Introduction

We have seen a number of times where we induce sparsity in models using l1 penalties. In this section we give some motivation why this works.

Preamble

Before we begin this section, we introduce subset selection for linear regression models.

Subset Selection

Although the least squares estimate is the linear unbiased estimate with minimum variance, it is possible that a biased estimate will give us a better mean squared error.

Consider a case where the true model is

\[ Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \]

and that \( X_1 \) and \( X_2 \) are almost perfectly correlated (statisticians say \( X_1 \) and \( X_2 \) are co-linear). What happens if we leave \( X_2 \) out?

Then the model is very well approximated by

\[ Y = \beta_0 + (\beta_1 + \beta_2)X_1 + \epsilon \]

and we may get a good estimate of \( Y \) estimating 2 parameters instead of 3. Our estimate will be a bit biased but we may lower our variance considerably creating an estimate with smaller expected prediction error than the least squares estimate.

We won’t be able to interpret the estimated parameter, but our prediction may be good.

In subset selection regression we select a number of covariates to include in the model. Then we look at all possible combinations of covariates and pick the one with the smallest RSS.

Consider the prostate cancer data set presented in the HTF book, available in the ElemStatLearn R package (Figure ??). Notice that residual sum of squares consistently drops for the training set as larger models are used, whereas smaller models tend to do better for test set residual sum of squares.

For a given number of predictors, how do we find the model that gives the smallest RSS? There are algorithms that do this, but you do not really want to use this.
Figure 1: Prediction error (RSS) for all possible subset models for training and test sets for prostate cancer data. The solid lines denote the minimums.
Sidebar: Ridge Regression

By only considering some of the covariates we were able to improve our prediction error. However, the choice of one covariate over another can sometimes be a very arbitrary decision as including either works well but both together do no work as well (this happens often with correlated predictors).

Notice that in subset-selection for linear regression, we are estimating models of the form \( \hat{f}_\lambda(x) = x'\beta \) where \( \beta \) is constrained to have exactly (say \( \lambda \)) non-zero \( \beta \)s. The selection problem is, having chosen \( \lambda \), select which \( p - \lambda \) coefficients \( \beta \) will be exactly zero.

Thus, we can think of the subset selection procedure as one that *shrinks* some of the coefficients to 0. But what if we do this in a smoother way? Instead of either keeping it (multiply by 1) or not (multiply by 0), let’s permit smoother shrinkage (multiply by a number between 0 and 1).

For ridge regression instead of minimizing least squares we *penalize* for having too many \( \beta \) that are big by considering the following minimization criteria:

\[
\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2.
\]

We will denote the parameter vector that minimizes this \( \hat{\beta}_{\text{ridge}} \). Here *complexity parameter* \( \lambda \) is a penalty. We saw a similar penalty parameter in tree-based methods.

One can demonstrate mathematically that minimizing the above expression is equivalent to minimizing the regular RSS

\[
\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq s
\]

where \( s \) is inversely proportional to lambda.

Notice that when \( \lambda \) is 0, we get the least squares estimate. However, as \( \lambda \) gets bigger, over fitting gets more expensive as larger values of \( \beta \) penalize the criterion more. The smallest penalty occurs when all the \( \beta \)s are 0. This gives us an estimate with small variance but likely large bias.

Although this problems looks complicated it turns out the resulting predictor is a linear estimate!

One can show that the solution is (in linear algebra notation)

\[
\hat{\beta}_{\text{ridge}} = (X'X + \lambda I)^{-1}X'y
\]

As with subset-selection, we can write the estimated \( \hat{f}_\lambda = S_\lambda y \), using again a *hat* matrix. Later on we’ll start calling these *smoother* matrices.

In Figure 3 we see the RSS in a test and training set for the prostate cancer data for various values of \( \lambda \).

