Regularized Regression Methods - Demo

Examples of LASSO

STAT 32950-24620

Spring 2023 (5/9-11)

Review - LS regression method

Least squares (LS) linear regression model

$$\boldsymbol{Y} = \beta_0 + \boldsymbol{Z}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

LS parameters β_0, $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_r)$ are obtained by minimizing

$$\|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|_2^2$$

over all possible values of parameter vector $\boldsymbol{\beta} \in \mathbb{R}^r$.

$$\hat{\boldsymbol{\beta}}_{LS} = \text{argmin}_{\boldsymbol{\beta}\in\mathbb{R}^r} \left\{\|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|_2^2\right\}$$

$\hat{\boldsymbol{\beta}}_{LS}$ has the smallest variance among all linear unbiased estimators of $\boldsymbol{\beta}$.

Shrinkage method, Ridge regression

Ridge regression shrinks the regression coefficients by imposing a penalty (also called regularization) on their 2-norm size

$$\|\boldsymbol{\beta}\|_2^2 = \boldsymbol{\beta}'\boldsymbol{\beta} = \beta_1^2 + \cdots + \beta_r^2$$

Ridge regression parameters $\beta_0, \beta_1, \cdots, \beta_r$ are obtained by

$$\min_{\beta} \left\{\|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|_2^2\right\} \qquad \textit{when} \quad \|\boldsymbol{\beta}\|_2^2 \le s_r$$

Or equivalently,

$$\min_{\beta} \left\{\|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|_2^2 + \lambda_{ridge}\|\boldsymbol{\beta}\|_2^2\right\}$$

where $s_r \propto 1/\lambda_{ridge}$, tuning parameters aim to minimize MSE.

Least absolute shrinkage and selection operator (LASSO)

LASSO regression regularizes the regression coefficients by imposing a penalty or a regularizaion on their 1-norm size

$$\|\boldsymbol{\beta}\|_1 = |\beta_1| + \cdots + |\beta_r|$$

LASSO regression parameters $\beta_0, \beta_1, \cdots, \beta_r$ are obtained by

$$\min_{\beta} \left\{\|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|_2^2\right\} \qquad \textit{when} \quad \|\boldsymbol{\beta}\|_1 \le s_l$$

Or equivalently,

$$\min_{\beta} \left\{\|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|_2^2 + \lambda_{lasso}\|\boldsymbol{\beta}\|_1\right\}$$

where $s_l \propto 1/\lambda_{lasso}$, tuning parameters aim to minimize MSE.

Elastic Net

Elastic Net linear regression

$$\min_{\beta} \left\{ \|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|_2^2 \right\}$$

when

$$\|\boldsymbol{\beta}\|_2^2 \leq s_r \quad \textit{and} \quad \textit{when} \quad \|\boldsymbol{\beta}\|_1 \leq s_l$$

Or equivalently,

$$\min_{\beta} \left\{ \|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|_2^2 + \lambda_{ridge}\|\boldsymbol{\beta}\|_2^2 + \lambda_{lasso}\|\boldsymbol{\beta}\|_1 \right\}$$

Another common form:

$$\min_{\beta} \left\{ \|\boldsymbol{y} - \beta_0 - \boldsymbol{Z}\boldsymbol{\beta}\|_2^2 + \lambda \left[\alpha\|\boldsymbol{\beta}\|_2^2 + (1-\alpha)\|\boldsymbol{\beta}\|_1 \right] \right\}$$

Example for LASSO Regression

Example — Prostate cancer dataset

- Number of explanatory variables $p = 9$, number of observations $n = 96$.
- measuring the correlation between the level of a prostate-specific antigen and some covariates.
- Used in Hastie, Tibshirani and Friedman (2009)
- from a study by Stamey et al. (1989) of prostate cancer

Example data

The covariates are

1. lcavol : log-cancer volume
2. lweight : log-prostate weight
3. age : age of patient
4. lbhp : log-amount of benign hyperplasia
5. svi : seminal vesicle invasion
6. lcp : log-capsular penetration
7. gleason : Gleason Score,
8. pgg45 : percent of Gleason scores 4 or 5
9. lpsa: is the response variable, log-psa.

Partition the data into Training and Calibration parts

```
pcancer=read.table("pcancer.dat",header=T)
str(pcancer)
'data.frame':  97 obs. of  10 variables:
$ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
$ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
$ age    : int  50 58 74 58 62 50 64 58 47 63 ...
$ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
$ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
$ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
$ gleason: int  6 6 7 6 6 6 6 6 6 6 ...
$ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
$ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
$ train  : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...

train <- pcancer[which(pcancer$train),1:9]
calibrate <- pcancer[-which(pcancer$train),1:9]
str(train)  # 67 obs. of  9 variables
```

Fit LASSO regression model on the training set

Training data: 70%
Calibrating data: 30%

Note: Calibration data is used for validation, just not "cross validation".

