Ridge, Lasso and ElasticNet

Hakan Gogtas

11/22/2020

Ridge regression

This package provides extremely efficient procedures for fitting the Ridge, Lasso, and ElasticNet regularization paths for linear regression, logistic and multinomial regression models, Poisson regression, and the Cox model. The algorithm is extremely fast and exploits sparsity in the input matrix where it exists. A variety of predictions can be made from the fitted models. The main function in the package is glmnet(). This function fits a generalized linear model (GLM) via penalized maximum likelihood. The regularization path is computed for the Ridge, Lasso, or ElasticNet penalty at a grid of values for the regularization parameter lambda. Can deal with all shapes of data, including very large sparse data matrices.

```
library(glmnet)
```

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

```
## Loaded glmnet 4.0-2
```

```
BodyFat <- read.csv("/cloud/project/BodyFat.csv")
#need to define X and Y matrix for glmnet
#take out Y and law indicator from data for X (all independent variable)
x <- model.matrix(Y~., BodyFat)[,-c(1)]
y <- BodyFat$Y
RidgeMod <- glmnet(x, y, alpha=0, nlambda=100,lambda.min.ratio=0.0001)
```

In the glmnet function are used the following arguments:

-nlambda=100: Set the number of lambda values (the default is 100)

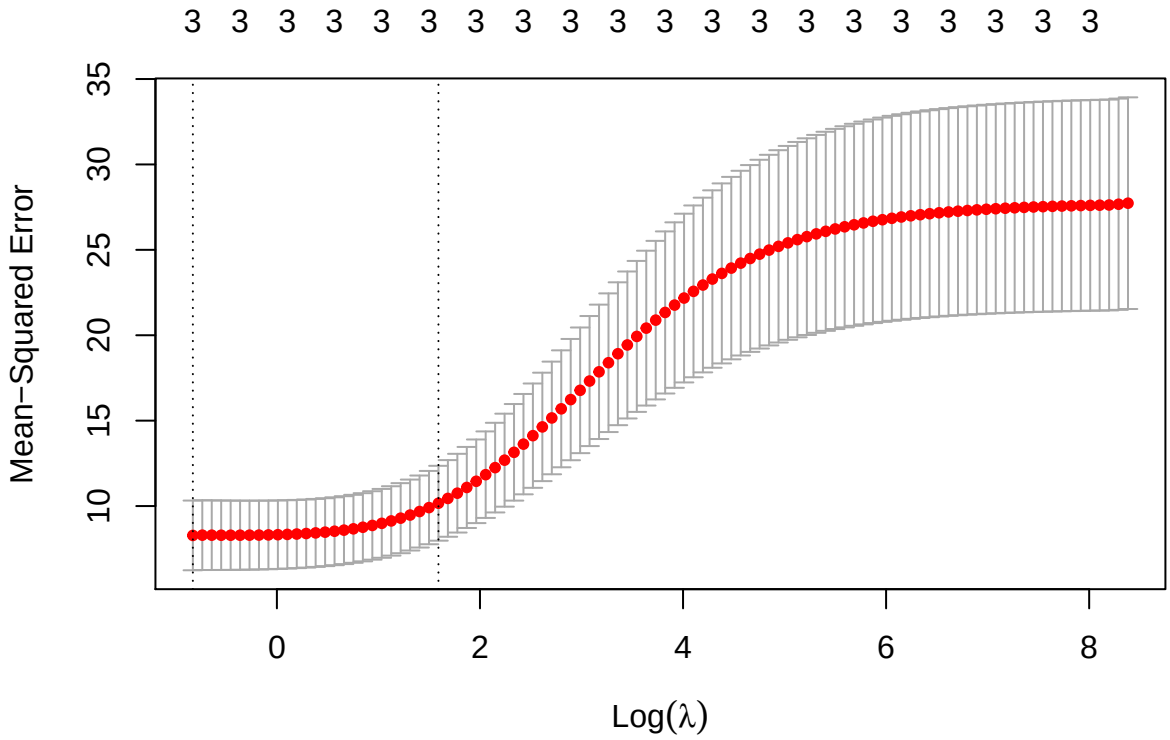
-lambda.min.ratio=0.0001: Set the smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (that is, the smallest value for which all coefficients are zero)

we will use cross valdidation to select the best lamda

```
CvRidgeMod <- cv.glmnet(x, y, alpha=0, nlambda=100,lambda.min.ratio=0.0001)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
par(mfrow=c(1,1))
plot(CvRidgeMod)
```




```
best.lambda.ridge <- CvRidgeMod$lambda.min
best.lambda.ridge
```

```
## [1] 0.4370159
```

The figure includes the cross-validation curve (red dotted line) and upper and lower standard deviation curves along the λ sequence (error bars). In the beginning of the procedure (to the right of the figure), the MSE is very high, and the coefficients are restricted to be too small; and then at some point, it kind of levels off. This seems to indicate that the full model is doing a good job.

