Penalized Regression

36-290 – Statistical Research Methodology

Week 6 Thursday – Fall 2021

Shrinkage Methods

The lasso and ridge regression are *shrinkage methods* that are alternatives to best subset selection (and to forward or backward stepwise selection). They differ from BSS in that they penalize models in a different manner than BSS does:

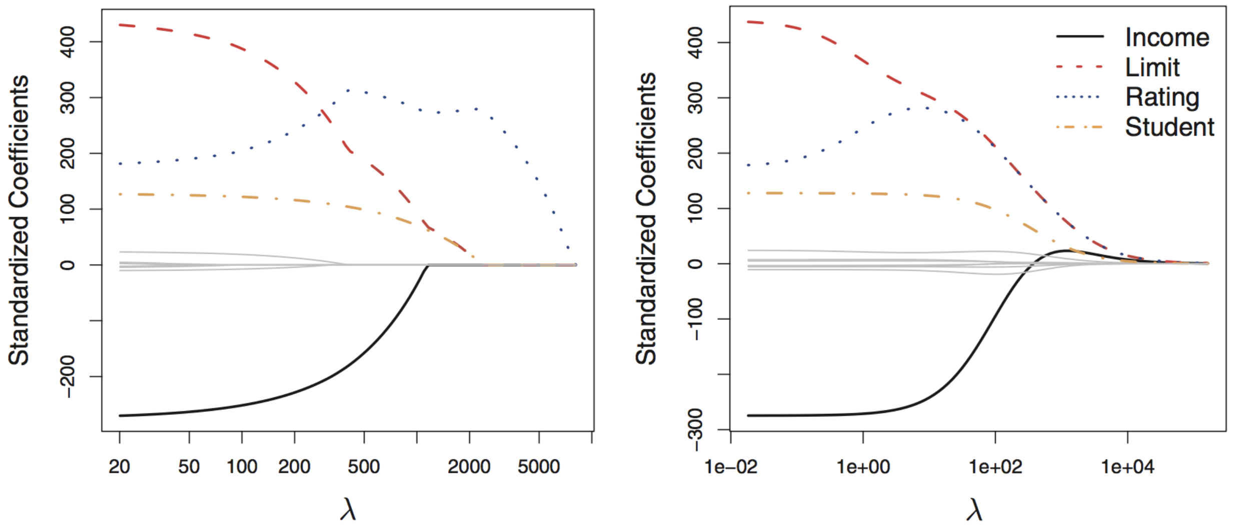
$$\text{lasso}: \ \text{RSS} + \lambda \sum_{i=1}^{p} |\beta_i|$$
$$\text{ridge}: \ \text{RSS} + \lambda \sum_{i=1}^{p} \beta_i^2$$

The effects of the additive penalty (or regularization) terms is dictated by the magnitude of the tuning parameter λ. If $\lambda \rightarrow 0$, then the penalty terms have no effect, and thus the use of lasso and ridge regression is equivalent to simply performing regular old linear regression.

However, if $\lambda \rightarrow \infty$, then...

- lasso: to balance the large value of λ, all the linear regression coefficients shrink toward zero, but some go to zero more quickly than others; hence lasso performs its own version of subset selection
- ridge: to balance the large value of λ, all the coefficients shrink toward zero

Shrinkage Methods



For example: note how the magnitude of the coefficient for `Income` trends as $\lambda \to \infty$.

In the context of lasso, the coefficient goes to zero at $\lambda \sim 1000$; if the optimum value of λ is larger, then `Income` would not be included in the learned model.

In the context of ridge regression, the coefficient shrinks towards zero, but never actually reaches it. `Income` is always a variable in the learned model, regardless of the value of λ.

While it may seem obvious that lasso is the preferred learning model (since it performs variable selection), realize that ridge regression may yield a small test-set MSE! Depending on circumstance, you may prefer to utilize a model that has p predictor variables, but yields better predictions, over a model that has $k < p$ predictor variables, but yields worse predictions.

Shrinkage Methods: Caveats

- If you use either the lasso or ridge regression, you should *standardize* your data. While there is no unique way to standardize, the most common convention is to, within each column of data, compute and subtract off the sample mean, and compute and subtract off the sample standard deviation:

$$\tilde{x}_i = \frac{x_i - \bar{x}_i}{s_{x,i}} \,.$$

(Note that if you utilize the `glmnet` package, standardization is performed by default.)

- λ is a tuning parameter. This means that you have to split your training data into training and validation sets, or perform cross-validation on the training data.

(Note that if you utilize the `glmnet` package, the `cv.glmnet()` package will perform the necessary cross-validation for you.)

Lasso: Regression Example

We will load up a 100-row dataset. `pred.train` and `pred.test` will have 70 and 30 rows respectively; `resp.train` and `resp.test`, 70 and 30 elements. We do this because lasso and ridge regression are most effective when $n \sim p$; applying these methods to a dataset with thousands of rows will not produce any interesting results!

