CS599-Project1

Anuraag Srivastava (as4378)

20th September, 2019

1 Profiles of ridge coefficients for prostate cancer data as lambda value is varied

1.1 Figure

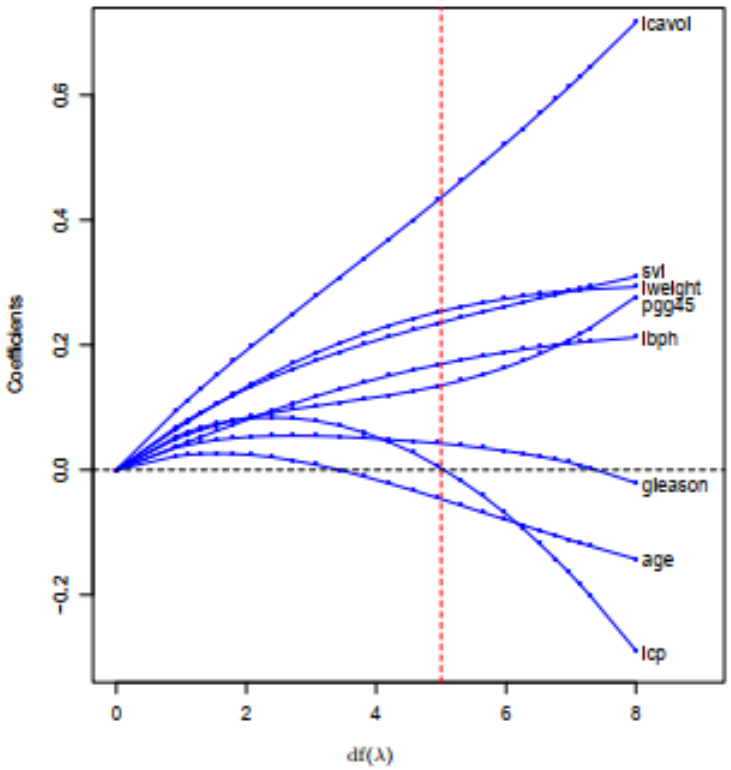

FIGURE 3.8. *Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter λ is varied. Coefficients are plotted versus $\mathrm{df}(\lambda)$, the effective degrees of freedom. A vertical line is drawn at $\mathrm{df} = 5.0$, the value chosen by cross-validation.*

1.2 Citation

Hastie, et al. Elements of Statistical Learning, Figure 3.8.

1.3 Problem Setting

The inputs to the algorithm are the vector of observed values $\mathbf{Y}$, feature matrix $\mathbf{X}$ and a list of tuning parameters or penalties L.

The output would be coefficient values for different features in $\mathbf{X}$ plotted as a function of λ values in $\mathbf{L}$.

The desired function is suppose to perform ridge regression for all the λ values in $\mathbf{L}$ and show the plot of coefficient value of features as the λ values is varied.

1.4 Data Sources

For the project we will use prostate cancer data available in package called faraway in R. This data set was used collected to examine the correlation between the level of prostate-specific antigen and a number of clinical measures in 97 men with prostate cancer who were about to receive a radical prostatectomy.

The prostate data frame has 97 rows and 9 columns. The observed values are in column named **lpsa** (log(prostate specific antigen)) and the rest of the columns are used as features.

The first few observations are as shown:

```
> data(prostate, package="faraway")
> head(prostate)
      lcavol lweight age      lbph svi      lcp gleason pgg45     lpsa
1 -0.5798185  2.7695  50 -1.386294   0 -1.38629       6     0 -0.43078
2 -0.9942523  3.3196  58 -1.386294   0 -1.38629       6     0 -0.16252
3 -0.5108256  2.6912  74 -1.386294   0 -1.38629       7    20 -0.16252
4 -1.2039728  3.2828  58 -1.386294   0 -1.38629       6     0 -0.16252
5  0.7514161  3.4324  62 -1.386294   0 -1.38629       6     0  0.37156
6 -1.0498221  3.2288  50 -1.386294   0 -1.38629       6     0  0.76547
```

1.5 Algorithm

Ridge regression shrinks the regression coefficients by imposing a penalty on their size. The solution to ridge coefficients are found by solving the penalized residual sum of squares as follows:

$$\hat{\beta^{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{i=1}^{p} \beta_j^2 \right. \tag{1}$$

Where, $\lambda \geq 0$ is the complexity parameter which controls the amount of shrinkage and p is the number of predictors or features.

