ECON 950 — Winter 2020

Prof. James MacKinnon

3. Methods Based on Linear Regression

The methods discussed in this section are primarily intended for high-dimensional situations, where p is large relative to N, perhaps much larger than N.

We can write

$$f(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}\boldsymbol{\beta} = \beta_0 + \sum_{j=1}^{p} \beta_j x_j, \tag{1}$$

treating the constant and other regressors separately. In the notation of ETM, p denotes $k-1$, and N denotes n.

Of course, the x_j can include transformations of inputs, such as square roots, logs, squares, or higher powers, various types of dummy variables, and products of inputs or of transformations of them.

So a "linear" regression model can actually allow for very nonlinear relationships.

3.1. Regression by Successive Orthogonalization

When the $\boldsymbol{x}_j$ are orthogonal, the $\hat{\beta}_j = \boldsymbol{x}_j^\top \boldsymbol{y} / \boldsymbol{x}_j^\top \boldsymbol{x}_j$ obtained by univariate regressions of $\boldsymbol{y}$ on each of the $\boldsymbol{x}_j$ are equal to the multiple regression least squares estimates.

We can accomplish something similar even when the data are not orthogonal. This is called **Gram-Schmidt Orthogonalization**. It is related to the FWL Theorem.

1. Set $\boldsymbol{z}_0 = \boldsymbol{x}_0 = \boldsymbol{\iota}$.
2. Regress $\boldsymbol{x}_1$ on $\boldsymbol{z}_0$ to obtain a residual vector $\boldsymbol{z}_1$ and coefficient $\hat{\gamma}_{01}$.
3. For $j = 2, \ldots, p$, regress $\boldsymbol{x}_j$ on $\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{j-1}$ to obtain a residual vector $\boldsymbol{z}_j$ and coefficients $\hat{\gamma}_{\ell j}$ for $\ell = 0, \ldots, j-1$.
4. Regress $\boldsymbol{y}$ on $\boldsymbol{z}_p$ to obtain $\hat{\beta}_p = \boldsymbol{z}_p^\top \boldsymbol{y} / \boldsymbol{z}_p^\top \boldsymbol{z}_p$.

The scalar $\hat{\beta}_p$ is the coefficient on $\boldsymbol{x}_p$ in the multiple regression of $\boldsymbol{y}$ on a constant and all the $\boldsymbol{x}_j$.

Note that step 3 simply involves j univariate regressions.

If we did this $p+1$ times, changing the ordering so that each of the $\boldsymbol{x}_j$ came last, we could obtain all the OLS coefficients.

The **QR decomposition** of $\boldsymbol{X}$ is

$$\boldsymbol{X} = \boldsymbol{Z}\boldsymbol{D}^{-1}\boldsymbol{D}\boldsymbol{\Gamma} = \boldsymbol{Q}\boldsymbol{R}, \tag{2}$$

where $\boldsymbol{Z}$ contains the columns $\boldsymbol{z}_0$ to $\boldsymbol{z}_p$ (in order), $\boldsymbol{\Gamma}$ is an upper triangular matrix containing the $\hat{\gamma}_{\ell j}$, and $\boldsymbol{D}$ is a diagonal matrix with typical diagonal element $||\boldsymbol{z}_j||$. Thus $\boldsymbol{Q} = \boldsymbol{Z}\boldsymbol{D}^{-1}$ and $\boldsymbol{R} = \boldsymbol{D}\boldsymbol{\Gamma}$.

It can be seen that $\boldsymbol{Q}$ is an $N \times (p+1)$ orthogonal matrix with the property that $\boldsymbol{Q}^\top\boldsymbol{Q} = \mathbf{I}$, and $\boldsymbol{R}$ is a $(p+1) \times (p+1)$ upper triangular matrix.

The QR decomposition provides a convenient orthogonal basis for $\mathcal{S}(\boldsymbol{X})$.

