value.
- In this case, the result is $d = 3$, which seems a bit dubious.
- This requires computing standard errors of AIC, or BIC, or the cross-validation MSE, but ISLR/ISLP does not explain how.
- There may be analytic methods, or we could obtain a bootstrap standard error.
- But this seems like a lot of work to implement a rule of thumb.

Ridge Regression

When there are many coefficients, it can be better to shrink them all towards zero than to set some of them to zero. In general, this approach is called **regularization**.

The classic method is **ridge regression**, where we minimize

$$\sum_{i=1}^{n}\Bigl(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\Bigr)^2 + \lambda\sum_{j=1}^{p}\beta_j^2. \tag{9}$$

Here λ is a tuning parameter that controls the amount of shrinkage. The `glmnet` package performs ridge regression and related methods.

This type of regularization is called ℓ_2-regularization. Note that $\sum_{j=1}^{p}\beta_j^2 = \|\boldsymbol{\beta}\|^2$ if we do not treat β_0 as part of $\boldsymbol{\beta}$.

Importantly, ridge regression does not shrink the constant term.

If we ridge-regress demeaned y on demeaned X, the constant is the average of the y_i, minus the average of the fitted values.

Minimizing (9) is equivalent to solving the constrained minimization problem

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n}\Big(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\Big)^2 \text{ subject to } \sum_{j=1}^{p}\beta_j^2 \leq t. \tag{10}$$

In (9), λ plays the role of a Lagrange multiplier for the constraint in (10). Shrinkage increases as $\lambda \uparrow$ and as $t \downarrow$.

For any value of t in (10), there is an associated value of λ, which might be 0 if t is sufficiently large.

Since it makes no sense to penalize all the β_j in the same way if the magnitudes of the x_j differ substantially, it is essential to rescale all the non-dummy regressors to have variance 1. `glmnet` does this.

During the computation, the x_{ij} and the y_i are also recentred, so as to have mean 0. Thus we first estimate $\boldsymbol{\beta}$ and then β_0.

Dummy regressors are generally not rescaled.

We have to choose λ (or, equivalently, t) in some way. It is common to use cross-validation. Replacing λ by λn in (9) can make this easier.

The book uses 10-fold cross-validation.

It seems more common to choose λ than t, perhaps because we get OLS when $\lambda = 0$.

For t, we get OLS whenever the inequality in (10) is strict. So we set $t = \|\hat{\boldsymbol{\beta}}\|^2$ as the smallest value that gives the OLS estimates.

ISLR/ISLP graphs coefficient estimates and other things, like MSE, against both λ and $\|\hat{\boldsymbol{\beta}}_\lambda^R\|^2/\|\hat{\boldsymbol{\beta}}\|^2$. Here $t = \|\hat{\boldsymbol{\beta}}_\lambda^R\|^2$, so it is being normalized to lie between 0 and 1.

The graphs look quite different even though what is on the vertical axis is the same, and what is on the horizontal axis contains the same information.

Putting normalized t on the horizontal axis has the advantage that the scale is known in advance: It runs from 0 to 1.

A number that is often more interesting than λ is the **effective degrees of freedom** of the ridge regression fit.

If the d_j are the **singular values** of $\boldsymbol{X}$ (see below), then

$$\mathrm{df}(\lambda) \equiv \mathrm{Tr}\Big(\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\,\mathbf{I}_p)^{-1}\boldsymbol{X}^\top\Big) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}. \tag{11}$$

When λ is big, $\mathrm{df}(\lambda)$ is small.

As can be seen from (11), the effective degrees of freedom for the ridge regression fit, $\mathrm{df}(\lambda)$, never exceeds p, the actual degrees of freedom for the corresponding OLS regression.

The limiting case occurs when $\lambda = 0$, so that $\mathrm{df}(\lambda) = p$.

If we wanted to, we could use $\mathrm{df}(\lambda)$ in a criterion function like C_p, AIC, or BIC as an alternative to cross-validation.

The Singular Value Decomposition

The **singular value decomposition** of $\boldsymbol{X}$ is

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top, \tag{12}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are $n \times p$ and $p \times p$ orthogonal matrices, with $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{V}^\top\boldsymbol{V} = \mathbf{I}_p$ and $\mathcal{S}(\boldsymbol{U}) = \mathcal{S}(\boldsymbol{X})$.

The columns of $\boldsymbol{V}$ are the **eigenvectors** of $\boldsymbol{X}^\top\boldsymbol{X}$.

The **eigen decomposition** of $\boldsymbol{X}^\top\boldsymbol{X}$ is

$$\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}^\top, \tag{13}$$

where $\boldsymbol{D}$ is a diagonal matrix.

