**STA 545 ASSIGNMENT #2**

**SURUCHI JAIKUMAR AHUJA**

**1)**

**GAUSS-MARKOV THEOREM**

(Actual population)

Variance ( ) ≤ Variance ()

i.e E () =

is a linear function of response vector y , where x is fixed.

Using Triangular Inequality,

y → linear function is when x is fixed,

and a linear estimator is an unbiased estimator of

if and only if

E () =

E () = β

=

**=**

Where so that

is a linear estimator.

is an unbiased estimator other than the OLS estimator

= y

Then we know that,

Variance ( ) = Variance ( )

Variance ( ) = Variance (

= Variance () + Variance () + 2 Covariance ()

Variance ( -) = Variance ()

= Variance () y = Variance (y

= ( Variance y (

= ( ( ) (

= ( ( = *( ( > 0*

Covariance ( ) = Covariance (- ,)

= Covariance ( (y , ) = ( Variance y

=

= ] = 0

=

Variance ( ) = Variance ( ) + Variance ()

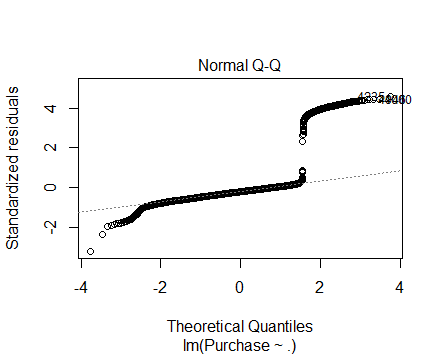
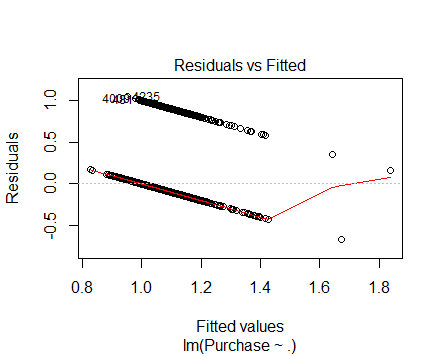
Hence ,

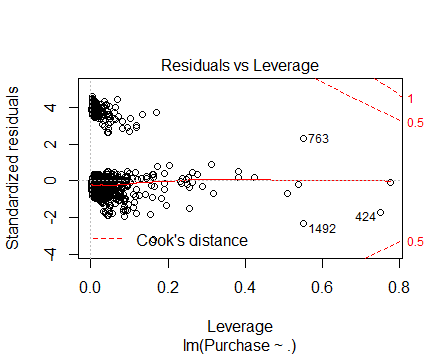
**Variance ( ) > Variance ()**

**…………………………………………..**

**2)**

The packages ISLR, MASS, leaps and glmnet were installed. The Caravan data was loaded into the R Console. The Caravan data set contains 86 variables on product-usage data and sociodemographic data derived from zip area codes. There are 5,822 customers in the training set and another 4,000 in the test set. The linear model was fitted in the training data set and the ordinary least squares was estimated.





***OUTPUT:***

**FORWARD ALGORITHM :** The estimates of 50 variables was taken

**>** forward <- regsubsets(Purchase~., data=Caravan, nvmax=86, method = "forward")

summary(forward)$outmat[50, ]

MOSTYPE MAANTHUI MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE MRELGE MRELSA

" " "\*" " " "\*" " " "\*" "\*" "\*" " " "\*" " "

MRELOV MFALLEEN MFGEKIND MFWEKIND MOPLHOOG MOPLMIDD MOPLLAAG MBERHOOG MBERZELF MBERBOER MBERMIDD

"\*" " " "\*" " " "\*" " " "\*" " " " " "\*" "\*"

MBERARBG MBERARBO MSKA MSKB1 MSKB2 MSKC MSKD MHHUUR MHKOOP MAUT1 MAUT2

"\*" " " " " " " " " "\*" " " "\*" "\*" "\*" "\*"

MAUT0 MZFONDS MZPART MINKM30 MINK3045 MINK4575 MINK7512 MINK123M MINKGEM MKOOPKLA PWAPART

"\*" "\*" "\*" " " " " "\*" " " "\*" "\*" "\*" "\*"

PWABEDR PWALAND PPERSAUT PBESAUT PMOTSCO PVRAAUT PAANHANG PTRACTOR PWERKT PBROM PLEVEN

" " "\*" "\*" " " " " "\*" "\*" "\*" "\*" " " "\*"

PPERSONG PGEZONG PWAOREG PBRAND PZEILPL PPLEZIER PFIETS PINBOED PBYSTAND AWAPART AWABEDR

" " "\*" "\*" "\*" "\*" "\*" " " " " " " "\*" " "

AWALAND APERSAUT ABESAUT AMOTSCO AVRAAUT AAANHANG ATRACTOR AWERKT ABROM ALEVEN APERSONG

" " " " "\*" " " " " "\*" "\*" " " " " "\*" " "

AGEZONG AWAOREG ABRAND AZEILPL APLEZIER AFIETS AINBOED ABYSTAND

"\*" "\*" "\*" "\*" "\*" "\*" " " "\*"

**BACKWARD ALGORITHM:** The estimates of the 50 variables was taken.

> backward <- regsubsets(Purchase~., data=Caravan, nvmax=86, method = "backward")

