Exercise 7 - Dimensionality

Ridge, LASSO, PCR and PLS

O’Malley, Conie

2025-02-12

Table of Contents

## Preparation

Download the **Ex7\_Dimensionality\_YourLastName.Rmd** R Markdown file to your working directory, rename it to include your last name and follow the instructions below. When you finish, knit it into an Word or PDF file and upload it.

Also, enter the following commend to set the Random Number Generator default:

RNGkind(sample.kind="default")

## 1. Ridge Regressions

**1.1 Data Work**

In class, we fitted a Ridge regression model with a quantitative outcome, which is an alternative to OLS when there are a large number of predictors, high dimensionality and multi-collinearity. In this exercise, you will do the same, but with a Ridge Logistic regression. We will use the **Heart.csv** South African Heart Disease data we used earlier to predict **chd** (1 - coronary heart disease; 0 no heart disease). You can view the dataset documentation at: <https://web.stanford.edu/~hastie/ElemStatLearn/datasets/SAheart.info.txt>

Let’s start by loading the glmnet() package, reading the data and creating the **x predictor matrix** and **y outcome**. As mentioned in class use [,-1] in the model.matrix() function to remove the first column, which contains 1’s representing the intercept, otherwise you will have 2 intercepts in the Ridge output.

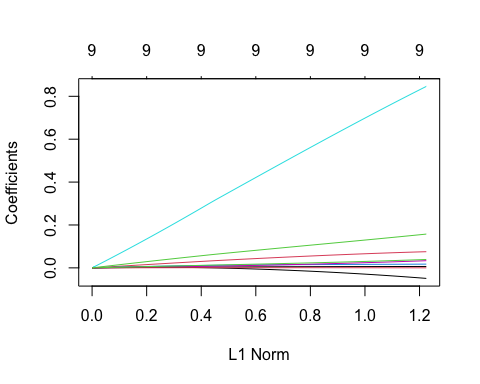
# Done for you  
  
options(scipen=4)  
  
library(glmnet) # Contains functions for Ridge and LASSO  
heart <- read.table("/Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Main/023. Programming Tools/R Studio/PredictiveAnalytics/R\_Exercises/data/Heart.csv", sep=",", head=T)  
  
x=model.matrix(chd~.,data=heart)[,-1]  
y=heart$chd

**1.2 Fit Ridge Model**

Fit the Ridge model with x and y, using the attribute family="binomial". Name the resulting glmnet() object **ridge.logit** and then plot() it. This plot graphs the **L2 Norm** graph (incorrectly labeled as L1 Norm) showing how the coefficients shrink (towards the left) as lambda goes up (to the left). To the far right are the OLS coefficients. To the far left is the null model. Notice how none of the coefficients become 0 until lambda is infitity (far left).

Technical note: when we fit logit models with glm() we use family=binomial(link="logit"), but the glmnet() function is different and only requires family="binomial".

ridge.logit <- glmnet::glmnet(x, y, alpha = 0, family = "binomial") # fit ridge logit model  
plot(ridge.logit) # plot model



**1.3 Cross-Validation**

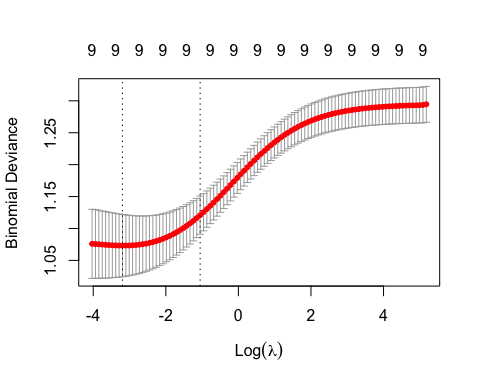
Enter set.seed(1) to get the same results shown in the solution.

Then use the cv.glmnet() function to create a **10-Fold Cross Validation** (default) object for this model. Don’t forget to include the family="binomial" attribute. Name this object **ridge.logit.cv10**.

