MATH 537 Final Exam

Gustavo Esparza

8/1/2019

```r
#LIBRARIES
require(pls)
library(nnet)
library(MASS)
library(glmnet)
```

SETUP

First, we must set up our data. Using our three given responses, we will create a new response that is the percent of rejected applicants plus the percent of accepted students that enrolled. Then we will modify our private university variable so that it has a binary numerical response (for the purpose of building models). Then we will create a dataframe that contains our new Exclusivity response and all of our potential predictors.

```r
College = read.csv("College.csv")
Apps = College$Apps
Accept = College$Accept
Enroll = College$Enroll
Exclusivity = 100*(Apps - Accept)/(Apps) + 100*(Enroll/Accept)

Priv = ifelse(College$Private=="Yes", 1, 0)

Y = Exclusivity
X = College[,6:19]
data3 = as.data.frame(cbind(Y,Priv,X))
```

Next, we will standardize our data to make computations simpler:

```r
data4 = as.data.frame(scale(data3))
Y=data4$Y
```

We can create a Design Matrix that will be used for Ridge,Lasso, and Elastic Net

```r
x= as.matrix(data4[,-1])
y= data4[,1]
```

For our Cross Validated measure of R^2, we will be splitting our original data into testing and training data (using 60% of data for training and the other 40% for testing)

```r
#CV Set Up
set.seed(100)
trainingRows = sample(1:nrow(data4), 0.6*nrow(data4))
training = data4[trainingRows, ]
test = data4[-trainingRows, ]

x.train = as.matrix(training[,-1])
y.train = training[,1]
x.test = as.matrix(test[,-1])
y.test = test[,1]
```

Now, we can begin our analysis of the 6 competing models.

Simple Linear Regression

First, we will build a linear model using all of our predictors. Here are the results:

```
model = lm(Y~.,data=data4)
summary(model)
```

```
##
## Call:
## lm(formula = Y ~ ., data = data4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3364 -0.5363 -0.0861  0.4472  5.2063
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.737e-16  3.097e-02   0.000  1.00000
## Priv        -1.454e-01  5.089e-02  -2.858  0.00437 **
## Top10perc    5.197e-01  8.065e-02   6.443 2.07e-10 ***
## Top25perc   -7.950e-02  7.317e-02  -1.087  0.27758
## F.Undergrad -1.321e-01  4.805e-02  -2.750  0.00610 **
## P.Undergrad  1.504e-01  4.025e-02   3.737  0.00020 ***
## Outstate    -4.033e-01  6.262e-02  -6.441 2.10e-10 ***
## Room.Board  -8.321e-03  4.350e-02  -0.191  0.84836
## Books        1.011e-01  3.263e-02   3.097  0.00202 **
## Personal    -6.719e-04  3.532e-02  -0.019  0.98483
## PhD         -8.711e-02  6.274e-02  -1.389  0.16538
## Terminal    -3.558e-02  6.213e-02  -0.573  0.56704
## S.F.Ratio    1.252e-01  4.267e-02   2.935  0.00344 **
## perc.alumni  7.060e-03  4.158e-02   0.170  0.86522
## Expend       2.618e-01  5.340e-02   4.902 1.16e-06 ***
## Grad.Rate    7.665e-02  4.165e-02   1.840  0.06611 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8633 on 761 degrees of freedom
## Multiple R-squared:  0.2691, Adjusted R-squared:  0.2547
## F-statistic: 18.68 on 15 and 761 DF,  p-value: < 2.2e-16
```

We can see that a good amount (8) of our predictors are deemed statistically significant in predicting our response. We can also note that a good amount of our coefficients are relatively small in scale. As for our R^2 value, we can see that it is about .2691, which is quite low. This is saying that all of our predictors combined are explaining about 27% of the variance in our response of Exclusicivity.

