Rännar et al. (1994) constructed a kernel based on the predictor matrix and response that had dimension $n \times n$. A usual PLS analysis can then be performed using this kernel, the outer products of the predictors, and the outer products of the response (each with dimension $n \times n$). Hence, this algorithm is computationally more efficient when there are more predictors than samples.

As noted in Fig. 6.9, the PLS components summarize the data through linear substructures (i.e., hyperplanes) of the original predictor space that are related to the response. But for many problems, the underlying structure in the predictor space that is optimally related to the response is not linear but curvilinear or nonlinear. Several authors have attempted to address this shortcoming of PLS in order to find this type of predictor space/response relationship. While many methods exist, the most easily adaptable approaches using the algorithms explained above are provided by Berglund and Wold (1997) and Berglund et al. (2001). In Berglund and Wold (1997), the authors show that adding squared predictors (and cubic, if necessary) can be included with the original predictors. PLS is then applied to the augmented data set. The authors also show that there is no need to add cross-product terms, thus greatly reducing the number of new predictors added to the original data. Subsequently, Berglund et al. (2001) employ the use of the GIFI approach (Michailidis and de Leeuw 1998) which splits each predictor into two or more bins for those predictors that are thought to have a nonlinear relationship with the response. Cut points for the bins are selected by the user and are based on either prior knowledge or characteristics of the data. The original predictors that were binned are then excluded from the data set that includes the binned versions of the predictors. PLS is then applied to the new predictor set in usual way.

Both of these approaches have successfully found nonlinear relationships between the predictors and the response. But there can be a considerable amount of effort required in constructing the data sets for input to PLS, especially as the number of predictors becomes large. As we will show in subsequent sections, other predictive modeling techniques can more naturally identify nonlinear structures between predictors and the response without having to modify the predictor space. Therefore, if a more intricate relationship between predictors and response exists, then we suggest employing one of the other techniques rather than trying to improve the performance of PLS through this type of augmentation.

6.4 Penalized Models

Under standard assumptions, the coefficients produced by ordinary least squares regression are unbiased and, of all unbiased linear techniques, this model also has the lowest variance. However, given that the MSE is a

combination of variance and bias (Sect. 5.2), it is very possible to produce models with smaller MSEs by allowing the parameter estimates to be biased. It is common that a small increase in bias can produce a substantial drop in the variance and thus a smaller MSE than ordinary least squares regression coefficients. One consequence of large correlations between the predictor variances is that the variance can become very large. Combatting collinearity by using biased models may result in regression models where the overall MSE is competitive.

One method of creating biased regression models is to add a penalty to the sum of the squared errors. Recall that original least squares regression found parameter estimates to minimize the sum of the squared errors:

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

When the model over-fits the data, or when there are issues with collinearity (as in Table 6.1), the linear regression parameter estimates may become inflated. As such, we may want to control the magnitude of these estimates to reduce the SSE. Controlling (or *regularizing*) the parameter estimates can be accomplished by adding a penalty to the SSE if the estimates become large. *Ridge regression* (Hoerl 1970) adds a penalty on the sum of the squared regression parameters:

$$\text{SSE}_{L_2} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} \beta_j^2.$$

The "L_2" signifies that a second-order penalty (i.e., the square) is being used on the parameter estimates. The effect of this penalty is that the parameter estimates are only allowed to become large if there is a proportional reduction in SSE. In effect, this method *shrinks* the estimates towards 0 as the λ penalty becomes large (these techniques are sometimes called "shrinkage methods").

By adding the penalty, we are making a trade-off between the model variance and bias. By sacrificing some bias, we can often reduce the variance enough to make the overall MSE lower than unbiased models.

For example, Fig. 6.15 shows the *path* of the regression coefficients for the solubility data over different values of λ. Each line corresponds to a model parameter and the predictors were centered and scaled prior to this analysis so that their units are the same. When there is no penalty, many parameters have reasonable values, such as the predictor for the number of multiple bonds (shown in orange). However, some parameter estimates are abnormally large, such as the number of non-hydrogen atoms (in green) and the number of non-hydrogen bonds (purple) previously singled out in Table 6.1. These large values are indicative of collinearity issues. As the penalty is increased, the parameter estimates move closer to 0 at different rates. By the time

