Supervised Learning - Shrinkage Methods

Dr. Sameera Viswakula

June, 2021

Shrinkage methods

As an alternative to selecting a subset of predictors using a least squares fit, we can fit a model containing all p predictors using a technique that ***constrains*** or ***regularizes*** the coefficient estimates. This approach:

- improves the fit
- reduce the variance of the estimates

The two best-known techniques for shrinking the regression coefficients towards zero are ***ridge regression*** and the ***lasso***.

The Ridge Regression

Recall:

In least squares fitting procedure, we estimate $\beta_0, \beta_1, \ldots, \beta_p$ by minimizing RSS, where

$$RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2$$

.

Ridge regression is very similar to least squares, except that the coefficients are estimated by minimizing a slightly different quantity. Here we minimize

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

where$\lambda >= 0$ is a ***tuning*** parameter.

The Ridge Regression ..

Here $\lambda \sum_j \beta_j^2$ is called a ***shrinkage penalty***. This penalty is small when $\beta_1, \beta_2, \ldots, \beta_p$ are close to zero.

- when $\lambda = 0$, ridge regression produces least squares estimates.
- as $\lambda \to 0$ the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero.

Recall that least squares method generates only one set of coefficient estimates. Here in ridge regression, it produces a different set of coefficient estimates, β^R, for each value of λ. Selecting a good value for λ is critical.

The shrinkage penalty is applied to $\beta_1, \ldots, \beta_p$, but not to the intercept β_0.

The Lasso

- In the ridge regression, all **p** predictors are in the final model.
- The penalty, $\lambda \sum_j \beta_j^2$ shrinks the coefficients towards zero.
- But not exactly to zero.
- This leads to model interpretations difficulties when p is quite large.
- Lasso overcomes this disadvantage.

The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

Lasso uses l_1 penalty. l_1 norm is defined as $||\beta||_1 = \sum|\beta_j|$.

- l_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.
- Hence, the lasso performs variable selection.

Example: Ridge Regression

Here we use the ***Hitters*** data in the ***ISLR*** package. We want to predict a baseball player's salary on the basis of various statistics associated with performance in the previous year.

```
library (ISLR)
```

```
Warning: package 'ISLR' was built under R version 4.0.5
```

```
data(Hitters)
```

- What are the variable names?
- How many records, variables are there?
- How many missing values are there in the *Salary* variable?

```
# removing missing values
myHitters = na.omit(Hitters )
```

Example: Ridge Regression..

Here we use the *glmnet* package and its *glmnet*() function to fit ridge regression models.

```
x = model.matrix(Salary~.,myHitters )[,-1]
y = myHitters$Salary
```

The *glmnet*() function has an ***alpha*** argument that determines what type of model is fit. If ***alpha=0*** then a ridge regression model is fit, and if ***alpha=1*** then a lasso model is fit.

```
library(glmnet)
```

```
Warning: package 'glmnet' was built under R version 4.0.5

Loading required package: Matrix

Loaded glmnet 4.1-1
```

Example: Ridge Regression..

```
ridge.mod = glmnet(x,y,alpha=0)
ridge.mod
```

```
Call:  glmnet(x = x, y = y, alpha = 0)

     Df   %Dev Lambda
1    19   0.00 255300
2    19   1.16 232600
3    19   1.27 211900
4    19   1.39 193100
5    19   1.53 176000
6    19   1.67 160300
7    19   1.83 146100
8    19   2.00 133100
9    19   2.19 121300
10   19   2.39 110500
11   19   2.61 100700
12   19   2.85  91740
```

Example: Ridge Regression..glmnet()

Defaults of ***glmnet()*** function:

- performs ridge regression for an automatically selected range of λ values.
- standardizes the variables so that they are on the same scale.
- stores regression coefficients for each value of λ in a matrix.
- estimated coefficients can be accessed by $coef()$.

Example: Ridge Regression..glmnet()

For this example we will be using a grid of λ values ranging from $\lambda = 10^{10}$ to $\lambda = 10^{-2}$.

```
grid = 10^seq(10,-2, length=100)
ridge.mod =glmnet(x, y, alpha=0, lambda=grid)
```

Due to the l_2 norm, the coefficient estimates are much smaller, for large value of λ, as compared to a small value of λ is used.

