Regularized Regression

Shrinkage Methods

Shrinkage Methods

- Instead of Least Squares, a model is fitted using some p predictors with a technique that shrinks the coefficient estimates towards zero
- Shrinking the coefficient estimates can reduce their variance
- Ridge Regression and Lasso Regression are the two shrinkage methods which will be covered in this topic

ℓ_1 & ℓ_2 Forms

- Consider the elements $\beta_1, \beta_2 \dots \beta_p$.
- The ℓ_1 form of the elements, also denoted by $||\beta||_1$ is given by the expression $||\beta||_1 = \sum_{j=1}^{p} |\beta_j|$
- The ℓ_2 form of the elements, also denoted by $||\beta||_2$ is given by the expression $||\beta||_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$

Least Squares (Quick Recap)

- In Least Squares Method, the estimates of $\beta_0, \beta_1, \beta_2 \ldots \beta_p$ are calculated by minimizing the following expression

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

Where

- y_i : Response value for ith observation
- x_{ij} : Value of jth observation in ith predictor

Ridge Regression

- For Ridge Regression, regression coefficients $\beta_0, \beta_1, \beta_2 \ldots \beta_p$ are calculated by minimizing the following expression

  $RSS + \lambda \sum_{j=1}^{p} \beta_j^2$

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

  where $\lambda \geq 0$ is a tuning parameter
- The term $\sum_{j=1}^{p} \beta_j^2$ is called shrinkage penalty. It is small if $\beta_1, \beta_2 \ldots \beta_p$ are close to zero.
- For selecting optimal value for λ, cross-validation can be used

Lasso Regression

- Ridge Regression selects all the predictors in the final model. This is a disadvantage of it.
- The Lasso overcomes this disadvantage. It finds the coefficients estimates of $\beta_0, \beta_1, \beta_2 \ldots \beta_p$ by minimizing the quantity,

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

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

- The ℓ_1 penalty of Lasso has the effect of forcing some of the coefficient estimates to zero when tuning parameter λ is sufficiently large.

Sane's STATS Academy of Statistics

Variable Selection

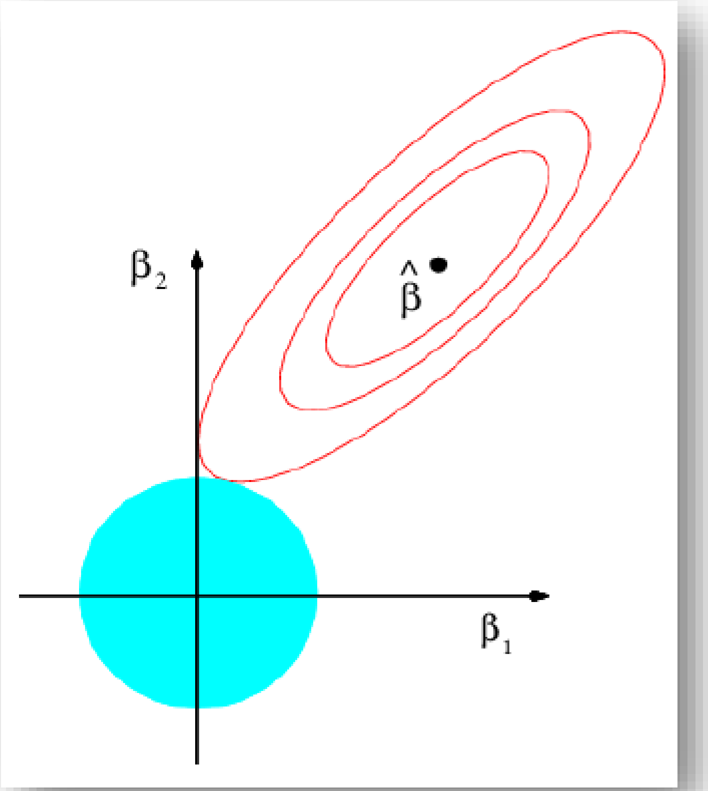
- It can be proved that the regularized regression coefficient estimates solve the optimization problems namely,
- Minimizing $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{i=1}^{p} \beta_i x_{ij})^2$

  subject to $\sum_{j=1}^{p} |\beta_j| \leq$ s for some finite value s (Lasso)
- Minimizing $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{i=1}^{p} \beta_i x_{ij})^2$