There are two vertical lines: one is at the minimum, and the other vertical line is within one standard error of the minimum. The second line is a slightly more restricted model that does almost as well as the minimum, and sometimes, we'll go for that. These lines then lie at two lambda values:

-lambda.min is the value of λ that gives the minimum mean cross-validated error

-lambda.1se, gives the most regularized model such that the error is within one standard error of the minimum

At the top of the plot, you actually see how many nonzero variables' coefficients are in the model. There are all six variables in the model (five variables, plus the intercept), and no coefficient is zero.

Once we have the best lambda, we can use predict to obtain coefficients:

```
predict(RidgeMod, s=best.lambda.ridge, type="coefficients")[1:4, ]
```

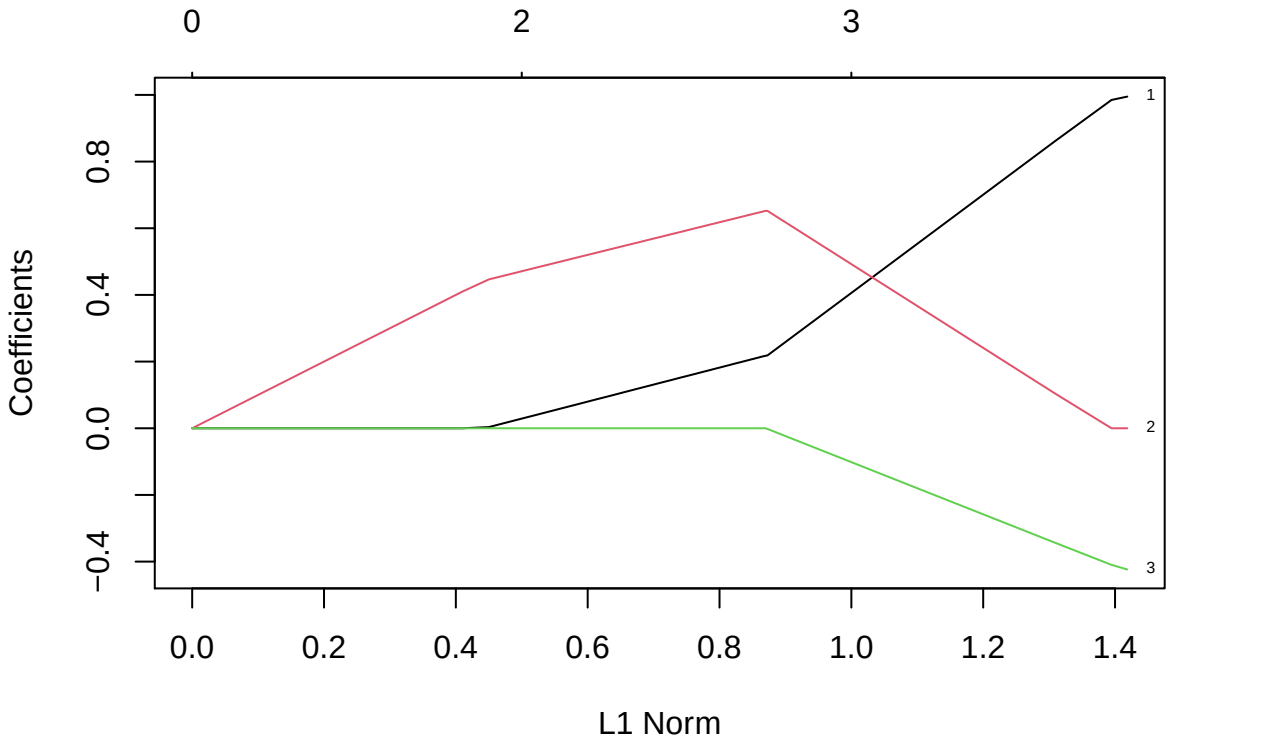
```
## (Intercept)          X1          X2          X3
## -10.0097587   0.4358802   0.4385841  -0.1183033
```

This is the best regression model for our data.

Lasso Regression

To perform Lasso regression, we will use the same dataset. On the R function below $\alpha = 1$ for Lasso regression. Everyting is very similar.