It is easy to see that

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} = \boldsymbol{R}^{-1}\boldsymbol{Q}^\top\boldsymbol{y}, \tag{3}$$

because

$$(\boldsymbol{X}^\top\boldsymbol{X})^{-1} = (\boldsymbol{R}^\top\boldsymbol{Q}^\top\boldsymbol{Q}\boldsymbol{R})^{-1} = (\boldsymbol{R}^\top\boldsymbol{R})^{-1} = \boldsymbol{R}^{-1}(\boldsymbol{R}^\top)^{-1},$$

and

$$\boldsymbol{X}^\top\boldsymbol{y} = \boldsymbol{R}^\top\boldsymbol{Q}^\top\boldsymbol{y}.$$

Since $\boldsymbol{R}$ is triangular, it is very easy to invert. Also,

$$\hat{\boldsymbol{y}} = \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{Q}^{\top}\boldsymbol{y}. \tag{4}$$

so that

$$\text{SSR}(\hat{\boldsymbol{\beta}}) = (\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{Q}^{\top}\boldsymbol{y})^{\top}(\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{Q}^{\top}\boldsymbol{y}) = \boldsymbol{y}^{\top}\boldsymbol{y} - \boldsymbol{y}^{\top}\boldsymbol{Q}\boldsymbol{Q}^{\top}\boldsymbol{y}. \tag{5}$$

The diagonals of the hat matrix are the diagonals of $\boldsymbol{Q}\boldsymbol{Q}^{\top}$.

The QR decomposition is easy to implement and numerically stable. It requires $O(Np^2)$ operations.

An alternative is to form $\boldsymbol{X}^{\top}\boldsymbol{X}$ and $\boldsymbol{X}^{\top}\boldsymbol{y}$ efficiently, use the Cholesky decomposition to invert $\boldsymbol{X}^{\top}\boldsymbol{X}$, and then multiply $\boldsymbol{X}^{\top}\boldsymbol{y}$ by $(\boldsymbol{X}^{\top}\boldsymbol{X})^{-1}$.

The Cholesky approach requires $O(p^3 + Np^2/2)$ operations. For $p << N$, Cholesky wins, but for large p QR does. Never use a general matrix inversion routine, because such routines are not as stable as QR or Cholesky.

In extreme cases, QR can yield reliable results when Cholesky does not. If $\boldsymbol{X}^{\top}\boldsymbol{X}$ is not well-conditioned (i.e., if the ratio of the largest to the smallest eigenvalue is large), then it is much safer to use QR than Cholesky. Intelligent scaling helps.

3.2. Subset Selection

When p is large, OLS tends to have low bias but large variance.

By setting some coefficients to zero, we can often reduce variance substantially while increasing bias only modestly.

It can be hard to make sense of models with a great many coefficients.

Best-subset regression tries every subset of size $k = 1, \ldots, p$ to find the one that, for each k, has the smallest SSR.

There is an efficient algorithm, but it becomes infeasible once p becomes even moderately large (say much over 40).

Forward-stepwise selection and **backward-stepwise selection** are feasible even when p is large. The former works even if $p > N$.

To begin the forward-stepwise selection procedure, we regress $\boldsymbol{y}$ on each of the $\boldsymbol{x}_j$ and pick the one that yields the smallest SSR.

Next, we add each of the remaining $p - 1$ regressors, one at a time, and pick the one that yields the smallest SSR.

Then we add each of the remaining $p-2$ regressors, one at a time, and pick the one that yields the smallest SSR.

If we are using the QR decomposition, we can do this efficiently. We are just adding one more column to $\boldsymbol{Z}$ and hence $\boldsymbol{Q}$, and one more row to $\boldsymbol{\Gamma}$ and hence $\boldsymbol{R}$.

We do this for each of the candidate regressors and calculate the SSR from (5) for each of them. We can reuse most of the elements of the old $\boldsymbol{Q}$ matrix.

For backward-stepwise regression, we start with the full model (assuming that $p < N$) and then successively drop regressors. At each step, we drop the one with the smallest t statistic.

Smart stepwise programs will recognize that regressors often come in groups. It generally does not make sense to

- Drop a regressor X if the model includes X^2 or XZ;
- Drop some dummy variables that are part of a set (e.g. some province dummies but not all, or some categorical dummies but not all).

In cases like this, variables should be added, or subtracted, in groups.

If we use either forward-stepwise or backward-stepwise selection, how do we decide which model to choose?

The model with all p regressors will always fit best, in terms of SSR.

We either need to penalize the number of parameters (d) or use cross-validation.

$$C_p: \quad \frac{1}{N}(\text{SSR} + 2d\hat{\sigma}^2) = (1 + 2d/N)\hat{\sigma}^2. \tag{6}$$

Thus C_p adds a penalty of $2d\hat{\sigma}^2$ to SSR.

