Feature selection and regularization with Lasso, Ridge and Elastic Net regression

Modestas Filipavicius

May 17, 2018

The **yeastStorey.rda** data frame contains marker and gene expression information of 112 F1 segregants derived from a yeast genetic cross of two strains.

The first column is a binary marker (response) denoting presence (1) or absence (0) of a SNP and the remaining columns correspond to the gene expression values across the segregants (predictors).

Load the data and construct the design matrix X and response variable y, respectively. Randomly split the data into training set (70%) and test set (30%).

Design matrix has 231 genes, or features, and 112 samples.

Corresponding design matrix is $X \in R^{n*p}$, where n = 112 and p = 231.

```r
# standardize data to zero-mean and unit-variance
data = cbind(data[, 1], scale(data[, -1]))
colMeans(data[, 1:5]) # double-check the zero-means
```

```
##                     YAL046C       YAL061W       YAR029W       YBL009W
##  4.910714e-01 -1.908196e-17 -2.695017e-18  9.334981e-17 -1.516334e-17
```

```r
set.seed(1)
mask_train = createDataPartition(data[, 1], times=1, p=0.70, list=F)
X_train = as.matrix(data[mask_train, -1])
X_test = as.matrix(data[-mask_train, -1])

y_train = as.matrix(data[mask_train, 1])
y_test = as.matrix(data[-mask_train, 1])

# double-check if correct dims
cat("Dimensions of X_train are:", dim(X_train))
```

```
## Dimensions of X_train are: 79 231
```

```r
X_train[1:5, 1:5]
```

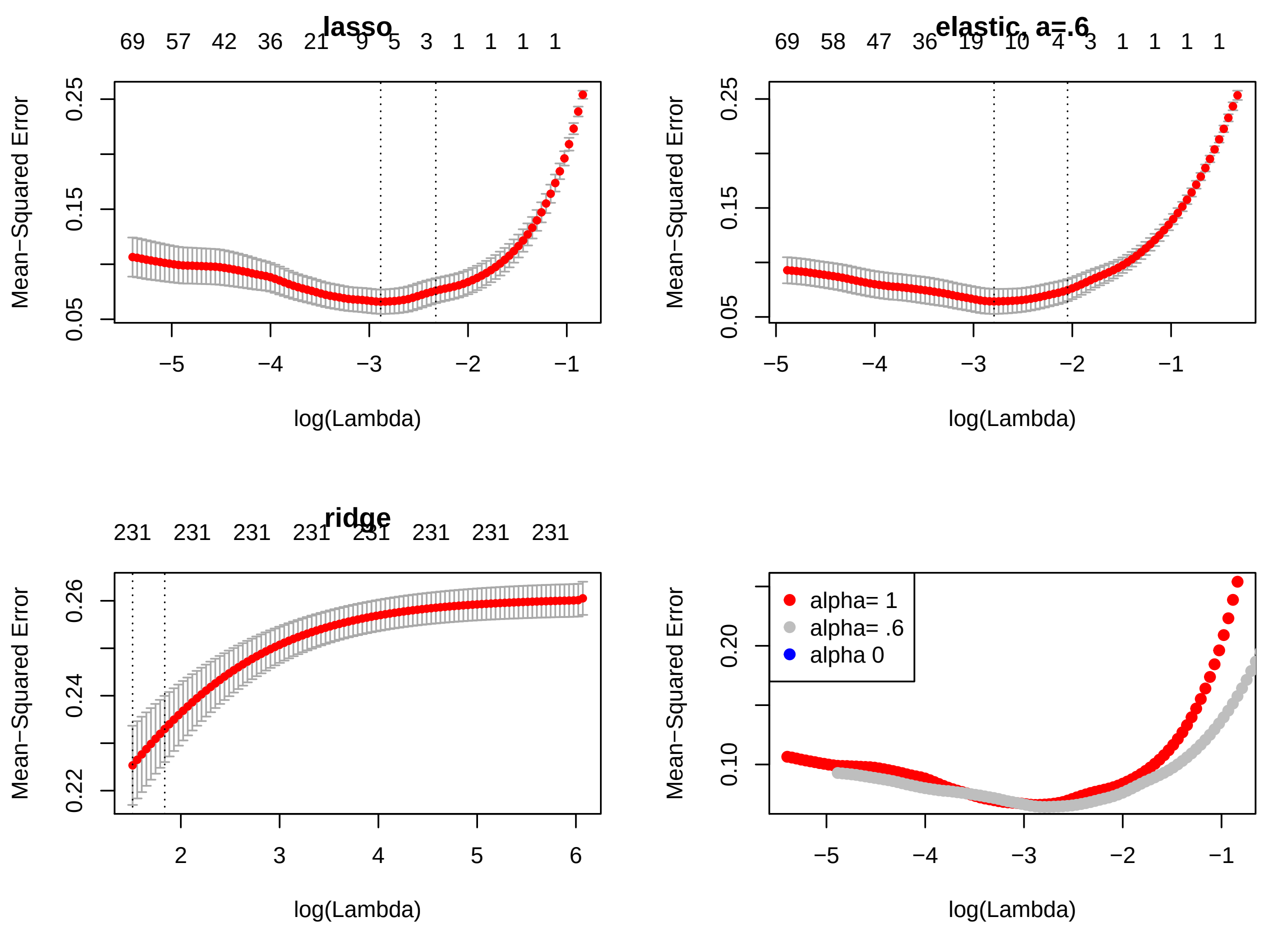
```
##              YAL046C    YAL061W     YAR029W     YBL009W    YBL059W
## sample_2  0.2356292 -0.1652986 -0.43898816 -0.09987781  0.1547758
## sample_4 -0.7943615  0.8327249  1.22583078  0.44415262 -1.1660886
## sample_5 -1.0988952 -1.1724250  0.07826706 -0.86364474 -4.0290406
## sample_6  2.4645449  1.4498544 -0.16616807 -1.15142654 -0.3988218
## sample_7 -1.3587941 -0.7102633 -1.68111438  0.67915783 -0.9038582
```