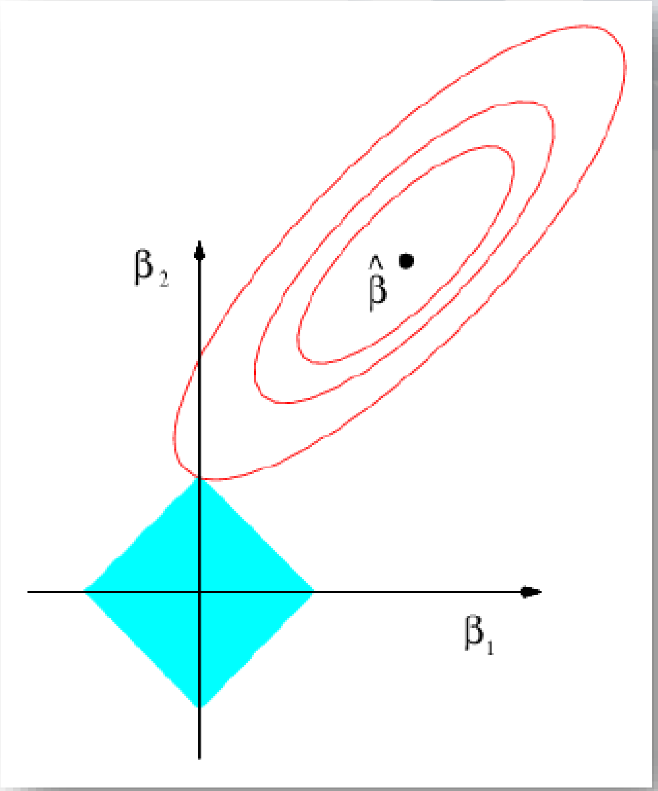
  subject to$\sum_{j=1}^{p} \beta_j^2 \leq s$ for some finite value s (Ridge)

Comparison (Considering 2-dimensional space)

- Ridge



- Lasso



Hence, we see that there can be a possibility of any coefficient in Lasso regression being reduced to zero.

Regularized Regression in R

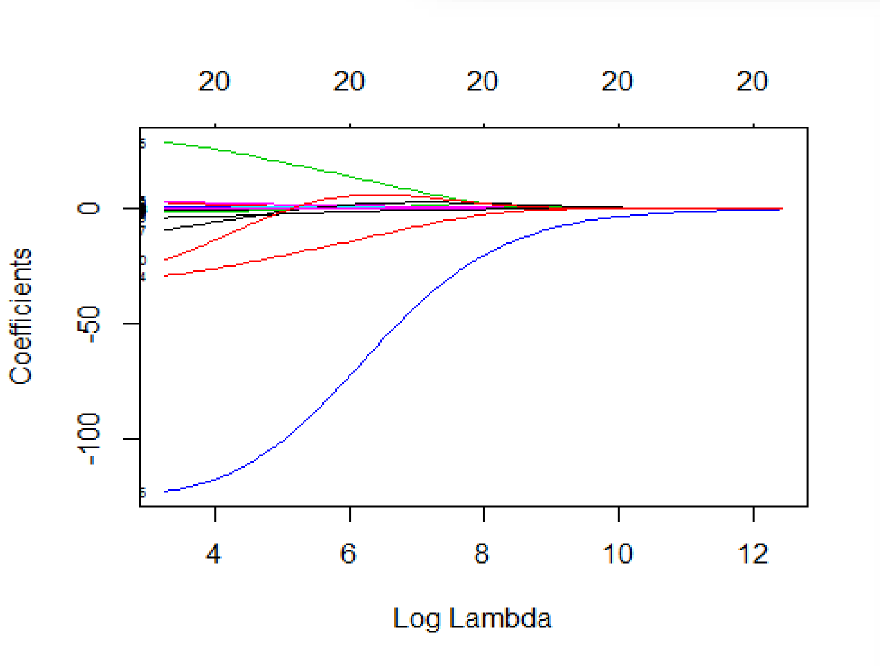
- Ridge and Lasso can be implemented in R using function glmnet and cross-validation using function cv.glmnet from package glmnet
- Syntax :
- glmnet(x, y, alpha, family,…)
- cv.glmnet(x, y, …)
- Where
  - x : Matrix of predictors. Can be a sparse matrix format
  - y : response variable vector
  - alpha : 1 for lasso and 0 for ridge
  - family : Any of "gaussian", "binomial", "poisson", "multinomial", "cox", "mgaussian“ (See help for more info on y and family options)