The **Akaike information criterion**, or **AIC**, is defined for all models estimated by maximum likelihood. For linear regression models with Gaussian errors,

$$\text{AIC} = \frac{1}{N\hat{\sigma}^2}(\text{SSR} + 2d\hat{\sigma}^2), \tag{7}$$

which is proportional to C_p.

More generally, with the sign reversed, the AIC equals $2 \log L(\boldsymbol{\theta} \mid \boldsymbol{y}) - 2d$. Notice that the penalty for AIC does not increase with N.

The **Bayesian information criterion**, or **BIC**, for linear regression models is

$$\text{BIC} = \frac{1}{N}\big(\text{SSR} + \log(N) d \hat{\sigma}^2\big). \tag{8}$$

More generally, with the sign reversed, $\text{BIC} = 2 \log L(\boldsymbol{\theta} \mid \boldsymbol{y}) - d \log N$.

When N is large, BIC places a much heavier penalty on d than does AIC, because $\log(N) >> 2$. Therefore, the chosen model is very likely to have lower complexity.

If the correct model is among the ones we estimate, BIC chooses it with probability 1 as $N \to \infty$. The other two criteria sometimes choose excessively complex models.

3.3. Ridge Regression

As we have seen, ridge regression minimizes the objective function

$$\sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2, \tag{9}$$

where λ is a complexity parameter that controls the amount of shrinkage. This is an example of ℓ_2-regularization.

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

$$\min \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq t. \tag{10}$$

In (9), λ plays the role of a Lagrange multiplier. For any value of t, there is an associated value of λ, which might be 0 if t is sufficiently large.

Notice that the intercept is not included in the penalty term. The simplest way to accomplish this is to recenter all variables before running the regression.

It would seem very strange to impose the same penalty on each of the β_j^2 if the associated regressors were not scaled similarly. Therefore, it is usual to rescale all the regressors to have variance 1. They already have mean 0 from the recentering.

We have seen that

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}_p)^{-1} \boldsymbol{X}^\top \boldsymbol{y}, \tag{11}$$

where every column of $\boldsymbol{X}$ now has sample mean 0 and sample variance 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

$$\mathrm{df}(\lambda) \equiv \mathrm{Tr}\big(\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbf{I}_p)^{-1}\boldsymbol{X}^\top\big) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}\,. \tag{12}$$

As $\lambda \to 0$, $\mathrm{df}(\lambda) \to p$.

Thus the effective degrees of freedom for the ridge regression is never greater than the actual degrees of freedom for the corresponding OLS regression.

The ridge regression estimator (11) can be obtained as the mode (and also the mean) of a Bayesian posterior. Suppose that

$$y_i \sim \mathrm{N}(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}, \sigma^2), \tag{13}$$

and the β_j are believed to follow independent $\mathrm{N}(0, \tau^2)$ distributions. Then expression (9) is minus the log of the posterior, with $\lambda = \sigma^2/\tau^2$.

Thus minimizing (9) yields the posterior mode, which is $\hat{\boldsymbol{\beta}}_{\mathrm{ridge}}$.

Evidently, as τ increases, λ becomes smaller. The more diffuse our prior about the β_j, the less it affects the posterior mode.

Similarly, as σ^2 increases, so that the data become noisier, the more our prior influences the posterior mode.

A simple way to compute ridge regression for any given λ is to form the augmented dataset

$$\boldsymbol{X}_a = \begin{bmatrix} \boldsymbol{X} \\ \sqrt{\lambda}\mathbf{I}_p \end{bmatrix} \quad \text{and} \quad \boldsymbol{y}_a = \begin{bmatrix} \boldsymbol{y} \\ \mathbf{0} \end{bmatrix}, \tag{14}$$

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

$$\boldsymbol{y}_a = \boldsymbol{X}_a\boldsymbol{\beta} + \boldsymbol{u}_a \tag{15}$$

yields the ridge regression estimator (11), because

$$\boldsymbol{X}_a^\top\boldsymbol{X}_a = \boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbf{I}_p \quad \text{and} \quad \boldsymbol{X}_a^\top\boldsymbol{y}_a = \boldsymbol{X}^\top\boldsymbol{y}. \tag{16}$$

In this case, it is tempting to use Cholesky, at least when p is not too large, because it is easy to update $\boldsymbol{X}_a^\top\boldsymbol{X}_a$ and $\boldsymbol{X}_a^\top\boldsymbol{y}_a$ as we vary λ.

