Lab 03 - Lasso

[Sicily Xie]

2022-10-12

Brief description

In the lectures last week, you learned about the concept of effective degrees of freedom as a measure of the complexity of a regression model. The LASSO was also introduced as another regularization method.

This week, we will continue to work on the prostate cancer data. Recall that the 9th variable, `lpsa`, is the response while the rest are potential predictors. the variable `lpsa` in the 9th column is the response while the rest are potential predictors. The data set is available in the `{ElemStatLearn}` package and a description can be found at http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/prostate.info.txt

Questions

Recall that the ridge regression imposes an L_2 penalty on the magnitudes of the coefficients (i.e., the $\lambda\boldsymbol{\beta}^\intercal\boldsymbol{\beta}$ term in the objective function). The LASSO, on the other hand, uses an L_1 penalty instead. For centred response $y_1, \ldots, y_n$ and vector of predictors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, we have

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^\intercal \boldsymbol{x}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right],$$

for $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\intercal$ and some penalty λ.

Q1 Give one reason why you might use LASSO instead of the ridge regression.

Since not all our predictors appear to have explanatory power on the response variable 'lpsa', and LASSO does model selection and select variables (by setting coefficients to zero), we may use LASSO instead of ridge regression.

Like last time, read the data into R using the following command (with correct file path and name):

```
data(prostate, package = "ElemStatLearn")
prostate <- subset(prostate, select = -train)
library(glmnet)
```

Again, we use the function `glmnet` in the package of the same name. Assuming that you have already installed the package, type `library(glmnet)` to load it into the current R session. Read the documentation of the function by typing `?glmnet` in the console. We are interested in these three arguments of the function: `x`, `y`, `alpha`.

Q2 Fit the LASSO using the `glmnet` function. Write down the code you used for the fitting (just the line involving the `glmnet` function).

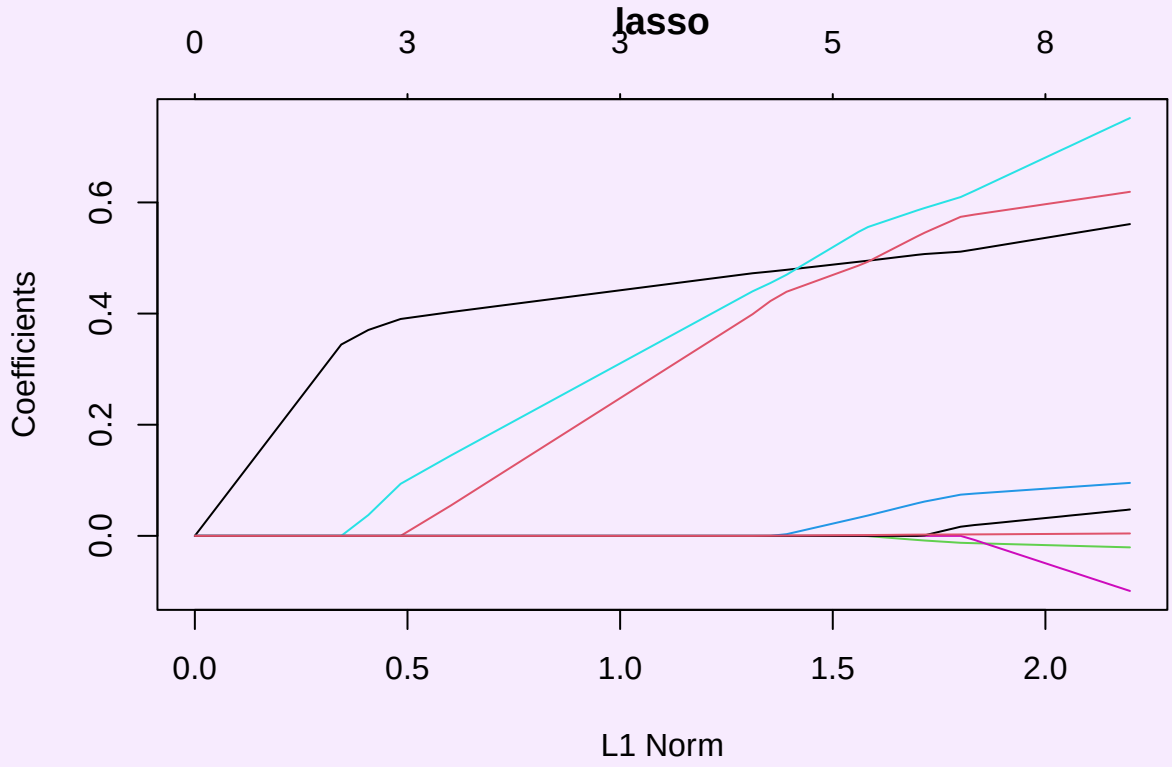
Hint: Recall the `as.matrix` function you used on the design matrix last time.

```
# some code
library(dbplyr)
X <- prostate %>% dplyr::select(-lpsa) %>% as.matrix()
Y <- prostate$lpsa
Lasso <- glmnet(x = X, y = Y, alpha = 1)
```