2. Using 10-fold cross-validation, find the optimum λ for each of lasso, ridge and elastic net ($\alpha = 0.6$) penalized regression models on the training set.

```
cvfit_lasso = cv.glmnet(X_train, y_train, nfolds = 10, alpha=1, type.measure = "mse")
cvfit_elastic = cv.glmnet(X_train, y_train, nfolds = 10, alpha=.6, type.measure = "mse")
cvfit_ridge = cv.glmnet(X_train, y_train, nfolds = 10, alpha=0, type.measure = "mse")
```

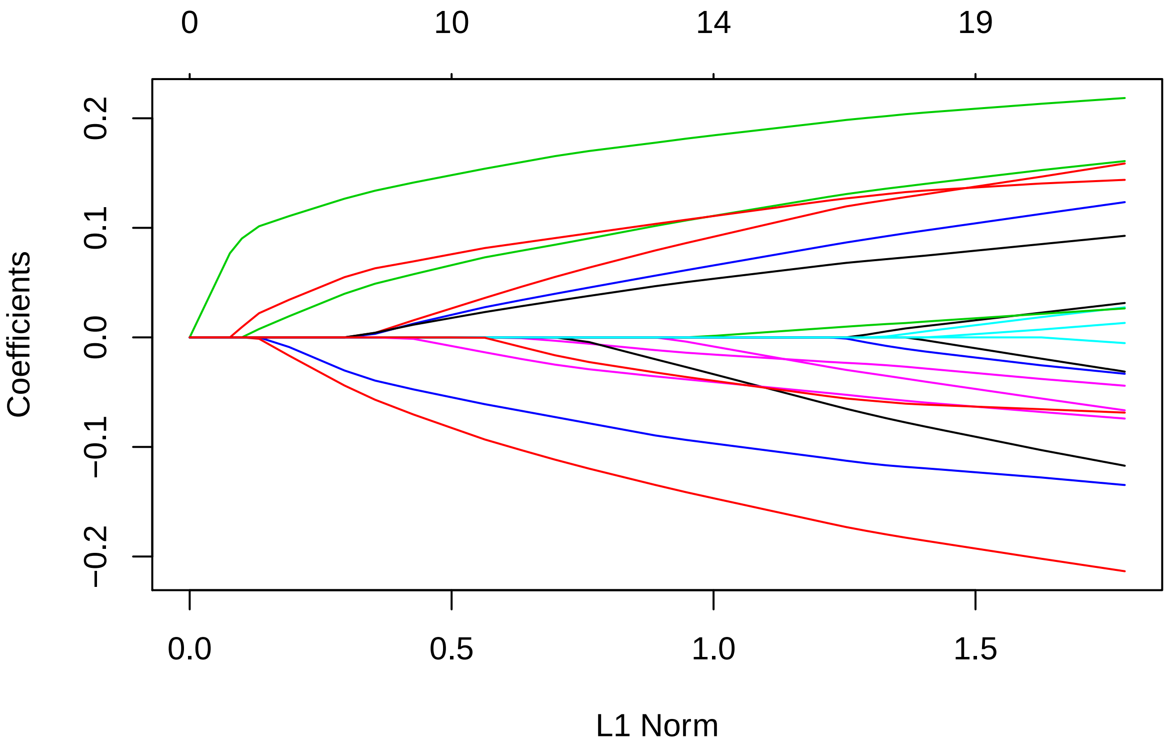
Plot the cross-validation error as a function of $log\lambda$ and trace curves of coefficients as a function of $log\lambda$.

```
# plot CV errors for 3 models

par(mfrow=c(2,2))
plot(cvfit_lasso, main="lasso");plot(cvfit_elastic, main="elastic, a=.6");plot(cvfit_ridge, main="ridge
plot(log(cvfit_lasso$lambda), cvfit_lasso$cvm, pch=19, col="red", xlab="log(Lambda)", ylab=cvfit_lasso$n
points(log(cvfit_elastic$lambda), cvfit_elastic$cvm, pch=19, col="grey")
points(log(cvfit_ridge$lambda), cvfit_ridge$cvm, pch=19, col="blue")
legend("topleft", legend=c("alpha= 1", "alpha= .6", "alpha 0"), pch=19, col=c("red", "grey", "blue"))
```