We must still use our CV training/test data to get an R^2 value that can be useful for comparisons across all models. Here is the code for that CV R^2 measure:

```
#CV R^2
m_lm = lm(Y~., data = training)
p_lm = predict(m_lm, test[,-1]) - test[,1]
sst   = test[,1] - mean(test[,1])
```

```
SSE = sum(p_lm^2)
SST = sum(sst^2)
r_sqrd = 1-SSE/SST

cat("CV R^2 for PCR = ",r_sqrd,"")
```

```
## CV R^2 for PCR =  0.1871986
```

we can now see that using our training data to explain the variance in our test response, we are now explaining about 19% of the variance. This is a bit lower than our original R^2, and we will keep this in mind as we move onto our next model.

PCR

Now, we will be using Principal Component Regression to see if our Predictors can be represented as orthogonal components of a quantity less than our original predictor size that still contains the majority of their information. Here is the summary for our Principal Component regression:

```
pcr_model = pcr(Y~.,data=data4,validation= "CV")
summary(pcr_model)
```

```
## Data:    X dimension: 777 15
##  Y dimension: 777 1
## Fit method: svdpc
## Number of components considered: 15
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.001    1.003   0.9633   0.9609   0.9270   0.9312   0.9290
## adjCV        1.001    1.003   0.9630   0.9605   0.9256   0.9307   0.9287
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.9247   0.9199   0.9237    0.9288    0.9273     0.895    0.8827
## adjCV   0.9232   0.9192   0.9230    0.9281    0.9298     0.894    0.8815
##        14 comps  15 comps
## CV       0.8872    0.8799
## adjCV    0.8858    0.8785
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X  35.8602   55.413   63.335    69.54    75.19    79.84    83.92    87.77
## Y   0.1171    8.101    8.681    14.93    14.93    15.66    17.64    17.76
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    90.79     93.35     95.39     97.25     98.47     99.43    100.00
## Y    17.81     17.93     18.24     23.44     25.71     25.72     26.91
```

From our summary, we can see that 7 components allow for about 80% of the variance in our predictors to be explained. This is a safe component level to stop at in order to ensure that a majority of our data is still being captured. 80% appears to be an agreed upon value, and any extra components will not be contributing all that much more to the model. The intent of PCR is to reduce our dimensioanlity, so we will proceed with a 7 component model.

For our R^2 comparison, we can look at our summary table and use the Y % of variance explained value for 7 components = 17.64. Thus, with our Principal Component model using 7 components, we have explained about 18% of the variance in our response. We can also see this computation more in depth as follows:

```
SST = var(data4$Y*(776))
SSR = var(pcr_model$residuals[4663:5439]*776)
R2 = (SST-SSR)/(SST)
R2
```

```
## [1] 0.1763937
```

Again, we must still compute a Cross Validated measure of R^2 for comparisons amongst our models, which can be seen as follows:

```
#CV R^2
m_pcr = pcr(Y~., data = training)
p_pcr = predict(m_pcr, test[,-1], ncomp = 7) - test[,1]
sst   = test[,1] - mean(test[,1])

SSE = sum(p_pcr^2)
SST = sum(sst^2)
r_sqrd = 1-SSE/SST

cat("CV R^2 for PCR = ",r_sqrd,"")
```

```
## CV R^2 for PCR =  0.1077957
```

Our cross validated percent of explained variance in our response is now about 11 percent. Through PCR, we attempted to reduce our dimensionality of predictors and also fix any issues of multicollinearitym but it appears that our model is not as competitive as OLS so far.

We could attempt to add more components, but that would seem to defeat the purpose of PCR. Perhaps this is implying that multicollinearity may not be a primary issue amongst our predictors.