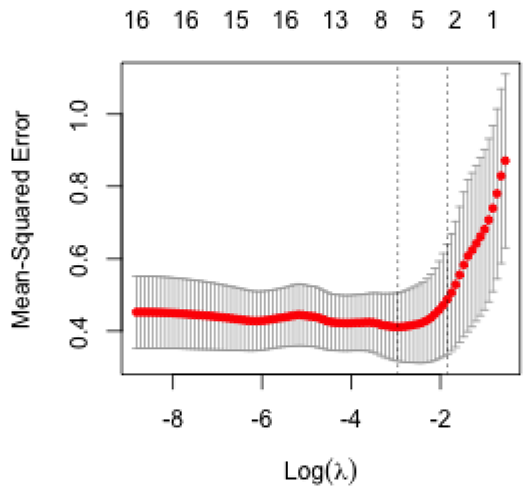
We will utilize the `glmnet` package. The functions of `glmnet` are a tad "prickly" in that they don't abide by the usual rules that apply to model functions in `R`. Note that to perform ridge regression, one can perform all the same steps below, except changing the argument `alpha=1` to `alpha=0`.

```
suppressMessages(library(glmnet))
x = model.matrix(resp.train~.,pred.train)[,-1] ; y = resp.train
out.lasso = glmnet(x,y,alpha=1)  # alpha = 0: ridge regression; alpha = 1: lasso
plot(out.lasso,xvar="lambda")
```

Lasso: Regression Example

What is the optimal value of λ?

```
set.seed(301)  # cv.glmnet() performs random sampling...so set the seed!
cv = cv.glmnet(x,y,alpha=1)
plot(cv)
```



```
cv$lambda.min ; log(cv$lambda.min)
```

```
## [1] 0.05174164
```

```
## [1] -2.961492
```

Lasso: Regression Example

Given the optimal value of λ, what variables are retained?

```
coef(out.lasso,cv$lambda.min)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  1.9837391
## mag.i        0.3386817
## col.Vi       0.1932016
## col.iJ      -1.0565253
## col.JH      -0.2604068
## V.G          1.0365429
## V.M20        .
## V.C          .
## V.size       .
## J.G          0.1771393
## J.M20        .
## J.C          0.0271853
## J.size       .
## H.G          .
## H.M20        .
## H.C          .
## H.size       .
```

We see that 7 of 16 predictor variables are selected.

```
x.test    = model.matrix(resp.test~.,pred.test)[,-1]
resp.pred = predict(out.lasso,s=cv$lambda.min,newx=x.test)
mean((resp.pred-resp.test)^2)
```

```
## [1] 0.1426921
```

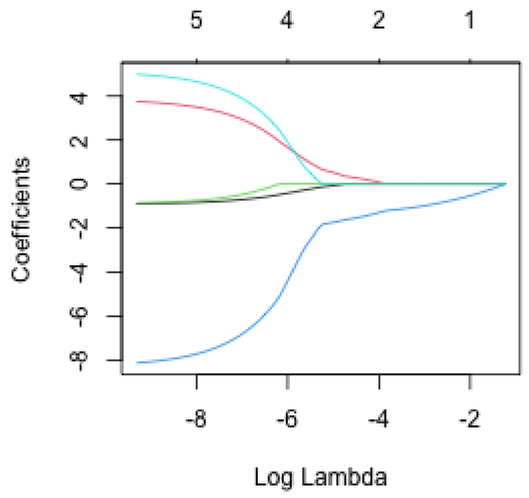
Lasso: Classification Example

The dataset that we read in now contains magnitude and redshift data for 500 quasars and 500 stars. The idea is to learn a classifier that can discriminate between quasars and stars with a low misclassification rate.

```
## [1] 1000    8
```

```
## [1] "u.mag"        "g.mag"        "r.mag"        "i.mag"        "z.mag"        "redshift"     "redshift.err" "clas
```

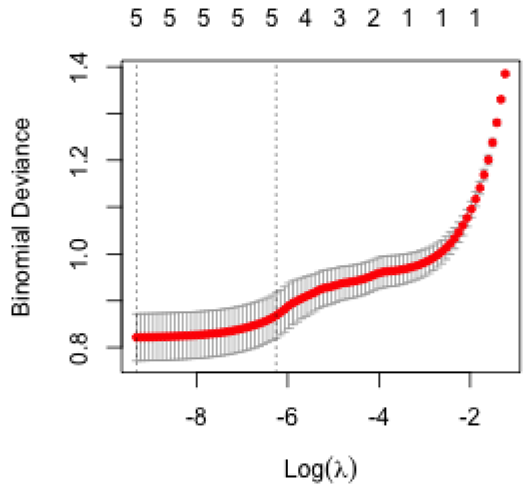
```
x = model.matrix(resp.train~.,pred.train)[,-1]
y = resp.train
out.lasso = glmnet(x,y,alpha=1,family="binomial")  # note the "family" argument
plot(out.lasso,xvar="lambda")
```



Lasso: Classification Example

What is the optimal value of λ? (Here, it is very small…meaning there is no reason to favor lasso over logistic regression. This is not surprising given the relative size of the training set [700] to the number of predictors [5].)

```
set.seed(302)
cv = cv.glmnet(x,y,alpha=1,family="binomial")
plot(cv) ; cv$lambda.min ; log(cv$lambda.min)
```



```
## [1] 8.933663e-05
```

```
## [1] -9.323099
```

Lasso: Classification Example

…and as we can see, no variables were removed from the predictor set.

```
coef(out.lasso,cv$lambda.min)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept) 22.0987059
## u.mag       -0.9004605
## g.mag        3.7503131
## r.mag       -0.8758846
## i.mag       -8.1310335
## z.mag        4.9978805
```

```
x.test    = model.matrix(resp.test~.,pred.test)[,-1]
resp.prob = predict(out.lasso,s=cv$lambda.min,newx=x.test,type="response")
resp.pred = ifelse(resp.prob>0.5,"STAR","QSO")
mean(resp.pred!=resp.test) # basically the same as for logistic regression
```

```
## [1] 0.1433333
```

```
table(resp.pred,resp.test)
```

```
##          resp.test
## resp.pred QSO STAR
##      QSO  147   25
##      STAR  18  110
```