3.4. The Singular Value Decomposition

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

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top, \tag{17}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are $N \times p$ and $p \times p$ orthogonal matrices, with $\mathcal{S}(\boldsymbol{U}) = \mathcal{S}(\boldsymbol{X})$. Thus $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{V}^\top\boldsymbol{V} = \mathbf{I}_p$. The columns of $\boldsymbol{V}$ are the **eigenvectors** of $\boldsymbol{X}^\top\boldsymbol{X}$.

Recall that λ is an **eigenvalue** of a matrix $\boldsymbol{A}$ if there exists a nonzero vector $\boldsymbol{\xi}$, called an **eigenvector**, such that

$$\boldsymbol{A}\boldsymbol{\xi} = \lambda\boldsymbol{\xi}. \tag{18}$$

Thus the action of $\boldsymbol{A}$ on $\boldsymbol{\xi}$ produces a vector with the same direction as $\boldsymbol{\xi}$, but a different length unless $\lambda = 1$.

In (17), $\boldsymbol{D}$ is a $p \times p$ diagonal matrix, with diagonal elements d_j that have the property

$$d_1 \geq d_2 \geq \ldots \geq d_p \geq 0. \tag{19}$$

These are the square roots of the eigenvalues of $\boldsymbol{X}^\top\boldsymbol{X}$.

The rank of the matrix $\boldsymbol{D}$, and also of the matrix $\boldsymbol{X}^\top\boldsymbol{X}$, is the number of d_j that are strictly positive.

Observe that

$$\begin{aligned}\boldsymbol{P_X y} &= \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} \\ &= \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top(\boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top)^{-1}\boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^\top\boldsymbol{y} \\ &= \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top(\boldsymbol{D}\boldsymbol{V}^\top)^{-1}(\boldsymbol{U}^\top\boldsymbol{U})^{-1}(\boldsymbol{V}\boldsymbol{D})^{-1}\boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^\top\boldsymbol{y} \\ &= \boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{y}.\end{aligned} \tag{20}$$

We can now see that

$$\begin{aligned}\boldsymbol{X}\hat{\boldsymbol{\beta}}_{\text{ridge}} &= (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbf{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y} \\ &= \boldsymbol{U}\boldsymbol{D}(\boldsymbol{D}^2 + \lambda\mathbf{I})^{-1}\boldsymbol{D}\boldsymbol{U}^\top\boldsymbol{y} \\ &= \sum_{j=1}^{p}\boldsymbol{u}_j\frac{d_j^2}{d_j^2+\lambda}\boldsymbol{u}_j{}^\top\boldsymbol{y},\end{aligned} \tag{21}$$

where $\boldsymbol{u}_j$ is the j^{th} column of $\boldsymbol{U}$.

When $\lambda = 0$, the scalars in the last line of (21) collapse to 1, and we see that

$$\boldsymbol{P_X y} = \boldsymbol{X}\hat{\boldsymbol{\beta}}_{\text{OLS}} = \sum_{j=1}^{p} \boldsymbol{u}_j \boldsymbol{u}_j^\top \boldsymbol{y}, \tag{22}$$

which is another way of writing the last line of (20).

When $\lambda > 0$, the estimates are shrunk. There is more shrinkage applied to the coordinates of basis vectors with small values of d_j.

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

$$\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{V} \boldsymbol{D}^2 \boldsymbol{V}^\top. \tag{23}$$

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

The largest eigenvector $\boldsymbol{v}_1$ has the property that $\boldsymbol{z}_1 = \boldsymbol{X}\boldsymbol{v}_1$ has the largest sample variance among all normalized linear combinations of the columns of $\boldsymbol{X}$.

Since $\boldsymbol{z}_1 = \boldsymbol{u}_1 d_1$, this sample variance is

$$\text{Var}(\boldsymbol{X}\boldsymbol{v}_1) = \text{Var}(\boldsymbol{u}_1 d_1) = d_1^2/N. \tag{24}$$

The first **principal component** of $\boldsymbol{X}$ is $\boldsymbol{z}_1$. Subsequent principal components are orthogonal to $\boldsymbol{z}_1$, and to each other. Their variance declines as j increases.

Thus small singular values d_j correspond to directions in $\mathcal{S}(\boldsymbol{X})$ that have small variance. Ridge regression shrinks these directions the most, because they provide relatively little information, which may easily be swamped by noise.