> summary(backward)$outmat[50,]

MOSTYPE MAANTHUI MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE MRELGE MRELSA

"\*" "\*" " " "\*" "\*" "\*" " " " " "\*" "\*" " "

MRELOV MFALLEEN MFGEKIND MFWEKIND MOPLHOOG MOPLMIDD MOPLLAAG MBERHOOG MBERZELF MBERBOER MBERMIDD

" " " " "\*" " " " " "\*" "\*" " " " " "\*" "\*"

MBERARBG MBERARBO MSKA MSKB1 MSKB2 MSKC MSKD MHHUUR MHKOOP MAUT1 MAUT2

" " "\*" " " " " " " "\*" " " "\*" "\*" "\*" "\*"

MAUT0 MZFONDS MZPART MINKM30 MINK3045 MINK4575 MINK7512 MINK123M MINKGEM MKOOPKLA PWAPART

" " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

PWABEDR PWALAND PPERSAUT PBESAUT PMOTSCO PVRAAUT PAANHANG PTRACTOR PWERKT PBROM PLEVEN

" " "\*" "\*" " " " " "\*" " " " " " " " " "\*"

PPERSONG PGEZONG PWAOREG PBRAND PZEILPL PPLEZIER PFIETS PINBOED PBYSTAND AWAPART AWABEDR

" " "\*" "\*" "\*" "\*" "\*" " " "\*" " " "\*" " "

AWALAND APERSAUT ABESAUT AMOTSCO AVRAAUT AAANHANG ATRACTOR AWERKT ABROM ALEVEN APERSONG

" " " " "\*" " " " " " " "\*" " " " " "\*" " "

AGEZONG AWAOREG ABRAND AZEILPL APLEZIER AFIETS AINBOED ABYSTAND

"\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

**RIDGE REGRESSION:** Estimates are given below

The ridge regression model was fit into the training set and the model is shown below:

> ridge.mod <- glmnet(X,Y,alpha = 0)

> ridge.mod

Call: glmnet(x = X, y = Y, alpha = 0)

