Penalized Regression

G. de los Campos

12/05/2022

Up to now, we considered methods that estimate parameters by either maximizing the likelihood function (ML) or by minimaizing the residual sum of squares (OLS). These methods have good statistical properties (e.g., OLS is unbiased and has minimum variance among the class of linear unbiased estiamtors, ML is asymptotically unbiased and asymptotically efficient). However, the performace of these methods can be sub-optimal when the number of parameters to be estimated (e.g., the number of regression coefficients) is large relative to sample size and when predictors are highly colinear. One way to circunvent this problem is by using variable seleciton procedures, some of which can be obtained by using penalized regresssions.

1) Penalized Regression

In a penalized regression, estimates are obtained by minimizing a penalized log-likelihood or, in the case of linear models, a penalized residual sum of squares:

$\hat{\beta} = argmin\{(y - X\beta)'(y - X\beta) + \lambda J(\beta)\}$

where $J(\beta)$ is a penalty function.

Choosing $\lambda = 0$ leads Ordinary Least Squares estimates; however, for any $\lambda > 0$ the solution won't be eqivalent to OLS.

Common choices for the penalty function are the

- L2-norm, $J(\beta) = \sum_j \beta_j^2$ (aka Ridge Regression, Hoerl and Kennard 1970),
- L-1 norm $J(\beta) = \sum_j |\beta_j|$ (aka Lasso, Tibshirani, 1996), and,
- Linear combinations of the two $J(\beta) = (1 - \alpha)\sum_j \beta_j^2 + \alpha \sum_j |\beta_j|$ (aka Elastic Net, Zhou and Hastie, 2005), for some $\alpha \in [0, 1]$.

Ridge regression shrunk OLS estimates towards zero, without making variable selection. Lasso and Elastic Net combine variable selection and shrinkage.

Commonly, these models are fitted over a grid of values of the regularization parameter (λ) and optimal value for the penalization parameter is often chosen by evaluating the ability of the fitted models to predict data that was not used to train the models (i.e., testing data).

Best subset selection is obtained using as a penality a function that counts the number of non-zero effects: $J(\beta) = \Sigma_j 1(\beta_j \neq 0))$. This approach identifies the best subset of predictors and estimate their coefficients without any shrinkage. However, the optimization problem of best-subset selction is non-convex.

Forward regression is an apporach where we select predictors sequentially until some stopping criteria is met. This approch can be viewed as an approximation to best-subset selection; however, a forward regression does not guarantee optimal best subset selection.

2) Data

To illustrate these methods we will use a famous prostate cancer data set which can be found in this link.

```r
DATA=read.table('https://hastie.su.domains/ElemStatLearn/datasets/prostate.data',header=TRUE)
head(DATA)
```

```
##       lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
## 1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
## 2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189
## 4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636
## 6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678
##   train
## 1  TRUE
## 2  TRUE
## 3  TRUE
## 4  TRUE
## 5  TRUE
## 6  TRUE
```

```r
train=DATA[,'train']
DATA=DATA[,-ncol(DATA)]
```

Our objective is to build a prediction model for the log of the PSA (`lpsa`) as a function of the other variables included in the data set.

The function takes a starting model (fm0 in the example), a `scope` argument defining the range of models to be examined, and a direction for the search (`forward`) in the example below.

The function prints, for each step fo the forward search the change produce in the model by each of the variables that were not yet included and picks one. The final model is returned.

3) OLS with all predictors

```r
fm=lm(lpsa~.,data=DATA)
summary(fm)
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = DATA)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76644 -0.35510 -0.00328  0.38087  1.55770
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.181561   1.320568   0.137  0.89096
## lcavol       0.564341   0.087833   6.425 6.55e-09 ***
## lweight      0.622020   0.200897   3.096  0.00263 **
## age         -0.021248   0.011084  -1.917  0.05848 .
## lbph         0.096713   0.057913   1.670  0.09848 .
## svi          0.761673   0.241176   3.158  0.00218 **
## lcp         -0.106051   0.089868  -1.180  0.24115
## gleason      0.049228   0.155341   0.317  0.75207
## pgg45        0.004458   0.004365   1.021  0.31000
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6995 on 88 degrees of freedom
## Multiple R-squared:  0.6634, Adjusted R-squared:  0.6328
## F-statistic: 21.68 on 8 and 88 DF,  p-value: < 2.2e-16
```

If we were to use pvalues, we would select `lcavol`, `lweight`, and `svi`; however, due to colinearity, we may be leaving out of the model important variables.