As expected the RSS in the training set is best when \( \lambda = 0 \) (no shrinkage, nothing stopping us from overfitting). However, for the training set the smallest RSS occurs for \( \lambda \approx 5 \)

The least squares estimates are given below. Notice age has a significant protective effect. This is at odds with out intuition.

|           | Est  | SEt  | Pr(>|t|)   |
|-----------|------|------|------------|
| (Intercept) | -0.10| 1.42 | 0.9434     |
| lcavol     | 0.59 | 0.10 | 9.58e-07 *** |
| lweight    | 0.73 | 0.28 | 0.0160 *   |
| age        | -0.03| 0.01 | 0.0257 *   |
Figure 2: Prediction error (RSS) for ridge regression with varying penalty parameters
Ridge regression shrinks the regression coefficients toward 0. Notice what happens to these coefficients as $\lambda$ grows. Notice in particular what happens to age.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Coefficient 1</th>
<th>Coefficient 2</th>
<th>Coefficient 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbph</td>
<td>0.18</td>
<td>0.07</td>
<td>0.0244*</td>
</tr>
<tr>
<td>svi</td>
<td>0.50</td>
<td>0.32</td>
<td>0.1329</td>
</tr>
<tr>
<td>lcp</td>
<td>-0.16</td>
<td>0.10</td>
<td>0.1299</td>
</tr>
<tr>
<td>gleason</td>
<td>0.07</td>
<td>0.17</td>
<td>0.6983</td>
</tr>
<tr>
<td>pgg45</td>
<td>0.01</td>
<td>0.004</td>
<td>0.1199</td>
</tr>
</tbody>
</table>

**Lasso**

The lasso’s definition is similar to that of ridge regression. However, we obtain very different results.
\[
\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j)^2 \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq s
\]

Like ridge regression it can be written as a penalized loss problem (also known as *lagrangian form*):

\[
\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|
\]

Unlike ridge regression, the lasso estimate \( \hat{\beta}_{\text{lasso}} \) does not have a closed-form solution. However, there has been an explosion of algorithms to efficiently solve the lasso problem for very large datasets (especially useful when \( n \ll p \)).

In practice one sees that the lasso makes more coefficients 0. This is sometimes nicer for interpret-ability. See the book and papers on lasso for more information. Figure ?? shows the path coefficients take as \( \lambda \) goes from infinity to zero. Notice coefficients move from *exactly* zero to non-zero for some \( \lambda \) value, and continue to grow as \( \lambda \) increases.

Since we don’t have a nice closed form solution, cross-validation has to be done directly. Figure ?? shows the estimated expected prediction error from 10-fold cross validation.

**SVD, PCA and ridge regression**

First, let’s revisit the geometry of linear regression as this will help us understand some of the issues in high-dimensional problems.

Consider data matrix \( \mathbf{X} \). A useful tool to visualize and predict is to look at the *principal components* of the variables in \( \mathbf{X} \).

Assume the \( \mathbf{X} \) are centered, i.e. \( \mathbf{X} \overline{1} = 0 \overline{1} \), the sample covariance matrix is given by \( \mathbf{XX}/N \) and \( \mathbf{X}'\mathbf{X} \) can be written as

\[
\mathbf{X}'\mathbf{X} = \mathbf{VDV}'
\]

Technical note: this is the eigen decomposition of \( \mathbf{X}'\mathbf{X} \).

The \( v_j \)s are called the eigen-vectors of the sample covariance matrix and also the principal component directions of \( \mathbf{X} \). Figure ?? shows a scatterplot of \( \mathbf{X} \) and the directions as red (solid) and blue (dashed) lines.

The first principal component \( \mathbf{z}_1 = \mathbf{X}v_1 \) has the property that it has the largest sample covariance among all normalized (coefficients squared add up to 1) linear combinations of \( \mathbf{X} \). The sample variance is \( d_1^2/N \).

The derived variable \( \mathbf{z}_1 = \mathbf{X}v_1 = \mathbf{u}_1d_1 \) is called the first principal component. Similarly \( \mathbf{z}_j = \mathbf{X}v_1 \) is called the \( j \)th principal component. \( \mathbf{XV} = \mathbf{UD} \) is a matrix with principal components in the columns. Figure ?? shows these.