3.5. Principal Components Regression

Recall from (24) that the principal components of $\boldsymbol{X}$ are $\boldsymbol{z}_m = \boldsymbol{X}\boldsymbol{v}_m$, where the $\boldsymbol{v}_m$ are the eigenvectors, that is, the columns of $\boldsymbol{V}$. These are $p \times 1$.

If we regress $\boldsymbol{y}$ on the first M principal components, we find that

$$\hat{\boldsymbol{y}}^{\text{pc}} = \bar{y}\boldsymbol{\iota} + \sum_{m=1}^{M} \hat{\theta}_m \boldsymbol{z}_m, \tag{25}$$

where, because the $\boldsymbol{z}_m$ are orthogonal,

$$\hat{\theta}_m = \frac{\boldsymbol{z}_m{}^\top \boldsymbol{y}}{\boldsymbol{z}_m{}^\top \boldsymbol{z}_m}. \tag{26}$$

As with other methods, it is a good idea to standardize the $\boldsymbol{x}_j$ so that they have mean 0 and variance 1.

The principal components are ordered in the same way as the eigenvalues:

$$||\boldsymbol{z}_1|| \geq ||\boldsymbol{z}_2|| \geq ||\boldsymbol{z}_3|| \geq \ldots \geq ||\boldsymbol{z}_p||. \tag{27}$$

Thus $\boldsymbol{z}_1$ is the direction in which the columns of $\boldsymbol{X}$ vary the most.

For principal components regression (PCR), we first regress $\boldsymbol{y}$ on $\boldsymbol{z}_1$, then on $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, and so on.

Because the $\boldsymbol{z}_m$ are orthogonal, adding another regressor, say $\boldsymbol{z}_g$, simply means regressing the current residual

$$\hat{\boldsymbol{\epsilon}}_{g-1} \equiv \boldsymbol{y} - \hat{\theta}_0 - \hat{\theta}_1 \boldsymbol{z}_1 - \hat{\theta}_2 \boldsymbol{z}_2 - \ldots - \hat{\theta}_{g-1} \boldsymbol{z}_{g-1} \tag{28}$$

on $\boldsymbol{z}_g$. Then we set $\hat{\boldsymbol{\epsilon}}_g = \hat{\boldsymbol{\epsilon}}_{g-1} - \hat{\theta}_g \boldsymbol{z}_g$. We keep going in this way until $g = M$.

The raw fit will improve as M increases, but criteria like AIC may get better or worse. As usual, we can also use cross-validation to choose M.

Of course, one can recover the implied β coefficients from the $\hat{\theta}_m$:

$$\hat{\boldsymbol{\beta}}^{\text{pc}} = \sum_{m=1}^{M} \hat{\theta}_m \boldsymbol{v}_m. \tag{29}$$

Especially when M is small, these are likely to be shrunk relative to the OLS estimates. When $M = p$, these will be equal to OLS estimates.

There is a close relationship between ridge regression and PCR. Both operate via the principal components of the input matrix.

In both cases, coefficients are shrunk but not set to zero.

For ridge, directions with small eigenvalues get shrunk more than ones with large eigenvalues; recall (21).

For PCR, directions with small eigenvalues are discarded, while ones with large eigenvalues are retained.

3.6. The Lasso

Lasso (Tibshirani, 1996) stands for **least absolute shrinkage and selection operator**.

The lasso can be obtained as the solution to

$$\min \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j \, x_{ij} \right)^2 \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq t. \tag{30}$$

This looks almost identical to (10). The main difference is that the constraint involves absolute values instead of squares. Thus the lasso involves ℓ_1-regularization.

As with ridge regression, solving (30) is equivalent to minimizing

$$\frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j \, x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{31}$$

where λ is a Lagrange multiplier; compare (9). Large values of λ are associated with small values of t.

Following SLS (but not ESL or ISLR), there is a factor of $1/(2N)$ in (30) and (31). This is not needed, but it makes the values of λ comparable across samples of different sizes.

If all variables are recentered to have mean zero, we can dispense with the β_0 parameter. We can recover it if necessary as

$$\hat{\beta}_0 = \bar{y} - \sum_{j=1}^{p} \hat{\beta}_j \bar{x}_j. \tag{32}$$

The obvious way to solve (30) is by quadratic programming, but there are better approaches, which we discuss below.