The solution to $\hat{\beta^{ridge}}$ is given as:

$$\hat{\beta^{ridge}} = (X^T X + \lambda I)^{-1} X^T y \tag{2}$$

The algorithm is then to solve the above equation for given λ values and plot the results. The pseudo code below depicts the idea.

1.5.1 Pseudo code

1: Input: feature matrix $\mathbf{X}$, observed output matrix $\mathbf{Y}$ list of tuning parameters L.
2: Output: A figure that plots ridge regression coefficients for different features as tuning parameter is varied.
3: $R \leftarrow$ empty list // List for storing ridge coefficients for different λ values
4: $X^* \leftarrow (X - mean(X))/sd(X)$ // Standardize the inputs
5: for λ in L:
6: $\quad Xt \leftarrow transpose(X^*)$
7: $\quad dim \leftarrow nrows(X^*)$
8: $\quad I \leftarrow diag(dim)$
9: $\quad sum \leftarrow XtX^* + \lambda I$
10: $\quad coefficients \leftarrow inverse(sum) * Xt * Y$
11: $\quad R[\lambda] \leftarrow rbind(coefficients, \lambda)$
12: $plot \leftarrow ggplot(coefficients \sim \lambda, data = R)$
13: print $plot$

This will help me learn about shrinkage methods for variable selection which is useful when dealing with high dimensional data with large number of features as techniques like best subset selection become infeasible.

2 Profiles of lasso coefficients for prostate cancer data as lambda value is varied

2.1 Figure

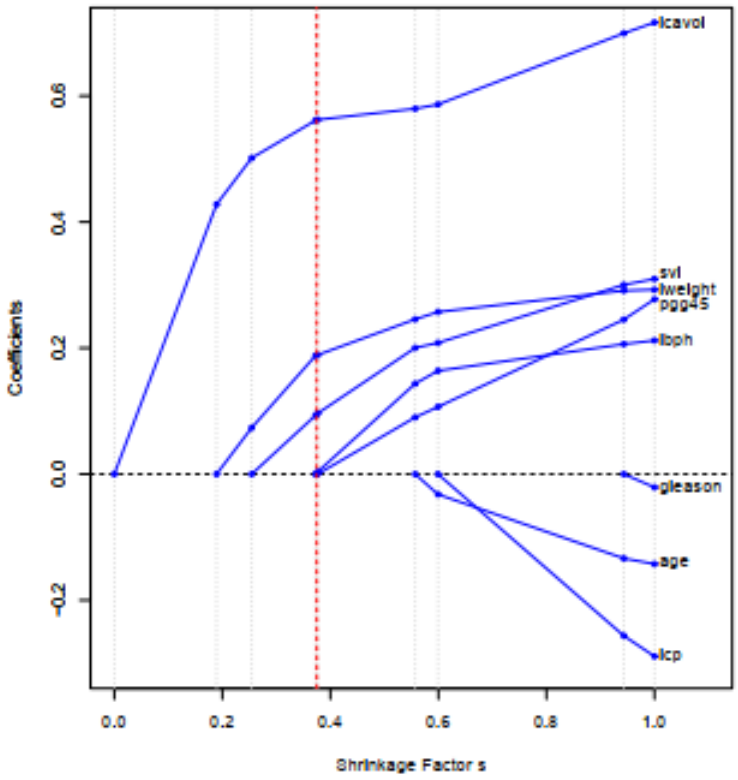

FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus* $s = t/\sum_1^p |\hat{\beta}_j|$. *A vertical line is drawn at* $s = 0.36$, *the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*

2.2 Citation

Hastie, et al. Elements of Statistical Learning, Figure 3.10.

2.3 Problem Setting

The inputs to the algorithm are the vector of observed values $\mathbf{Y}$, feature matrix $\mathbf{X}$ and a list of tuning parameters T.

The output would be coefficient values for different features in $\mathbf{X}$ plotted as a function of shrinkage parameter $s = \frac{t}{\sum_1^p |\hat{\beta}_j|}$ for all t values in $\mathbf{T}$.

The desired function is suppose to perform shrinkage using lasso penalty for all the t values in $\mathbf{T}$ and show the plot of coefficient value of features as the s value is varied.

2.4 Data Sources

For the project we will use prostate cancer data available in package called faraway in R. This data set was used collected to examine the correlation between the level of prostate-specific antigen and a number of clinical measures in 97 men with prostate cancer who were about to receive a radical prostatectomy.

The prostate data frame has 97 rows and 9 columns. The observed values are in column named **lpsa** (log(prostate specific antigen)) and the rest of the columns are used as features.