Using `glmnet` to fit a LASSO regression model on the training set.

```
library(MASS)
library(glmnet)
y <- as.numeric(train[,9])
x <- as.matrix(train[,1:8])
trainfit = glmnet(x,y)

plot(trainfit,label=T)
```


Set of models from a LASSO regression fit

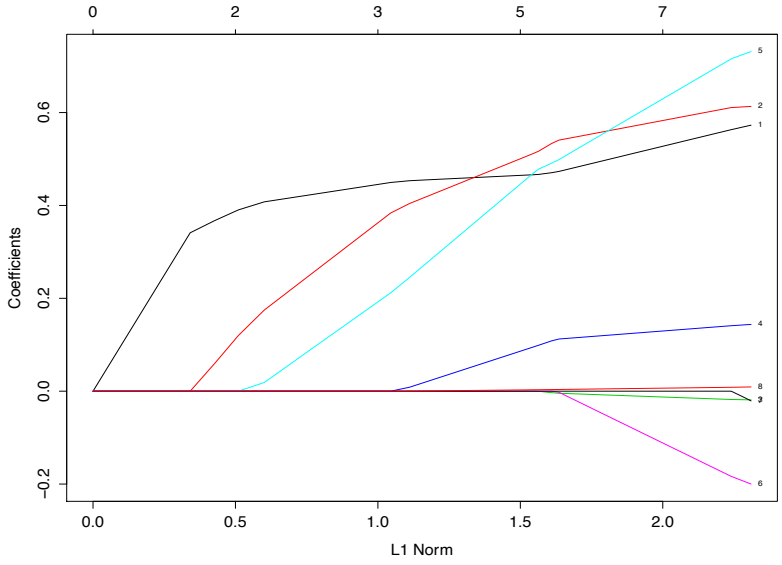
For a set of possible λ values, model fit is evaluated on the calibration set.

Different λ values apply different levels of regularization on the L1 norm $\|\boldsymbol{\beta}\|_1$, resulting in different numbers of explanatory variables kept in the model.

In the following plots:

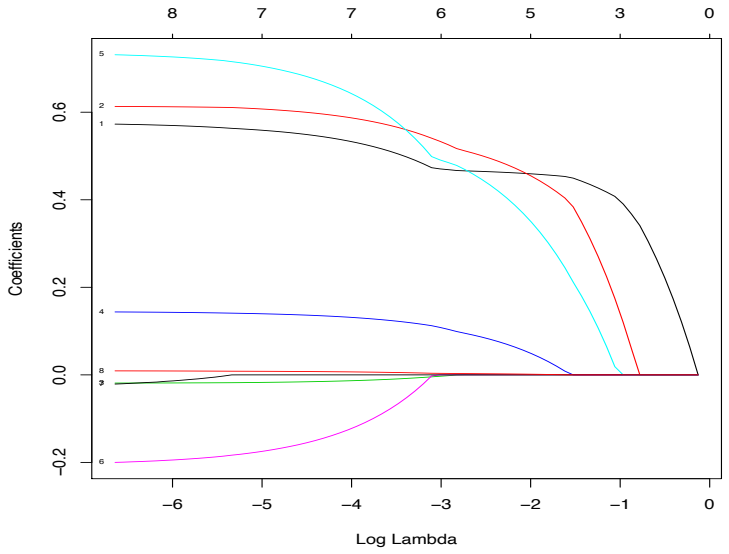
Top axis: DF = number of explanatory variables kept in the model
Bottom axis: L1 norm of $\boldsymbol{\beta}$ (can ask for xvar="lambda" to get log(λ))
Vertical axis: values of coefficients β_i

Visualization of the coefficients in the model (on L1 norm)

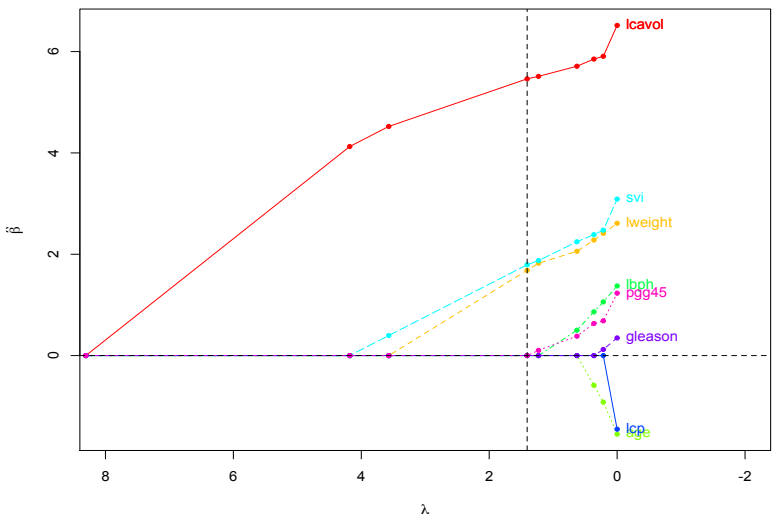


Visualization of the coefficients in the model (on log(λ))

plot(trainfit,label=T,xvar="lambda")



