

TP 4 : Le lasso (suite)

Stéphane Canu

16 septembre 2024, MLA, INSA Rouen

Le but du TP est d'étudier une méthode de sélection de variables, le Lasso¹, 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 entraînant 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: {:.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 $\lambda = 10^{-3}n$.

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

- b) Vérifiez que la solution obtenue est bien optimale. Pour ce faire on vérifiera que $0 \in \partial J_\lambda(\beta)$
- c) Résoudre, à l'aide de CVX, le problème dual du Lasso

$$\begin{cases} \min_{\beta} & \frac{1}{2} \|X\beta\|^2 \\ \text{s.t.} & \|X^\top(X\beta - y)\|_\infty \leq \lambda \end{cases}$$

- d) Résoudre, à l'aide de Sklearn, le problème du Lasso
- e) Résoudre, à l'aide d'une IA générative (et sans utiliser sklearn), le problème du Lasso. Précisez le prompt que vous avez utilisé.
- f) vérifiez que toutes les méthodes (celles de cette semaine et celles de la semaine dernière) donnent le même résultat et comparez les temps de calcul.
- g) Quelle est la plus rapide des méthodes? A quoi sert CVX?
- h) A la vu de ces résultats, quelles serait votre recommandation pour résoudre un problème de lasso?