Df %Dev Lambda

[1,] 20 1.004e-37 21.470000

[2,] 20 1.131e-03 19.560000

[3,] 20 1.233e-03 17.820000

[4,] 20 1.345e-03 16.240000

[5,] 20 1.465e-03 14.800000

[6,] 20 1.596e-03 13.480000

[7,] 20 1.736e-03 12.290000

[8,] 20 1.888e-03 11.190000

[9,] 20 2.051e-03 10.200000

[10,] 20 2.227e-03 9.293000

[11,] 20 2.415e-03 8.468000

[12,] 20 2.616e-03 7.716000

[13,] 20 2.831e-03 7.030000

[14,] 20 3.060e-03 6.406000

[15,] 20 3.303e-03 5.837000

[16,] 20 3.561e-03 5.318000

[17,] 20 3.834e-03 4.846000

[18,] 20 4.122e-03 4.415000

[19,] 20 4.423e-03 4.023000

[20,] 20 4.739e-03 3.666000

[21,] 20 5.069e-03 3.340000

[22,] 20 5.411e-03 3.043000

[23,] 20 5.766e-03 2.773000

[24,] 20 6.131e-03 2.527000

[25,] 20 6.505e-03 2.302000

[26,] 20 6.888e-03 2.098000

[27,] 20 7.277e-03 1.911000

[28,] 20 7.671e-03 1.741000

[29,] 20 8.068e-03 1.587000

[30,] 20 8.466e-03 1.446000

[31,] 20 8.862e-03 1.317000

[32,] 20 9.255e-03 1.200000

[33,] 20 9.643e-03 1.094000

[34,] 20 1.002e-02 0.996500

[35,] 20 1.040e-02 0.908000

[36,] 20 1.076e-02 0.827300

[37,] 20 1.111e-02 0.753800

[38,] 20 1.145e-02 0.686900

[39,] 20 1.177e-02 0.625800

[40,] 20 1.208e-02 0.570200

[41,] 20 1.238e-02 0.519600

[42,] 20 1.266e-02 0.473400

[43,] 20 1.292e-02 0.431400

[44,] 20 1.317e-02 0.393000

[45,] 20 1.341e-02 0.358100

[46,] 20 1.363e-02 0.326300

[47,] 20 1.383e-02 0.297300

[48,] 20 1.402e-02 0.270900

[49,] 20 1.420e-02 0.246800

[50,] 20 1.437e-02 0.224900

[51,] 20 1.453e-02 0.204900

[52,] 20 1.467e-02 0.186700

[53,] 20 1.481e-02 0.170100

[54,] 20 1.493e-02 0.155000

[55,] 20 1.505e-02 0.141300

[56,] 20 1.516e-02 0.128700

[57,] 20 1.526e-02 0.117300

[58,] 20 1.535e-02 0.106900

[59,] 20 1.544e-02 0.097360

[60,] 20 1.552e-02 0.088710

[61,] 20 1.559e-02 0.080830

[62,] 20 1.566e-02 0.073650

[63,] 20 1.572e-02 0.067110

[64,] 20 1.578e-02 0.061140

[65,] 20 1.584e-02 0.055710

[66,] 20 1.589e-02 0.050760

[67,] 20 1.593e-02 0.046250

[68,] 20 1.598e-02 0.042140

[69,] 20 1.602e-02 0.038400

[70,] 20 1.606e-02 0.034990

[71,] 20 1.610e-02 0.031880

[72,] 20 1.613e-02 0.029050

[73,] 20 1.617e-02 0.026470

[74,] 20 1.620e-02 0.024120

[75,] 20 1.623e-02 0.021970

[76,] 20 1.626e-02 0.020020

[77,] 20 1.629e-02 0.018240

[78,] 20 1.632e-02 0.016620

[79,] 20 1.635e-02 0.015150

[80,] 20 1.637e-02 0.013800

[81,] 20 1.640e-02 0.012570

[82,] 20 1.643e-02 0.011460

[83,] 20 1.646e-02 0.010440

[84,] 20 1.648e-02 0.009512

[85,] 20 1.651e-02 0.008667

[86,] 20 1.654e-02 0.007897

[87,] 20 1.657e-02 0.007196

[88,] 20 1.660e-02 0.006556

[89,] 20 1.662e-02 0.005974

[90,] 20 1.665e-02 0.005443

[91,] 20 1.668e-02 0.004960

[92,] 20 1.670e-02 0.004519

[93,] 20 1.673e-02 0.004118

[94,] 20 1.676e-02 0.003752

[95,] 20 1.678e-02 0.003418

[96,] 20 1.681e-02 0.003115

[97,] 20 1.683e-02 0.002838

[98,] 20 1.685e-02 0.002586

[99,] 20 1.688e-02 0.002356

[100,] 20 1.690e-02 0.002147

names(cv.out)

[1] "lambda" "cvm" "cvsd" "cvup" "cvlo" "nzero" "name"

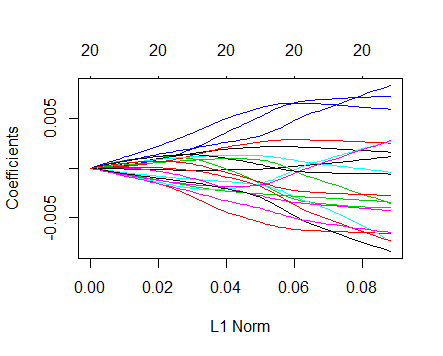
[8] "glmnet.fit" "lambda.min" "lambda.1se"

+

The best lambda value was chosen by cross validation.

> bestlam

[1] 0.1068519



The predicted model using ridge regression

> ridge.pred

21 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) 1.059704e+00

MOSTYPE -2.020452e-04

MAANTHUI -4.355174e-03

MGEMOMV 1.879565e-05

MGEMLEEF 3.463833e-03

MOSHOOFD -1.357419e-03

MGODRK -1.971980e-03

MGODPR 1.883007e-03

MGODOV 2.057512e-03

MGODGE -2.339150e-03

MRELGE 2.550196e-03

MRELSA -2.155630e-03

MRELOV -1.887090e-03

MFALLEEN -2.044175e-03

MFGEKIND -8.507241e-04

MFWEKIND 9.428973e-04

MOPLHOOG 4.910934e-03

MOPLMIDD 1.276708e-03

MOPLLAAG -3.262884e-03

MBERHOOG 1.027207e-03

MBERZELF -3.944817e-04

**LASSO MODEL:** Estimates are given below

The Lasso model was fit into the training data set and the best lambda value is chosen using cross validation.The model is shown below