Then use the cbind() function to list this object’s **cvm** (mean 2LL cross-validation). Name these two columns **“Lambda”** and **“10FCV”** respectively. Notice that the output shows 100 different lambda values, each with their respective 10FCV deviance.

Finally, plot() the **ridge.logit.cv10** object. Notice that the plot shows log(lambda) rather than lambda, which is for display purposes.

set.seed(1) # assign set seed  
ridge.logit.cv10 <- cv.glmnet(x, y, alpha = 0, family = "binomial") # fit cv ridge logit model  
cv10 <- cbind(Lambda = ridge.logit.cv10$lambda, `10FCV` = ridge.logit.cv10$cvm) # create cv10 dataframe  
plot(ridge.logit.cv10) # plot model



**1.4 Tuning the Model**

In the output above, you will notice that the models are ordered from high to low shrinkage, using 100 different lambdas. Also, note that the lowest **10FCV** deviance is for **Lambda=0.04099545** (your results may deviate from mine slightly due to sampling). This means that the best Ridge model is one with very little shrinkage, pretty close to a plain Logit model. The plot also shows that the best lambda is close to 0 (i.e., most negative log(lambda)). Let’s find this **Best Lambda**, but this time computationally.

Retrieve the $lambda.min value from **ridge.logit.cv10** and store it in an object named **ridge.best.lambda**.

Then, find the corresponding **10FCV** for this best lambda using the min() function on the **ridge.logit.cv10$cvm** vector. Store this value in an object named **min.cv.ridge**. This result should provide the **10FCV** value corresponding to the **Best Lambda**.

Finally, use the cbind function to display the "Best Lambda"=ridge.best.lambda, "Best Log Lambda"=log(ridge.best.lambda) (to be able to spot this value in the graph) and "Best 10FCV" = min.cv.ridge.

ridge.best.lambda <- ridge.logit.cv10$lambda.min # find min lambda value  
ridge.best.lambda

## [1] 0.04099545

min.cv.ridge <- min(ridge.logit.cv10$cvm) # find min 2LL average  
min.cv.ridge

## [1] 1.073231

cbind("Best Lambda" = ridge.best.lambda, "Best Log Lambda" = log(ridge.best.lambda), "Best 10FCV" = min.cv.ridge)

## Best Lambda Best Log Lambda Best 10FCV  
## [1,] 0.04099545 -3.194294 1.073231

**1.5 Comparing Coefficients**

Use the coef() function to retrieve the **Ridge Regression** coefficients from the **ridge.logit** model object for the **Best Lambda** using s=ridge.best.lambda and store the resulting coefficient vector in an object named **ridge.coef**.

Also, retrieve the **Plain Logit** coefficients from **ridge.logit**, this time using s=0 and naming the vector **ridge.coef.0**.

Now use the cbind() function to display 4 coefficient vectors: ridge.coef (log-odds, Ridge), exp(ridge.coef) (odds, ridge), ridge.coef.0 (log-odds plain Logit), and exp(ridge.coef.0) (odd, plain Logit) and store the results in an object named **all.coefs**. Then use colnames(all.coefs) to name the resulting columns **Best Ridge**, **Odds**, **0-Lambda Ridge**, **Odds**. Then display **all.coefs**.

ridge.coef <- coef(ridge.logit, s=ridge.best.lambda) # ridge regression coefficients for best lambda  
ridge.coef.0 <- coef(ridge.logit, s=0) # plain logit ridge regression coefficients  
cbind(ridge.coef, exp(ridge.coef), ridge.coef.0, exp(ridge.coef.0)) -> all.coefs # create matrix vector  
colnames(all.coefs) <- c("Best Ridge", "Odds", "0-Lambda Ridge", "Odds") # assign column names  
all.coefs

