Méthodes d'optimisation pour le Machine Learning

Stéphane Canu
scanu@insa-rouen.fr, asi.insa-rouen.fr\~scanu

Lundi 11 septembre 2023

The objectives of the lab

The purpose of this lab is to reproduce tables 3.1 and 3.2 from the third chapter of the book "Elements of Statistical Learning" from Hastie, Tibshirani and Friedman, as shown bellow. The objective of the homework is to reproduce partially table 3.3.

TABLE 3.1. *Correlations of predictors in the prostate cancer data.*

	lcavol	lweight	age	lbph	svi	lcp	gleason
lweight	0.300						
age	0.286	0.317					
lbph	0.063	0.437	0.287				
svi	0.593	0.181	0.129	−0.139			
lcp	0.692	0.157	0.173	−0.089	0.671		
gleason	0.426	0.024	0.366	0.033	0.307	0.476	
pgg45	0.483	0.074	0.276	−0.030	0.481	0.663	0.757

TABLE 3.2. *Linear model fit to the prostate cancer data. The Z score is the coefficient divided by its standard error (3.12). Roughly a Z score larger than two in absolute value is significantly nonzero at the $p = 0.05$ level.*

Term	Coefficient	Std. Error	Z Score
Intercept	2.46	0.09	27.60
lcavol	0.68	0.13	5.37
lweight	0.26	0.10	2.75
age	−0.14	0.10	−1.40
lbph	0.21	0.10	2.06
svi	0.31	0.12	2.47
lcp	−0.29	0.15	−1.87
gleason	−0.02	0.15	−0.15
pgg45	0.27	0.15	1.74

The following code is available on line with google colab at
`https://colab.research.google.com/drive/1-gaATpexHmUqJvChanNrjTeafOFI3e9k`

Ex. 1 — Tables 3.1 and 3.2

1. Prepare the data
   a) Raw data is available on line, download it from moodle (`Data.txt` file) or from the web at `https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data`.

```
import pandas as pd

url_data = "https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data"
df = pd.read_csv(url_data, delimiter='\t')
```

   b) Extract and normalize the explicative variables

```
variables = df.columns[1:9]
df[variables] = df[variables].apply(lambda x: (x - x.mean()) / x.std())
```

c) Is it wise to normalize these data?

d) Split the dataset into training and test data

```
# Get the training and test sets
Y_train = df.loc[df["train"]=="T", 'lpsa'].as_matrix()
X_train = df.loc[df["train"]=="T", variables].as_matrix()
print("Training set : n = {} samples and p = {} dimensions".format(X_train.shape[0],
     X_train.shape[1]))

Y_test = df.loc[df["train"]=="F", 'lpsa'].as_matrix()
X_test = df.loc[df["train"]=="F", variables].as_matrix()
print("Test set : n = {} samples and p = {} dimensions".format(X_test.shape[0], X_
    test.shape[1]))
```

2. Compute the correlations of predictors in the prostate cancer data as presented Table 3.1

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
Xn = (X_train - X_train.mean(axis=0))/X_train.std(axis=0)
n, p = Xn.shape
C = Xn.T@Xn/n
# or C = np.corrcoef(X_train.T)
plt.figure(figsize=(6, 4))
sns.heatmap(C, annot=True, fmt=".3f", vmin=-1)
```

3. Reproduce the results presented Table 3.2

a) Compute the coefficients of the linear regression model, without using the `lm` function (but you can use it validate your code)

```
X = np.concatenate( (np.ones((X_train.shape[0],1)), X_train), axis=1)
b_ls = np.linalg.solve(X.T@X, X.T@Y_train)
```

b) Compute the prediction error

```
n, p = X.shape
y_hat = X@b_ls
err = Y_train - y_hat
```

c) Compute the standard error for each variable

```
sigma_square = np.dot(Y_train - y_hat, Y_train - y_hat)/(n-p)
vector_v = np.diag(np.linalg.inv(X.T@X))
std_error_coef = np.sqrt(sigma_square * vector_v)
```

d) compute the Z score for each variable

```
z_score = b_ls/(std_error_coef)
```

e) visualize the results and compare with table 3.2