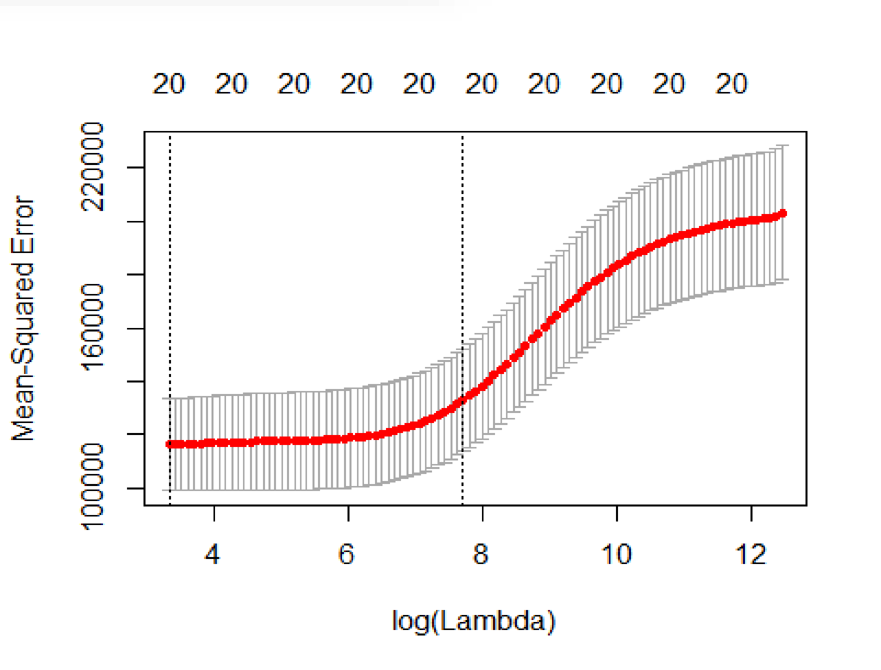
Example : Baseball Data (Hitters)

- Hitters is a Major League Baseball Data from the 1986 and 1987 seasons
- Features: 19 , Response: Salary
- As the algorithm cannot handle NA values those have been removed from our example

Ridge Regression Program & Output

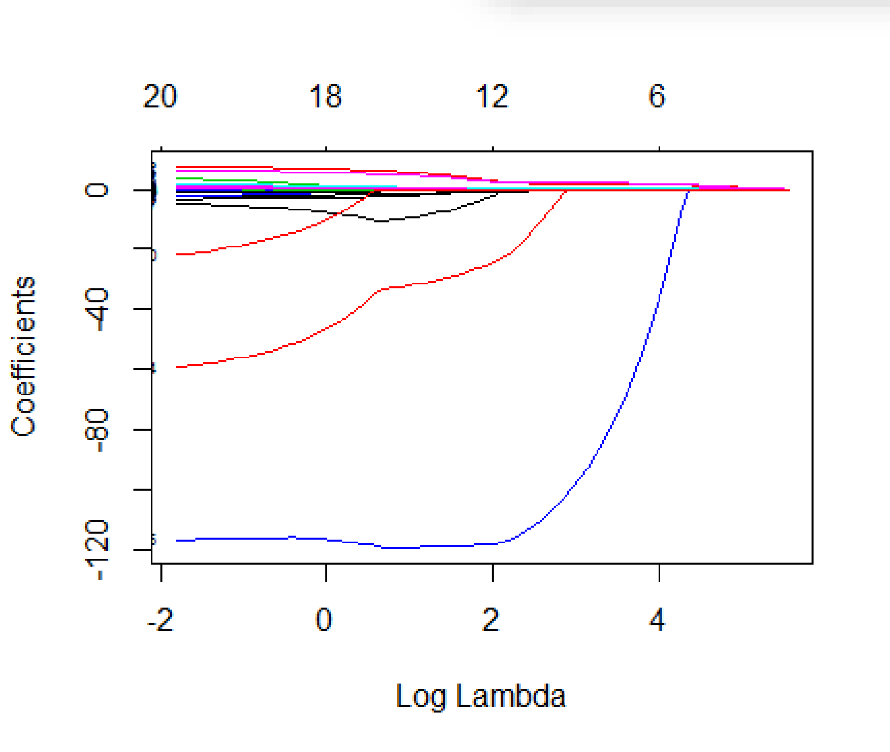
```
x=model.matrix(Salary~.-1,data=Hitters)
y=Hitters$Salary
fit.ridge <- glmnet(x,y,alpha = 0)
plot(fit.ridge,xvar = "lambda",label = TRUE)
cv.ridge <- cv.glmnet(x,y, alpha=0)
plot(cv.ridge)
```