## 10 x 4 Matrix of class "dgeMatrix"  
## Best Ridge Odds 0-Lambda Ridge Odds  
## (Intercept) -5.3186664789 0.004899283 -5.7221730429 0.003272592  
## sbp 0.0062893712 1.006309191 0.0064556098 1.006476492  
## tobacco 0.0704757286 1.073018526 0.0753889131 1.078303436  
## ldl 0.1421137485 1.152707760 0.1574967615 1.170576967  
## adiposity 0.0167321735 1.016872940 0.0174298051 1.017582591  
## famhistPresent 0.7664077959 2.152021851 0.8461807201 2.330728128  
## typea 0.0280980909 1.028496566 0.0334345091 1.033999724  
## obesity -0.0376375020 0.963061986 -0.0491364514 0.952051212  
## alcohol 0.0003693287 1.000369397 0.0002284054 1.000228431  
## age 0.0339089740 1.034490437 0.0391317985 1.039907533

**1.6 Discussion**

Do you see much difference between the Ridge coefficients for Best Lambda and plain Logit? Why do you think that is?

#### Analysis

The Ridge regression coefficients for Best Lambda and plain Logit are quite similar, which suggests that the regularization parameter is not having a significant impact on the model. This could be because the data is not highly correlated or because the sample size is large enough to provide stable estimates of the coefficients.

**1.7 Interpretation**

Provide a brief interpretation for the **famhistPresent** (binary) in the **Ridge Regression**, both in terms of **log-odds** and **odds**. The interpretation is similar to one for a regular logit model.

#### Analysis

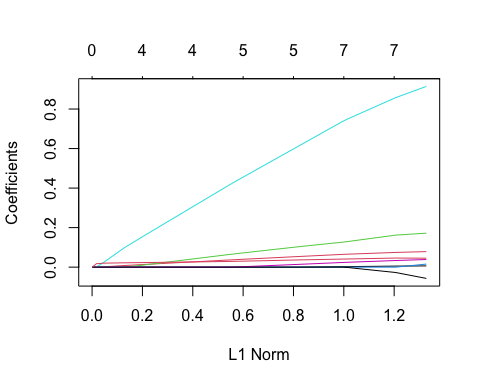
The log-odds of having coronary heart disease increases by 0.766 for each unit increase in famhistPresent. This means that the odds increases by a factor of for each unit increase in famhistPresent.

## 2. LASSO Regression

**2.1 LASSO**

Repeat the entire analysis in 1 above, but this time using LASSO. But notice in the plot that some coefficients drop out of the model as lambda grows, before becoming infinity. That’s the key difference with Ridge.

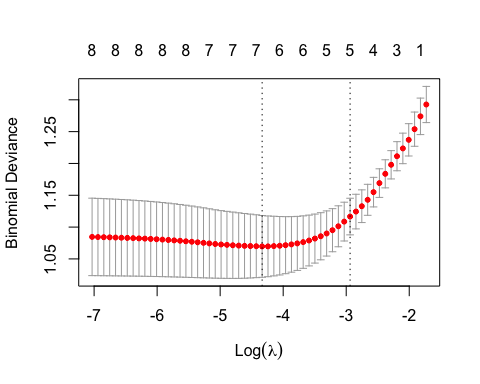
# Done for you  
  
# LASSO Logit model:  
lasso.logit <- glmnet(x, y, alpha=1, family="binomial")   
plot(lasso.logit)



set.seed(1)  
  
lasso.logit.cv10 <- cv.glmnet(x, y, alpha=1, family="binomial")  
cbind("Lambda"=lasso.logit.cv10$lambda, "10FCV"=lasso.logit.cv10$cvm)