A fancier visualization of the coefficients, x-axis in λ



Model fitting process

```
> print(trainfit)  # no of nonzero coeff(DF), %dev explained, lambda value

Call:  glmnet(x = x, y = y)
      Df    %Dev   Lambda
 [1,]  0 0.00000 0.87890
 [2,]  1 0.09126 0.80080
 [3,]  1 0.16700 0.72970
 [4,]  1 0.22990 0.66480
 [5,]  1 0.28220 0.60580
 [6,]  1 0.32550 0.55200
 [7,]  1 0.36150 0.50290
 [8,]  1 0.39140 0.45820
 [9,]  2 0.42810 0.41750
[10,]  2 0.45980 0.38040
[11,]  3 0.48770 0.34660
[12,]  3 0.51310 0.31590
[13,]  3 0.53420 0.28780
[14,]  3 0.55180 0.26220
[15,]  3 0.56630 0.23890
[16,]  3 0.57840 0.21770
[17,]  5 0.59170 0.19840
[18,]  5 0.60450 0.18070
  :
[24,]  5 0.64650 0.10340
[25,]  5 0.64990 0.09424
  :
[31,]  6 0.66320 0.05393
[32,]  6 0.66530 0.04914
  :
[69,]  8 0.69430 0.00157
[70,]  8 0.69430 0.00143
[71,]  8 0.69430 0.00130
```


Model selection: Many choices of tuning parameters (s,or λ)

```
> coef(trainfit,s=0.1)      # 6 term (including intercept)
9 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) -0.064140435
lcavol       0.462712201
lweight      0.483361985
age          .
lbph         0.072282456
svi          0.410200201
lcp          .
gleason      .
pgg45        0.002245818

> coef(trainfit,s=0.05)     # 7 terms (including intercept)
9 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) -0.112643640
lcavol       0.470217454
lweight      0.532127691
age         -0.002943140
lbph         0.107622441
svi          0.489955346
lcp          .
gleason      .
pgg45        0.003463712
```


Parameter-dependent model prediction on validation set

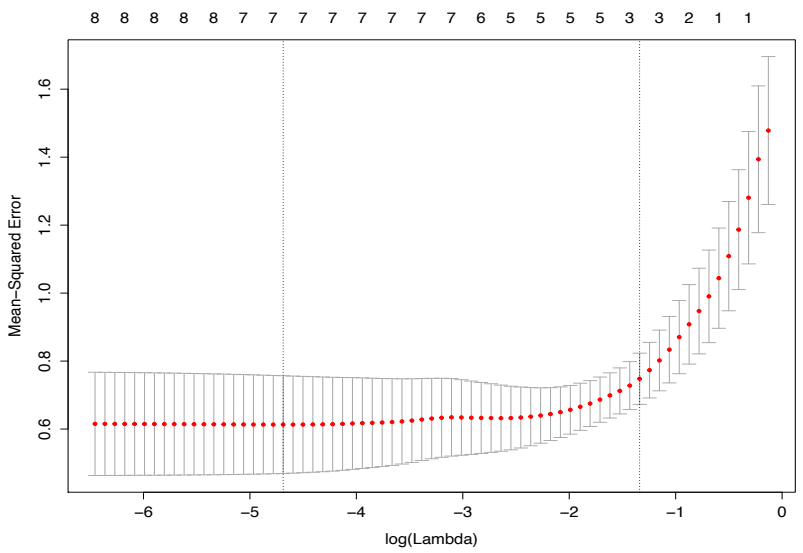
```
nx=as.matrix(calibrate[,1:8])                  # saved validation data
cbind(predict(trainfit,newx=nx, s=c(0.1,0.05)),calibrate[,9])  # predict of s=.1, .05, true y
           1        2
7   2.000388 1.960186 0.7654678
9   1.187210 1.118166 1.0473190
10  1.507195 1.384180 1.0473190
15  2.068642 1.986382 1.3987169
22  2.732481 2.770052 1.6582281
25  2.002415 1.989092 1.7316555
26  2.137355 2.062555 1.7664417
28  1.795435 1.823814 1.8164521
32  1.982366 1.989699 2.0082140
34  1.419724 1.322725 2.0215476
36  2.700799 2.757486 2.0856721
42  2.305471 2.210431 2.3075726
44  2.552468 2.486081 2.3749058
48  2.638200 2.704350 2.5687881
49  2.334230 2.293865 2.5915164
50  2.324743 2.301932 2.5915164
53  2.194626 2.276820 2.6844403
54  3.129746 3.209202 2.6912431
55  3.005596 2.935674 2.7047113
57  1.680437 1.596517 2.7880929
62  3.275113 3.423275 2.8535925
64  3.460785 3.648120 2.8820035
65  2.546161 2.452543 2.8820035
66  2.605518 2.646575 2.8875901
73  2.701967 2.701174 3.0563569
74  2.995277 3.146681 3.0750055
80  3.082012 3.067295 3.5130369
84  3.267951 3.360083 3.5709402
95  3.255211 3.284046 5.1431245
97  3.950538 4.041404 5.5829322
```