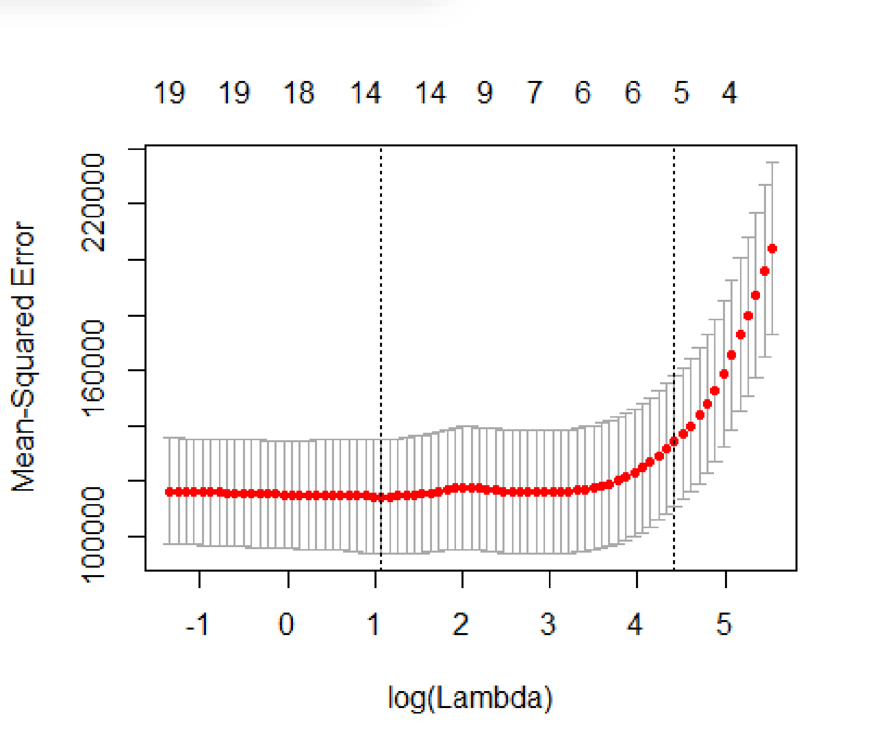




Lasso Regression Program & Output

```
fit.lasso <- glmnet(x,y,alpha = 1)
plot(fit.lasso,xvar = "lambda",label = TRUE)
cv.lasso <- cv.glmnet(x,y, alpha=1)
plot(cv.lasso)
coef.cv.glmnet(cv.lasso)
```





Coefficients of Models (λ) with minimum error

- In the figure the missing values indicate the coefficients being reduced to zero

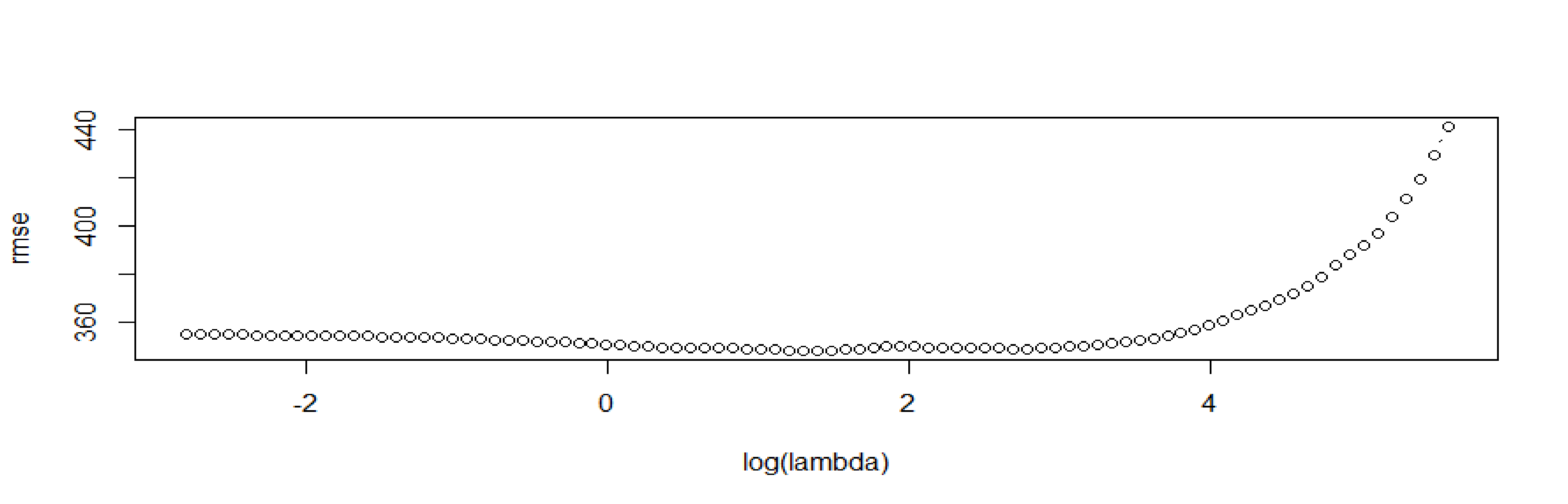
```
> coef.cv.glmnet(cv.ridge,s="lambda.min")
21 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  1.041375e+02
AtBat       -6.301512e-01
Hits         2.642007e+00
HmRun       -1.384250e+00
Runs         1.049276e+00
RBI          7.318299e-01
Walks        3.276489e+00
Years       -8.705697e+00
CAtBat       1.136403e-04
CHits        1.319492e-01
CHmRun       6.898036e-01
CRuns        2.831928e-01
CRBI         2.512166e-01
CWalks      -2.603598e-01
LeagueA     -2.871264e+01
LeagueN      2.874200e+01
DivisionW   -1.223809e+02
PutOuts      2.621883e-01
Assists      1.628961e-01
Errors      -3.669810e+00
NewLeagueN  -2.108745e+01
```

```
> coef.cv.glmnet(cv.lasso,s="lambda.min")
21 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  1.491497e+02
AtBat       -1.474290e+00
Hits         5.499426e+00
HmRun        .
Runs         .
RBI          .
Walks        4.599165e+00
Years       -9.191831e+00
CAtBat       .
CHits        .
CHmRun       4.806743e-01
CRuns        6.354799e-01
CRBI         3.956153e-01
CWalks      -4.993240e-01
LeagueA     -3.162382e+01
LeagueN      3.364107e-14
DivisionW   -1.192516e+02
PutOuts      2.704287e-01
Assists      1.594997e-01
Errors      -1.942636e+00
NewLeagueN   .
```