## Lambda 10FCV  
## [1,] 0.1774595083 1.292630  
## [2,] 0.1616944946 1.274015  
## [3,] 0.1473300014 1.253870  
## [4,] 0.1342416101 1.237113  
## [5,] 0.1223159555 1.223631  
## [6,] 0.1114497431 1.211369  
## [7,] 0.1015488551 1.197902  
## [8,] 0.0925275347 1.183698  
## [9,] 0.0843076435 1.169120  
## [10,] 0.0768179848 1.154929  
## [11,] 0.0699936868 1.142871  
## [12,] 0.0637756406 1.132823  
## [13,] 0.0581099884 1.124372  
## [14,] 0.0529476572 1.116556  
## [15,] 0.0482439333 1.108631  
## [16,] 0.0439580752 1.101266  
## [17,] 0.0400529610 1.095121  
## [18,] 0.0364947663 1.089966  
## [19,] 0.0332526718 1.085612  
## [20,] 0.0302985961 1.081948  
## [21,] 0.0276069524 1.078923  
## [22,] 0.0251544268 1.076431  
## [23,] 0.0229197768 1.074383  
## [24,] 0.0208836470 1.072815  
## [25,] 0.0190284014 1.071601  
## [26,] 0.0173379707 1.070708  
## [27,] 0.0157977131 1.070192  
## [28,] 0.0143942878 1.069764  
## [29,] 0.0131155390 1.069669  
## [30,] 0.0119503906 1.069949  
## [31,] 0.0108887509 1.070209  
## [32,] 0.0099214243 1.070518  
## [33,] 0.0090400323 1.070927  
## [34,] 0.0082369407 1.071475  
## [35,] 0.0075051936 1.072073  
## [36,] 0.0068384529 1.072700  
## [37,] 0.0062309436 1.073448  
## [38,] 0.0056774037 1.074366  
## [39,] 0.0051730388 1.075255  
## [40,] 0.0047134803 1.076105  
## [41,] 0.0042947477 1.076942  
## [42,] 0.0039132142 1.077737  
## [43,] 0.0035655750 1.078476  
## [44,] 0.0032488191 1.079142  
## [45,] 0.0029602030 1.079769  
## [46,] 0.0026972267 1.080356  
## [47,] 0.0024576125 1.080900  
## [48,] 0.0022392850 1.081402  
## [49,] 0.0020403531 1.081863  
## [50,] 0.0018590938 1.082276  
## [51,] 0.0016939370 1.082648  
## [52,] 0.0015434523 1.082988  
## [53,] 0.0014063363 1.083305  
## [54,] 0.0012814012 1.083603  
## [55,] 0.0011675651 1.083884  
## [56,] 0.0010638418 1.084140  
## [57,] 0.0009693330 1.084375  
## [58,] 0.0008832202 1.084592

plot(lasso.logit.cv10)



lasso.best.lambda <- lasso.logit.cv10$lambda.min   
min.cv.lasso <- min(lasso.logit.cv10$cvm)  
  
cbind("Best Lambda"=lasso.best.lambda, "Best Log Lambda"=log(lasso.best.lambda), "Best 10FCV" = min.cv.lasso)

## Best Lambda Best Log Lambda Best 10FCV  
## [1,] 0.01311554 -4.333958 1.069669

lasso.coef <- coef(lasso.logit, s=lasso.best.lambda)  
lasso.coef.0 <- coef(lasso.logit, s=0)  
  
# Bind all coefficients  
  
all.coefs <- cbind(lasso.coef, exp(lasso.coef), lasso.coef.0, exp(lasso.coef.0))  
  
# Label the coefficients  
  
colnames(all.coefs) <- c("Best LASSO", "Odds", "0-Lambda LASSO", "0dds")  
  
# Display them  
  
all.coefs

## 10 x 4 Matrix of class "dgeMatrix"  
## Best LASSO Odds 0-Lambda LASSO 0dds  
## (Intercept) -5.542386621 0.003917167 -6.129416690 0.002177851  
## sbp 0.003381401 1.003387124 0.006320764 1.006340782  
## tobacco 0.067870752 1.070226975 0.078673786 1.081851349  
## ldl 0.137863725 1.147819120 0.171882898 1.187538762  
## adiposity 0.000000000 1.000000000 0.015465995 1.015586212  
## famhistPresent 0.777513058 2.176053812 0.914486995 2.495494723  
## typea 0.026945871 1.027312194 0.038607316 1.039362263  
## obesity -0.008357962 0.991676869 -0.056958635 0.944633143  
## alcohol 0.000000000 1.000000000 0.000000000 1.000000000  
## age 0.042535589 1.043453191 0.045341285 1.046384915

**2.2 Interpretation**

Use the cbind() function to display the **Best Lambda** and **lowest 10FCV** for both, the Ridge and LASSO regressions in 4 columns side by side. Label each column as shown below.