4) Forward Regression

The `step()` function can be used to implement forward regression (as well as backgward elimination and stepwise methods that combine both forward and backward regression).

```
fm0=lm(lpsa~1,data=DATA)
fullModel='lpsa ~ lcavol+lweight+age+lbph+svi+lcp+gleason+pgg45'
fwd=step(fm0,scope=fullModel,direction='forward',data=DATA)
```

```
## Start:  AIC=28.84
## lpsa ~ 1
##
##           Df Sum of Sq     RSS     AIC
## + lcavol   1    69.003  58.915 -44.366
## + svi      1    41.011  86.907  -6.658
## + lcp      1    38.528  89.389  -3.926
## + lweight  1    24.019 103.899  10.665
## + pgg45    1    22.814 105.103  11.783
## + gleason  1    17.416 110.502  16.641
## + lbph     1     4.136 123.782  27.650
## + age      1     3.679 124.239  28.007
## <none>                 127.918  28.838
##
## Step:  AIC=-44.37
## lpsa ~ lcavol
##
##           Df Sum of Sq    RSS     AIC
## + lweight  1    7.1726 51.742 -54.958
## + svi      1    5.2375 53.677 -51.397
## + lbph     1    3.2658 55.649 -47.898
## + pgg45    1    1.6980 57.217 -45.203
## <none>                 58.915 -44.366
## + lcp      1    0.6562 58.259 -43.452
## + gleason  1    0.4156 58.499 -43.053
## + age      1    0.0025 58.912 -42.370
##
## Step:  AIC=-54.96
## lpsa ~ lcavol + lweight
##
##           Df Sum of Sq    RSS     AIC
## + svi      1    5.1737 46.568 -63.177
## + pgg45    1    1.8158 49.926 -56.424
## <none>                 51.742 -54.958
## + lcp      1    0.8187 50.923 -54.506
```

```
## + gleason  1    0.7163 51.026 -54.311
## + age      1    0.6456 51.097 -54.176
## + lbph     1    0.4440 51.298 -53.794
##
## Step:  AIC=-63.18
## lpsa ~ lcavol + lweight + svi
##
##           Df Sum of Sq    RSS     AIC
## + lbph     1   0.97296 45.595 -63.226
## <none>                 46.568 -63.177
## + age      1   0.62301 45.945 -62.484
## + pgg45    1   0.50069 46.068 -62.226
## + gleason  1   0.34449 46.224 -61.898
## + lcp      1   0.06937 46.499 -61.322
##
## Step:  AIC=-63.23
## lpsa ~ lcavol + lweight + svi + lbph
##
##           Df Sum of Sq    RSS     AIC
## + age      1   1.15879 44.437 -63.723
## <none>                 45.595 -63.226
## + pgg45    1   0.33173 45.264 -61.934
## + gleason  1   0.20691 45.389 -61.667
## + lcp      1   0.10115 45.494 -61.441
##
## Step:  AIC=-63.72
## lpsa ~ lcavol + lweight + svi + lbph + age
##
##           Df Sum of Sq    RSS     AIC
## <none>                 44.437 -63.723
## + pgg45    1   0.66071 43.776 -63.176
## + gleason  1   0.47674 43.960 -62.769
## + lcp      1   0.13040 44.306 -62.008
```

Remarks:

- For large-scale screenings the step function can be too slow. As an altern ative you can use the `FWD()` function of the `BGDataExt` package.
- The functions stepAIC() and stepBIC() of the MASS package peform forward regression by minimizing AIC and BIC, respectively.

5) Lasso regression using the glmnet package

The following example shows how to fit a Lasso regression using the glmnet package.

- $\alpha = 1$ is used to fit Lasso (use $\alpha = 0$ for Ridge Regression, and any value betwen 0 and 1 for Elastic Net).
- By default `glmnet()` runs models over a grid of 100 values of the penalization parameter (λ)
- The object returned by `glmnet()` is a list containing:
  - `$lambda` the grid of values of λ,
  - `$beta` a matrix with estimated coefficients (rows) for each value of λ in the grid (columns).
  - `$df`, the number of non-zero effects in the solution for each value of λ.
  - `$a0`, the estimated intercept for each value of λ.