A related decomposition is the singular value decomposition (SVD) of the centered input matrix \( \mathbf{X} \). This decomposition is extremely useful in many statistical analysis methods. We will see it again later.

The SVD of an \( N \times p \) matrix \( \mathbf{X} \) is

\[
\mathbf{X} = \mathbf{UDV}'
\]

with \( \mathbf{U} \) \( N \times p \) and \( p \times p \) orthogonal matrices and \( \mathbf{D} \) a \( p \times p \) diagonal matrix with entries \( d_1 \geq d_2 \geq \ldots d_p \geq 0 \) called the singular values of \( \mathbf{X} \).

Technical Note: \( \mathbf{U} \) is an orthogonal basis for the space defined by the columns of \( \mathbf{X} \) and \( \mathbf{V} \) is an orthogonal basis for the space defined by the rows of \( \mathbf{X} \).
Figure 4: Path of coefficients in lasso estimate, prostate cancer data
Figure 5: 10-fold cross-validation estimate of prediction error for lasso in prostate cancer data
Figure 6: Plot of two predictors, $X_2$ versus $X_1$, and the principal component directions.
Figure 7: Plot of two principal components of $X$. 

![Plot of two principal components of $X$.](image-url)
We can show that the least squares predictor for linear regression is

\[ \hat{y} = X\hat{\beta}_{ls} = X(X'X)^{-1}X'y = UU'y \]

Technical Note: \( U'y \) are the coordinates of \( y \) with respect to the orthogonal basis \( U \)

Recall the ridge regression problem, written in Lagrangian form:

\[
\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \beta'x_i) + \lambda \sum_{j=1}^{p} \beta_j^2
\]

The ridge solution can be expressed as

\[
X\hat{\beta}_{ridge} = X(X'X + \lambda I)^{-1}X'y = UD(D^2 + \lambda I)^{-1}DU'y
\]

\[ = \sum_{j=1}^{p} u_j \frac{d_j}{d_j + \lambda} u'_j y \]

Notice that because \( \lambda > 0 \), \( \frac{d_j}{d_j + \lambda} \leq 1 \). Like linear regression, ridge regression computes the coordinates of \( y \) with respect to the orthogonal basis \( U \). It then shrinks these coordinates by the factors \( \frac{d_j}{d_j + \lambda} \). This means that a greater amount of shrinkage occurs when \( \lambda \) is big and for smaller \( d_j \)s.

We now see that ridge regression shrinks coefficients related to principal components with small variance. This makes sense because we have less information about them.

In the case of Figure ??, we can think of it as weight and height, we are saying predict with the sum and ignore the difference. In this case, the sum give much more info than the difference.

**Ridge and Lasso comparison**

Suppose data matrix \( X \) is orthonormal, i.e. the predictors are not correlated and in, loosely, in the same scale. Then from the above we can see what best subset regression, ridge regression and the lasso do to the least-squares estimate \( \hat{\beta}_j \) (recall that in this case the least squares estimate are “decoupled” and are simple projections of the outcome onto each predictor).

Ridge regression *shrinks* the estimate as we saw above: \( \hat{\beta}^{ridge}_j = \frac{\hat{\beta}_j}{1 + \lambda} \). The best subset (of size \( M \)) includes predictor \( j \) if it’s least squares estimate \( \hat{\beta}_j \) is one of the \( M \) largest in absolute value. That is, the best subset estimate is \( \hat{\beta}_j \cdot I\{\text{rank}(|\beta_j|) \leq M}\).