Then comment, which of the two modeling methods would you select and why?

cbind("Best Lambda (Ridge)" = ridge.best.lambda, "Best 10FCF (Ridge)" = min.cv.lasso, "Best Lambda (Lasso)" = lasso.best.lambda, "Lowest 10FCV (Lasso)" = min.cv.ridge)

## Best Lambda (Ridge) Best 10FCF (Ridge) Best Lambda (Lasso)  
## [1,] 0.04099545 1.069669 0.01311554  
## Lowest 10FCV (Lasso)  
## [1,] 1.073231

#### Analysis

The cross validation results of both are very similar, with the Ridge slightly outperforming the Lasso. The Lasso has a lower best lambda value. I would select the Lasso model because it excludes the most insignificant variables, aligning with the principle of model simplicity. I would fit a model based on this and test for multicollinearity in the new model before proceeding.

Also, you can compare the Ridge and LASSO coefficients side by side:

# Done for you  
  
all.coefs <- cbind(lasso.coef, lasso.coef.0, ridge.coef, ridge.coef.0)  
colnames(all.coefs) <- c("Best LASSO", "0-Lambda LASSO", "Best Ridge", "0-Lambda Ridge")  
all.coefs

## 10 x 4 sparse Matrix of class "dgCMatrix"  
## Best LASSO 0-Lambda LASSO Best Ridge 0-Lambda Ridge  
## (Intercept) -5.542386621 -6.129416690 -5.3186664789 -5.7221730429  
## sbp 0.003381401 0.006320764 0.0062893712 0.0064556098  
## tobacco 0.067870752 0.078673786 0.0704757286 0.0753889131  
## ldl 0.137863725 0.171882898 0.1421137485 0.1574967615  
## adiposity . 0.015465995 0.0167321735 0.0174298051  
## famhistPresent 0.777513058 0.914486995 0.7664077959 0.8461807201  
## typea 0.026945871 0.038607316 0.0280980909 0.0334345091  
## obesity -0.008357962 -0.056958635 -0.0376375020 -0.0491364514  
## alcohol . . 0.0003693287 0.0002284054  
## age 0.042535589 0.045341285 0.0339089740 0.0391317985

## 3. Principal Components Regression (PCR)

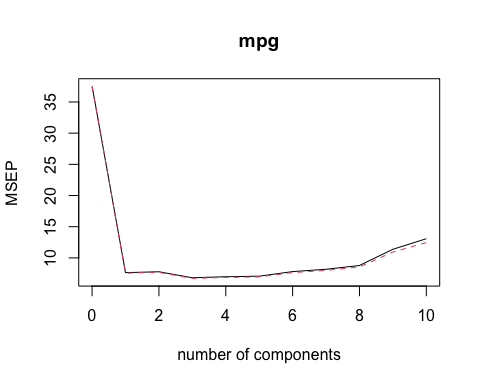
**3.1 Fit the PCR Model**

Load the **pls** library. Then enter set.seed(1) to get repeatable results and fit a **PCR** regression on all the data in the **mtcars** data set to predict **mpg** using all other variables as predictors. Use the pcr() function with scale=T, validation="CV" (for 10FCV). Name this PCR model **pcr.fit**. Then display the summary() of **pcr.fit** and draw a validation plot with val.type="MSEP" (MSE of the predictor).