The eigenvectors v_j are the columns of $\boldsymbol{V}$.

The **eigenvalues** of $\boldsymbol{X}^\top\boldsymbol{X}$ are the diagonal elements of $\boldsymbol{D}^2$, and the **singular values** of $\boldsymbol{X}$ are the diagonal elements of $\boldsymbol{D}$.

Computing the Ridge Estimator

First, we find the means of all variables and subtract them; that is, we centre all variables. Then

$$\hat{\boldsymbol{\beta}}^R_\lambda = \left((\boldsymbol{X} - \bar{\boldsymbol{X}})^\top (\boldsymbol{X} - \bar{\boldsymbol{X}}) + \lambda \mathbf{I}_p\right)^{-1} (\boldsymbol{X} - \bar{\boldsymbol{X}})^\top (\boldsymbol{y} - \bar{\boldsymbol{y}}). \tag{14}$$

If $\lambda = 0$, this gives us OLS on the centred data.

Here $\bar{\boldsymbol{y}}$ is a vector with $\bar{y}$ in each of the n positions, and $\bar{\boldsymbol{X}}$ is an $n \times p$ matrix with $\bar{\boldsymbol{x}}^\top$ in each of the n rows.

The constant term is just the mean of the y_i minus the mean of the ridge fitted values.

$$\hat{\beta}^R_{0\lambda} = \bar{y} - \bar{\boldsymbol{x}}^\top \hat{\boldsymbol{\beta}}^R_\lambda. \tag{15}$$

Importantly, the matrix $\left((\boldsymbol{X} - \bar{\boldsymbol{X}})^\top (\boldsymbol{X} - \bar{\boldsymbol{X}}) + \lambda \mathbf{I}_p\right)^{-1}$ is nonsingular even when $\boldsymbol{X} - \bar{\boldsymbol{X}}$ does not have full rank.

Thus we can use ridge regression even on ill-conditioned datasets and when $p > n$.

We can trick OLS into giving us ridge estimates. Form the augmented dataset

$$\boldsymbol{X}_a = \begin{bmatrix} \boldsymbol{X} - \bar{\boldsymbol{X}} \\ \sqrt{\lambda}\,\mathbf{I}_p \end{bmatrix} \quad \text{and} \quad \boldsymbol{y}_a = \begin{bmatrix} \boldsymbol{y} - \bar{\boldsymbol{y}} \\ \mathbf{0} \end{bmatrix}, \tag{16}$$

which has $n + p$ observations. Then OLS estimation of the regression

$$\boldsymbol{y}_a = \boldsymbol{X}_a \boldsymbol{\beta} + \boldsymbol{u}_a \tag{17}$$

yields the ridge regression estimator, because

$$\boldsymbol{X}_a^\top \boldsymbol{X}_a = (\boldsymbol{X} - \bar{\boldsymbol{X}})^\top (\boldsymbol{X} - \bar{\boldsymbol{X}}) + \lambda \mathbf{I}_p \tag{18}$$

and

$$\boldsymbol{X}_a^\top \boldsymbol{y}_a = (\boldsymbol{X} - \bar{\boldsymbol{X}})^\top (\boldsymbol{y} - \bar{\boldsymbol{y}}). \tag{19}$$

Of course, this might not be computationally attractive when $p >> n$.

Examples of Ridge Regression

In Section 6.5.2, ISLR/ISLP goes through an empirical example of ridge regression, using the `Hitters` dataset.

The object is to predict `Salary`. Why not the log of `Salary`?

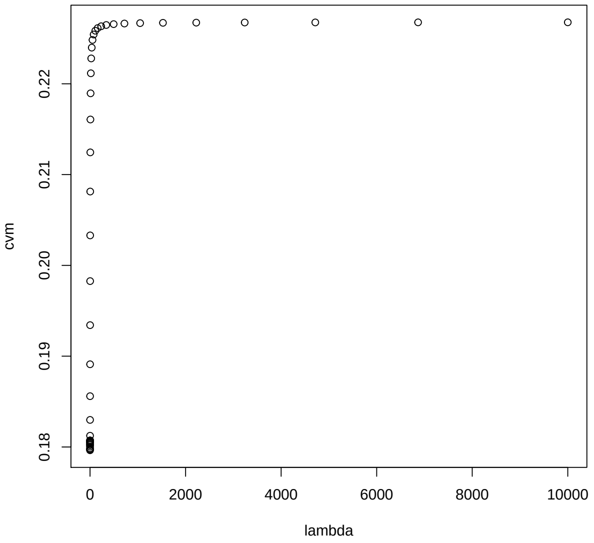
I studied another dataset, used in ESL, that concerns heart disease.