```
dash = '-' * 50
print(dash)
print("{:<11s}{:<15s}{:<13s}{:<10s}".format("Term", "Coef", "Std. error","Z score"))
print(dash)
for k in range(variables.shape[0]+1):
  if k==0:
    print("{:<10s}{:>12.2f}{:>12.2f}{:>12.2f}".format("Intercept", b_ls[k], std_
        error_coef[k], z_score[k]))
  else:
    print("{:<10s}{:>12.2f}{:>12.2f}{:>12.2f}".format(variables[k-1], b_ls[k], std_
        error_coef[k], z_score[k]))
print(dash)
```

Ex. 2 — Questions of means and preprocessing

1. (Hastie et al. Ex. 3.5 augmented; ridge regression/Lasso with intercept, $\Leftrightarrow$ a ridge regression/Lasso without intercept).
   Consider the ridge regression problem with intercept term (3.41)

$$J(\beta_0, \beta) = \sum_{i=1}^{N} \left[y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right]^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q.$$

   Show that this problem is equivalent to the minimization of

$$J^c(\beta_0^c, \beta^c) = \sum_{i=1}^{N} \left[y_i - \beta_0^c - \sum_{j=1}^{p} (x_{ij} - \overline{x}_j)\beta_j^c \right]^2 + \lambda \sum_{j=1}^{p} \left|\beta_j^c\right|^q.$$

   Give the correspondence between $[\beta_0^c; \beta^c]$ and the original $[\beta_0; \beta]$. Characterize the solution of this modified criterion. Propose a procedure to solve (1) by using

$$\widehat{\beta}^{\mathrm{R}} = (X^\top X + \lambda I)^{-1} X^\top \mathrm{y}.$$

   Show that similar result holds for the Lasso task in terms of the $[\beta_0^c; \beta^c]$ and $[\beta_0; \beta]$ correspondence.

2. Weighted and normalized Lasso
   - Weighted Lasso $\Leftrightarrow$ a standard Lasso: Consider the weighted Lasso optimization problem (on centered data with no intercept):

$$\min_{\beta} \|\mathrm{y} - X\beta\|_2^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j|,$$

     with given positive weight $w_j > 0$ $(j = 1, \ldots, p)$. Show that this problem is equivalent to a standard Lasso task [(3.52)]

$$\min_{\beta'} \|y - X'\beta'\|_2^2 + \lambda \sum_{j=1}^{p} |\beta'_j|.$$

   - Normalized Lasso $\Leftrightarrow$ a weighted Lasso task with intercept: A typical way of using the Lasso is to apply it on normalized data as follows:
     1. Normalize y and X: $\mathrm{y}^r = \frac{y - \overline{y}}{\sigma_y}$ and $x_{ij}^r = \frac{x_{ij} - \overline{x}_j}{\sigma_j}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, p$.
     2. Minimize $\|\mathrm{y}^r - X^r\beta^r\|_2^2 + \lambda \|\beta^r\|_1$ with respect to β^r.

     Show that this problem is equivalent to a weighted Lasso with intercept:

$$\min_{\beta} \|\mathrm{y} - \beta_0 - X\beta\|_2^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j|.$$

     Explain how to retrieve β and β_0 from β^r.

3. (Hastie et al exercice 3.12 ridge regression $\Leftrightarrow$ a least squares problem) Show that the ridge regression estimate can be obtained by ordinary least squares regression on an augmented data set. We augment the centered matrix X with p additional rows containing $\sqrt{\lambda} I_p$, and augment y with p zeros. By introducing artificial data having response value zero, the fitting procedure is forced to shrink the coefficients toward zero. This is related to the idea of hints due to Abu-Mostafa (1995), where model constraints are implemented by adding artificial data examples that satisfy them.

Ex. 3 — Your Turn

1. Reproduce in python Table 3.3, at least the first five columns that is LS, Best Subset, Ridge, Lasso and PCR, implementing in python your own version of the LS, Best Subset, Ridge, Lasso and PCR functions. You are allowed to cheat on the choice of the hyper-parameters, and for instance assume that the best subset is given by the two variables `lcavol` and `lweight`, for the ridge $\lambda = 24.25$, for the lasso $\lambda = 23.39$ and by taking 7 components for the PCR.
2. to solve the Lasso problem you can use CVXpy

```
import cvxpy as cp

Xn = (X_train - X_train.mean(axis=0))/X_train.std(axis=0)
Yn = (Y_train - Y_train.mean())/Y_train.std()
t = .7015
n, p = Xn.shape

b = cp.Variable(p)
o = cp.Minimize(cp.sum_squares(Xn@b-Yn))
c = [cp.norm(b, 1) <= t]
problem = cp.Problem(o, c)
problem.solve(solver=cp.SCS,eps=1e-5)

b_lasso = b.value*Y_train.std()/X_train.std(axis=0)
np.set_printoptions(formatter={'float': '{: 0.3f}'.format})
print("Estimation: ",b_lasso)
```

TABLE 3.3. *Estimated coefficients and test error results, for different subset and shrinkage methods applied to the prostate data. The blank entries correspond to variables omitted.*

Term	LS	Best Subset	Ridge	Lasso	PCR	PLS
Intercept	2.465	2.477	2.452	2.468	2.497	2.452
lcavol	0.680	0.740	0.420	0.533	0.543	0.419
lweight	0.263	0.316	0.238	0.169	0.289	0.344
age	−0.141		−0.046		−0.152	−0.026
lbph	0.210		0.162	0.002	0.214	0.220
svi	0.305		0.227	0.094	0.315	0.243
lcp	−0.288		0.000		−0.051	0.079
gleason	−0.021		0.040		0.232	0.011
pgg45	0.267		0.133		−0.056	0.084
Test Error	0.521	0.492	0.492	0.479	0.449	0.528
Std Error	0.179	0.143	0.165	0.164	0.105	0.152