Ridge And Lasso Regressions

Jai Vrat Singh

23/07/2018

Data used in this example can be downloaded from kaggle: https://www.kaggle.com/c/boston-housing

For myself, I have temporarily saved it into /Users/jvsingh/.kaggle/competitions/boston-housing using the kaggle api.

Ridge Regression

```
filename="/Users/jvsingh/.kaggle/competitions/boston-housing/train.csv"
df = read.csv(file = filename, header = TRUE, stringsAsFactors = FALSE)
#uppercase columns
colnames(df) <- toupper(colnames(df))
head(df)
```

```
##   ID    CRIM   ZN INDUS CHAS   NOX    RM  AGE    DIS RAD TAX PTRATIO
## 1  1 0.00632 18.0  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3
## 2  2 0.02731  0.0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8
## 3  4 0.03237  0.0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7
## 4  5 0.06905  0.0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7
## 5  7 0.08829 12.5  7.87    0 0.524 6.012 66.6 5.5605   5 311    15.2
## 6 11 0.22489 12.5  7.87    0 0.524 6.377 94.3 6.3467   5 311    15.2
##    BLACK LSTAT MEDV
## 1 396.90  4.98 24.0
## 2 396.90  9.14 21.6
## 3 394.63  2.94 33.4
## 4 396.90  5.33 36.2
## 5 395.60 12.43 22.9
## 6 392.52 20.45 15.0
```

```
#we do not need ID
df[["ID"]] = NULL
head(df)
```

```
##      CRIM   ZN INDUS CHAS   NOX    RM  AGE    DIS RAD TAX PTRATIO  BLACK
## 1 0.00632 18.0  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0.0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
## 3 0.03237  0.0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
## 4 0.06905  0.0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
## 5 0.08829 12.5  7.87    0 0.524 6.012 66.6 5.5605   5 311    15.2 395.60
## 6 0.22489 12.5  7.87    0 0.524 6.377 94.3 6.3467   5 311    15.2 392.52
##   LSTAT MEDV
## 1  4.98 24.0
## 2  9.14 21.6
## 3  2.94 33.4
## 4  5.33 36.2
## 5 12.43 22.9
## 6 20.45 15.0
```

```r
X = df[,colnames(df)[-length(colnames(df))]]
Y = df[,length(colnames(df))]
```

Ridge Model

```r
#Ridge Model Fit
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.4
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```r
ridge.mod = glmnet(x = as.matrix(X), y=Y, alpha=0, lambda = 0.1)
coef(ridge.mod)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  32.313334399
## CRIM         -0.049073407
## ZN            0.044489648
## INDUS         0.034325021
## CHAS          3.833756311
## NOX         -14.628904073
## RM            3.814889858
## AGE          -0.005497041
## DIS          -1.487442301
## RAD           0.288706424
## TAX          -0.010886780
## PTRATIO      -0.838062519
## BLACK         0.011615203
## LSTAT        -0.589050130
```

```r
predict (ridge.mod, s=1, type = "coefficients")
```