Alternative model selection: Cross validation

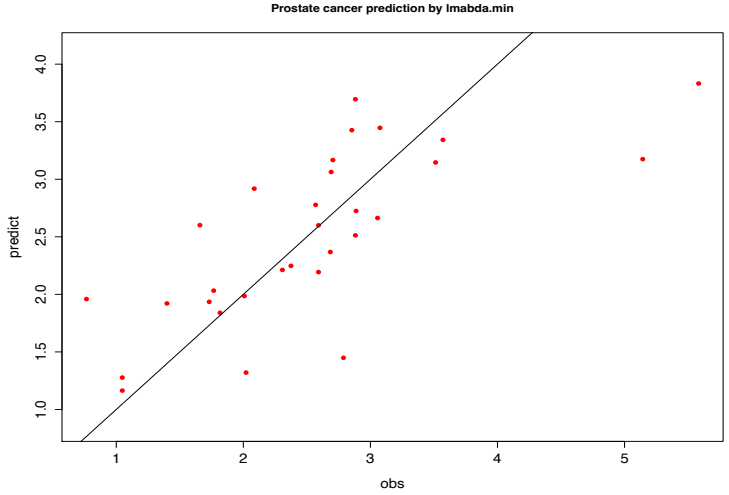
```
> tcvfit = cv.glmnet(x,y)  # cross-validation, on training data; nfold=10 default

> tcvfit$lambda.min    # .0092, log = -4.68776, for example
> tcvfit$lambda.1se    # 0.26, log=-1.338545, for example

> coef(tcvfit, s="lambda.min") #8 x's
9 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept)  0.193668298
lcavol       0.553290578
lweight      0.603005047
age         -0.016384627
lbph         0.137824087
svi          0.691699046
lcp         -0.163512605
gleason      .
pgg45        0.007860603

> coef(tcvfit, s="lambda.1se") #4 x's
9 x 1 sparse Matrix of class "dgCMatrix"
                     1
(Intercept) 0.7185225
lcavol      0.4350437
lweight     0.3115982
age         .
lbph        .
svi         0.1452013
lcp         .
gleason     .
pgg45       .
```


Choose tuning parameters by Mean Squared Error

```
plot(tcvfit)
```

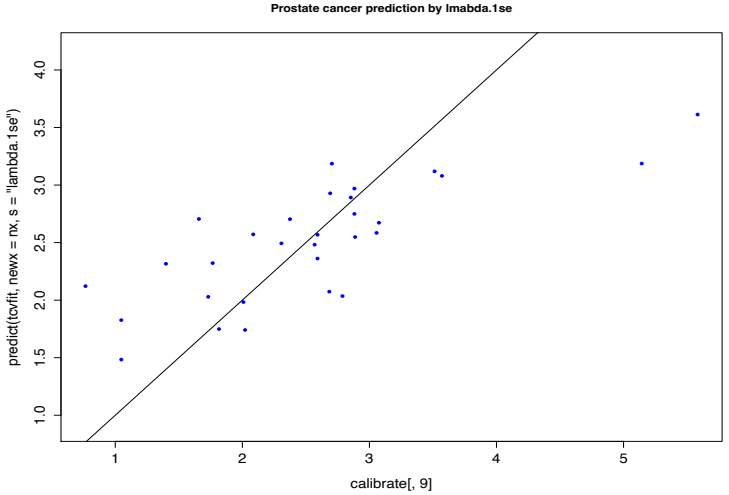


`lambda.min`: The λ minimizing cross-validated mean squared error




```
plot(calibrate[,9],predict(tcvfit,newx=nx,s="lambda.min"),asp=1,xlab="obs",ylab="predict",col=2,pch=16)
abline(0,1)
title(main="Prostate cancer prediction by lmabda.min",cex.main=.8)
```


`lambda.1se`: The λ of cross-validated MSE within one s.e. of the min




```
plot(calibrate[,9],predict(tcvfit,newx=nx,s="lambda.1se"),asp=1,,col=4,pch=16,cex=.8)
abline(0,1)
title(main="Prostate cancer prediction by lmabda.1se",cex.main=.8)
```