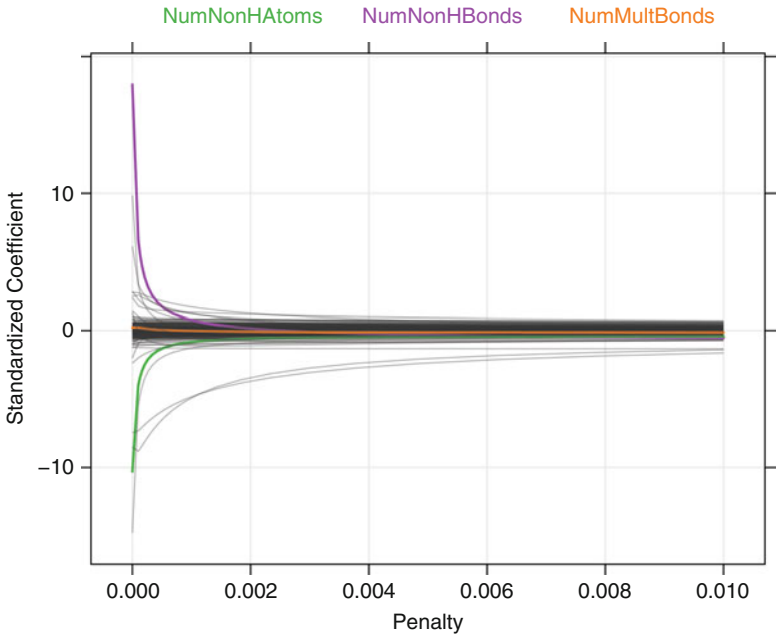

Fig. 6.15: The ridge-regression coefficient path

the penalty has a value of $\lambda = 0.002$, these two predictors are much more well behaved, although other coefficient values are still relatively large in magnitude.

Using cross-validation, the penalty value was optimized. Figure 6.16 shows how the RMSE changes with λ. When there is no penalty, the error is inflated. When the penalty is increased, the error drops from 0.72 to 0.69. As the penalty increases beyond 0.036, the bias becomes to large and the model starts to under-fit, resulting in an increase in MSE.

While ridge regression shrinks the parameter estimates towards 0, the model does not set the values to absolute 0 for any value of the penalty. Even though some parameter estimates become negligibly small, this model does not conduct *feature selection*.

A popular alternative to ridge regression is the *least absolute shrinkage and selection operator* model, frequently called the *lasso* (Tibshirani 1996). This model uses a similar penalty to ridge regression:

$$\text{SSE}_{L_1} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} |\beta_j|.$$

While this may seem like a small modification, the practical implications are significant. While the regression coefficients are still shrunk towards 0,

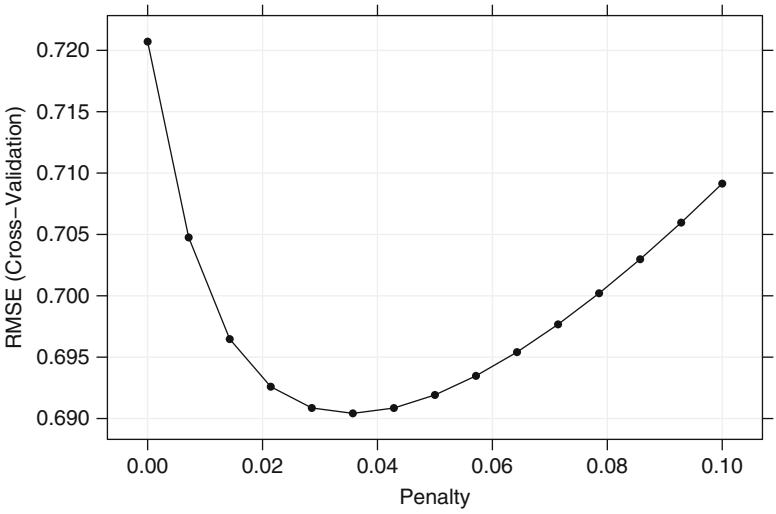

Fig. 6.16: The cross-validation profiles for a ridge regression model

a consequence of penalizing the absolute values is that some parameters are actually set to 0 for some value of λ. Thus the lasso yields models that simultaneously use regularization to improve the model and to conduct feature selection. In comparing, the two types of penalties, Friedman et al. (2010) stated

> "Ridge regression is known to shrink the coefficients of correlated predictors towards each other, allowing them to borrow strength from each other. In the extreme case of k identical predictors, they each get identical coefficients with $1/k$th the size that any single one would get if fit alone.[. . .]
>
> lasso, on the other hand, is somewhat indifferent to very correlated predictors, and will tend to pick one and ignore the rest."