|  |
| --- |
| > lasso.mod <- glmnet(X,Y,alpha = 1)  > lasso.mod  Call: glmnet(x = X, y = Y, alpha = 1)  Df %Dev Lambda  [1,] 0 0.000000 2.147e-02  [2,] 1 0.001392 1.956e-02  [3,] 1 0.002548 1.782e-02  [4,] 2 0.003690 1.624e-02  [5,] 3 0.005298 1.480e-02  [6,] 3 0.006653 1.348e-02  [7,] 3 0.007776 1.229e-02  [8,] 3 0.008708 1.119e-02  [9,] 3 0.009481 1.020e-02  [10,] 4 0.010160 9.293e-03  [11,] 5 0.010840 8.468e-03  [12,] 5 0.011510 7.716e-03  [13,] 5 0.012070 7.030e-03  [14,] 5 0.012530 6.406e-03  [15,] 5 0.012920 5.837e-03  [16,] 7 0.013260 5.318e-03  [17,] 7 0.013560 4.846e-03  [18,] 7 0.013820 4.415e-03  [19,] 7 0.014030 4.023e-03  [20,] 7 0.014210 3.666e-03  [21,] 7 0.014350 3.340e-03  [22,] 8 0.014500 3.043e-03  [23,] 10 0.014700 2.773e-03  [24,] 10 0.014890 2.527e-03  [25,] 11 0.015060 2.302e-03  [26,] 11 0.015210 2.098e-03  [27,] 11 0.015340 1.911e-03  [28,] 13 0.015450 1.741e-03  [29,] 13 0.015560 1.587e-03  [30,] 13 0.015700 1.446e-03  [31,] 13 0.015820 1.317e-03  [32,] 13 0.015900 1.200e-03  [33,] 13 0.015970 1.094e-03  [34,] 13 0.016020 9.965e-04  [35,] 13 0.016070 9.080e-04  [36,] 13 0.016110 8.273e-04  [37,] 13 0.016140 7.538e-04  [38,] 14 0.016170 6.869e-04  [39,] 14 0.016210 6.258e-04  [40,] 14 0.016250 5.702e-04  [41,] 14 0.016280 5.196e-04  [42,] 14 0.016300 4.734e-04  [43,] 14 0.016320 4.314e-04  [44,] 15 0.016340 3.930e-04  [45,] 15 0.016360 3.581e-04  [46,] 15 0.016370 3.263e-04  [47,] 16 0.016380 2.973e-04  [48,] 16 0.016390 2.709e-04  [49,] 17 0.016490 2.468e-04  [50,] 18 0.016580 2.249e-04  [51,] 18 0.016660 2.049e-04  [52,] 19 0.016720 1.867e-04  [53,] 19 0.016790 1.701e-04  [54,] 19 0.016840 1.550e-04  [55,] 19 0.016890 1.413e-04  [56,] 20 0.016930 1.287e-04  [57,] 20 0.016970 1.173e-04  [58,] 20 0.016990 1.069e-04  [59,] 20 0.017020 9.736e-05  [60,] 20 0.017040 8.871e-05  [61,] 20 0.017060 8.083e-05  [62,] 20 0.017070 7.365e-05  [63,] 20 0.017090 6.711e-05  [64,] 20 0.017100 6.114e-05  [65,] 20 0.017110 5.571e-05  [66,] 20 0.017110 5.076e-05  [67,] 20 0.017120 4.625e-05  [68,] 20 0.017130 4.214e-05  [69,] 20 0.017130 3.840e-05  [70,] 20 0.017140 3.499e-05  [71,] 20 0.017140 3.188e-05  [72,] 20 0.017140 2.905e-05  [73,] 20 0.017150 2.647e-05  [74,] 20 0.017150 2.412e-05  [75,] 20 0.017150 2.197e-05  [76,] 20 0.017150 2.002e-05  [77,] 20 0.017150 1.824e-05  [78,] 20 0.017160 1.662e-05  [79,] 20 0.017160 1.515e-05  [80,] 20 0.017160 1.380e-05  [81,] 20 0.017160 1.257e-05  [82,] 20 0.017160 1.146e-05  [83,] 20 0.017160 1.044e-05  [84,] 20 0.017160 9.512e-06  [85,] 20 0.017160 8.667e-06  [86,] 20 0.017160 7.897e-06  [87,] 20 0.017160 7.196e-06  [88,] 20 0.017160 6.556e-06  [89,] 20 0.017160 5.974e-06  [90,] 20 0.017160 5.443e-06  [91,] 20 0.017160 4.960e-06  [92,] 20 0.017160 4.519e-06  [93,] 20 0.017170 4.118e-06  [94,] 20 0.017170 3.752e-06  [95,] 20 0.017170 3.418e-06  [96,] 20 0.017170 3.115e-06  [97,] 20 0.017170 2.838e-06  [98,] 20 0.017170 2.586e-06  [99,] 20 0.017170 2.356e-06  [100,] 20 0.017170 2.147e-06 |
| The best lambda value is chosen by cross validation   |  | | --- | | bestlam1  [1] 0.0009965045    The predicted lasso model is  > lasso\_pred <- predict(lasso.mod,s=bestlam1,type= "coefficients")  > lasso\_pred  21 x 1 sparse Matrix of class "dgCMatrix"  1  (Intercept) 1.0670649532  MOSTYPE .  MAANTHUI -0.0037933770  MGEMOMV .  MGEMLEEF 0.0048337937  MOSHOOFD -0.0019786527  MGODRK -0.0033149534  MGODPR 0.0008327947  MGODOV 0.0011139120  MGODGE -0.0036496949  MRELGE 0.0051571253  MRELSA .  MRELOV .  MFALLEEN -0.0029806570  MFGEKIND -0.0018554258  MFWEKIND .  MOPLHOOG 0.0055310637  MOPLMIDD .  MOPLLAAG -0.0054627047  MBERHOOG .  MBERZELF -0.0008264202 | |  | | |  | | --- | |  | |   …………………………………………………. |
| |  | | --- | |  | |

3)

**Conjugate Gradient Algorithms:**

Conjugate gradient algorithm is one of the most prominent method iterative method to find out the solution for large linear system of equations. It is the most effective for systems of the form:

A x = b

This is done by finding the quadratic form which is scalar and a quadratic function with the form