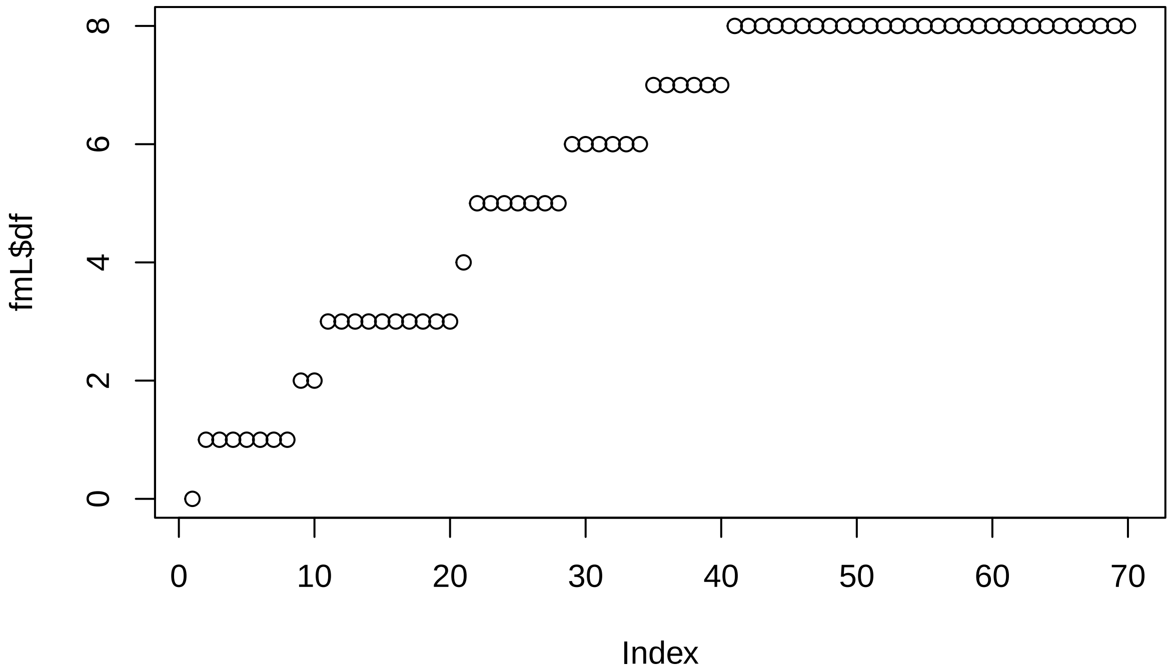
```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```r
y=DATA[,'lpsa']
X=as.matrix(DATA[,colnames(DATA)!='lpsa'])
fmL=glmnet(y=y,x=X,alpha=1)

plot(fmL$df)
```



There were changes in the model DF at the follwoing steps

```r
which(diff(fmL$df)!=0)
```

```
## [1]  1  8 10 20 21 28 34 40
```

Finding what predictors were active at each step in the grid

```r
for(i in which(diff(fmL$df)!=0)){
    message('----- step ',i,' --------')
    print(colnames(X)[fmL$beta[,i]!=0])
}
```

```
## ----- step 1 --------
```

```
## character(0)
```

```
## ----- step 8 --------
```

```
## [1] "lcavol"
```

```
## ----- step 10 --------
```

```
## [1] "lcavol" "svi"
```

```
## ----- step 20 --------
```

```
## [1] "lcavol"  "lweight" "svi"
## ----- step 21 --------
## [1] "lcavol"  "lweight" "svi"     "pgg45"
## ----- step 28 --------
## [1] "lcavol"  "lweight" "lbph"    "svi"     "pgg45"
## ----- step 34 --------
## [1] "lcavol"  "lweight" "age"     "lbph"    "svi"     "pgg45"
## ----- step 40 --------
## [1] "lcavol"  "lweight" "age"     "lbph"    "svi"     "gleason" "pgg45"
```