Recall the lasso in Lagrangian form:

\[
\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - x'_i\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|
\]

The lasso does a soft version of best subset: \( \hat{\beta}_{lasso} = \text{sign}(\hat{\beta}_j)(\hat{\beta}_j - \lambda)_+ \), where \( (x)_+ \) is the *positive part* of \( x \). Whereas best subset does *hard thresholding*, the lasso does *soft thresholding*. Figure ?? illustrates these transformations on the least square estimate.
Coordinate descent

The above observation leads to a highly efficient algorithm to solving the lasso regression problem. Suppose you have current guesses for parameters $\hat{\beta}_0$ and $\hat{\beta}_1, \ldots, \hat{\beta}_{p-1}$, and want to minimize the lasso objective with respect to parameter $\beta_p$. The solution in this case is given by

$$\hat{\beta}_p = \text{sign}(z_p)(|z_p| - \lambda)_+$$

where $z_p$ is the least-squares fit of $X_p$ to the partial residual

$$r_{pi} = y_i - \hat{\beta}_0 - \sum_{j=1}^{p-1} \hat{\beta}_j x_{ij}.$$

Cycling through covariates until convergence gives a solution to the lasso problem.

Penalized Classification

Logistic regression

Assume

$$\log \frac{Pr(G = 1|X = x)}{Pr(G = K|X = x)} = \beta_{10} + \beta_1' x$$

$$\log \frac{Pr(G = 2|X = x)}{Pr(G = K|X = x)} = \beta_{20} + \beta_2' x$$

$$\vdots$$

$$\log \frac{Pr(G = K - 1|X = x)}{Pr(G = K|X = x)} = \beta_{(K-1)0} + \beta_{K-1}' x.$$
Notice \( g(p) = \log \frac{p}{1-p} \) is the logistic link and is \( g : (0,1) \to \mathbb{R} \).

A simple calculation gives

\[
Pr(G = k | X = x) = \frac{\exp(\beta_k + \beta'_k x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_l + \beta'_l x)}, \quad k = 1, \ldots, K - 1,
\]

\[
Pr(G = K | X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_l + \beta'_l x)}
\]

When \( K = 2 \) this has a very simple form (only one set of covariates) and is a very popular model used in biostatistical applications.

With this probability model we can now write the log-likelihood

\[
l(\theta) = \sum_{i=1}^{N} \log p_g(x_i; \theta)
\]

where \( p_g(x_i; \theta) = Pr(G = k | X = x; \theta) \). In the two-class case, use \( y_i = 1 \) for \( g_i = 1 \) and \( y_i = 0 \) for \( g_i = 2 \); let \( p_1(x_i; \theta) = p(x_i; \theta) \), so \( p_2(x_i; \theta) = 1 - p(x_i; \theta) \). The log-likelihood is then

\[
l(\beta) = \sum_{i=1}^{N} \left\{ y_i \log p(x_i; \beta) + (1 - y_i) \log (1 - p(x_i; \beta)) \right\}
\]

\[
= \sum_{i=1}^{N} \left\{ y_i \beta'_i x_i - \log(1 + e^{\beta'_i x_i}) \right\}
\]

Our estimate of \( \beta \) will be the maximum likelihood estimate (MLE), obtained by maximizing the log-likelihood with respect to \( \beta \).

We do this by setting the partial derivatives of the log-likelihood to zero:

\[
\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i; \beta)) = 0
\]

This results in a nonlinear system of \( p + 1 \) equations. These are also called the score equations. Notice that for the intercept \( (x_0 = 1) \), its score equation \( \sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p(x_i; \beta) \) states that for \( \beta \) to be an MLE solution, the expected number of observations in class 1, must match the observed number of observations.

To solve the set of score equations, we can use the Newton-Raphson method, which starting from an initial guess \( \beta_{old} \) iteratively updates the estimate using:

\[
\beta_{new} \leftarrow \beta_{old} - \left( \frac{\partial^2 l(\beta)}{\partial \beta \partial \beta'} \right)^{-1} \frac{\partial l(\beta)}{\partial \beta},
\]

with derivatives evaluated at \( \beta_{old} \). The matrix of second derivatives (Hessian matrix) is given by

\[
\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta'} = -\sum_{i=1}^{N} x_i x'_i p(x_i; \beta)(1 - p(x_i; \beta)).
\]

By writing the gradient and Hessian in matrix notation, we can see a neat by-product of using the Newton method in this case. The gradient and Hessian are given by
\[
\frac{\partial l(\beta)}{\partial \beta} = X'(y - p)
\]
\[
\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta'} = -X'WX
\]

where vector \( y \) is the vector of \( y_i \) values, \( X \) the \( N \times (p + 1) \) matrix of \( x_i \) values, \( p \) the vector of fitted probabilities and \( W \) a \( N \times N \) diagonal matrix with \( i \)th entry \( p(x_i; \beta)(1 - p(x_i; \beta)) \).