Example: Ridge Regression..

```
ridge.mod$lambda [50]
```

```
[1] 11497.57
```

```
coef(ridge.mod)[,50]
```

```
  (Intercept)         AtBat          Hits         HmRun
407.356050200   0.036957182   0.138180344   0.524629976
          RBI         Walks         Years        CAtBat
  0.239841459   0.289618741   1.107702929   0.003131815
       CHmRun         CRuns          CRBI        CWalks
  0.087545670   0.023379882   0.024138320   0.025015421
    DivisionW       PutOuts       Assists        Errors
 -6.215440973   0.016482577   0.002612988  -0.020502690
```

```
sqrt(sum(coef(ridge.mod)[-1 ,50]^2) )
```

```
[1] 6.360612
```

Example: Ridge Regression..

```
ridge.mod$lambda[60]
```

```
[1] 705.4802
```

```
coef(ridge.mod)[,60]
```

```
 (Intercept)       AtBat        Hits       HmRun
 54.32519950  0.11211115  0.65622409  1.17980910   0.937
       Walks       Years      CAtBat       CHits
  1.31987948  2.59640425  0.01083413  0.04674557   0.337
        CRBI      CWalks     LeagueN   DivisionW
  0.09780402  0.07189612 13.68370191 -54.65877750  0.118
      Errors  NewLeagueN
 -0.70358655  8.61181213
```

```
sqrt(sum(coef(ridge.mod)[-1,60]^2) )
```

```
[1] 57.11001
```

Example: Ridge Regression..predict()

We can use the predict() function to obtain the ridge regression coefficients for a new value of λ, say 50.

```
predict(ridge.mod, s=50, type="coefficients")[1:20 ,]
```

```
  (Intercept)         AtBat          Hits         HmRun
 4.876610e+01 -3.580999e-01  1.969359e+00 -1.278248e+00  1.
          RBI         Walks         Years        CAtBat
 8.038292e-01  2.716186e+00 -6.218319e+00  5.447837e-03  1.
       CHmRun         CRuns          CRBI        CWalks
 6.244860e-01  2.214985e-01  2.186914e-01 -1.500245e-01  4.
    DivisionW       PutOuts       Assists        Errors
-1.182011e+02  2.502322e-01  1.215665e-01 -3.278600e+00 -9.
```

Example: Splitting the dataset

Let's first set a random seed so that the results obtained will be reproducible.

```
set.seed (1)
train = sample (1: nrow(x), nrow(x)/2)
test = (- train )
y.test = y[test]
```

Next we fit a ridge regression model on the training set, and evaluate its MSE on the test set, using $\lambda = 4$.

```
ridge.mod = glmnet (x[train ,], y[train], alpha=0, lambda=g
thresh = 1e-12)
ridge.pred = predict (ridge.mod, s=4, newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
[1] 142199.2
```

Example: Ridge Regression..

The test MSE is 101036.8. Now let's fit a model with just an intercept. Note that here the predicted value for each test observation is the mean of the training dataset.

```
mean((mean(y[train ])-y.test)^2)
```

```
[1] 224669.9
```

We could also get the same result by fitting a ridge regression model with a very large value of λ.

```
ridge.pred = predict(ridge.mod, s=1e10, newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
[1] 224669.8
```

Example: Ridge Regression..

##Which model is better?

Intercept only model or the ridge regression model with $\lambda = 4$.

Ex: Check whether there is any benefit to performing ridge regression with $\lambda = 4$ instead of just performing least squares regression. Recall that least squares is simply ridge regression with $\lambda = 0$.

Example: Ridge Regression..

Answer

```
ridge.pred = predict (ridge.mod, s=0, newx=x[test ,])
mean((ridge.pred-y.test)^2)
```

```
[1] 167789.8
```

Example: Ridge Regression..

Comparing estimates

```r
lm(y~x, subset=train)
```

```

Call:
lm(formula = y ~ x, subset = train)

Coefficients:
(Intercept)       xAtBat        xHits       xHmRun          x
   274.0145      -0.3521      -1.6377       5.8145          1.
     xWalks       xYears      xCAtBat       xCHits        xCH
     3.7287     -16.3773      -0.6412       3.1632          3.
      xCRBI      xCWalks     xLeagueN   xDivisionW       xPut
    -0.6005       0.3379     119.1486    -144.0831          0.
    xErrors  xNewLeagueN
    -4.7128     -71.0951
```

Example: Ridge Regression..