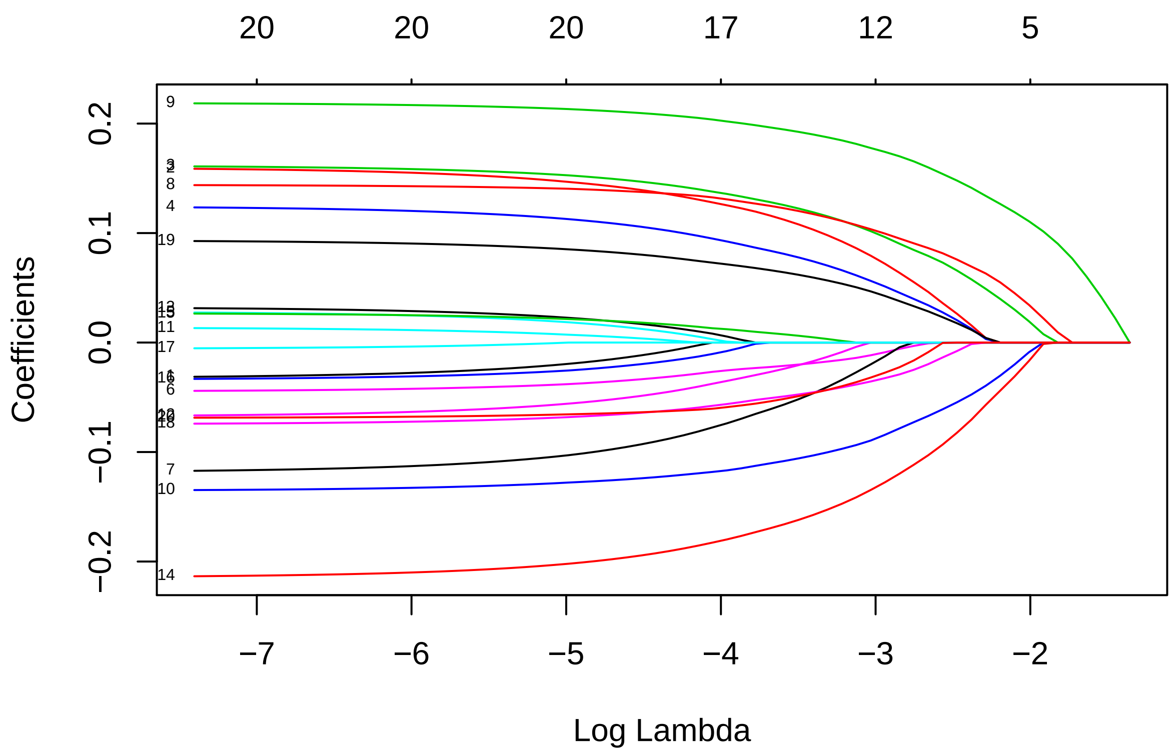



```
# plot coeff trace curves
# CANNOT BE DONE, I GET AN ERROR - "xvar" is not a graphical parameter

# plot(cvfit_lasso, xvar = "lambda", label=TRUE)
# plot(cvfit_elastic, xvar = "lambda")
# plot(cvfit_ridge, xvar = "lambda")

# above gives me error, even if test case work
# test case has a diferent object class "glmnet",
```

```
# compared to "cv.glmnet"
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
g2=sample(1:2,100,replace=TRUE)
g4=sample(1:4,100,replace=TRUE)
fit1=glmnet(x,y)
plot(fit1)
```




```
plot(fit1,xvar="lambda",label=TRUE)
```




```r
class(fit1)
```

```
## [1] "elnet"  "glmnet"
```

```r
class(cvfit_elastic)
```

```
## [1] "cv.glmnet"
```

```r
# Instead I can show non-zero features, for ridge - all features shrinked, but none to 0.
features_lasso = which(as.matrix(coef(cvfit_lasso, s = "lambda.min")) != 0)
length(features_lasso)
```

```
## [1] 9
```

```r
#which(as.matrix(coef(cvfit_ridge, s = "lambda.min")) != 0)
features_elastic = which(as.matrix(coef(cvfit_elastic, s = "lambda.min")) != 0)
length(features_elastic)
```

```
## [1] 16
```

As expected, lasso has the least number of non-zero features, that's why I would expect lasso's trace curve to shrink to 0 faster than elastic net's when log(lambda) changes.

Fit the final model on the training set and predict the response on the test dataset.

```r
# predict on test

y_hat_lasso = as.matrix(predict(cvfit_lasso, newx = X_test, s = "lambda.min", type="response"))
y_hat_elastic = as.matrix(predict(cvfit_elastic, newx = X_test, s = "lambda.min", type="response"))

# normalize to range [0, 1] to get probabilities of predicting label "1"
y_hat_lasso = (y_hat_lasso - min(y_hat_lasso)) / (max(y_hat_lasso) - min(y_hat_lasso))
```

```r
y_hat_elastic = (y_hat_elastic - min(y_hat_elastic)) / (max(y_hat_elastic) - min(y_hat_elastic))

# binarize predictions to either "0"" or "1""
y_hat_lasso = ifelse(y_hat_lasso >= 0.5, 1, 0)
y_hat_elastic = ifelse(y_hat_elastic >= 0.5, 1, 0)

# determine accuracy of both predictors
accur_lasso = length(which(y_hat_lasso == y_test))/length(y_test)
accur_lasso
```

