Exploratory Data Mining via Search Strategies Lab #2

Ross Jacobucci & Kevin J. Grimm

Outline

The second lab will go over some more recent techniques in regression –
1. Multivariate Adaptive Regression Splines
2. Regularized Regression

Ridge and Lasso Regression Including a penalty on the β parameters, and by varying the penalty we can shrink some of the $\beta's$ to zero, doing a form of "automatic" subset selection. \ Althought there a number of packages to do this, maybe the best is *glmnet*
Note, for glmnet, your data has to be set up in two separate matrices. Doing this can be accomplished by:

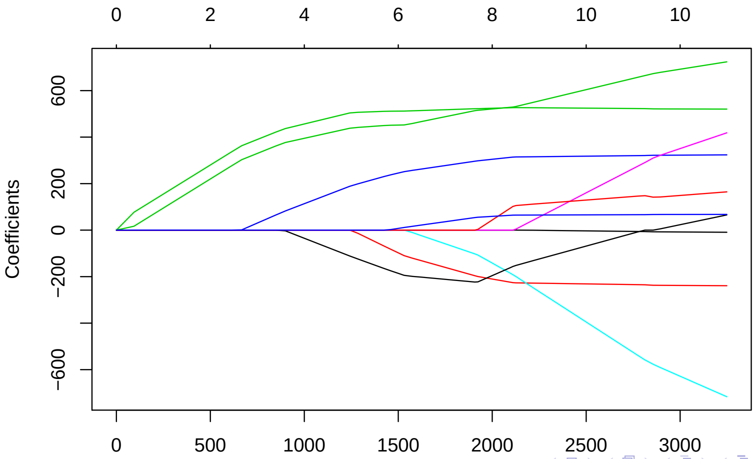
```
library(elasticnet);data(diabetes);
X <- diabetes$x;Y <- diabetes$y
diabetes2 <- data.frame(cbind(Y,X))
YY <- as.matrix(diabetes2$Y)
XX <- as.matrix(diabetes2[,2:11])
```

Two things to note:

1. Because we are doing regression with a continuous outcome, we specify the family(distribution) as "gaussian"
2. Shrinkage in lasso and ridge is sensitive to the scale of the variables, therefore, it is best to standardize the predictors before entering. glmnet does this by default (look at ?glmnet).

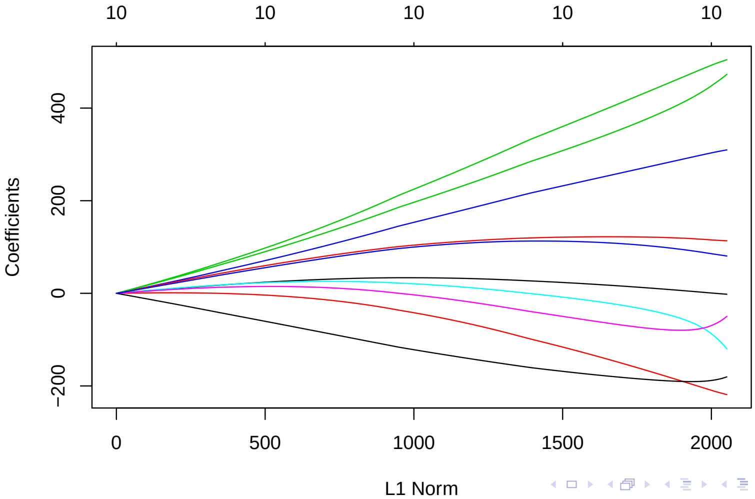
Lasso

```
library(glmnet)
# ?glmnet
lasso.out <- glmnet(XX,YY,family="gaussian",alpha=1)
plot(lasso.out)
```



Ridge

```
ridge.out <- glmnet(XX,YY,family="gaussian",alpha=0)
#plot(ridge.out,type.coef="2norm")
plot(ridge.out)
```



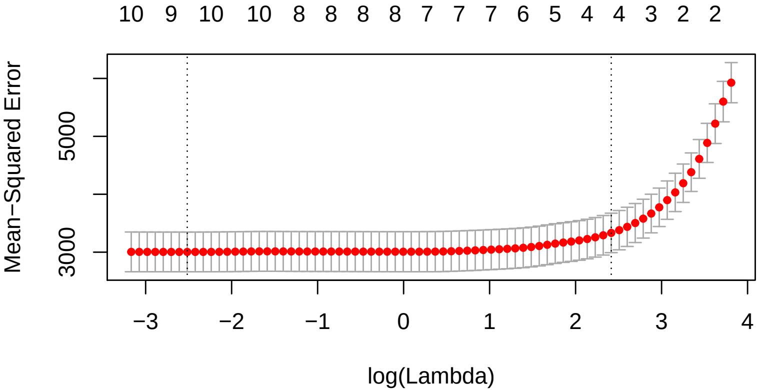
Regularized Regression Continued

Since ridge regression does not shrink the β coefficients to 0 with increase penalization, it does not do an "automatic" form of subset selection

The problem now becomes, which value of λ (amount of shrinkage) do we choose?