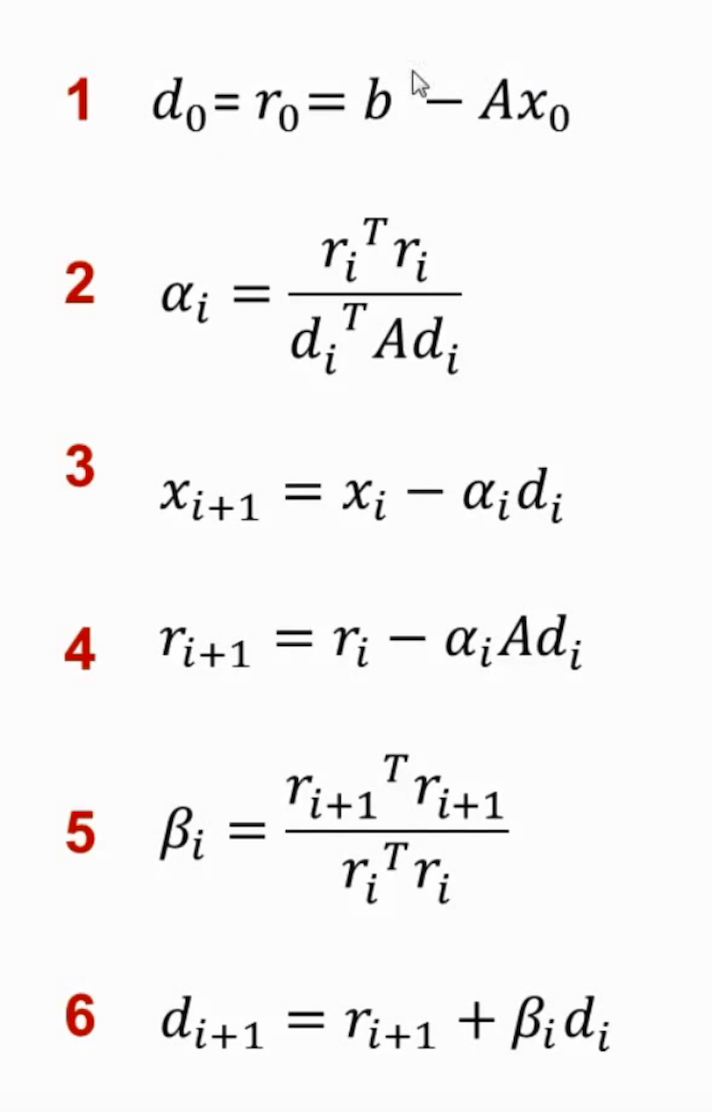
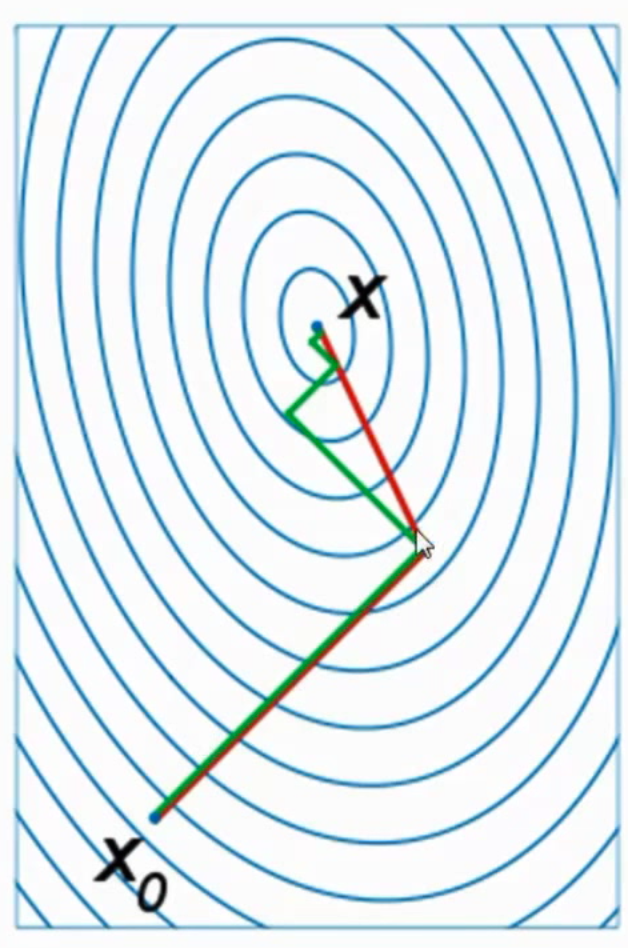
(A is a matrix, x and b are vectors, c is a scalar constant)

The above condition holds true if and only if matrix A is a sparse symmetric positive-definite.

Then

When we set the gradient to 0, we get Ax=b, and on solving we get the solution for the linear system.

The algorithm for this method:



The above figure is taken from Tom Carlon Presentation.