Comparing estimates

```
predict(ridge.mod, s=0, type="coefficients") [1:20 ,]
```

```
 (Intercept)        AtBat         Hits        HmRun
 274.2089049   -0.3699455   -1.5370022    5.9129307    1.48
       Walks        Years       CAtBat        CHits        C
   3.7577989  -16.5600387   -0.6313336    3.1115575    3.32
        CRBI       CWalks      LeagueN    DivisionW       Pu
  -0.5694414    0.3300136  118.4000592 -144.2867510    0.19
      Errors   NewLeagueN
  -4.6833775  -70.1616132
```

Example: Ridge Regression..Selecting λ

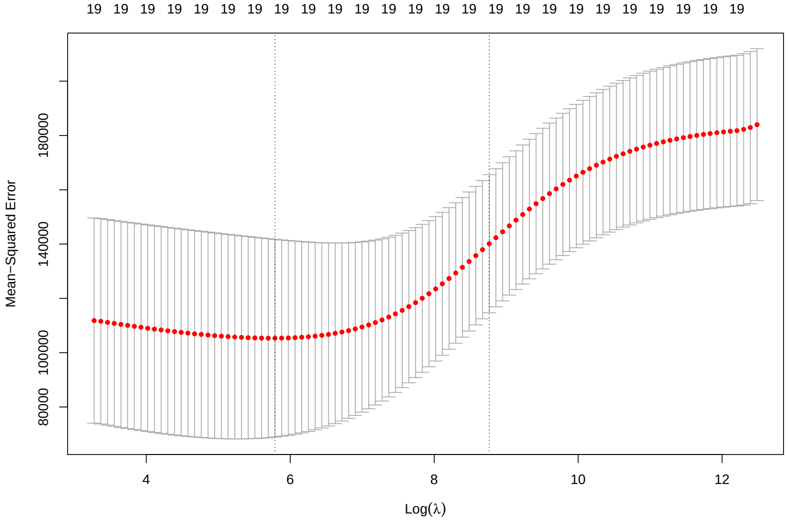
How do we select λ?

Instead of arbitrarily choosing $\lambda = 4$, it would be better to use cross-validation to choose the tuning parameter. This can be easily done using *cv.glmnet*() function. By default, *cv.glmnet*() function performs ten-fold cross-validation.

```
set.seed (1)
cv.out = cv.glmnet (x[train,],y[train],alpha=0)
```

Example: Ridge Regression.. Cross-validation

```
plot(cv.out)
```



Example: Ridge Regression.. Cross-validation

```
bestlam = cv.out$lambda.min
bestlam
```

```
[1] 326.0828
```

Therefore, the value of λ that results in the smallest cross-validation error is 211.7416.

What is the test MSE associated with this value of λ?

```
ridge.pred = predict (ridge.mod, s=bestlam, newx=x[test ,])
mean((ridge.pred-y.test)^2)
```

```
[1] 139856.6
```

Example: Ridge Regression..

This means we have improves the test MSE.

Finally, we refit our ridge regression model on the full data set, using the value of λ chosen by cross-validation, and examine the coefficient estimates.

```
out = glmnet (x,y,alpha =0)
predict(out, type="coefficients",s=bestlam )[1:20 ,]
```

```
 (Intercept)        AtBat         Hits        HmRun
 15.44383135   0.07715547   0.85911581   0.60103107    1.063
       Walks        Years       CAtBat        CHits        C
  1.62444616   1.35254780   0.01134999   0.05746654    0.406
        CRBI       CWalks      LeagueN    DivisionW       Pu
  0.12116504   0.05299202  22.09143189 -79.04032637    0.166
      Errors   NewLeagueN
 -1.36092945   9.12487767
```

Example: Ridge Regression..Done

As expected, none of the coefficients are zero.

Ridge regression does not perform **variable selection!**

Example: The Lasso

We saw that ridge regression with a wise choice of λ can outperform least squares as well as the null model on the *Hitters* data set.