PLSR

Now, we will be using Partial Least Squares Regression to see if our Predictors can be represented as orthogonal components of a quantity less than our original predictor size that still contains the majority of their information. In addition to this, PLSR also attempts to provide for some correlation to our dependent variable. Here is the summary for our Partial Least Squares regression:

```
plsr_model = plsr(Y~.,data=data4,validation="CV")
summary(plsr_model)
```

```
## Data:    X dimension: 777 15
##  Y dimension: 777 1
## Fit method: kernelpls
## Number of components considered: 15
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.001   0.9332   0.9001   0.8934   0.8835   0.8802   0.8794
## adjCV        1.001   0.9325   0.9004   0.8927   0.8821   0.8790   0.8781
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.8782   0.8777   0.8781    0.8779    0.8778    0.8777    0.8777
## adjCV   0.8769   0.8765   0.8768    0.8766    0.8765    0.8765    0.8765
##        14 comps  15 comps
## CV       0.8777    0.8777
## adjCV    0.8765    0.8765
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    18.88    37.01    60.50    64.26    69.63    74.47    76.56    79.65
## Y    14.86    20.68    23.36    26.01    26.52    26.70    26.87    26.90
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    82.04     84.78     88.99     92.87     95.65     97.78    100.00
## Y    26.91     26.91     26.91     26.91     26.91     26.91     26.91
```

From our summary, we can see that 8 components allow for about 80% of the variance in our predictors to be explained. This is a safe component level to stop at in order to ensure that a majority of our data is still being captured. 80% appears to be an agreed upon value, and any extra components will not be contributing all that much more to the model. The intent of PLSR is to reduce our dimensionality, so we will proceed with a 8 component model.

For our R^2 comparison, we can look at our summary table and use the Y % of variance explained value for 8 components = 26.9. Thus, with our Principal Component model using 8 components, we have explained about 27% of the variance in our response. This is just about the R^2 value from our simple linear regression. We can also see this computation more in depth as follows:

```
SST = var(data4$Y*(776))
SSR = var(plsr_model$residuals[5440:6216]*776)
R2 = (SST-SSR)/(SST)
R2
```

```
## [1] 0.2689861
```

Again, we must still compute a Cross Validated measure of R^2 for comparisons amongst our models, which can be seen as follows:

```
#CV R^2
m_plsr = plsr(Y~., data = training)

p_plsr = predict(m_plsr, test[,-1], ncomp = 7) - test[,1]
sst    = test[,1] - mean(test[,1])
SSE = sum(p_plsr^2)
SST = sum(sst^2)

r_sqrd = 1-SSE/SST

cat("CV R^2 for PLSR = ",r_sqrd,"")
```

```
## CV R^2 for PLSR =  0.1859897
```

Our cross validated percent of explained variance in our response is now about 19 percent. Through PLSR, we attempted to reduce our dimensionality of predictors and also fix any issues of multicollinearity and also account for some correlation to our response. We have performed better than PCR, which can be attributed to the fact that PLSR seems to consider our response to some extent, while PCR does not at all.

We could attempt to add more components(this was attempted and did not produce a much better result), but that would seem to defeat the purpose of PLSR. While we have improved our model to some extent, perhaps creating new components is not quite the path we should be taking and we will begin to focus on shrinkage/regularization as another possible method.

RIDGE

For Ridge regression, we will be considering a penalty for insignificant/mildly effective predictors to see if new coeffecients can improve our model.

Here are the coeffecients from our omptimized Ridge Regression:

```
#Cross Validated Lambda with all data
#fit.ridgeCV = cv.glmnet(x, y, alpha=0,family = 'gaussian',standardize=F)

lambdaRidgeCV = cv.glmnet(data.matrix(x.train),y.train,alpha=0,nfolds=10)$lambda.min
lambdaRidgeCV = 0.06085495 #For consistency
cat("Ridge Lambda = ",lambdaRidgeCV,"\n\n")
```

```
## Ridge Lambda =  0.06085495
```

```
#For Coeffecients
fit.ridge =  glmnet(x, y, alpha=0,lambda=lambdaRidgeCV,family='gaussian',standardize=F)
coef(fit.ridge)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  3.108119e-16
## Priv        -1.410198e-01
## Top10perc    3.835511e-01
## Top25perc    1.406406e-02
## F.Undergrad -9.293888e-02
## P.Undergrad  1.263986e-01
## Outstate    -3.200545e-01
## Room.Board  -2.507861e-02
## Books        9.910647e-02
## Personal     5.485188e-03
## PhD         -6.578419e-02
## Terminal    -5.177019e-02
## S.F.Ratio    1.056256e-01
## perc.alumni  1.333015e-03
## Expend       2.333716e-01
## Grad.Rate    5.848419e-02
```