The key difference between ridge and lasso is that, for lasso, some of the coefficients will be 0 when t is sufficiently small (equivalently, when λ is sufficiently large).

If t is sufficiently large—greater than $t_0 = \sum_{j=1}^{p} |\hat{\beta}_j|$—then $\hat{\boldsymbol{\beta}}^{\text{lasso}} = \hat{\boldsymbol{\beta}}^{\text{ols}}$. In this case, the constraints do not bind.

For $t < t_0$, the lasso coefficients are shrunk, and some of them may well be 0. Ultimately, as $t \to 0$, they will all be 0.

For the tuning parameter, we can use λ, t, or the **standardized tuning parameter**

$$s \equiv t \Big/ \sum_{j=1}^{p} |\hat{\beta}_j|. \tag{33}$$

Whichever one we use, it is typically chosen by cross-validation.

The above statement is true when lasso is used for prediction, but not when it is used for inference. Other methods are proposed in papers by Chernozhukov et al.

ESL and SLS recommend using K-fold cross-validation, where $K = 5$ or $K = 10$. We will discuss these methods in detail later.

For K-fold cross-validation, we divide the training sample into K folds (subsamples) of equal or roughly equal size.

Then we sequentially omit one fold at a time, applying the lasso to the other $K - 1$ folds. This gives us K sets of estimates.

For $k = 1, \ldots, K$, the estimates using all folds except the k^{th} are then used to compute fitted values for observations in fold k.

We use these fitted values to compute the mean squared prediction error for all observations. This is then plotted against t, s, or λ to choose the optimal value.

As usual, when the bound is too tight, the bias is large, and when the bound is too loose, the variance is large.

In the former case, there will be few non-zero coefficients, and in the latter case there will be many.

See SLS-fig-2.3.pdf for cross-validated estimate of prediction error.

If $\boldsymbol{X}$ is orthonormal, there is an explicit solution for lasso:

$$\hat{\beta}_j^{\text{lasso}} = \text{sign}(\hat{\beta}_j)\big(|\hat{\beta}_j| - \lambda\big)_+, \tag{34}$$

where $(z)_+ = z$ if $z > 0$ and $(z)_+ = 0$ if $z \leq 0$.

Thus $\hat{\beta}_j^{\text{lasso}}$ either has the same sign as $\hat{\beta}_j$ or equals 0.

Like ridge, lasso is defined even when $p + 1 > N$.

The equivalent solution for ridge in the orthonormal case is

$$\hat{\beta}_j^{\text{ridge}} = \hat{\beta}_j/(1+\lambda). \tag{35}$$

So it always has the same sign as $\hat{\beta}_j$ and never equals 0.

Like ridge regression, the lasso has a Bayesian interpretation. The prior for β_j is a Laplace (double exponential) distribution:

$$p(\beta_j) = \frac{\lambda}{2}\exp\bigl(-\lambda|\beta_j|\bigr). \tag{36}$$

The lasso has been generalized in many ways. One important one is the **elastic net**, which is discussed below.

ESL-fig3.10.pdf plots the profiles of lasso coefficients against s — see (33) — for the prostate cancer data.

This may be compared with ESL-fig3.08.pdf, which plots the profiles of ridge coefficients against $\text{df}(\lambda)$ for the same data.

The lasso shrinkage causes the estimates of the non-zero coefficients to be biased towards zero and inconsistent.

This bias can be reduced in at least three ways.

- Use the lasso to identify the set of predictors with non-zero coefficients. Then run another regression on just that set of predictors. This is called **post-lasso** estimation. Its properties were studied in various Chernozhukov et al. papers.
- Use the lasso twice. This is called the **relaxed lasso**. In the second step, include only the variables with non-zero coefficients in the first step.

  If cross-validation is used at each step, the penalty parameter λ in the first step is likely to be larger than in the second step, because there are more noise variables to eliminate in the first step.

  Since bias increases with λ, there should be less bias after the second step.
- Use the **adaptive lasso** (Zou, 2006). It replaces the penalty in (31) with

$$\lambda \sum_{j=1}^{p} w_j |\beta_j|, \quad w_j = 1/|\tilde{\beta}_j|^{\nu}, \quad \nu > 0. \tag{37}$$

Here the $\tilde{\beta}_j$ are **pilot estimates**, perhaps OLS estimates if $p << N$ or univariate regression coefficients otherwise.

The penalty on small coefficients is larger than the penalty on large ones. Thus the former tend to get driven to zero, and the latter get shrunk less than ordinary lasso would do.