Figure 6.17 shows the paths of the lasso coefficients over different penalty values. The x-axis is the fraction of the full solution (i.e., ordinary least squares with no penalty). Smaller values on the x-axis indicate that a large penalty has been used. When the penalty is large, many of the regression coefficients are set to 0. As the penalty is reduced, many have nonzero coefficients. Examining the trace for the number of non-hydrogen bonds (in purple), the coefficient is initially 0, has a slight increase, then is shrunken towards 0 again. When the fraction is around 0.4, this predictor is entered back into the model with a nonzero coefficient that consistently increases (most likely due to collinearity). Table 6.2 shows regression coefficients for ordinary least squares, PLS, ridge-regression, and the lasso model. The ridge-regression penalty used in this table is 0.036 and the lasso penalty was 0.15. The ridge-regression model shrinks the coefficients for the non-hydrogen atom and non-hydrogen bond predictors significantly towards 0 in comparison to

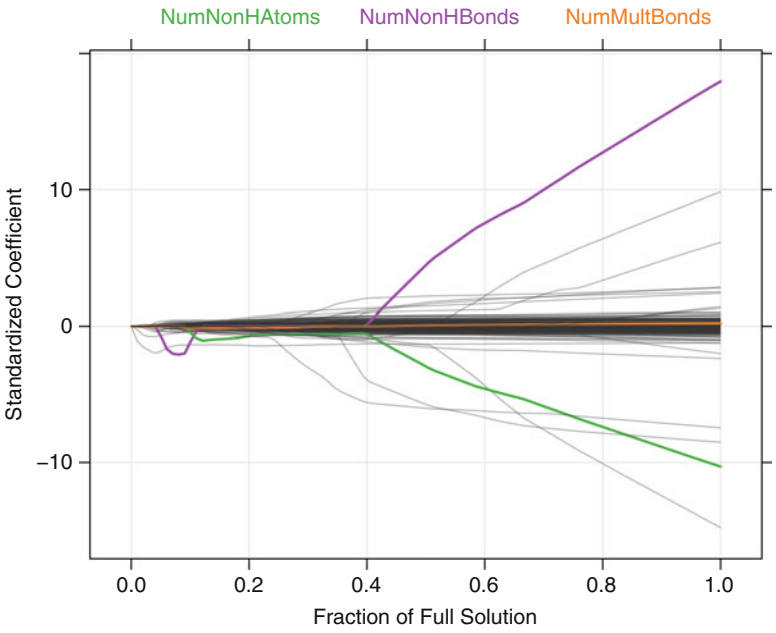

Fig. 6.17: The lasso coefficient path for the solubility data. The x-axis is the fraction of the full least squares solution. As the fraction increases, the lasso penalty (λ) decreases

the ordinary least squares models while the lasso model shrinks the non-hydrogen atom predictor out of the model. Between these models, the lasso model had the smallest cross-validation error of 0.67, slightly better than the PLS model (0.68) and ridge regression (0.69).

This type of regularization has been a very active area of research. The lasso model has been extended to many other techniques, such as linear discriminant analysis (Clemmensen et al. 2011; Witten and Tibshirani 2011), PLS (Chun and Keleş 2010), and PCA (Jolliffe et al. 2003; Zou et al. 2004). A significant advancement for this model was Efron et al. (2004). Their model, least angle regression (LARS), is a broad framework that encompasses the lasso and similar models. The LARS model can be used to fit lasso models more efficiently, especially in high-dimensional problems. Friedman et al. (2010) and Hesterberg et al. (2008) provide a survey of these techniques.

