Optimización Numérica: Laboratorio 4

rodrigo.mendoza@itam.mx

1 Octubre 2019

1 Programación Lineal

Supongamos que tenemos un conjunto de datos $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R} : i \in [m]\}$ y que $p \gg m$. Queremos ajustar un modelo lineal de la forma

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{y} + \boldsymbol{\varepsilon}. \tag{1}$$

En este caso, incluso cuando $\boldsymbol{\varepsilon} = 0$, hay un número infinito de soluciones. Queremos encontrar la solución *más simple* que se ajuste a los datos. Medimos simplicidad por medio del operador $\|\cdot\|_0 : \mathbb{R}^p \to \mathbb{N}$ dado por $\|\boldsymbol{\beta}\|_0 = |\{j \in [p] : \boldsymbol{\beta}_j \neq 0\}|$.

1. Demuestre que $\|\boldsymbol{\beta}\|_q \to \|\boldsymbol{\beta}\|_0$ si $q \to 0+$.

2. Comente si $\|\cdot\|_0$ en verdad define una norma.

3. Supongamos que $\boldsymbol{\varepsilon} = 0$. En este caso, la solución *más simple* a (1) puede formularse como:

$$\arg\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_0 \text{ sujeto a } \mathbf{X}\boldsymbol{\beta} = \mathbf{y}. \tag{2}$$

   Si quisiéramos encontrar la solución de (2) por *fuerza bruta*. ¿Cuántos patrones de soporte $\{j \in [p] : \beta_j \neq 0\}$ de $\boldsymbol{\beta}$ que tendríamos que checar en el peor de los casos? ¿Cómo se compara este número con el número de átomos en el universo observable para $p = 100$?

4. El *problema de cubiertas exactas de 3-conjuntos* es: Dada una colección $\{S_j : S_j \subset [m], |S_j| = 3, j \in [p]\}$, ¿Existe un subconjunto $J \subset [p]$ tal que $\cup_{j \in J} S_j = [m]$ y $S_{j_1} \cap S_{j_2} = \emptyset$ para $j_1 \neq j_2$?.

   Está demostrado que el *problema de cubiertas exactas de 3-conjuntos* es NP-duro. Úselo para demostrar que (2) es NP-duro[1].

5. Considere los siguiente problemas:

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_1 \quad \text{s.t.} \quad \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2 \leq \eta \tag{QCBP}$$

$$\min_{\boldsymbol{\beta}} \lambda\|\boldsymbol{\beta}\|_1 + \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 \quad (\lambda \geq 0) \tag{BPD}$$

$$\min_{\boldsymbol{\beta}} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_1 \leq \tau \quad (\tau \geq 0) \tag{LASSO}$$

   Tenemos que:

   - Si $\boldsymbol{\beta}^*$ es un minimizador de (BPD) con $\lambda > 0$, existe $\eta \geq 0$ tal que $\boldsymbol{\beta}^*$ minimiza (QCBP).
   - Si $\boldsymbol{\beta}^*$ es un minimizador único de (QCBP) con $\eta \geq 0$, existe $\tau \geq 0$ tal que $\boldsymbol{\beta}^*$ es un minimizador único de (LASSO).
   - Si $\boldsymbol{\beta}^*$ es un minimizador de (LASSO) con $\tau > 0$, existe $\lambda \geq 0$ tal que $\boldsymbol{\beta}^*$ es un minimizador de (BPD).

   Demuestre los primeros dos incisos.

6. ¿Bajo qué condiciones (QCBP) es equivalente a (LASSO)?

[1] *Pista:* Demuestre que puede resolver el problema de cubiertas de 3-conjuntos con un problema de la forma (2) y que este problema puede construirse en tiempo polinomial

7. En algunos casos, el problema (2) puede resolverse por medio (QCBP) con $\eta = 0$. Demuestre que en este caso (QCBP) puede formularse como un problema de programación lineal introduciendo variables auxiliares $\mathbf{z}^+, \mathbf{z}^- \in \mathbb{R}^p$ definidas por:

$$z_j^+ = \begin{cases} z_j & z_j > 0 \\ 0 & z_j \leq 0 \end{cases}, \quad z_j^- = \begin{cases} 0 & z_j > 0 \\ -z_j & z_j \leq 0 \end{cases} \tag{3}$$

¿Cómo podría recuperar la solución original a partir de estas variables?

8. Implemente la función `bp` en Python que resuelva (QCBP) con $\eta = 0$ usando `cvxopt.solvers.cone.lp` (http://cvxopt.org/userguide/coneprog.html?highlight=lp#linear-cone-programs)

```
def bp(X, y):
    '''(X,y):  Matriz de datos y vector de respuesta [X.shape=(m,p), y.shape=(m,)]
       (betahat): Soluci\'on [betahat.shape=(1,)]
    '''
    ...
    return betahat
```

9. Implmente la siguiente función de datos en Python.

```
import numpy as np
import random
def datos(m,p,k):
    np.random.seed(1111)
    random.seed(1111)
    X = np.random.normal(0, 1, (m,p))
    beta = np.random.normal(0, 1, (p,1))
    beta[random.sample(range(0,p), k=p-k)] = 0
    y = np.matmul(X, beta)
    return X, y.squeeze(), beta.squeeze()
```

10. Genere `(X,y,beta) = datos(75,150,5)`. Use la función `bp` para estimar una solución `betahat` a (2). Compare el soporte de `betahat` con el soporte de `beta`.