With this we can rewrite the Newton update as

\[
\beta^{\text{new}} = \beta^{\text{old}} + (X'WX)^{-1}X'(y - p)
\]

\[
= (X'WX)^{-1}X'W(X\beta^{\text{old}} + W^{-1}(y - p))
\]

Introducing notation \( z = X\beta^{\text{old}} + W^{-1}(y - p) \), we can see that the Newton update is the solution to a weighted least squares problem

\[
\beta^{\text{new}} \leftarrow \arg\min_{\beta} (z - X\beta)'W(z - X\beta)
\]

**Back to diagonal LDA**

There is an elegant connection between the lasso/ridge regression comparison just discussed and the shrunk centroids method (PAM). Diagonal LDA can be motivated using a slightly different approach as a Naive Bayes classifier, as pictured here:

![Figure 9: A naive Bayes classifier](image)

In this case \( g \) represents class, and \( x_j \) is expression for gene \( j \). The naive Bayes classifier assumes that \( x_j | g = k \sim N(\mu x_{kj}, \sigma^2) \), that is, expression is independent and normally distributed in each class. Given a training set \( \{(x_1, g_1), \ldots, (x_n, g_n)\} \), parameters for the naive Bayes model can be estimated by maximizing log-likelihood:

\[
\max \sum_{j} \log \{ P(G = g_i | x_j) \}.
\]

The estimates given above for diagonal LDA are exactly the maximizers of this optimization problem.
Now, suppose we re-parameterize the class means for each gene as \( \mu_{x,kj} = \mu_j + \mu_{kj} \) where \( \mu_j \) is the overall mean, and regularize the solution to the naive Bayes/diagonal LDA problem using an \( l_1 \)-penalty as follows:

\[
\min_{\mu_j, \mu_{kj}} \left\{ \frac{1}{2} \sum_{j=1}^{p} \sum_{k=1}^{K} \sum_{i \in C_k} \frac{(x_{ij} - \mu_j - \mu_{kj})^2}{s_j^2} + \lambda \sum_{j=1}^{K} \sqrt{N_k} |\mu_{kj}| s_j \right\},
\]

You can show that the solution to this regularized optimization problem is the nearest-shrunken centroid. That is, PAM, or the nearest-shrunken centroid method, is the solution to an \( l_1 \)-regularized naiveBayes classifier, where deviation of class means from the overall mean is penalized.

**sparsePCA**

Recall our discussion of PCA analysis and the singular value decomposition. We described the principal components as eigen-genes or eigen-arrays, where each of these is the result of a linear combination of genes/arrays in our dataset. We saw that it was difficult to ascertain the contribution of each specific gene to principal components (eigen-genes).

We briefly mentioned that the SVD is the solution to an optimization problem based on a low-rank matrix approximation objective:

\[
\max_v v'X'Xv \\
\text{s.t. } v'v = 1
\]

Where vector \( v \) contains the coefficient for each gene used to construct each principal component (eigen-gene). As before, we can use a regularization technique to find a low-rank matrix approximation, where only a few genes are used to construct principal components (eigen-genes), by using an \( l_1 \)-penalty on this optimization problem:

\[
\max_v v'X'Xv + \lambda \sum_j |v_j| \\
\text{s.t. } v'v = 1
\]

This figure shows the two-dimensional scatter plot for regular PCA on the docetaxel sensitivity dataset:

This figure shows the scatter plot for PCA (using only 29 genes).

The following figure plots the coefficients for the first eigen-gene for each method:

Sparse PCA is implemented in R in package **PMA** by Witten and Tibshirani.
Figure 10: PCA scatter plot

Figure 11: Sparse PCA scatter plot
Figure 12: Sparse PCA scatter plot