TP 9 : MIP

MLA, Stéphane Canu

17 novembre 2025, ITI, INSA Rouen

Le but du TP est d'étudier l'usage de la programmation entière mixte *mixed integer programming* (MIP), dans le cadre de la régression. Vous êtes supposé avoir déjà installé CVX.

Ex. 1 — Régression linéaire : les moindres déviations absolues (*least absolute deviations*)

1. Écrire une fonction python $(\widehat{\beta}, \widehat{\beta}_0) = \texttt{Estime\_beta}(X, y)$ permettant d'estimer la valeur des paramètres d'un modèles linéaire à partir d'un échantillon de n observations y et d'une matrice $x \in \mathbb{R}^{n \times p}$ en minimisant

$$\sum_{i=1}^{n} |X_i^\top \widehat{\beta} + \widehat{\beta}_0 - y_i|$$

   Estimez les paramètres du modèle à partir des 13 observations suivantes.

```
import numpy as np
X = np.array([ [-1.6000, -0.8095, -0.1649, 1.1174],
[ 0.4600, -2.9443, 0.6277, -1.0891],
[ 1.3549, 1.4384, 1.0933, 0.0326],
[ 0.5519, 0.3252, 1.1093, 0.5525],
[ 0.8152, -0.7549, -0.8637, 1.1006],
[-1.4367, 1.3703, 0.0774, 1.5442],
[ 0.8371, -1.7115, -1.2141, 0.0859],
[ 0.0246, -0.1022, -1.1135, -1.4916],
[-1.5218, -0.2414, -0.0068, -0.7423],
[-0.9986, 0.3192, 1.5326, -1.0616],
[-0.2226, 0.3129, -0.7697, 2.3505],
[ 0.9647, -0.8649, 0.3714, -0.6156],
[ 0.9861, -0.0301, -0.2256, 0.7481] ])
y = np.array([ 2.1250, -0.8500, -1.0410, 1.1469, 2.7257, -0.2190,1.8182, 1.2390,
    -0.5833, -0.0358, 1.2243, 2.3874, 2.4112 ])
```

2. Donnez une estimation de espérance de la valeur absolue des erreurs

$$\mathbb{E}\Big(|X^\top \widehat{\beta} + \widehat{\beta}_0 - Y|\Big)$$

   où $X \in \mathbb{R}^4$, $Y \in \mathbb{R}$ deux variables aléatoires et $(\widehat{\beta}, \widehat{\beta}_0)$ les estimations des paramètres β et β_0.

3. a) Écrire une fonction python $y_p = \texttt{predlin}(x, \widehat{\beta}, \widehat{\beta}_0)$ permettant de prédire la valeur de la variable Y connaissant $x \in \mathbb{R}^4$ à l'aide d'un modèle linéaire $Y = X^\top \beta + \varepsilon$. Donnez une prédiction pour la valeur de la variable y à partir des 27 observations suivantes.

```
Xt =np.array([ [ -1.2504 0.1825 -0.2730 -0.7519
   -1.0300 -1.5651 1.5763 1.5163
   -0.8261 -0.0845 -0.4809 -0.0326
   -0.1811 1.6039 0.3275 1.6360
   -1.5184 0.0983 0.6647 -0.4251
    1.5848 0.0414 0.0852 0.5894
    1.7378 -0.7342 0.8810 -0.0628
    0.0866 -0.0308 0.3232 -2.0220
    0.0808 0.2323 -0.7841 -0.9821
   -0.4705 0.4264 -1.8054 0.6125
```

```
 1.5751 -0.3728 1.8586 -0.0549
-0.3558 -0.2365 -0.6045 -1.1187
-1.2945 2.0237 0.1034 -0.6264
 1.1393 -2.2584 0.5632 0.2495
-0.2812 2.2294 0.1136 -0.9930
-0.8198 0.3376 -0.9047 0.9750
-0.2297 1.0001 -0.4677 -0.6407
-1.3481 -1.6642 -0.1249 1.8089
-1.2189 -0.5900 1.4790 -1.0799
 1.7279 -0.2781 -0.8608 0.1992
 1.7791 0.4227 0.7847 -1.5210
 0.3934 -1.6702 0.3086 -0.7236
-1.4815 0.4716 -0.2339 -0.5933
-0.8449 -1.2128 -1.0570 0.4013
-0.4143 0.0662 -0.2841 0.9421
 1.2882 0.6524 -0.0867 0.3005
 0.2908 0.9248 0.3501 0.7596] ])
```

b) Calculez la moyenne des carrés des erreurs

$$\frac{1}{27}\sum_{i=1}^{27} |Xt_i^\top \widehat{\beta} + \widehat{\beta}_0 - yt_i|$$

avec

```
yt = np.array([-0.2504, -0.03 , 0.1739, 0.8189, -0.5184, 2.5848, 2.7378, 1.0866,
     1.0808, 0.5295, 2.5751, 0.6442, -0.2945, 2.1393, 0.7188, 0.1802, 0.7703,
    -0.3481, -0.2189, 2.7279, 2.7791, 1.3934, -0.4815, 0.1551, 0.5857, 2.2882,
    1.2908])
```