Lasso with Supervised Learning

- Data divided
- Cross-validation applied while evaluating RMSE

```
lasso.tr <- glmnet(x[intrain,],y[intrain])
pred <- predict(lasso.tr,x[-intrain,])
dim(pred)
rmse <- sqrt(apply((y[-intrain]-pred)^2,2,mean))
```



Coefficients of Best Model

```
> lam.best <- lasso.tr$lambda[order(rmse)[1]]
> coef(lasso.tr,s=lam.best)
21 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  170.30004050
AtBat         -1.75444276
Hits           5.77582252
HmRun          .
Runs           .
RBI            .
Walks          4.83453532
Years         -6.43674901
CAtBat         .
CHits          .
CHmRun         2.05832497
CRuns          0.48020476
CRBI           0.01994523
CWalks        -0.28581829
LeagueA        .
LeagueN        .
DivisionW   -103.94595248
PutOuts        0.29494229
Assists        0.32453860
Errors        -1.55372780
NewLeagueN   -10.76797783
```

K-fold Cross-validation

- K-fold cross-validation can be done for optimal value of lambda for which the error is minimum
- The model can be fitted using function cv.glmnet the object of which also returns optimal lambda for minimum error
- The prediction can be done by using function predict.cv.glmnet and calling the optimal lambda in it.

Program for CV

Help Doc

`lambda`	the values of `lambda` used in the fits.
`cvm`	The mean cross-validated error - a vector of length `length(lambda)`.
`cvsd`	estimate of standard error of `cvm`.
`cvup`	upper curve = `cvm+cvsd`.
`cvlo`	lower curve = `cvm-cvsd`.
`nzero`	number of non-zero coefficients at each `lambda`.
`name`	a text string indicating type of measure (for plotting purposes).
`glmnet.fit`	a fitted glmnet object for the full data.
`lambda.min`	value of `lambda` that gives minimum `cvm`.
`lambda.1se`	largest value of `lambda` such that error is within 1 standard error of the minimum.

```
> regular.tr.cv <- cv.glmnet(x[intrain,],y[intrain])
> names(regular.tr.cv)
 [1] "lambda"     "cvm"        "cvsd"       "cvup"       "cvlo"       "nzero"
 [7] "name"       "glmnet.fit" "lambda.min" "lambda.1se"
> pred.regular <- predict.cv.glmnet(regular.tr.cv , x[-intrain,], s="lambda.min")
> postResample(pred = pred.regular[,1] , obs = y[-intrain])
      RMSE    Rsquared
348.4114039   0.3783999
```