We can see that our coeffecients have been resized, some are smaller than our linear regression counterparts while others are larger. This was done in hopes that adding more or less importance to certain predictors will prove to be more beneficial in explaining the variance of our response variable.

Now, we will use our Ridge regression model to compute an overall R^2, as well as a Cross Validated R^2 to compare amongst our models. Here is the code:

```
#fit.ridge = fit.ridgeCV$glmnet.fit

#full R^2
y_predicted = predict(fit.ridge, s = lambdaRidgeCV, newx = x)

#Sum of Squares Total and Error
sst <- sum((y - mean(y))^2)
```

```r
sse <- sum((y_predicted - y)^2)

#R-squared
rsq <- 1 - sse / sst
cat("Full Model R^2 = ",rsq,"\n\n")
```

```
## Full Model R^2 =  0.2633336
```

```r
#cv R^2
fit.ridge =  glmnet(data.matrix(x.train), y.train, alpha=0,lambda=lambdaRidgeCV,
                    family='gaussian',standardize=F)
y_predicted = predict(fit.ridge, s = lambdaRidgeCV, newx = x.test)

#Sum of Squares Total and Error
sst <- sum((y.test - mean(y.test))^2)
sse <- sum((y_predicted - y.test)^2)

#R-squared
rsq <- 1 - sse / sst
cat("Cross Validated R^2 = ",rsq,"\n")
```

```
## Cross Validated R^2 =  0.1976479
```

We can see that our overall and Cross Validated R^2 are somewhat close to each other and also a bit closer to our original R^2 value from the linear regression model. What we are seeing is, by adding more and less importance to certain predictors, we are better explaining our response variance in a cross validated environment. Thus far, our CV R^2 for ridge regression appears to be our top competitor.

LASSO

For Lasso regression, we will be considering a penalty for insignificant/mildly effective predictors to see if new coeffecients can improve our model. This is similar to Ridge, but Lasso uses a penalty that often times will eliminate certain predictors entirely rather than simply reducing their impact on the overall model.

Here are the coeffecients from our omptimized Lasso Regression:

```r
#Cross Validated Lambda with all data
#fit.lassoCV = cv.glmnet(x, y, alpha=1,family = 'gaussian',standardize=F)

lambdalassoCV = cv.glmnet(data.matrix(x.train),y.train,alpha=1,nfolds=10)$lambda.min
lambdalassoCV = 0.00685636 #For Consistency
cat("Lasso Lambda = ",lambdalassoCV,"\n\n")
```

```
## Lasso Lambda =  0.00685636
```

```r
#For Coeffecients
fit.lasso =  glmnet(x, y, alpha=1,lambda=lambdalassoCV,family='gaussian',standardize=F)
coef(fit.lasso)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.911518e-16
## Priv        -1.268766e-01
## Top10perc    4.496653e-01
## Top25perc   -1.759771e-02
## F.Undergrad -1.004132e-01
## P.Undergrad  1.281620e-01
## Outstate    -3.854004e-01
## Room.Board  -1.284366e-03
## Books        9.328497e-02
## Personal     .
## PhD         -7.306011e-02
## Terminal    -3.811747e-02
## S.F.Ratio    1.077400e-01
## perc.alumni  .
## Expend       2.431197e-01
## Grad.Rate    5.419414e-02
```