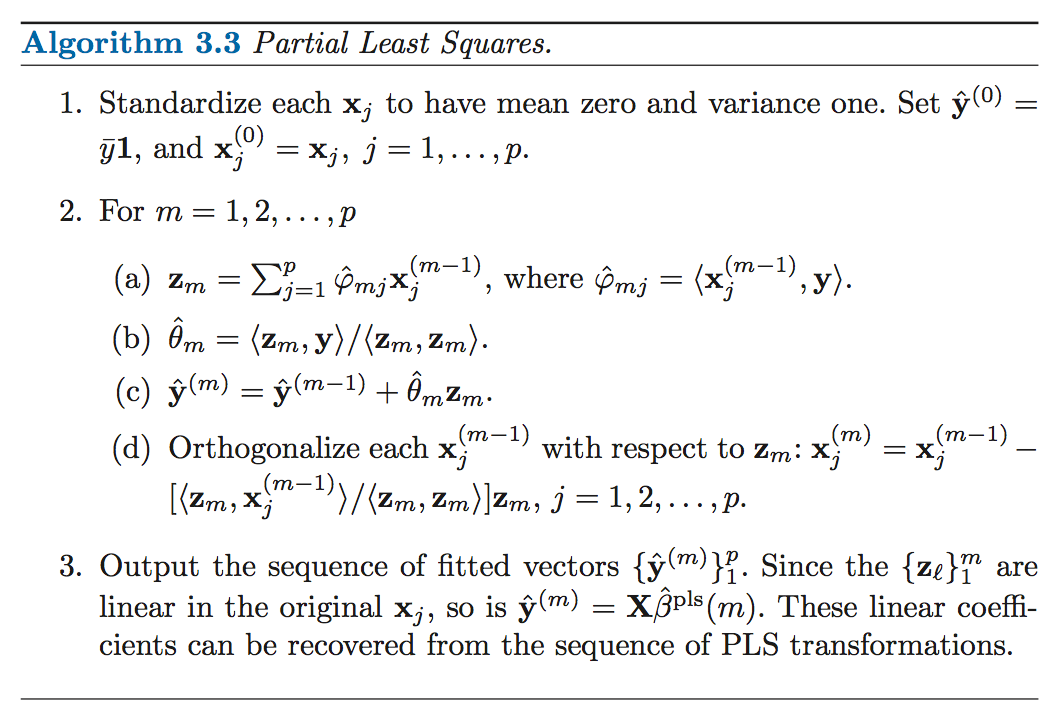
Conjugate Gradient algorithm optimizes the steepest decent algorithm by avoiding the repetitious steps. It starts at an arbitrary point and slide down the paraboloid.This is done by considering the directions of decent that are orthogonal to the previous steps. This way it avoid redundancy. The answer is obtained in exactly n steps from our initial guess.

**Partial Least Square:**

Partial least squares (PLS) is a method for constructing predictive models when the factors are many and highly collinear. Partial least square is dimension reduction technique to reduce **p+1 -> M+1** dimensions where M < p.

Partial least square is the alternative to principle component regression. It uses the response to

choose the input variables.



The above algorithm is taken from the ELS Textbook.

The Partial Least Squares direction does not fit the predictors as closely as does Principal Component Algorithm, but it is comparatively better when it comes to explaining the response. It generalizes and combines features from PCR and multiple regression.

Supervised dimension reduction of PLS can reduce bias. Thus by basis variance trade off we tend to increase the variance.

Both partial least square and conjugate gradient algorithm are used for solving the linear system of equations. They both have similarities in which they choose directions that are orthogonal to the previous steps. PLS use less number of dimensions than in the original data set. While Conjugate Gradient uses coefficient of all the predictors. They should sparse symmetric positive definite matrix.

……………………………………………………………

**4)**

In this, first the packages ISLR, glmnet, and pls (partial least squares) were installed in the working directory. The college data set was loaded into the R console.

(a)

The college data set was then split into the training set and test set. The training set consisted of 80% of the data in the College dataset and the testing set consisted of 20% of the data in the College data set. The Apps ( Number of applications received ) is present in the second row of the College data set and that is removed from both the train and test set. So there are two new test sets and train sets, one containing only the Apps and the other containing all the rows other than Apps. A linear model is fitted in the training set using the least square method (lm) and the linear model was predicted where the previous linear model was fitted in the testing data. The mean error i.e the RSS was calculated using the formula . Where is the predicted linear model.

***OUTPUT :***

*The mean error for the linear model*

1242119

(b)

Ridge Regression is a shrinkage method and is used to shrink regression coefficients by imposing penalty on their size. Ridge Regression is a proportional shrinkage method. In this method λ is a complexity parameter, which controls the amount of shrinkage and in this particular question the best λ value is chosen by cross validation ( cv.out ). A ridge regression model was fit into the training set and the best λ value is chosen. Here the test error is calculated by the formula . Here is the predicted model.

***OUTPUT :***

> ridge.mod = glmnet(X.train, Y.train, alpha=0)

> names(ridge.mod)

[1] "a0" "beta" "df" "dim" "lambda" "dev.ratio" "nulldev" "npasses"

[9] "jerr" "offset" "call" "nobs"

> ridge.mod$lambda[10]

[1] 1663966

> coef(ridge.mod)[,10]

(Intercept) Private Accept Enroll Top10perc Top25perc F.Undergrad

2.946296e+03 -9.446310e+00 3.612783e-03 8.864017e-03 1.839614e-01 1.768183e-01 1.640808e-03

P.Undergrad Outstate Room.Board Books Personal PhD Terminal

2.474871e-03 1.202234e-04 1.539856e-03 6.878930e-03 2.884554e-03 2.358459e-01 2.421852e-01

S.F.Ratio perc.alumni Expend Grad.Rate

2.526041e-01 -7.775917e-02 4.542694e-04 8.146252e-02

> l2 <- sqrt(coef(ridge.mod)[2:18,19]^2)

> l2

Private Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad Outstate

2.150845e+01 8.253707e-03 2.021861e-02 4.183886e-01 4.018867e-01 3.741265e-03 5.628185e-03 2.724438e-04

Room.Board Books Personal PhD Terminal S.F.Ratio perc.alumni Expend

3.512415e-03 1.560951e-02 6.546638e-03 5.348353e-01 5.489944e-01 5.766761e-01 1.781374e-01 1.034304e-03

Grad.Rate

1.859967e-01

names(cv.out)

[1] "lambda" "cvm" "cvsd" "cvup" "cvlo" "nzero" "name"

[8] "glmnet.fit" "lambda.min" "lambda.1se"

> bestlam

[1] 421.876

The best lambda value is found to be 421.876 by cross validation method.