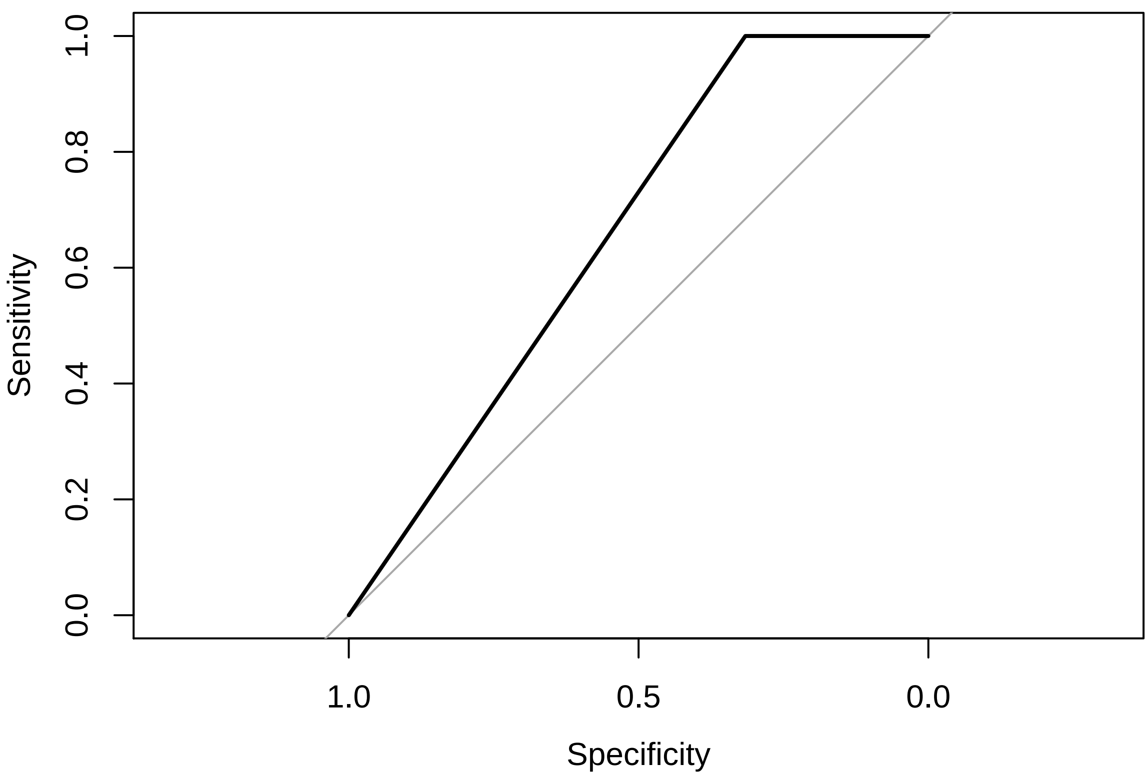
```
## [1] 0.6060606
```

```r
accur_elastic = length(which(y_hat_elastic == y_test))/length(y_test)
accur_elastic
```