The first few observations are as shown:

```
> data(prostate, package="faraway")
> head(prostate)
      lcavol lweight age      lbph svi      lcp gleason pgg45     lpsa
1 -0.5798185  2.7695  50 -1.386294   0 -1.38629       6     0 -0.43078
2 -0.9942523  3.3196  58 -1.386294   0 -1.38629       6     0 -0.16252
3 -0.5108256  2.6912  74 -1.386294   0 -1.38629       7    20 -0.16252
4 -1.2039728  3.2828  58 -1.386294   0 -1.38629       6     0 -0.16252
5  0.7514161  3.4324  62 -1.386294   0 -1.38629       6     0  0.37156
6 -1.0498221  3.2288  50 -1.386294   0 -1.38629       6     0  0.76547
```

2.5 Algorithm

Lasso is a shrinkage method like ridge with the following difference:

$$\hat{\beta^{lasso}} = \arg\min_{\beta} \left\{ \frac{1}{2} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{i=1}^{p} |\beta_j| \right. \tag{3}$$

Thus the penalty term is linear in coefficient values as compared to quadratic penalty term in ridge regression.

To implement Lasso we will implement least angle regression (LARS) algorithm described below.

1. standardize the predictors to have mean zero and unit norm. Start with the residual $r = y - \bar{y}, \beta_1 = \ldots \beta_p = 0$

2. Find the predictor x_j most correlated with r

3. Move β_j from 0 towards its least-squares coefficient $dot(X[j,], r)$ until some other competitor $X[k,]$ has as much correlation with the current residual as does $X[j,]$

4. Move $\beta[j]$ and $\beta[k]$ in the direction defined by the joint least squares coefficient of the current residual on $(X[j,], X[k,])$ until some other competitor $X[l]$ has as much correlation with the current residual.

5. Continue in this way until all p predictors have been entered. After $min(N-1, p)$ steps, we arrive at the full least-squares solution.

A more detailed implementation is described in pseudo code below:

2.5.1 Pseudo code

Input: feature matrix $\mathbf{X}$, observed output matrix $\mathbf{Y}$ list of tuning parameters T.
Output: A figure that plots lasso coefficients for different features as shrinkage parameter is varied.
$R \leftarrow$ empty list // List for storing lasso coefficients for different λ values
$N \leftarrow nrow(X)$ //Number of observations
$p \leftarrow dim(X[2])$ //Number of predictors
for t in T:
  $X^* \leftarrow (X - mean(X))/sd(X)$ //standardize the predictors
  $activeSet \leftarrow emptylist$ //Initially no predictor is chosen
  $inactiveSet \leftarrow c(1:p)$ //All the predictors are initially inactive
  $mu \leftarrow zeros(N)$ //regressed version of y, initially 0 as there are no predictors
  $\beta \leftarrow zeros(p+1, p)$
  initially all the coefficients have 0 value
  $corr \leftarrow zeros(p+1, p)$
  initially the correlation will be 0 with residuals
  $D \leftarrow zeros(0, 0)$
  matrix for storing the predictors captured, initially 0 as active set is 0
  $r \leftarrow Y - mean(Y)$
  initial residual
  for k in predictors
    //Find the current correlation
    $c \leftarrow dotProduct(X^*, r)$
    //Store the result in correlation matrix
    $corr \leftarrow c$
    //find the predictor with max correlation to r
    $pmax \leftarrow \arg\max_i \left\{ c[i] \right.$
    // add the found predictor to active set and remove from inactive set
    $rbind(activeSet, pmax)$
    $inactiveSet[-c(pmax),]$
    //add it to captured predictor matrix
    $D.insertColumn(pmax)$
    //get the sign of correlation for current active predictors
    $sign \leftarrow sign(c[activeSet])$
    $sign \leftarrow sign.reshape(length(s), 1)$
    //move in the direction of least squares solution for the active set
    $W \leftarrow solve(R, solve(R^T * sign))$
    //Still working on implementation for following steps
    Find the new correlation with residuals
    Check if a new predictor has greater correlation with current residual
    Add it to $activeSet$, D and remove from $inactiveSet$
  $s \leftarrow t/sum(abs(\beta))$
  $R \leftarrow rbind(\beta, t)$
$plot \leftarrow ggplot(coefficients \sim s, data = R)$
print $plot$

This will again be a useful shrinkage method to learn for high dimensional data. It also has an added advantage of dropping the predictors when their value hits 0.