3.7. Properties of the Lasso

The lasso objective function is a special case of

$$\frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|^q. \tag{38}$$

For lasso, $q = 1$, and for ridge, $q = 2$. When $q = 0$, we have best-subset regression.

The smallest value of q for which the constraint set is convex is $q = 1$. Convexity makes optimization much easier.

The maximum number of non-zero coefficients is $\min(N, p)$.

Various consistency results have been proved for the lasso. The true parameter vector needs to be sparse relative to $N/\log(p)$, so it must contain a lot of zeros. See SLS, Chapter 11.

The obvious way to obtain standard errors for lasso is to use the pairs bootstrap. Resample from $(y_i, \boldsymbol{x}_i)$ pairs and apply the procedure to each bootstrap sample.

We can use the standard deviations of the bootstrap lasso estimates around their mean as standard errors.

We can also calculate the proportion of the time that each parameter estimate is non-zero.

Better inference methods have been developed in papers by Chernozhukov and others. These include **double selection** and **double machine learning**. However, they work only for coefficients on regressors that are *known* to be in the model.

3.8. Least Angle Regression

Least angle regression (Efron, Hastie, Johnstone, and Tibshirani, 2004) can be thought of as an alternative to stepwise regression, but it also provides an efficient way to compute the lasso.

Before doing anything else, standardize all predictors to have mean 0 and unit norm, so that $||\boldsymbol{x}_j|| = 1$.

1. Let $\boldsymbol{r} = \boldsymbol{y} - \bar{\boldsymbol{y}}$.
2. Find the predictor $\boldsymbol{x}_j$ most correlated with $\boldsymbol{r}$.
3. Move β_j from 0 towards $\boldsymbol{x}_j^\top \boldsymbol{r} = (\boldsymbol{x}_j^\top \boldsymbol{x}_j)^{-1} \boldsymbol{x}_j^\top \boldsymbol{r}$ until some other $\boldsymbol{x}_k$ has as much correlation with the current residual as does $\boldsymbol{x}_j$.
4. Move both β_j and β_k together in the direction of the least squares coefficient of the current residual on $[\boldsymbol{x}_j, \boldsymbol{x}_k]$ until some other $\boldsymbol{x}_\ell$ has as much correlation with the current residual.
5. Continue in this way until all p predictors have been included in the regression. After $\min(N-1, p)$ steps, we arrive at the least squares solution.

The direction of the active step after variable k is entered is

$$\boldsymbol{\delta}_k = (\boldsymbol{X}_{[k]}^\top \boldsymbol{X}_{[k]})^{-1} \boldsymbol{X}_{[k]} (\boldsymbol{y} - \boldsymbol{X}_{[k]} \boldsymbol{\beta}_{[k]}). \tag{39}$$

Here $\boldsymbol{X}_{[k]}$ denotes the columns of $\boldsymbol{X}$ that are now included, and $\boldsymbol{\beta}_{[k]}$ denotes the coefficients, one of which is 0.

The coefficients then evolve as

$$\boldsymbol{\beta}_{[k]}(\alpha) = \boldsymbol{\beta}_{[k]} + \alpha \boldsymbol{\delta}_k, \tag{40}$$

and the fit vector evolves as

$$\hat{\boldsymbol{f}}_k(\alpha) = \hat{\boldsymbol{f}}_k + \boldsymbol{X}_{[k]} \boldsymbol{\delta}_k. \tag{41}$$

The exact step length and the identity of the next variable to add can be calculated analytically, but ESL does not give details.

If we start at $t = 0$, the lasso coefficient(s) look just like the LAR ones until one of the non-zero lasso coefficients hits zero again.

Therefore, a modified version of the LAR algorithm can be used to compute lasso for all values of t or λ.

Add an additional step between 4. and 5.

4a. If a non-zero coefficient hits 0, drop its variable from the active set and recompute the current least-squares directions.

Note that dropped variables may reappear later.

LAR always takes p steps to obtain the OLS estimates. The lasso version may take more than that, but it tends to be similar.

This procedure works even when $p >> N$. It provides a whole sequence of lasso estimates for $0 \leq \lambda < \infty$.

Chapter 5 of SLS discusses optimization methods for lasso and other convex optimization problems in much greater detail. It suggests that, although LAR is theoretically attractive, it is not the fastest method.