Q3 Inspect how the coefficients evolve as the penalty is relaxed by plotting the fitted object using the `plot` function. Write down the first three variables that enter the model.

Hint: `plot` is a method function. When invoked, it calls the plotting function that corresponds to the class of the object supplied. The relevant documentation in this case is `?plot.glmnet`.

```
# some code
plot(Lasso, main = "lasso")
```



```
coef(Lasso)
```

```
## 9 x 70 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 70 column names 's0', 's1', 's2' ... ]]

##
## (Intercept) 2.478387 2.39211797 2.313513 2.2418910 2.1766317 2.1171699
## lcavol      .        0.06390244 0.122128 0.1751809 0.2235207 0.2675662
## lweight     .        .          .        .         .         .
## age         .        .          .        .         .         .
## lbph        .        .          .        .         .         .
## svi         .        .          .        .         .         .
## lcp         .        .          .        .         .         .
## gleason     .        .          .        .         .         .
## pgg45       .        .          .        .         .         .
##
## (Intercept) 2.0629905 2.0136243 1.97009261 1.93129218 1.70556859 1.4850051
## lcavol      0.3076988 0.3442661 0.37049910 0.39023472 0.40270479 0.4136549
## lweight     .         .         .          .          0.05452806 0.1084723
## age         .         .         .          .          .          .
## lbph        .         .         .          .          .          .
## svi         .         .         0.03749248 0.09364696 0.14450117 0.1907849
## lcp         .         .         .          .          .          .
## gleason     .         .         .          .          .          .
## pgg45       .         .         .          .          .          .
##
## (Intercept) 1.2840409 1.1009298 0.9340858 0.7820637 0.6435469 0.5173355
## lcavol      0.4236222 0.4327040 0.4409790 0.4485189 0.4553889 0.4616486
## lweight     0.1576257 0.2024125 0.2432206 0.2804033 0.3142829 0.3451527
## age         .         .         .         .         .         .
## lbph        .         .         .         .         .         .
## svi         0.2329720 0.2714114 0.3064360 0.3383490 0.3674270 0.3939218
## lcp         .         .         .         .         .         .
## gleason     .         .         .         .         .         .
## pgg45       .         .         .         .         .         .
##
## (Intercept) 0.4023364 0.2975535 0.1984562118 0.1248350124 0.0770691525
## lcavol      0.4673523 0.4725492 0.4756432483 0.4787402803 0.4817230671
## lweight     0.3732801 0.3989087 0.4225418494 0.4392275909 0.4489707416
## age         .         .         .            .            .
## lbph        .         .         .            0.0030306359 0.0091805326
## svi         0.4180628 0.4400593 0.4549684954 0.4698725647 0.4859221771
## lcp         .         .         .            .            .
## gleason     .         .         .            .            .
## pgg45       .         .         0.0002431869 0.0004629353 0.0006388846
##
## (Intercept) 0.0334727776 -0.0062505838 -0.042445029 -0.075424060 -0.105473324
## lcavol      0.4844638281  0.4869613120  0.489236925  0.491310379  0.493199634
## lweight     0.4578643115  0.4659677414  0.473351285  0.480078894  0.486208841
```

```
## age          .            .            .            .            .
## lbph         0.0147746767 0.0198718560 0.024516216  0.028747984  0.032603813
## svi          0.5004651429 0.5137160994 0.525789879  0.536791058  0.546814922
## lcp          .            .            .            .            .
## gleason      .            .            .            .            .
## pgg45        0.0007993421 0.0009455417 0.001078753  0.001200131  0.001310726
##
## (Intercept) -0.1112475272 -0.054930095 -0.003724989  0.042931876  0.085443855
## lcavol       0.4951681653  0.497594361  0.499896523  0.501993222  0.503903655
## lweight      0.4933432034  0.504244149  0.514104337  0.523088488  0.531274515
## age         -0.0004409811 -0.002097352 -0.003601995 -0.004972962 -0.006222137
## lbph         0.0366675517  0.041886589  0.046648497  0.050987510  0.054941055
## svi          0.5557469463  0.562836673  0.569322656  0.575234035  0.580620262
## lcp          .             .            .            .            .
## gleason      .             .            .            .            .
## pgg45        0.0014357637  0.001623422  0.001792333  0.001946244  0.002086482
##
## (Intercept)  0.124179187  0.1537694867  0.161190508  0.169279155  0.176764617
## lcavol       0.505644371  0.5071477800  0.508157222  0.509085165  0.509932399
## lweight      0.538733318  0.5455934491  0.552486888  0.558643448  0.564250670
## age         -0.007360338 -0.0084065349 -0.009374709 -0.010256773 -0.011060542
## lbph         0.058543379  0.0618168146  0.064736545  0.067418867  0.069862537
## svi          0.585527991  0.5899942923  0.594693521  0.598979114  0.602871895
## lcp          .            .             .            .            .
## gleason      .            0.0009732887  0.004797183  0.008139093  0.011167617
## pgg45        0.002214262  0.0023140828  0.002351087  0.002387475  0.002421009
##
## (Intercept)  0.183600037  0.189830165  0.188758180  0.189219422  0.188572690
## lcavol       0.510704641  0.511408314  0.515331944  0.519714141  0.523679178
## lweight      0.569359455  0.574014350  0.578176649  0.582064106  0.585614172
## age         -0.011792915 -0.012460227 -0.013195979 -0.013915220 -0.014566826
## lbph         0.072089068  0.074117794  0.076108313  0.077938717  0.079606407
## svi          0.606417244  0.609647425  0.620999448  0.633529800  0.644914372
## lcp          .            .           -0.007367663 -0.016226062 -0.024209154
## gleason      0.013924953  0.016437056  0.019456896  0.021951213  0.024369822
## pgg45        0.002451612  0.002479503  0.002614410  0.002783472  0.002932379
##
## (Intercept)  0.187557496  0.187004619  0.186519809  0.186079199  0.185677798
## lcavol       0.527313407  0.530603038  0.533600234  0.536331183  0.538819523
## lweight      0.588813498  0.591762578  0.594450526  0.596899703  0.599131302
## age         -0.015155012 -0.015696168 -0.016189387 -0.016638797 -0.017048282
## lbph         0.081126028  0.082510813  0.083772457  0.084922017  0.085969453
## svi          0.655172784  0.664633113  0.673253871  0.681108816  0.688265952
## lcp         -0.031389445 -0.038019638 -0.044063231 -0.049570041 -0.054587648
## gleason      0.026618791  0.028629958  0.030459988  0.032127290  0.033646464
## pgg45        0.003064293  0.003187926  0.003300706  0.003403473  0.003497111
##
## (Intercept)  0.18531206  0.184432043  0.184120033  0.183886618  0.183679367
## lcavol       0.54108681  0.543163791  0.545045098  0.546759224  0.548321154
## lweight      0.60116465  0.602993677  0.604682444  0.606222590  0.607625966
## age         -0.01742139 -0.017755653 -0.018065566 -0.018348272 -0.018605889
## lbph         0.08692384  0.087790912  0.088583648  0.089305806  0.089963800
```

```
## svi          0.69478727  0.700587694  0.706011030  0.710955705  0.715461304
## lcp         -0.05915950 -0.063221072 -0.067019360 -0.070486225 -0.073645635
## gleason      0.03503068  0.036351369  0.037502711  0.038545162  0.039494276
## pgg45        0.00358243  0.003656148  0.003726989  0.003791853  0.003850985
##
## (Intercept)  0.183491098  0.183319614  0.183163371  0.182421537  0.18222956
## lcavol       0.549744336  0.551041088  0.552222641  0.553298030  0.55427829
## lweight      0.608904673  0.610069783  0.611131388  0.612090141  0.61297073
## age         -0.018840623 -0.019054505 -0.019249385 -0.019421708 -0.01958331
## lbph         0.090563338  0.091109614  0.091607361  0.092055982  0.09246970
## svi          0.719566658  0.723307305  0.726715643  0.729671679  0.73250603
## lcp         -0.076524427 -0.079147480 -0.081537509 -0.083613290 -0.08559377
## gleason      0.040358996  0.041146889  0.041864786  0.042584093  0.04318924
## pgg45        0.003904867  0.003953962  0.003998696  0.004035688  0.00407250
##
## (Intercept)  0.18214871  0.182092582  0.182044637  0.18126230  0.18104474
## lcavol       0.55517187  0.555986376  0.556728588  0.55738917  0.55800352
## lweight      0.61377457  0.614507048  0.615174459  0.61578961  0.61634272
## age         -0.01973112 -0.019865876 -0.019988673 -0.02009487 -0.02019606
## lbph         0.09284663  0.093190065  0.093502991  0.09377964  0.09403963
## svi          0.73509623  0.737457117  0.739608397  0.74138737  0.74316684
## lcp         -0.08740909 -0.089064822 -0.090573757 -0.09183363 -0.09307040
## gleason      0.04372856  0.044217631  0.044662827  0.04514679  0.04554048
## pgg45        0.00410659  0.004137745  0.004166149  0.00418803  0.00421067
##
## (Intercept)  0.18101456  0.181040084  0.181079433  0.181120127  0.180367787
## lcavol       0.55856495  0.559077561  0.559545008  0.559971046  0.560325276
## lweight      0.61684742  0.617307081  0.617725797  0.618107280  0.618476646
## age         -0.02028920 -0.020374307 -0.020451913 -0.020522644 -0.020581422
## lbph         0.09427699  0.094493332  0.094690472  0.094870103  0.095022449
## svi          0.74480675  0.746304045  0.747669098  0.748913107  0.749851879
## lcp         -0.09421656 -0.095265858 -0.096223368 -0.097096241 -0.097775686
## gleason      0.04587810  0.046178753  0.046450556  0.046697568  0.047003757
## pgg45        0.00423222  0.004252122  0.004270335  0.004286953  0.004298425
##
## (Intercept)  0.180062440  0.179993096
## lcavol       0.560672762  0.560993437
## lweight      0.618795888  0.619084904
## age         -0.020638683 -0.020692037
## lbph         0.095170395  0.095306738
## svi          0.750853157  0.751799943
## lcp         -0.098467238 -0.099122783
## gleason      0.047253050  0.047454995
## pgg45        0.004310552  0.004322661
```