Ex. 2 — Sélection de variables et détection d'*outliers*

1. En sachant que k_v variables sont pertinentes et que k_o observation sont aberrantes, proposez une méthode permettant de prendre en compte ces informations pour améliorer vos estimations.
2. La modalisation : soit $X = (\mathrm{x}_1, \ldots, \mathrm{x}_n)^t$ une matrice $n \times p$ et $\mathrm{y} \in \mathbb{R}^n$ un vecteur de réponses. Soit $\tau \in \mathbb{R}^n$ une nouvelle variable modélisant les observations douteuses de la manière suivante :

$$\forall i \in \{1, \ldots, n\}, \qquad \tau_i = \begin{cases} y_i - \mathrm{x}_i^t\beta - \varepsilon_i & \text{si l'observation } i \text{ est un point aberrant} \\ 0 & \text{si l'observation } i \text{ est conforme.} \end{cases}$$

Nous considérons le modèle suivant

$$\mathrm{y} = X\beta + \varepsilon + \tau,$$

où $\beta \in \mathbb{R}^p$ est le vecteur des paramètres inconnus à estimer, et $\varepsilon \in \mathbb{R}^n$ un vecteur de bruit.
Étant donnés k_v le nombre de variables nécessaires et k_o le nombre d'*outliers*, nous cherchons à résoudre le problème d'optimisation suivant :

$$\begin{cases} \min\limits_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n} & \|X\beta + \tau - \mathrm{y}\|_1 \\ \text{s.t.} & \|\beta\|_0 \leq k_v \\ & \|\tau\|_0 \leq k_o, \end{cases} \tag{1}$$

a) Montrez que, pour des constantes M_v et M_o assez grandes, ce problème est équivalent à

$$\begin{cases} \min\limits_{\beta\in\mathbb{R}^p,\tau\in\mathbb{R}^n} & \|X\beta+\tau-\mathrm{y}\|_1 \\ & \sum\limits_{i=1}^{n}(1-t_i)\le k_o \\ & \sum\limits_{j=1}^{p} z_j \le k_v \\ & -z_jM_v \le \beta_j \le z_jM_v \qquad j=1,p \\ & -t_iM_o \le \tau_i \le t_iM_o \qquad i=1,n. \end{cases} \tag{2}$$

b) Écrire une fonction $(\widehat{\beta},\widehat{\beta}_0)$ =`ma_regression_robuste`(X,y,k_v,k_o) pour mettre ne ouvre le code ci-dessus. Proposez une manière de choisir k_v et k_o.

Ex. 3 — Et chatGPT dans tout ca ?

1. Voici la réponse de ChatGPT à la première question : qu'en pensez-vous ?
On reformule le problème en programmation linéaire en introduisant des variables auxiliaires $(u_i \ge 0)$ telles que

$$(u_i \ge |X_i^\top\beta+\beta_0-y_i|).$$

Cela donne :

$$\min_{\beta,\beta_0,u}\sum_{i=1}^{n} u_i$$

sous contraintes, pour tout (i) :

$$X_i^\top\beta+\beta_0-y_i\le u_i \quad \text{et} \quad -X_i^\top\beta-\beta_0+y_i\le u_i$$

On code avec `scipy.optimize.linprog`

```
import numpy as np
from scipy.optimize import linprog

def Estime_beta(X, y):
   X = np.asarray(X)
   y = np.asarray(y)
   n, p = X.shape

# Variables: [beta (p), beta0 (1), u (n)]
   c = np.hstack([np.zeros(p + 1), np.ones(n)]) # minimise sum(u_i)

   A_ub = []
   b_ub = []

   for i in range(n):
      xi = X[i]
      yi = y[i]

# 1) X_i^T beta + beta0 - y_i <= u_i
# <=> X_i^T beta + beta0 - u_i <= y_i (on met tout ôéct gauche sauf y_i)
      row1 = np.zeros(p + 1 + n)
      row1[:p] = xi
      row1[p] = 1.0 # beta0
      row1[p + 1 + i] = -1.0 # -u_i
      A_ub.append(row1)
      b_ub.append(yi)
```

```
# 2) -X_i^T beta - beta0 + y_i <= u_i
# <=> -X_i^T beta - beta0 - u_i <= -y_i
      row2 = np.zeros(p + 1 + n)
      row2[:p] = -xi
      row2[p] = -1.0 # -beta0
      row2[p + 1 + i] = -1.0 # -u_i
      A_ub.append(row2)
      b_ub.append(-yi)

   A_ub = np.array(A_ub)
   b_ub = np.array(b_ub)

# bornes : beta, beta0 libres ; u_i >= 0
   bounds = [(-np.inf, np.inf)] * (p + 1) + [(0, np.inf)] * n

   res = linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=bounds, method="highs")
   if not res.success:
      raise RuntimeError("Le solveur n'a pas éconverg : " + res.message)

   beta_hat = res.x[:p]
   beta0_hat = res.x[p]

   return beta_hat, beta0_hat
```

Application aux 13 observations données
En utilisant tes données :

```
beta_hat, beta0_hat = Estime_beta(X, y)
print(beta_hat)
print(beta0_hat)
```

On obtient numériquement (arrondis à 3 décimales) :

$$\hat{\beta} \approx (0.337,\ -0.065,\ -0.123,\ 0.038) \quad \text{et} \quad \hat{\beta}_0 \approx 0.524$$

Donc le modèle linéaire L1 estimé est :

$$\hat{y} \approx 0{,}337, x_1 - 0{,}065, x_2 - 0{,}123, x_3 + 0{,}038, x_4 + 0{,}524.$$