3.9. Pathwise Coordinate Optimization

We can rewrite (31) as

$$\frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{k\neq j}\tilde{\beta}_k(\lambda)x_{ik} - \beta_j x_{ij}\right)^2 + \lambda\sum_{k\neq j}|\tilde{\beta}_k(\lambda)| + \lambda|\beta_j|, \qquad (42)$$

where $\tilde{\beta}_k(\lambda)$ denotes the current estimate of β_k at penalty parameter λ.

Minimizing (42) is a one-dimensional problem. It is effectively a univariate lasso problem with response variable

$$\tilde{u}_i^{(j)} = y_i - \sum_{k \neq j} \tilde{\beta}_k(\lambda) x_{ik}. \tag{43}$$

The solution to this problem is

$$\tilde{\beta}_j(\lambda) = \text{sign}(t)\big(|t| - \lambda\big)_+, \quad t = \sum_{i=1}^{N} x_{ij} \tilde{u}_i^{(j)}; \tag{44}$$

recall (34).

Repeated iteration of (43) and (44) over all j yields the lasso estimate $\hat{\boldsymbol{\beta}}(\lambda)$; see Friedman, Hastie, Hoefling, and Tibshirani (2007). This approach is an example of **coordinate descent**.

Pathwise coordinate optimization is extremely simple. But one disadvantage, relative to LAR, is that it just yields estimates for a single value of λ.

However, estimates for a large value of λ can be used as starting points for a smaller value, and so on. When λ is sufficiently large, $\hat{\boldsymbol{\beta}}^{\text{lasso}} = \mathbf{0}$. We can start at the smallest value of λ for which that is true.

We still only get results for a grid of values, rather than all values as with LAR, but that is often sufficient.

Observe that the actual computation of the $\tilde{u}_i^{j)}$ in (43) and of $\tilde{\beta}_j(\lambda)$ in (44) are extremely simple. Moreover, in many cases, when $\tilde{\beta}_j(\lambda) = 0$, it just stays at 0.

SLS reports simulation results which suggest that coordinate descent is the fastest way to obtain lasso estimates, both when $N >> p$ and when $p >> N$.

3.10. Elastic-Net Penalization

The elastic-net penalty (Zou and Hastie, 2005) has the form

$$\sum_{j=1}^{p} \big(\alpha|\beta_j| + (1-\alpha)\beta_j^2\big). \tag{45}$$

This reduces to the lasso penalty when $\alpha = 1$ and the ridge penalty when $\alpha = 0$.

As usual, expression (45) is either multiplied by a tuning parameter λ and added to the sum of squares, or it is required to be less than an upper limit t.

Since there are two tuning parameters, cross-validation is a lot more work than for lasso or ridge. To reduce effort, people often just choose α and condition on it.

Elastic-net tends to yield more non-zero coefficients than lasso, but they are shrunk more towards zero.

Unlike lasso, it can yield more than N non-zero coefficients when $p > N$.

We can use any program for lasso to compute the elastic net estimator by using the augmentation trick discussed above to compute ridge regression.

Conversely, if we have a program for elastic net (such as `elasticnet` in Stata 16), we can use it to obtain ridge estimates by setting $\alpha = 0$.

If we multiply (45) by λ and then separate the two penalties, the ridge one becomes

$$(1 - \alpha)\lambda \sum_{j=1}^{p} \beta_j^2. \tag{46}$$

As in (14), we can form the augmented dataset

$$\boldsymbol{X}_a = \begin{bmatrix} \boldsymbol{X} \\ \sqrt{(1-\alpha)\lambda}\mathbf{I}_p \end{bmatrix} \quad \text{and} \quad \boldsymbol{y}_a = \begin{bmatrix} \boldsymbol{y} \\ \mathbf{0} \end{bmatrix}. \tag{47}$$

Then we can obtain the elastic-net estimates by solving the lasso problem

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\frac{1}{2}(\boldsymbol{y}_a - \boldsymbol{X}_a\boldsymbol{\beta})^\top(\boldsymbol{y}_a - \boldsymbol{X}_a\boldsymbol{\beta}) + \alpha\lambda \sum_{j=1}^{p} |\beta_j| \right). \tag{48}$$

This just depends on one tuning parameter, $\alpha\lambda$. However, since $\boldsymbol{X}_a$ depends on $(1-\alpha)\lambda$, we ultimately have to choose two tuning parameters.