Using cross-validation is one of the better ways, and is implemented the glmnet package

```
cv.lasso <- cv.glmnet(XX,YY,family="gaussian",alpha=1)
plot(cv.lasso)
```



Choosing the Optimal Lambda (Penalty)

Two-strategies for selecting λ: either pick the lowest CV error, or the best solution within 1 standard error.
I don't think that there is a clear best choice. The one advantage of using the 1SE rule is that you need fewer predictors.In our example 4 instead of 7.

```
#str(cv.lasso)
(lmin <- cv.lasso$lambda.min)
```

```
## [1] 0.08077547
```

```
(lminSE <- cv.lasso$lambda.1se)
```

```
## [1] 11.18648
```

```
lasso.out2 = glmnet(XX,YY,family="gaussian",alpha=1,lambda=lminSE)
lasso.out2
```

```
##
## Call:  glmnet(x = XX, y = YY, family = "gaussian", alpha = 1, lambda = lminS
##
##       Df   %Dev Lambda
## [1,]   4 0.4504  11.19
```

Highly correlated predictors

```
library(lavaan)
sim.mod <- '
y ~ 1*x1 + 1*x2
x1~~0.999*x2
'
set.seed(3)
dat <- simulateData(sim.mod, model.type="sem",sample.nobs=100)

out <- lm(y ~ ., data=dat)
summary(out)
```

```
##
## Call:
## lm(formula = y ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.9320 -0.7527 -0.1309  0.7976  2.5258
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02609    0.10487   0.249    0.804
## x1           1.58969    2.30343   0.690    0.492
## x2           0.16792    2.30542   0.073    0.942
##
```

Ridge Regression

```r
library(glmnet)
X <- matrix(cbind(dat$x1,dat$x2),100,2)
Y <- data.matrix(dat$y)
ridge <- glmnet(X,Y,family="gaussian",alpha=0)
coef(ridge,s=0.2)
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 0.02293853
## V1          0.84523667
## V2          0.82491602
```

Lasso for correlated variables

```r
#library(glmnet)
lasso <- glmnet(X,Y,family="gaussian",alpha=1) # change alpha
coef(lasso,s=0.01)
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 0.02680360
## V1          1.71635450
## V2          0.02973456
```

Lasso doesn't have the same properties as ridge for collinear predictors. This is the rationale for the elastic net

Elastic Net

```
enet1 <- glmnet(X,Y,family="gaussian",alpha=0.5) # mixture
coef(enet1,s=0.01)
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept) 0.0232833
## V1          1.0197302
## V2          0.7301989
```

```
enet2 <- glmnet(X,Y,family="gaussian",alpha=0.5) # mixture
coef(enet2,s=0.2)
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 0.02326065
## V1          0.82358307
## V2          0.78180790
```

```
enet3 <- glmnet(X,Y,family="gaussian",alpha=0.5) # mixture
coef(enet3,s=2)
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 0.02767924
## V1          0.25954355
## V2          0.25609747
```

More P's than People

```
set.seed(1)
N <- 30; P <- 100
X <- matrix(rnorm(N*P),N,P)
Y <- rnorm(N)

out <- lm(Y ~ X)
head(summary(out)$coefficients)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.37504720        NaN     NaN      NaN
## X1          -0.08138609        NaN     NaN      NaN
## X2           4.79719809        NaN     NaN      NaN
## X3          -6.98146901        NaN     NaN      NaN
## X4          -4.93789272        NaN     NaN      NaN
## X5           1.50237557        NaN     NaN      NaN
```

Other parts of the summary list the errors and non-singularity of the information matrix. Can't invert a matrix that is wider than long.

Regularized Regression for P > N

Both Ridge and Lasso (& Enet) can handle this case

```r
# just use lasso
lasso2 <- glmnet(X,Y,alpha=1)
head(coef(lasso2,0.0001))
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) -0.16476453
## V1          -0.04102075
## V2           .
## V3           .
## V4           .
## V5           .
```

The addition of penalties effectively reduces the dimensionality of the parameter space. In this case, don't need much for penalty because a lot of coefficients are set immediately to zero.

P-values for Lasso

The traditional lasso does not output p-values. Only really assessing “importance” in the sense of what relationships we think with generalize through the use of cross-validation.

Need two new packages

```
library(lars)
library(covTest)
```

Relaxed Lasso

So the lasso has found to be biased in that it shrinks non-zero coefficients too much. To compensate for this, the relaxed lasso is a two step procedure in that the steps include:

1. Fit lasso, select non-zero coefficients
2. Re-fit linear regression with only non-zero coefficients included

Step 1

```
y <- data.matrix(mtcars$mpg)
x <- as.matrix(mtcars[,2:11],nrow(mtcars),10)
lasso.out1 <- cv.glmnet(x,y)
coef(lasso.out1,lasso.out1$lambda.1se)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 33.471391885
## cyl         -0.833427049
## disp          .
## hp          -0.005886408
## drat          .
## wt          -2.287815524
## qsec          .
## vs            .
## am            .
## gear          .
## carb          .
```

Relaxed Lasso Step 2

```r
lm.out <- lm(mpg ~ cyl + hp + wt,mtcars)
coef(lm.out)
```

```
## (Intercept)         cyl          hp          wt
##  38.7517874  -0.9416168  -0.0180381  -3.1669731
```

The coefficients are larger in step 2.