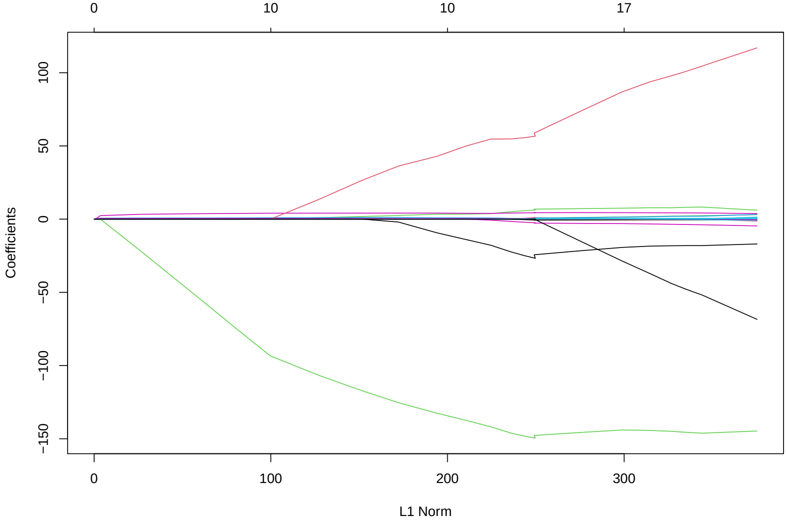
Can the lasso yield either a more accurate or a more interpretable model than ridge regression?

We use the ***glmnet()*** function with ***alpha=1***.

```
lasso.mod = glmnet(x[train,],y[train],alpha=1, lambda=grid
plot(lasso.mod)
```

Example: Lasso ..

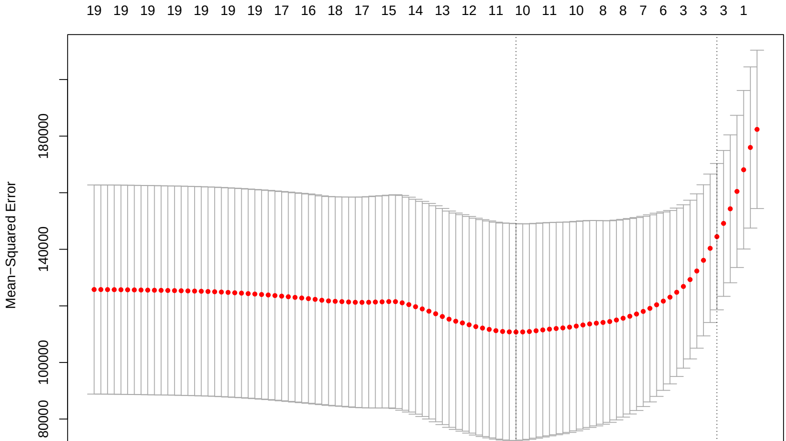
```
Warning in regularize.values(x, y, ties, missing(ties), na.
collapsing to unique 'x' values
```



Example: Lasso ..

Let's perform cross-validation and compute the associated test error.

```
set.seed (1)
cv.out = cv.glmnet(x[train,], y[train], alpha=1)
plot(cv.out)
```



Example: Lasso ..

```
bestlam = cv.out$lambda.min
bestlam
```

```
[1] 9.286955
```

```
lasso.pred = predict(lasso.mod, s=bestlam, newx=x[test ,])
mean((lasso.pred-y.test)^2)
```

```
[1] 143673.6
```

Example: Lasso ..

Estimates

```
out = glmnet (x, y, alpha=1, lambda=grid)
lasso.coef = predict (out, type="coefficients", s=bestlam )
lasso.coef
```

```
  (Intercept)        AtBat         Hits        HmRun
   1.27479059  -0.05497143   2.18034583   0.00000000
          RBI        Walks        Years       CAtBat
   0.00000000   2.29192406  -0.33806109   0.00000000
       CHmRun        CRuns         CRBI       CWalks
   0.02825013   0.21628385   0.41712537   0.00000000
    DivisionW      PutOuts      Assists       Errors
-116.16755870   0.23752385   0.00000000  -0.85629148
```

Example: Lasso ..

Estimates

```r
lasso.coef[lasso.coef !=0]
```

```
  (Intercept)          AtBat           Hits          Walks    -
   1.27479059    -0.05497143     2.18034583     2.29192406    -
       CHmRun          CRuns           CRBI        LeagueN
   0.02825013     0.21628385     0.41712537    20.28615023 -1
      PutOuts         Errors
   0.23752385    -0.85629148
```

```r
length(lasso.coef) # number of coefficients
```

```
[1] 20
```

```r
length(lasso.coef[lasso.coef !=0]) # number of non-zero co
```

```
[1] 12
```