We can see our penalty λ is abour .006 and it has indeed impacted our model coeffecients significantly. We can see that certain predictors are either increased or decreased by a power of ten or so, and two predictors (Personal and perc.alumni) have been removed entirely. This new rearrangement will hopefully provide similar or better results than our Ridge Regression (via our R^2 comparison).
Now, we will use our Lasso regression model to compute an overall R^2, as well as a Cross Validated R^2 to compare amongst our models. Here is the code:

```r
#fit.lasso = fit.lassoCV$glmnet.fit

#full R^2
y_predicted = predict(fit.lasso, s = lambdalassoCV, newx = x)

#Sum of Squares Total and Error
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#R-squared
rsq <- 1 - sse / sst
cat("Full Model R^2 = ",rsq,"\n\n")
```

```
## Full Model R^2 =  0.2669876
```

```r
#cv R^2
fit.lasso =  glmnet(data.matrix(x.train), y.train, alpha=0,lambda=lambdalassoCV,
                    family='gaussian',standardize=F)
y_predicted = predict(fit.lasso, s = lambdalassoCV, newx = x.test)

#Sum of Squares Total and Error
sst <- sum((y.test - mean(y.test))^2)
sse <- sum((y_predicted - y.test)^2)

#R-squared
rsq <- 1 - sse / sst
cat("Cross Validated R^2 = ",rsq,"\n")
```

```
## Cross Validated R^2 =  0.1898308
```

Again, we can see that Lasso (similar to Ridge) regression has improved the overall explanation of the variance in our Exclusicivity response, as compared to the Cross Validated methods evaluated before. Our cross validated R^2 is slightly lower than our Ridge regression counterpart, although the difference is negligible. What is important to note in this comparison is the fact that Lasso is using less predictors yet still explaining our response variance just as effective. This can be seen as implying that those two predictors (Personal and Perc.alumni) are rather weak and can be dropped from the overall model without losing any information if the weights of our coeffecients are redistributed.

ELASTIC NET

Finally, we will be evaluating the effectiveness of our last model : Elastic Net. Elastic Net can be seen as somewhat of a middle ground between Ridge and Lasso Regression. We are still using a penalty for our predictor coeffecients, but our scale can be a value (α and λ) that resides between the prior two penalties. The idea is the same, but our penalty severity will be adjusted to a certain extent.

Here are the coeffecients from our omptimized Elastic Net Regression:

```
#Cross Validated Lambda with all data
#fit.elasticCV = cv.glmnet(x, y, alpha=.5,family = 'gaussian',standardize=F)
lambdaelasticCV = cv.glmnet(data.matrix(x.train),y.train,alpha=0.5,nfolds=5)$lambda.min
lambdaelasticCV = 0.02683862 # for consistency

#For Coeffecients
fit.elastic =  glmnet(x, y, alpha=.5,lambda=lambdaelasticCV,family='gaussian',standardize=F)
cat("Elastic Net Lambda = ",lambdaelasticCV,"\n\n")
```

```
## Elastic Net Lambda =  0.02683862
```

```
coef(fit.elastic)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.786586e-16
## Priv        -1.106844e-01
## Top10perc    4.050382e-01
## Top25perc    .
## F.Undergrad -5.848386e-02
## P.Undergrad  1.035402e-01
## Outstate    -3.477771e-01
## Room.Board   .
## Books        8.717967e-02
## Personal     .
## PhD         -5.481541e-02
## Terminal    -3.862017e-02
## S.F.Ratio    8.733580e-02
## perc.alumni  .
## Expend       2.109585e-01
## Grad.Rate    3.126791e-02
```

We can see our penalty λ is abour .026 and it has indeed impacted our model coeffecients significantly. We can see that certain predictors are either increased or decreased by a power of ten or so, and 4 predictors (Top 25perc , Room.Board, Personal and perc.alumni) have been removed entirely. It is important to note that all predictors dropped in Lasso are also dropped here in the Elastic Net regression. This is a good sign that Elastic Net is penalizing predictors in a manner that will produce similarly strong explanation of response variance.