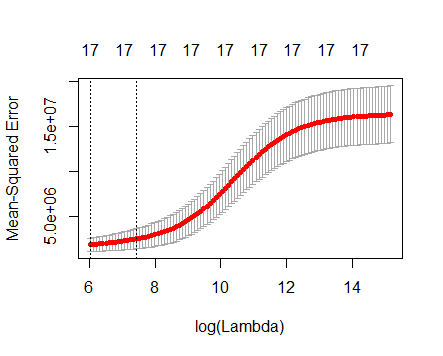
The test error is calculated to be :

test\_error <- sum((y\_hat-y\_true)^2)

> test\_error

[1] 301720080

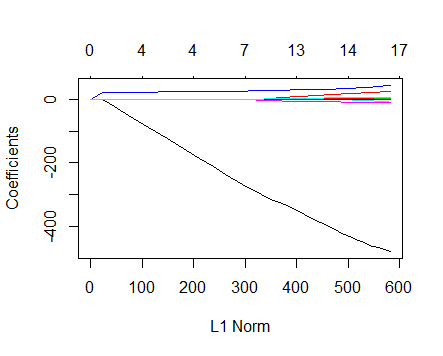
Mean- Squared error.



(c)

**Lasso** **Regression** is a shrinkage method but it is different from Ridge Regressions in a few ways. It follows a method called soft thresholding where it translates each coefficient by a constant factor λ, truncating at zero. Here the λ value is chosen by cross validation (cv.out)

***OUTPUT :***



lasso\_mod$lambda[10]

[1] 1663.966

The lambda value chosen by cross validation is chosen as 1663.966

The number of non- zero coefficient estimated are

|  |
| --- |
| Lasso.pred  1  18 x 1 sparse Matrix of class "dgCMatrix"                          1  (Intercept) -7.665194e+02  Private     -2.636265e+02  Accept       1.492322e+00  Enroll      -2.313685e-01  Top10perc    2.959490e+01  Top25perc    .  F.Undergrad  .  P.Undergrad  8.248746e-03  Outstate    -6.235982e-02  Room.Board   1.155013e-01  Books        .  Personal     8.185845e-02  PhD         -7.460613e+00  Terminal    -2.889644e-01  S.F.Ratio    1.821695e+01  perc.alumni  .  Expend       6.828947e-02  Grad.Rate    3.940208e+00 |

coef(lasso\_mod)[,10]

(Intercept) Private Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad

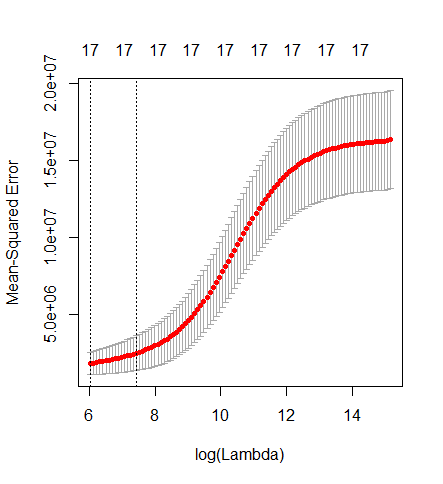
1297.7114849 0.0000000 0.8504165 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

Outstate Room.Board Books Personal PhD Terminal S.F.Ratio perc.alumni

0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

Expend Grad.Rate

0.0000000 0.0000000



test.error.lasso

[1] 193228181

(d)

**Principal Component Regression** is a regression analysis and the main technique is principal component analysis. Present in the pls package. A pcr model was fit in the training data set. The k is a complexity parameter and it is chosen by cross validation.

***OUTPUT :***

pcr.fit <- pcr(train$Apps~.,data=train,scale=TRUE,validation="CV")

> summary(pcr.fit)

Data: X dimension: 543 17

Y dimension: 543 1

Fit method: svdpc

Number of components considered: 17

VALIDATION: RMSEP

Cross-validated using 10 random segments.

(Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps

CV 4102 4017 2164 2192 1729 1678 1679 1673 1641 1600

adjCV 4102 4018 2162 2196 1689 1670 1673 1668 1634 1595

10 comps 11 comps 12 comps 13 comps 14 comps 15 comps 16 comps 17 comps

CV 1602 1605 1604 1624 1624 1424 1204 1125

adjCV 1597 1600 1599 1619 1620 1400 1193 1117

TRAINING: % variance explained

1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps 10 comps

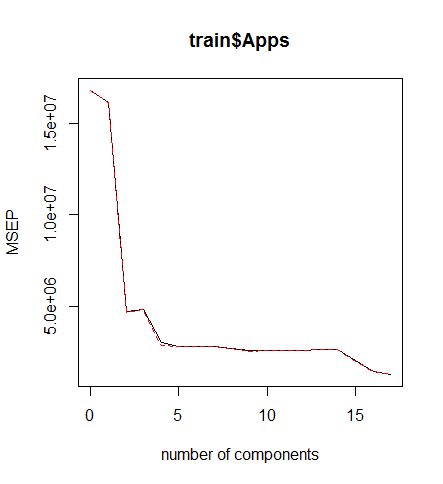
X 31.448 57.11 63.88 69.55 74.76 79.79 83.60 87.07 90.15 92.66