The first three predictors enter the model are 'lcavol', 'lweight', and 'svi'.

Now, we choose the optimal value of λ via cross validation. Recall how you used `cv.glmnet` last time with the `nfolds` argument in addition to those above (check the documentation of this function).

Carry out a 5-fold cross validation. Run `set.seed(400)` **immediately before** the `cv.glmnet` function to obtain reproducible results.

Q4 What is the value of λ that gives the smallest mean squared error?

```
set.seed(400)
cv_fit <- cv.glmnet(X,Y,nfolds =5,alpha = 1)
cv_fit$lambda.min
```

```
## [1] 0.0824037
```

The value of λ that gives the smallest mean squared error is 0.0824037.

Q5 How many predictors are in the model with this value of λ?

```
# some code
coef(cv_fit, s = cv_fit$lambda.min)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -0.042445029
## lcavol       0.489236925
## lweight      0.473351285
## age          .
## lbph         0.024516216
## svi          0.525789879
## lcp          .
## gleason      .
## pgg45        0.001078753
```

There are 5 predictors are in the model with this value of λ.

We compare this and the optimal ridge regression model obtained last time. For design matrix $\boldsymbol{X}$ (without intercept) with singular value decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix of singular values $d_1, \ldots, d_p$, the effective degrees of freedom (edf) of the ridge regression model is

$$\text{edf} = \sum_{i=1}^{p} \left(\frac{d_i^2}{d_i^2 + \lambda} \right),$$

where λ is the penalty.

Q6 The optimal ridge regression model (from Lab 2) has $\lambda = e^{-2.4}$. What is the edf of this model?

Hint: Centre the $\boldsymbol{X}$ matrix first. You will find the `svd` function useful.

```r
# some code
s <- svd(X)
rid_df <- sum((s$d)^2/((s$d)^2+exp(-2.4)))
rid_df
```

```
## [1] 7.978551
```

The edf of ridge regression model is 7.978551.

Q7 From the results in **Q5** and **Q6**, which model is more complex based on their edf's?

The ridge regression model is more complex.