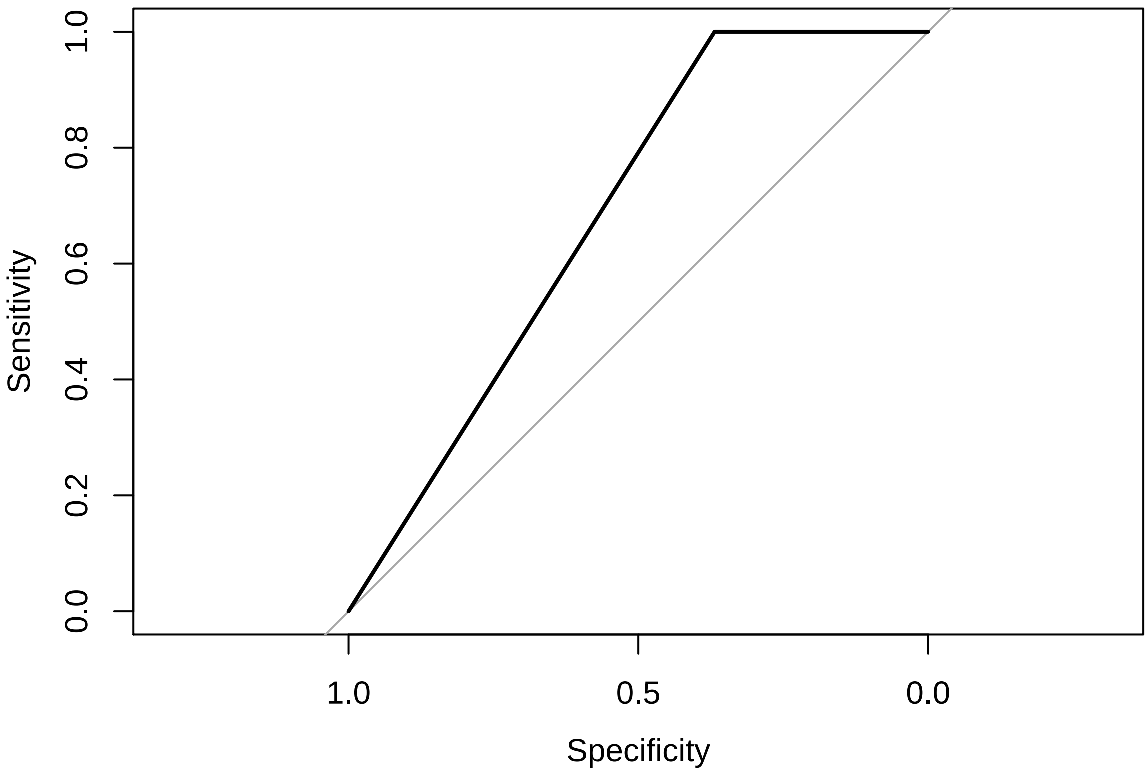
```
## [1] 0.6363636
```

Finally, plot the ROC curves for each case.

```r
# ROC lasso
roc_lasso = roc(response = as.vector(y_test), predictor = as.vector(y_hat_lasso), plot = T)
```




```r
# ROC elastic
roc_elastic = roc(response = as.vector(y_test), predictor = as.vector(y_hat_elastic), plot = T)
```



Comparison of lasso and elastic-net regularization and feature selection.

Lasso selected 9 features out of 231, and elastic net (a=0.6) selected 16. Unsurprisingly, the top 9 features (heaviest weight) in elastic net were the very same 9 features selected by lasso.

```
features_lasso
```

```
## [1]   1  16  23  76  89 125 132 138 232
```

```
features_elastic
```

```
##  [1]   1  16  23  31  65  76  89  96  98 104 112 125 132 138 164 232
```

```
cat("Intersection between lasso and elastic feature sets is:", intersect(features_lasso, features_elast
```

```
## Intersection between lasso and elastic feature sets is: 1 16 23 76 89 125 132 138 232
```

```
mask = which(as.matrix(coef(cvfit_lasso)) != 0)
cat("Coefficient values are:", as.matrix(coef(cvfit_lasso))[mask])
```

```
## Coefficient values are: 0.4874483 -0.01227717 -0.3816148
```

In terms of prediction performance, lasso's accuracy was: 0.6060606, compared to elastic net's: 0.6363636. However, after fitting the training data multiple times with both models, I can conclude that their predictions on a test set were almost identical, and would change dependending on the seed instance. At least for this run, areas under ROC were extremely similar too:

lasso: 0.6578947 elastic: 0.6842105