train$Apps 4.702 72.59 72.59 83.62 84.19 84.24 84.45 85.24 85.94 86.15

11 comps 12 comps 13 comps 14 comps 15 comps 16 comps 17 comps

X 94.82 96.66 97.78 98.68 99.34 99.82 100.00

train$Apps 86.20 86.25 86.26 86.31 92.47 93.59 94.11



The pcr test error was calculated using the formula

pcr.test.error <- mean((data.frame(pcr.pred)-Y.test)^2)

> pcr.test.error

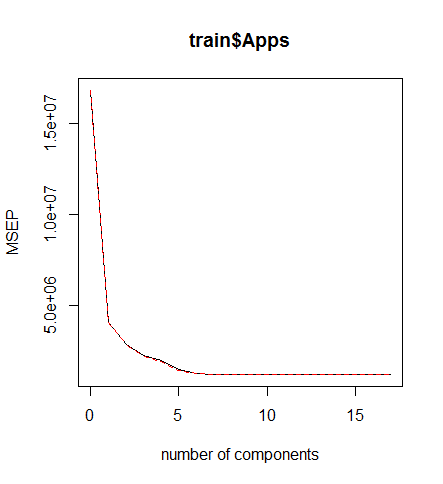
[1] 1519910

(e)

**The Partial Least Squares** is not a scale invariant method and the k is a complexity parameter. Here k is selected by cross validation. A pls model was fit into the training data set and the test error was calculated using the formula

|  |
| --- |
| > pls.fit <- plsr(train$Apps~.,data=train,scale=TRUE,validation="CV")  > summary(pls.fit)  Data: X dimension: 543 17  Y dimension: 543 1  Fit method: kernelpls  Number of components considered: 17  VALIDATION: RMSEP  Cross-validated using 10 random segments.  (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps  CV 4102 2027 1704 1507 1423 1230 1147 1123 1121 1119  adjCV 4102 2019 1702 1500 1398 1204 1137 1115 1111 1112  10 comps 11 comps 12 comps 13 comps 14 comps 15 comps 16 comps 17 comps  CV 1119 1116 1116 1114 1114 1114 1114 1114  adjCV 1111 1109 1109 1107 1107 1107 1107 1107  TRAINING: % variance explained  1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps 10 comps  X 26.04 43.32 62.46 64.69 66.93 71.03 74.14 76.71 81.30 84.41  train$Apps 77.76 84.76 88.28 92.00 93.76 93.88 93.95 94.03 94.07 94.09  11 comps 12 comps 13 comps 14 comps 15 comps 16 comps 17 comps  X 86.65 89.47 92.50 95.14 96.87 98.80 100.00  train$Apps 94.10 94.11 94.11 94.11 94.11 94.11 94.11 |
|  |
|  |

***OUTPUT:***



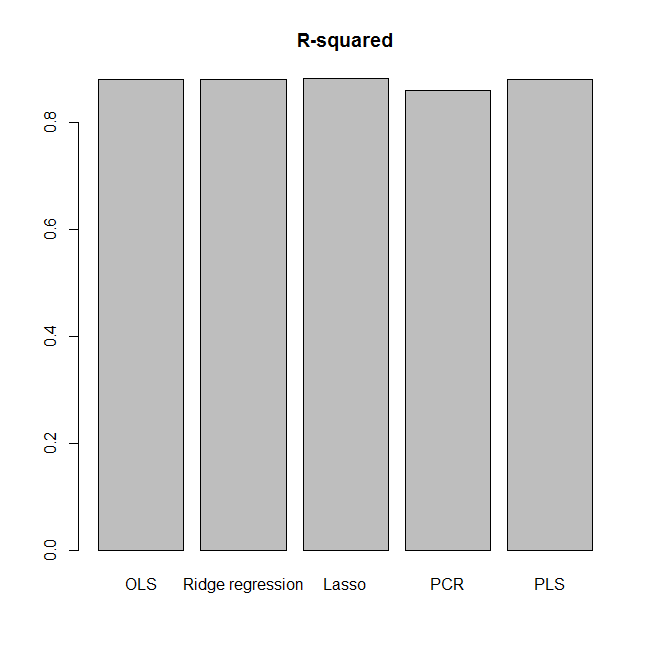
|  |
| --- |
|  |
|  |

(f)

**R-Squared test** (Goodness of Fit) is done, it is a statistical measure of how close the data are fitted to the regression line.

Given below is the graph for the R-squared test, the results show that PCR value is lesser compared to the other models. For this data set; the models of OLS, Ridge Regression, Lasso Regression, Partial Least Squares will be able to calculate the number of college applications recieved accurately.

|  |  |
| --- | --- |
| METHOD | TEST ERROR |
| Linear Model | 1242119 |
| Ridge Regression | 301720080 |
| Lasso Regression | 193228181 |
| Principal Component Regression | 1519910 |
| Partial Least Squares | 1287035 |



--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------THE END----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------