The output is binary, so I should have used logistic regression, but I just estimated a linear probability model.

- There are nine predictors, of which five are "significant" in an OLS regression using conventional standard errors.
- I specified a grid of 50 λ values from 10^{-4} to 10^4. Each was larger than its predecessor by a factor of 1.4563.
- I used 10-fold cross-validation via the `cv.glmnet` command.

Figure 9.1 shows the CV mean squared error plotted against λ. Plotting anything against λ on a linear scale is evidently a bad idea!

Figure 9.1



The values of λ vary enormously, but the big effects on model fit occur over a fairly small range of them.

We should either plot against $\log(\lambda)$ or against $\|\hat{\boldsymbol{\beta}}_\lambda^R\|^2 / \|\hat{\boldsymbol{\beta}}\|^2$, as the book does in Figure 6.5.

When we plot against $\log(\lambda)$, in Figure 9.2, we get a much more readable figure. Using `plot(cvfit)` is a better way to do this; see Figure 9.2a. This also gives us confidence intervals.

- The CV-MSE minimizing value of $\log(\lambda)$ is -2.4436 (purple).
- A small amount of shrinkage helps a little. The CV-MSE drops from 0.1807, for OLS, to 0.1796 for the CV-MSE minimizing value.
- But if we shrink even a little bit too much, using $\log(\lambda) = -1.3156$, we do worse than OLS.

The program also reports the one-standard-error value of $\log(\lambda)$, which is -0.5639. It gives a CV-MSE of 0.1856, so way too much shrinkage.

Figure 9.2

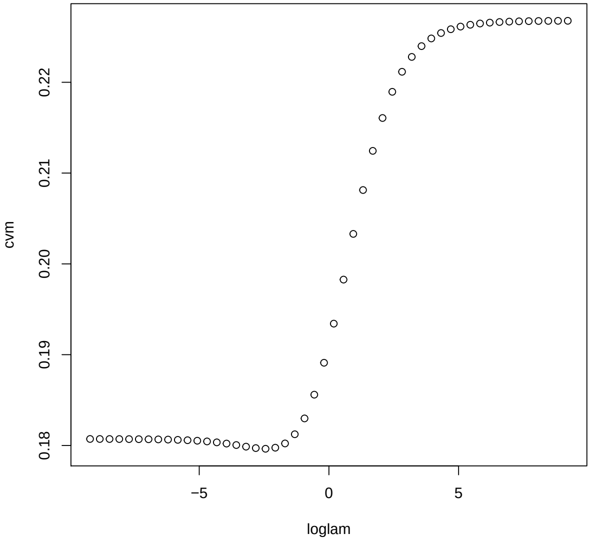


Figure 9.2a (produced by glmnet)

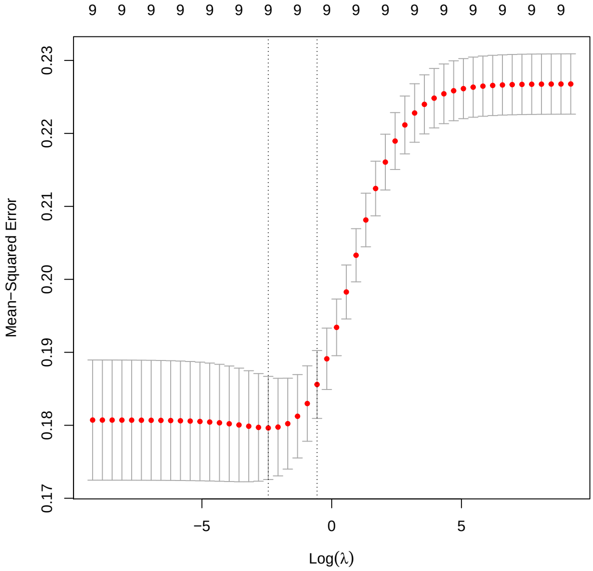


Figure 9.3 shows estimates for three predictors: family history (black), tobacco use (red), and low-density cholesterol (blue).

There is evidently a lot more shrinkage at $\log(\lambda) = -0.5639$ than at $\log(\lambda) = -2.4436$.

The latter is the value of λ chosen by cross-validation, and the former is the one-standard-error value.

- The coefficients in Figure 9.3 are not standardized.
- `glmnet` standardizes the predictors before performing ridge regression, but then reports coefficients for the original data.
- Thus you should *not* scale the regressors when using `glmnet`.
- Of course, when using any estimation method, it can be desirable to scale some regressors so that the coefficients are reasonable.