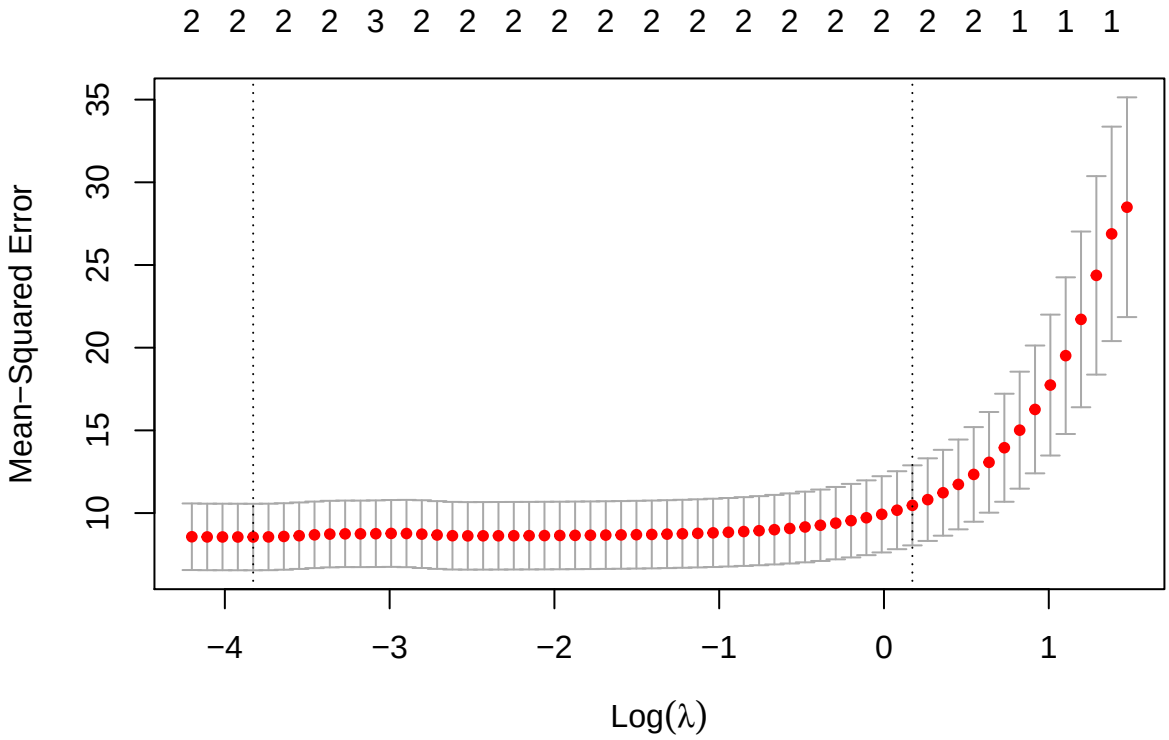
```
LassoMod <- glmnet(x, y, alpha=1, nlambda=100,lambda.min.ratio=0.0001)
plot(LassoMod,xvar="norm",label=TRUE)
```




```
CvLassoMod <- cv.glmnet(x, y, alpha=1, nlambda=100,lambda.min.ratio=0.0001)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
plot(CvLassoMod)
```




```
best.lambda.lasso <- CvLassoMod$lambda.min
best.lambda.lasso
```

```
## [1] 0.02175039
```

```
coef(CvLassoMod, s = "lambda.min")
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  6.6874068
## X1           0.9923757
## X2           .
## X3          -0.4201475
```

We have confirmed that the Lasso method is able to make a selection of variables. Ultimately, we can say that both Lasso and Ridge balance the trade-off bias-variance with the choice of λ. Lasso implicitly assumes that part of the coefficients are zero, or at least not significant. Lasso tends to have a higher performance than Ridge in cases where many predictors are not actually tied to the response variables. In opposite cases, the Ridge tends to have better performance. Both approaches can be compared by cross-validation.

ElasticNet

```
EnetMod <- glmnet(x, y, alpha=0.5, nlambda=100,lambda.min.ratio=0.0001)
CvElasticnetMod <- cv.glmnet(x, y,alpha=0.5,nlambda=100,lambda.min.ratio=0.0001)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
best.lambda.enet <- CvElasticnetMod$lambda.min
best.lambda.enet
```

```
## [1] 0.01883045
```

```r
coefficients(EnetMod,s=best.lambda.enet)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  6.7458229
## X1           0.9943108
## X2           .
## X3          -0.4240354
```

```r
coef(CvElasticnetMod, s = "lambda.min")
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  6.7458229
## X1           0.9943108
## X2           .
## X3          -0.4240354
```

```r
f<-lm(Y~.,data=BodyFat)
y_hat.ridge <- predict(RidgeMod, s = best.lambda.ridge, newx = x)
y_hat.lasso <- predict(LassoMod, s = best.lambda.lasso, newx = x)
y_hat.enet <- predict(CvElasticnetMod , s = best.lambda.enet, newx = x)
sst <- sum((y - mean(y))^2)
sse.ols<-sum(f$residuals^2)
sse.ridge <- sum((y-y_hat.ridge)^2)
sse.lasso <- sum((y-y_hat.lasso)^2)
sse.enet <- sum((y-y_hat.enet)^2)
cbind(sse.ols,sse.ridge,sse.lasso,sse.enet)
```

```
##       sse.ols sse.ridge sse.lasso sse.enet
## [1,] 98.40489  109.5129  105.9692 105.9521
```

```r
# R squared
rsq.ols<-1 - sse.ols / sst
rsq.ridge <- 1 - sse.ridge / sst
rsq.lasso <- 1 - sse.lasso / sst
rsq.enet  <- 1 - sse.enet  / sst
cbind(rsq.ols,rsq.ridge,rsq.lasso,rsq.enet)
```

```
##        rsq.ols rsq.ridge rsq.lasso  rsq.enet
## [1,] 0.8013586 0.7789357 0.7860892 0.7861237
```