Now, we will use our Elastic Net regression model to compute an overall R^2, as well as a Cross Validated R^2 to compare amongst our models. Here is the code:

```
#fit.elastic = fit.elasticCV$glmnet.fit
```

```r
#full R^2
y_predicted = predict(fit.elastic, s =lambdaelasticCV, newx = x)

#Sum of Squares Total and Error
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#R-squared
rsq <- 1 - sse / sst
cat("\n Full Model R^2 = ",rsq,"\n")
```

```
##
##  Full Model R^2 =  0.2607876
```

```r
#cv R^2
fit.elastic =  glmnet(data.matrix(x.train), y.train, alpha=0,
                      lambda=lambdaelasticCV,family='gaussian',standardize=F)
y_predicted = predict(fit.elastic, s = lambdaelasticCV, newx = x.test)

#Sum of Squares Total and Error
sst <- sum((y.test - mean(y.test))^2)
sse <- sum((y_predicted - y.test)^2)

#R-squared
rsq <- 1 - sse / sst
cat("Cross Validated R^2 = ",rsq,"\n")
```

```
## Cross Validated R^2 =  0.1947085
```

Again, we can see that Elastic Net (similar to Ridge and Lasso) regression has improved the overall explanation of the variance in our Exclusicivity response, as compared to the Cross Validated methods evaluated before. Our cross validated R^2 is slightly lower than our Ridge and Lasso regression counterpart, although the difference is negligible. What is important to note in this comparison is the fact that Elastic is using less predictors than both Ridge and Lasso yet still explaining our response variance just as effective. This can be seen as implying that those four predictors (Top 25perc , Room.Board, Personal and perc.alumni) are rather weak and can be dropped from the overall model without losing any information if the weights of our coeffecients are redistributed accordingly.

Discussion

When trying to decide which model would be most effective in explaining our response of Exclusivity, we essentially had three basic outcomes:

1.) Our full model was the most effective and no adjustments were neccessary.

2.) There was an issue of collinearity amongst our predictors and we needed new orthogonal components that could reduce the dimensionality of our predictor space will still effectively explaining our response.

3.) Rather than collinearity being a major issue, we simply had predictors that weren't too useful, thus reorganizing and/or dropping the impact of our predictors would improve our model.

We have ultimately observed that the third outcome was in fact what occured, as elastic net regression dropped four total predictors and still maintained a significantly large percent of response variance explained.

With our Simple Linear Regression (1) we saw that a Cross Validated R^2 computation proved to not be as strong as the value found when considering all of our data. This implied that there could be a better method in our inventory that could improve our model.

With a component regression method (PCR and PLSR from option 2), we wished to see if collinearity could be fixed (if existing) and reduce dimensionality while still maintaining a strong model. However, we saw that, even when keeping a number of components which explained 80% of the variance in our predictors, our cross validated R^2 measure was lower than desired. As such, our component models were ineffective in optimizing the percent of response variance explained. It should be noted that PSLR did perform better than PCR, and we can attribute this to the fact that PLSR also attempts to explain the correlation to our response, while PCR only attempts to create components solely considering our predictors.

Using a shrinkage/regularization (Ridge,Lasso,Elastic Net from option 3) proved to be the optimal route. Within all three of these methods, we saw our cross validated R^2 values being maximized closer to our overall computed R^2. These comparisons were not too difficult to make, Ridge, Lasso and Elastic Net were far more beneficial than a component regression or a simple linear regression.

While our three shrinkage/regularization methods were all effective, we should also consider how these methods chose to reorganize our predictor importance. Ridge regression chose to keep all original predictors, Lasso and Elastic Net dropped 2 and 4 overall predictors, respectively. A desired outcome is to optimally explain our response using a minimal amount of predictors, so Elastic Net appears to be the overall best method out of our choice of 6 methods.