library(pls)  
set.seed(1)  
pcr.fit <- pcr(mpg ~ ., data=mtcars, scale=T, validation="CV")  
summary(pcr.fit)

## Data: X dimension: 32 10   
## Y dimension: 32 1  
## Fit method: svdpc  
## Number of components considered: 10  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 6.123 2.763 2.789 2.610 2.642 2.658 2.793  
## adjCV 6.123 2.748 2.772 2.591 2.622 2.637 2.765  
## 7 comps 8 comps 9 comps 10 comps  
## CV 2.859 2.964 3.373 3.616  
## adjCV 2.827 2.924 3.307 3.528  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 57.60 84.10 90.07 92.77 94.99 97.09 98.42 99.23  
## mpg 82.53 82.63 85.40 85.41 85.47 85.56 85.58 85.85  
## 9 comps 10 comps  
## X 99.76 100.0  
## mpg 86.09 86.9

validationplot(pcr.fit, val.type="MSEP")



**3.2 Optimal Number of Components**

Based on the summary and the plot, what seems to be the optimal number of components?

#### Analysis

There is a clear elbow at 1 component and a subtle elbow at 3. After reviewing the RMSE and explained variance, the 3 component model exceeds the 1 component model because it has a lower RMSE (2.610 < 2.763) and has significant explanatory power for the x variables (90.07) and response variable (85.40).

## 4. Partial Least Squares Regression (PLSR)

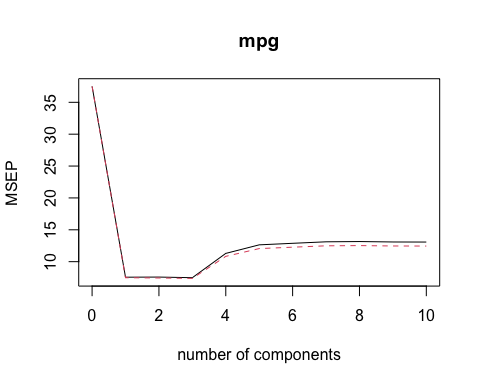
**4.1 Fit the PLSR Model**

Enter set.seed(1) again and repeat the analysis above, but using the plsr() function for **PLS** instead.

set.seed(1)  
plsr.fit <- plsr(mpg ~ ., data=mtcars, scale=T, validation="CV")  
summary(plsr.fit)

## Data: X dimension: 32 10   
## Y dimension: 32 1  
## Fit method: kernelpls  
## Number of components considered: 10  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 6.123 2.750 2.754 2.736 3.361 3.555 3.589  
## adjCV 6.123 2.735 2.728 2.716 3.291 3.472 3.503  
## 7 comps 8 comps 9 comps 10 comps  
## CV 3.622 3.627 3.618 3.616  
## adjCV 3.534 3.539 3.530 3.528  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 57.58 67.65 89.94 91.48 92.82 94.00 95.86 97.14  
## mpg 83.18 85.39 85.75 86.38 86.72 86.85 86.89 86.90  
## 9 comps 10 comps  
## X 98.51 100.0  
## mpg 86.90 86.9

validationplot(plsr.fit, val.type="MSEP")



**4.2 Interpretation**

Based on the summary and the plot, what seems to be the optimal number of components?

#### Analysis

There is again a clear elbow at 1 component, but after revieiwing the RMSE and explained variance, the 3 component model exceeds the 1 component model because it has significanly greater explanatory power for the x variables (89.94) and response variable (85.75). The 1 componenet model, while having a smaller RMSE (2.750 < 2.736), has a much lower explained variance of x (57.58 < 89.94). Therefore, I would select the 3 component model because it has a higher explained variance and is more interpretable than the 1 component model.

**4.3 Evaluation**

Based on the RMSE for 3 components, which method would you select, PCR or PLS? Briefly explain why.

#### Analysis

I would select the 3 Component PCR becase it has the lowest RMSE (2.610) and highest explanatory value of x and the response variable (90.07 & 85.40, respectively).