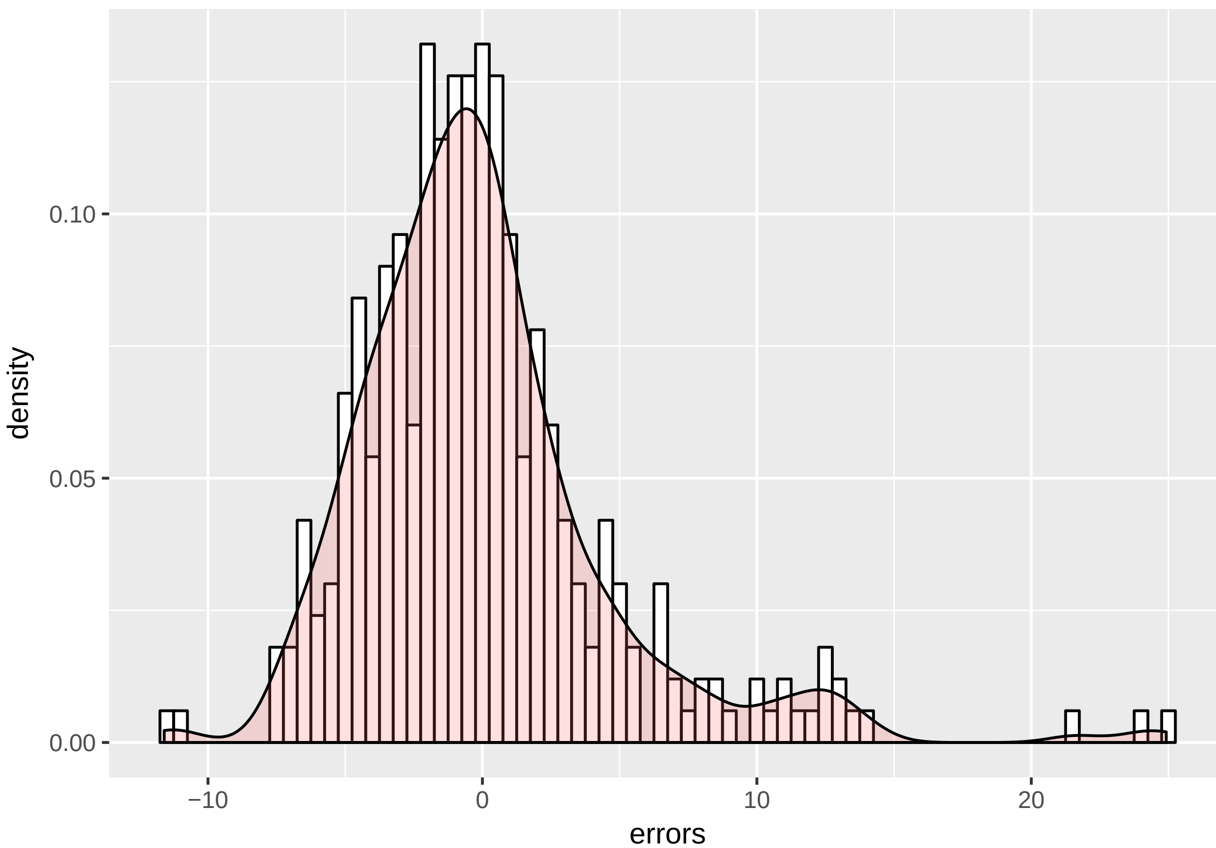
```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  32.313334399
## CRIM         -0.049073407
## ZN            0.044489648
## INDUS         0.034325021
## CHAS          3.833756311
## NOX         -14.628904073
## RM            3.814889858
## AGE          -0.005497041
## DIS          -1.487442301
## RAD           0.288706424
## TAX          -0.010886780
## PTRATIO      -0.838062519
## BLACK         0.011615203
## LSTAT        -0.589050130
```

```
#Prediction from API
predictions = predict (ridge.mod, s=1,newx = as.matrix(X))

#Prediction manually
thetas = as.matrix(predict (ridge.mod, s=1, type = "coefficients"))
predvals = cbind(rep(1,dim(X)[1]), as.matrix(X)) %*% thetas

#These values should be same
assertthat::are_equal(sum(abs(predictions - predvals)),0.0)
```

```
## [1] TRUE
```

Let us analyze errors

```
library(ggplot2)
# Histogram overlaid with kernel density curve
ggplot(data.frame(errors = as.numeric(Y - predictions)), aes(x=errors))  +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```

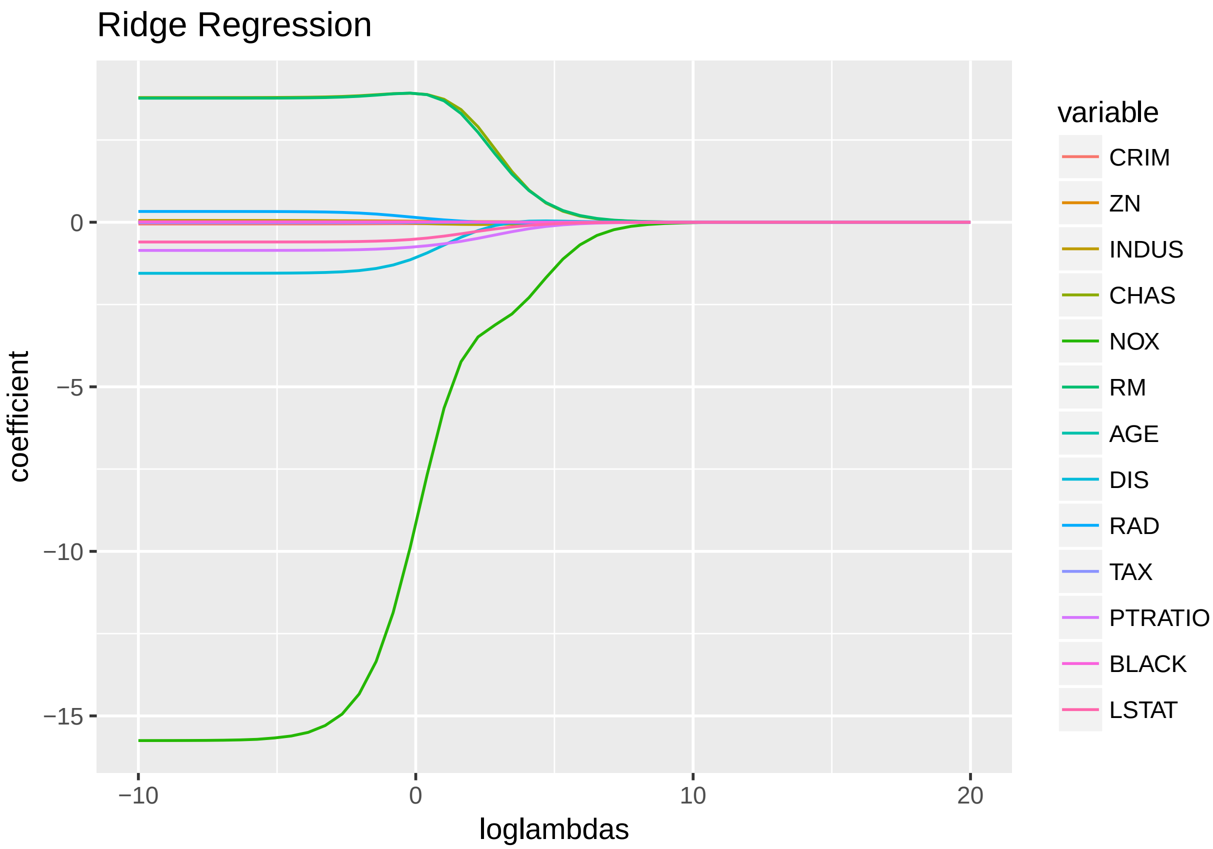



```
#Let us produce ridge coefficients with different alphas
loglambdas = seq(from = -10, to = 20, length.out = 50)
lambdas    = exp(loglambdas)

RidgeCoeffs <- function(lambda){
```

```r
  ridge.mod = glmnet(x = as.matrix(X), y=Y, alpha=0, lambda = lambda)
  as.matrix(coef(ridge.mod))
}

coeffsList = as.data.frame(t(sapply(lambdas, function(lambda) RidgeCoeffs(lambda))))

#prepare column names
colnames(coeffsList) = c("bias", colnames(df)[-length(colnames(df))])
coeffsDF = coeffsList
coeffsDF["loglambdas"] = loglambdas
head(coeffsDF)
```

```
##       bias        CRIM         ZN      INDUS     CHAS       NOX       RM
## 1 34.03654 -0.05236777 0.04744169 0.05256458 3.788303 -15.74995 3.769750
## 2 34.03581 -0.05236625 0.04744038 0.05255684 3.788324 -15.74949 3.769772
## 3 34.03446 -0.05236344 0.04743797 0.05254256 3.788362 -15.74865 3.769812
## 4 34.03197 -0.05235827 0.04743352 0.05251622 3.788433 -15.74709 3.769886
## 5 34.02738 -0.05234874 0.04742533 0.05246768 3.788563 -15.74421 3.770023
## 6 34.01891 -0.05233118 0.04741022 0.05237822 3.788803 -15.73890 3.770275
##            AGE       DIS       RAD         TAX    PTRATIO      BLACK
## 1 -0.004666253 -1.551365 0.3274956 -0.01278266 -0.8561569 0.01165637
## 2 -0.004666682 -1.551339 0.3274787 -0.01278182 -0.8561497 0.01165636
## 3 -0.004667474 -1.551291 0.3274475 -0.01278027 -0.8561364 0.01165635
## 4 -0.004668935 -1.551203 0.3273898 -0.01277741 -0.8561119 0.01165632
## 5 -0.004671627 -1.551040 0.3272836 -0.01277213 -0.8560666 0.01165626
## 6 -0.004676587 -1.550740 0.3270878 -0.01276242 -0.8559832 0.01165615
##        LSTAT loglambdas
## 1 -0.6000606 -10.000000
## 2 -0.6000560  -9.387755
## 3 -0.6000476  -8.775510
## 4 -0.6000321  -8.163265
## 5 -0.6000034  -7.551020
## 6 -0.5999506  -6.938776
```

```r
#plotting
library(reshape2)
coeffsDF[["bias"]] = NULL #Remove bias
coeffsDF.melt = melt(coeffsDF, id.vars = "loglambdas")
colnames(coeffsDF.melt) <- c("loglambdas", "variable", "coefficient")

ggplot(data=coeffsDF.melt, aes(x=loglambdas, y=coefficient, group=variable, colour=variable)) +
    geom_line() +
    ggtitle("Ridge Regression")
```



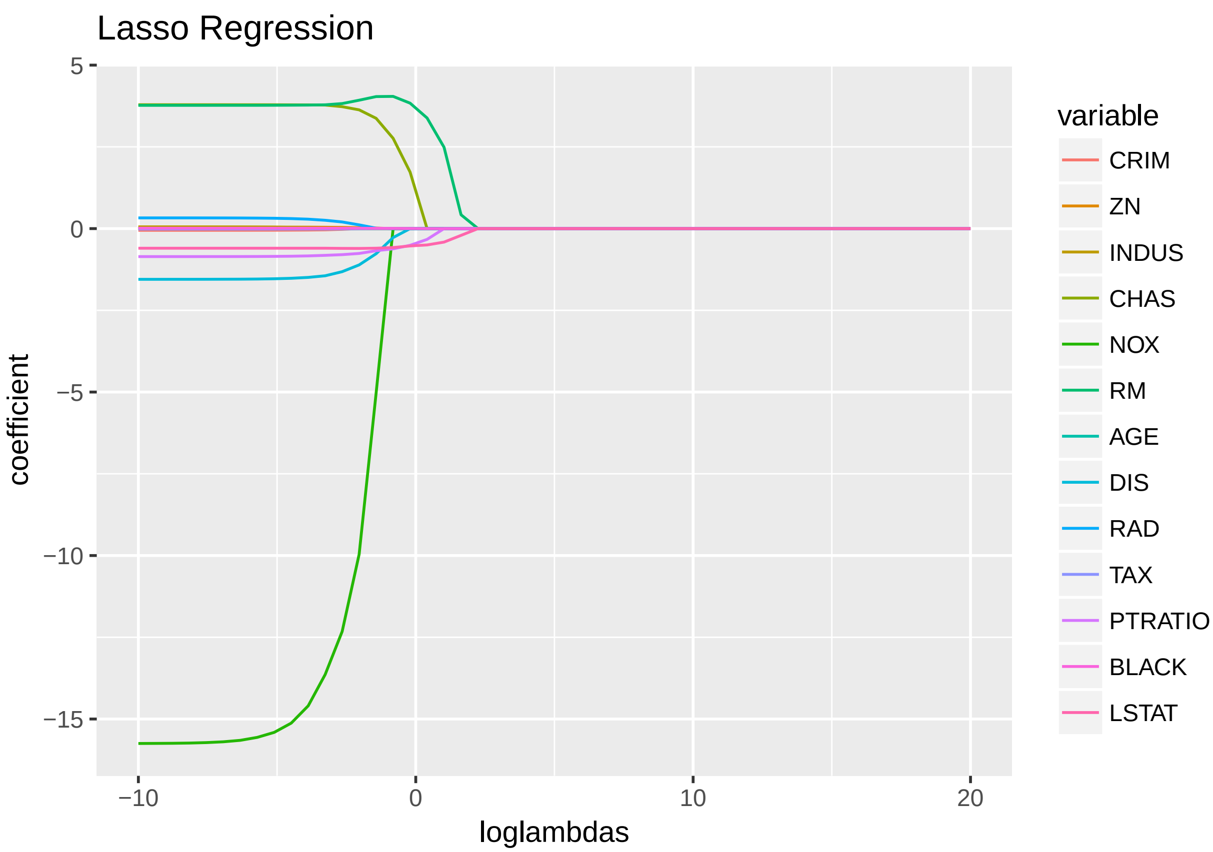
LASSO Regression

```r
#Let us produce lasso coefficients with different alphas

LassoCoeffs <- function(lambda){
  mod = glmnet(x = as.matrix(X), y=Y, alpha=1, lambda = lambda)
  as.matrix(coef(mod))
}
coeffsList = as.data.frame(t(sapply(lambdas, function(lambda) LassoCoeffs(lambda))))

#prepare column names
colnames(coeffsList) = c("bias", colnames(df)[-length(colnames(df))])
coeffsDF = coeffsList
coeffsDF["loglambdas"] = loglambdas
head(coeffsDF)
```

```
##       bias        CRIM         ZN      INDUS     CHAS       NOX       RM
## 1 34.03425 -0.05235184 0.04743584 0.05251643 3.788266 -15.74798 3.769748
## 2 34.03158 -0.05233686 0.04742960 0.05246803 3.788256 -15.74586 3.769767
## 3 34.02665 -0.05230924 0.04741808 0.05237873 3.788238 -15.74195 3.769804
## 4 34.01757 -0.05225829 0.04739683 0.05221403 3.788203 -15.73474 3.769871
## 5 34.00081 -0.05216432 0.04735764 0.05191022 3.788140 -15.72143 3.769996
## 6 33.96990 -0.05199097 0.04728536 0.05134982 3.788024 -15.69689 3.770225
##            AGE       DIS       RAD         TAX    PTRATIO      BLACK
## 1 -0.004662889 -1.551265 0.3274294 -0.01277939 -0.8561208 0.01165602
## 2 -0.004660478 -1.551155 0.3273564 -0.01277578 -0.8560831 0.01165570
```

```
## 3 -0.004656031 -1.550952 0.3272219 -0.01276913 -0.8560135 0.01165513
## 4 -0.004647827 -1.550577 0.3269738 -0.01275686 -0.8558852 0.01165406
## 5 -0.004632694 -1.549887 0.3265160 -0.01273423 -0.8556485 0.01165211
## 6 -0.004604782 -1.548612 0.3256718 -0.01269248 -0.8552119 0.01164849
##        LSTAT loglambdas
## 1 -0.6000661 -10.000000
## 2 -0.6000662  -9.387755
## 3 -0.6000664  -8.775510
## 4 -0.6000667  -8.163265
## 5 -0.6000674  -7.551020
## 6 -0.6000685  -6.938776
```

```
#plotting
coeffsDF[["bias"]] = NULL #Remove bias
coeffsDF.melt = melt(coeffsDF, id.vars = "loglambdas")
colnames(coeffsDF.melt) <- c("loglambdas", "variable", "coefficient")

ggplot(data=coeffsDF.melt, aes(x=loglambdas, y=coefficient, group=variable, colour=variable)) +
    geom_line() +
    ggtitle("Lasso Regression")
```



As we see above, the Lasso converges faster to zero as expected with increasing regularization penalty

As seen below, both ridge and lasso have same estimates and equal to that of Linear Regression if we use the regularization paramater 0, i.e lambda=0

```
ridge.mod = glmnet(x = as.matrix(X), y=Y, alpha=0, lambda = 0)
#as.matrix(t(coef(ridge.mod)))
lasso.mod = glmnet(x = as.matrix(X), y=Y, alpha=1, lambda = 0)
```

```r
#as.matrix(t(coef(lasso.mod)))
reg.mod <- lm(MEDV ~ ., data = df)
#coef(reg.mod)
rbind(as.matrix(t(coef(ridge.mod))), as.matrix(t(coef(lasso.mod))), coef(reg.mod))
```

```
##    (Intercept)        CRIM         ZN      INDUS     CHAS       NOX
## s0    34.03741 -0.05236957 0.04744323 0.05257375 3.788278 -15.75050
## s0    34.03741 -0.05236957 0.04744323 0.05257375 3.788278 -15.75050
##       34.04544 -0.05248934 0.04744487 0.05385524 3.784864 -15.73966
##          RM          AGE       DIS       RAD         TAX    PTRATIO
## s0 3.769724 -0.004665744 -1.551396 0.3275157 -0.01278366 -0.8561655
## s0 3.769724 -0.004665744 -1.551396 0.3275157 -0.01278366 -0.8561655
##    3.768832 -0.004626602 -1.548823 0.3289671 -0.01286650 -0.8569757
##         BLACK      LSTAT
## s0 0.01165639 -0.6000660
## s0 0.01165639 -0.6000660
##    0.01166590 -0.6003155
```