TP 3 : Le lasso

Stéphane Canu

16 septembre 2024, MLA, INSA Rouen

Le but du TP est d'étudier une méthode de sélection de variables, le Lasso[1], dans le cadre de la régression sur des données simulées. Pour le faire fonctionner, vous êtes supposé avoir déjà installé CVX (que vous pourrez télécharger à cette adresse : `http://cvxr.com/cvx/`)

Le code suivant est disponible en ligne avec google colab :

Ex. 1 — Le Lasso comme une méthode de sélection de variables

1. Génération des données du problème.
   a) Générez les données du problème. Une matrice X de taille $n = 200$ individus et $p = 2n$ variables. Vous prendrez soin de centrer la matrice et de la normaliser de sorte que $\sum_{i=1}^{n} X_{ij}^2 = 1$. Un vecteur de paramètre $w_{opt} \in \mathbb{R}^p$ dont $k = 5$ seulement sont non nulles. Un vecteur de réponses $y = Xw_{opt} + \varepsilon \in \mathbb{R}^n$ où ε est un bruit Gaussien entrainant un rapport signal sur bruit de 2.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cvxpy as cvx
import time

n = 200 # number of examples (you can try with n = 1000 and n = 5000)
p = 2*n # dimensionality of the problem
k = 5   # number of active variables

np.random.seed(0)
X = np.random.randn(n,p)  # creating features and normalizing them
X = (X - np.mean(X,axis = 0))/np.std(X,axis = 0)

t = np.arange(0,p)/(p-1); # bulding the variance matrix !
S = np.zeros((p,p))
nn = 0.00001
for i in range(p):
  S[i,:] = np.exp(-(t-t[i])**2/nn);
X = X@(S**.5)
X = X/np.linalg.norm(X,axis=0)

ind = np.random.choice(p, k, replace=False) # generating optimal weights
weights = np.random.randn(k)
weights += 0.1+np.sign(weights)             # to get large enough weight
wopt = np.zeros(p)
wopt[ind] = weights

rsnr = 2                                    # generating output by X@w + noise
z = X[:,ind]@weights
stdnoise = np.std(z)/rsnr
y = z + stdnoise*np.random.randn(n)
```

   b) Vérifiez que les données ont bien les propriétés attendues.
   c) Calculez l'erreur de généralisation "in sample" de la méthode des moindres carrés,

```
b_ls = np.linalg.solve(X.T@X,X.T@y)
e_ls = np.sum((X@b_ls-z)**2)
print("Test error for the LS regression: {:0.4f}".format(e_ls))
```

[1] `http://statweb.stanford.edu/~tibs/lasso.html`

d) Écrire une fonction Eval_coef(X,z,coeff), qui calcule l'erreur de généralisation "in sample"

2. Différentes manières de résoudre le problème du Lasso

a) Écrire un programme CVX résolvant, pour $\lambda = 10^{-3}n$.

$$\min_{\beta \in \mathbf{R}^p} J_\lambda(\beta) \quad \text{avec} \quad J_\lambda(\beta) = \tfrac{1}{2}\|X\beta - y\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

b) Écrire un programme CVX résolvant la formulation suivant de Lasso, avec une valeur de t permettant d'obtenir les mêmes résultats que le problème précédent.

$$\left\{ \begin{array}{ll} \min\limits_{\beta \in \mathbf{R}^p} & \tfrac{1}{2}\|X\beta - y\|^2 \\ \text{avec} & \sum\limits_{j=1}^{p} |\beta_j| \leq t \end{array} \right. \tag{1}$$

c) Résoudre le problème du Lasso (1) en réécrivant le cout comme une fonctionnelle quadratique de la forme

$$\tfrac{1}{2}\|X\beta - y\|^2 = \tfrac{1}{2}\beta^\top D\beta + \beta^\top e$$

où la matrice D et le vecteur e sont à préciser

d) réécrire le Lasso comme un programme quadratique sous sa forme standard.

$$\left\{ \begin{array}{ll} \min\limits_{x \in \mathbf{R}^n} & \tfrac{1}{2}\mathrm{x}^\top H\mathrm{x} + \mathrm{x}^\top c \\ \text{avec} & A\mathrm{x} \leq b \end{array} \right. \tag{2}$$

e) Proposez un code CVX permettant de résoudre le Lasso réécrit comme un QP standard.

f) Comment résoudre ce même QP de manière plus efficace ?

g) Résoudre, à l'aide de CVX, le problème dual du Lasso

$$\left\{ \begin{array}{ll} \min\limits_{\beta} & \tfrac{1}{2}\|X\beta\|^2 \\ s.t. & \|X^\top(X\beta - y)\|_\infty \leq \lambda \end{array} \right.$$

h) vérifiez que toutes les méthodes donnent le même résultat et comparez les temps de calcul.

i) Quelle est la plus rapide des méthodes ? A quoi sert CVX ?