Figure 9.3

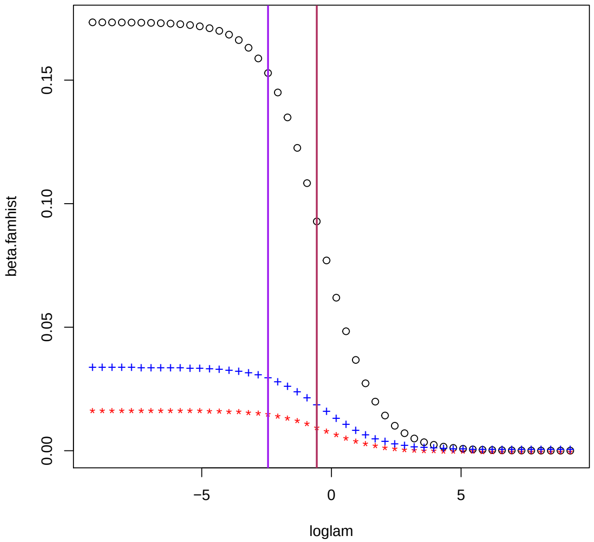


Figure 6.5 from ISLR uses simulated data to illustrate how the bias-variance tradeoff works for ridge regression.

Note that both panels show the same values of squared bias, variance, and (test) MSE. But what is on the horizontal axis differs.

In the left panel, λ is on the horizontal axis, but since the scale is logarithmic, it is really $\log \lambda$.

In the right panel, it is $\|\hat{\boldsymbol{\beta}}_\lambda^R\|^2/\|\hat{\boldsymbol{\beta}}\|$, presumably after the regressors have been scaled.

As expected, variance decreases with λ, and squared bias increases.

The optimal value of λ occurs where variance is decreasing fairly sharply and squared bias is increasing.

Equivalently, the optimal value of $\|\hat{\boldsymbol{\beta}}_\lambda^R\|^2/\|\hat{\boldsymbol{\beta}}\|$ occurs where variance is increasing and squared bias is decreasing sharply.

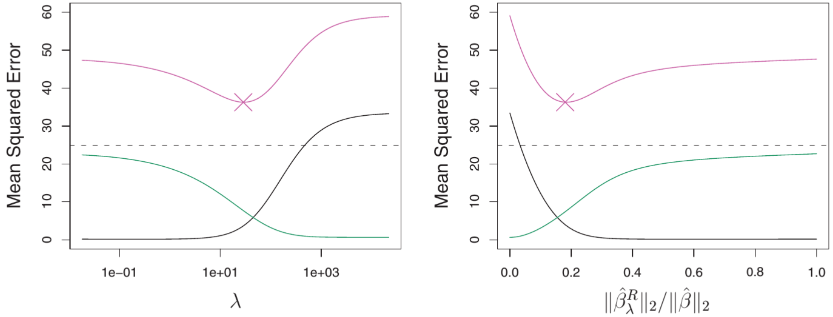


FIGURE 6.5. *Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on a simulated data set, as a function of λ and $\|\hat{\beta}_\lambda^R\|_2/\|\hat{\beta}\|_2$. The horizontal dashed lines indicate the minimum possible MSE. The purple crosses indicate the ridge regression models for which the MSE is smallest.*

Because ridge regression adds bias in order to reduce variance, it works best when the OLS variance is large.

- This tends to happen when p/n is large, or when there is a lot of collinearity among the regressors.
- If OLS yields precise coefficient estimates for all coefficients of interest, then it makes no sense to use ridge regression.

When only some of the coefficients are a problem, it may make sense to use more complicated forms of ℓ_2-regularization.

`glmnet` allows a user to specify a penalty factor for each coefficient.

The ordinary penalty term in (9) is replaced by

$$\lambda \sum_{j=1}^{p} \nu_j \beta_j^2, \quad \nu_j \geq 0. \tag{20}$$

Although the ν_j could be any non-negative numbers, the natural choices are 1 and 0.

Typically, the ν_j would be 1 for most coefficients but 0 for a small number of them that we do not want to shrink.

Instead of shrinking coefficients towards 0, we can shrink them towards each other in various ways.

Classic papers are Shiller (1973) and Gersovitz and MacKinnon (1978).

Both used **smoothness priors** to estimate subsets of coefficients. For Shiller, they were the coefficients of a "distributed lag." For G&M, they were the coefficients on seasonal dummy variables.

Instead of the coefficients themselves being shrunk towards zero, the differences between nearby ones are shrunk.

The algorithms use dummy observations more complicated than the ones in (16).

For example, if there are j coefficients to be smoothed, the penalty could be linear in the second differences of adjacent coefficients:

$$\beta_{j+1} - 2\beta_j + \beta_{j-1}, \quad j = 2, J-1. \tag{21}$$