A generalization of the lasso model is the *elastic net* (Zou and Hastie 2005). This model combines the two types of penalties:

$$\text{SSE}_{\text{Enet}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^{P} \beta_j^2 + \lambda_2 \sum_{j=1}^{P} |\beta_j|.$$

Table 6.2: Regression coefficients for two highly correlated predictors for PLS, ridge regression, the elastic net and other models

Model	NumNonHAtoms	NumNonHBonds
NumNonHAtoms only	−1.2 (0.1)	
NumNonHBonds only		−1.2 (0.1)
Both	−0.3 (0.5)	−0.9 (0.5)
All predictors	8.2 (1.4)	−9.1 (1.6)
PLS, all predictors	−0.4	−0.8
Ridge, all predictors	−0.3	−0.3
lasso/elastic net	0.0	−0.8

The ridge penalty used for this table was 0.036 and the lasso penalty was 0.15. The PLS model used ten components.

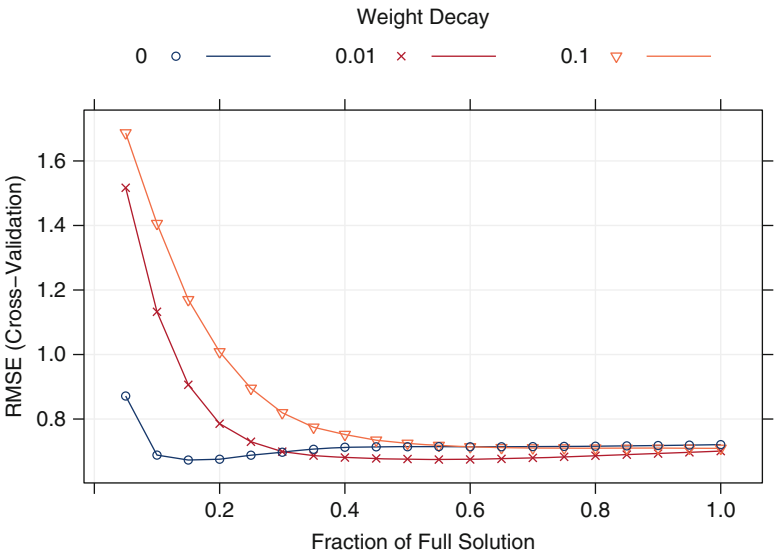

Fig. 6.18: The cross-validation profiles for an elastic net model

The advantage of this model is that it enables effective regularization via the ridge-type penalty with the feature selection quality of the lasso penalty. The Zou and Hastie (2005) suggest that this model will more effectively deal with groups of high correlated predictors.

Both the penalties require tuning to achieve optimal performance. Again, using resampling, this model was tuned for the solubility data. Figure 6.18 shows the performance profiles across three values of the ridge penalty and 20 values of the lasso penalty. The pure lasso model (with $\lambda_1 = 0$) has an initial

drop in the error and then an increase when the fraction is greater than 0.2. The two models with nonzero values of the ridge penalty have minimum errors with a larger model. In the end, the optimal performance was associated with the lasso model with a fraction of 0.15, corresponding to 130 predictors out of a possible 228.

6.5 Computing

The R packages elasticnet, caret, lars, MASS, pls and stats will be referenced.

The solubility data can be obtained from the AppliedPredictiveModeling R package. The predictors for the training and test sets are contained in data frames called `solTrainX` and `solTestX`, respectively. To obtain the data in R,

```
> library(AppliedPredictiveModeling)
> data(solubility)
> ## The data objects begin with "sol":
> ls(pattern = "^solT")
  [1] "solTestX"       "solTestXtrans"  "solTestY"       "solTrainX"
  [5] "solTrainXtrans" "solTrainY"
```

Each column of the data corresponds to a predictor (i.e., chemical descriptor) and the rows correspond to compounds. There are 228 columns in the data. A random sample of column names is

```
> set.seed(2)
> sample(names(solTrainX), 8)
  [1] "FP043"        "FP160"        "FP130"        "FP038"        "NumBonds"
  [6] "NumNonHAtoms" "FP029"        "FP185"
```

The "`FP`" columns correspond to the binary 0/1 fingerprint predictors that are associated with the presence or absence of a particular chemical structure. Alternate versions of these data that have been Box–Cox transformed are contained in the data frames `solTrainXtrans` and `solTestXtrans`. These modified versions were used in the analyses in this and subsequent chapters.

The solubility values for each compound are contained in numeric vectors named `solTrainY` and `solTestY`.

Ordinary Linear Regression

The primary function for creating linear regression models using simple least squares is `lm`. This function takes a formula and data frame as input. Because of this, the training set predictors and outcome should be contained in the same data frame. We can create a new data frame for this purpose: