Penalized Regression and LARS algorithm

Penalized Regression

Penalized regression methods keep all the predictor variables in the model but constrain (regularize) the regression coefficients by shrinking them toward zero. If the amount of shrinkage is large enough, these methods can also perform variable selection by shrinking some coefficients to zero.

Shrinkage Methods

1.Ridge Regression
2.Lasso Regression
3.Elastic net Regression

Ridge Regression

Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated. It has been used in many fields including econometrics, chemistry, and engineering.

The ridge regression minimizes,

$$\sum(Yi - Yi(hat))^2 + lambda\sum(\beta j)^2$$

To build the ridge regression in r, we use glmnet function from glmnet package in R.

Let's consider the mtcars dataset in R, and fit a ridge regression model to predict the mileage of the car

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-7
```

```
x <- data.matrix(mtcars[, c("hp", "wt", "drat")])
y <- mtcars[, "mpg"]
lambda_seq <- 10^seq(2, -2, by = -.1)
fit <- glmnet(x, y, alpha = 0, lambda  = lambda_seq)
summary(fit)
```

```
##           Length Class     Mode   
## a0         41    -none-    numeric
## beta      123    dgCMatrix S4     
## df         41    -none-    numeric
## dim         2    -none-    numeric
## lambda     41    -none-    numeric
## dev.ratio  41    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        5    -none-    call   
## nobs        1    -none-    numeric
```

```
# Get coefficients of all 100 models
ridge_coef <- coef(fit)

# Display coefficients for 12 models.
round(ridge_coef[, c(1:5,35:41)], 3)
```

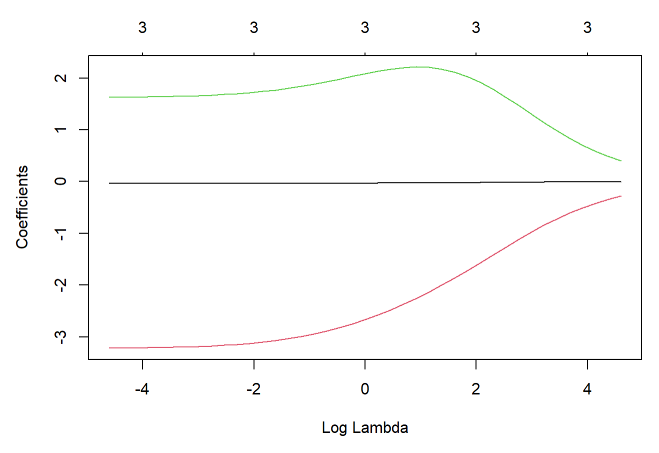
```
## 4 x 12 sparse Matrix of class "dgCMatrix"
```

```
##   [[ suppressing 12 column names 's0', 's1', 's2' ... ]]
```

```
##                                                                            
## (Intercept) 20.092 20.100 20.112 20.132 20.164 29.156 29.204 29.240 29.271
## hp          -0.004 -0.004 -0.005 -0.006 -0.008 -0.032 -0.032 -0.032 -0.032
## wt          -0.281 -0.344 -0.418 -0.505 -0.605 -3.193 -3.200 -3.205 -3.209
## drat         0.398  0.485  0.587  0.704  0.836  1.650  1.643  1.638  1.633
##                           
## (Intercept) 29.293 29.313 29.329
## hp          -0.032 -0.032 -0.032
## wt          -3.212 -3.215 -3.218
## drat         1.630  1.627  1.625
```

We can also produce a Trace plot to visualize how the coefficient estimates changed as a result of increasing λ

```
plot(fit, xvar ="lambda")
```



Choose an optimal value for λ

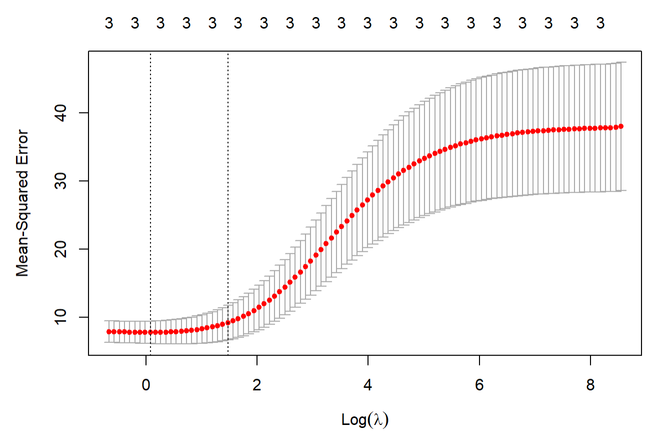
Identify the lambda value that produces the lowest MSE by using k-fold cross-validation.

glmnet has the function cv.glmnet() that performs k-fold cross validation using k = 10 folds.

```
set.seed(123)
#perform k-fold cross-validation to find optimal lambda value
cv_model <- cv.glmnet(x, y, alpha = 0)

#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model$lambda.min
best_lambda
```

```
## [1] 1.08339
```

```
plot(cv_model)
```



Final Model

```
final_model <- glmnet(x, y, alpha = 0, lambda = best_lambda)
coef(final_model)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) 25.49107983
## hp          -0.03051333
## wt          -2.63641838
## drat         2.10130824
```

R-SQUARED OF THE MODEL

```
y_predicted <- predict(final_model, s = best_lambda, newx = x)

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq
```

```
## [1] 0.8296018
```

LASSO REGRESSION

Lasso stands for Least Absolute Shrinkage and Selection Operator. It shrinks the regression coefficients toward zero by penalizing the regression model with a penalty term called L1-norm.

$$\sum(Yi - Yi(hat))^2 + lambda\sum|\beta j|$$

One advantage of lasso regression over ridge regression, is that it produces simpler and more interpretable models that incorporate only a reduced set of the predictors.

Generally, lasso might perform better in a situation where some of the predictors have large coefficients, and the remaining predictors have very small coefficients.

Ridge regression will perform better when the outcome is a function of many predictors, all with coefficients of roughly equal size.

```
# Find the best lambda using cross-validation
set.seed(123)
cvl <- cv.glmnet(x, y, alpha = 1)
# Display the best lambda value
cvl$lambda.min
```

```
## [1] 0.1034148
```

```
# Fit the final model on the training data
model_lasso <- glmnet(x, y, alpha = 1, lambda = cvl$lambda.min)
# Dsiplay regression coefficients
coef(model_lasso)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) 29.63708800
## hp          -0.03125498
## wt          -3.21282109
## drat         1.49439939
```

R-SQUARED OF THE LASSO REGRESSION MODEL

```
y_predicted <- predict(model_lasso, s = cvl$lambda.min, newx = x)

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq
```

```
## [1] 0.8364553
```

ELASTIC NET REGESSION

Elastic Net produces a regression model that is penalized with both the L1-norm and L2-norm.

The elastic net regression can be easily computed using the caret workflow, which invokes the glmnet package.

The objective of this method is to effectively shrink coefficients (like in ridge regression) and to set some coefficients to zero (as in LASSO).

FIND ALPHA AND LAMBDA

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
data.new <- data.frame(y,x)
set.seed(123)
model_ER <- train(
  y~.,data=data.new, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneLength = 10
)
# Best tuning parameter
model_ER$bestTune
```

```
##   alpha   lambda
## 6   0.1 0.834975
```

```
#Coefficient of the final model
coef(model_ER$finalModel, model_ER$bestTune$lambda)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 26.44660083
## hp          -0.03051303
## wt          -2.74813181
## drat         1.93555113
```

R-SQUARED OF THE ELASTIC NET REGRESSION MODEL

```
y_predicted <- predict(model_ER, newx = x)

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq
```

```
## [1] 0.8312703
```

LARS ALGORITHM

Least-angle regression (LARS) is an algorithm for fitting linear regression models to high-dimensional data.

It is a model selection method for linear regression.

At each step, LARS finds the attribute which is most highly correlated to the target value.

Can use the lars function from lars package in R.

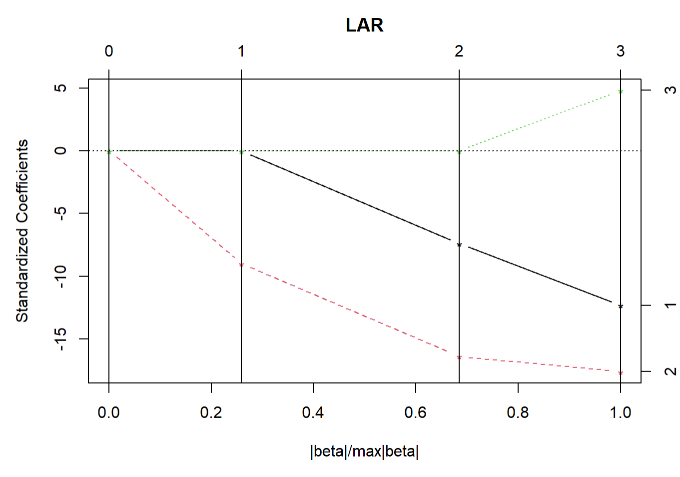
```
library(lars)
```

```
## Loaded lars 1.3
```

```
Lars_obj <- lars(x,y,type="lar")
Lars_obj
```

```
## 
## Call:
## lars(x = x, y = y, type = "lar")
## R-squared: 0.837 
## Sequence of LAR moves:
##      wt hp drat
## Var   2  1    3
## Step  1  2    3
```

```
plot(Lars_obj)
```



Obtain a penalized regression

model that best describes the variations in the response variable medv from the Boston dataset in MASS package.

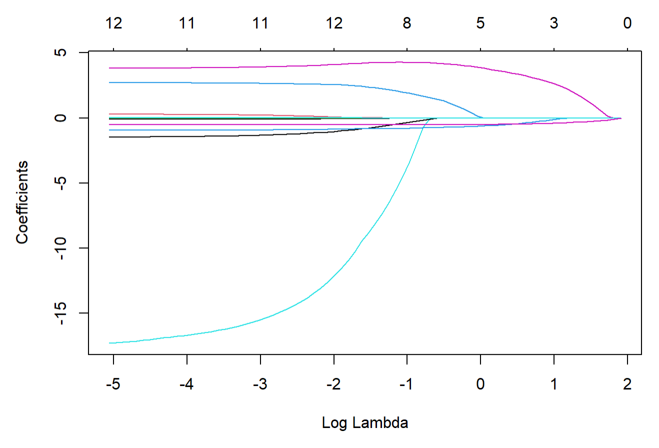
```
library(MASS)
library(glmnet)

# Load the Boston Housing dataset
data("Boston")

X <- as.matrix(Boston[, -14])  # Exclude the medv column
y <- Boston$medv
```

```
# Fit a LASSO regression model
lasso_model <- glmnet(X, y, alpha = 1)
summary(lasso_model)
```

```
##           Length Class     Mode   
## a0         76    -none-    numeric
## beta      988    dgCMatrix S4     
## df         76    -none-    numeric
## dim         2    -none-    numeric
## lambda     76    -none-    numeric
## dev.ratio  76    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        4    -none-    call   
## nobs        1    -none-    numeric
```

```
# Display regression coefficients
#coef(lasso_model)
```

```
# Plot coefficients for the LASSO model
plot(lasso_model, xvar = "lambda")
```




```
# Fit a ridge regression model
ridge_model <- glmnet(X, y, alpha = 0)
summary(ridge_model)
```

```
##           Length Class     Mode   
## a0         100   -none-    numeric
## beta      1300   dgCMatrix S4     
## df         100   -none-    numeric
## dim          2   -none-    numeric
## lambda     100   -none-    numeric
## dev.ratio  100   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         4   -none-    call   
## nobs         1   -none-    numeric
```

```
# Display regression coefficients
#coef(ridge_model)
```

```
# Plot coefficients for the ridge model
plot(ridge_model, xvar = "lambda")
```