

1 Implementácia a grafické znázornenie

1.1 Prevedenie úlohy LP do tvaru pre `scipy.optimize.linprog`

Metóda `linprog` z modulu `scipy.optimize` vyžaduje nasledujúci tvar úlohy LP:

$$\begin{aligned} \min \quad & c^T x \\ & A_{ub} x \leq b_{ub} \\ & A_{eq} x = b_{eq} \\ & x \in [l, u] \end{aligned} \quad l \leq u; \quad l, u \in (\mathbb{R} \cup \{-\infty, \infty\})^n$$

Hodnotami $-\infty$ a ∞ značíme neohraničenosť v danom smere, v zdrojovom kóde sa obe nahrádzajú hodnotou `None`. Upravme teda úlohy vyjadrené vyššie do predpísaného tvaru.

Pre L^1 lineárnu regresiu:

$$\begin{aligned} \min \quad & \left(\begin{array}{c|c} \mathbf{0}_{k+1}^T & \mathbf{1}_n^T \end{array} \right) \left(\frac{\beta}{t} \right) \\ & \left(\begin{array}{c|c} -\mathbf{A} & -\mathbb{I}_n \\ \hline \mathbf{A} & -\mathbb{I}_n \end{array} \right) \left(\frac{\beta}{t} \right) \leq \left(\frac{-y}{y} \right) \\ & \beta_i \in (-\infty, \infty) \\ & t_j \in [0, \infty) \end{aligned} \quad \begin{aligned} i &= 0, 1, \dots, k \\ j &= 1, \dots, n \end{aligned}$$

Pre L^∞ regresiu:

$$\begin{aligned} \min \quad & \left(\begin{array}{c|c} \mathbf{0}_{k+1}^T & \mathbf{1} \end{array} \right) \left(\frac{\beta}{\gamma} \right) \\ & \left(\begin{array}{c|c} -\mathbf{A} & -\mathbf{1}_n \\ \hline \mathbf{A} & -\mathbf{1}_n \end{array} \right) \left(\frac{\beta}{\gamma} \right) \leq \left(\frac{-y}{y} \right) \\ & \beta_i \in (-\infty, \infty) \\ & \gamma \in [0, \infty) \end{aligned} \quad i = 0, 1, \dots, k$$

Úlohy v zdrojovom kóde sú implementované práve v tomto tvare.

1.2 Implementovanie regresných LP úloh

Na implementáciu formulovaných LP úloh využívame tri knižnice:

- `numpy` - tvorenie matíc a vektorov, načítanie dát
- `scipy.optimize` - implementovaný LP solver
- `matplotlib.pyplot` na vykresľovanie grafov.

Dáta relevantné pre túto úlohu sú uložené v súbore `data/A04plotregres.npz`. Jedná sa o 16 bodov v \mathbb{R}^2 , kde prvá súradnica reprezentuje nezávislú premennú (vektor týchto súradníc označíme x) a druhá závislú premennú (označíme y).

Vytvorme si potrebné štruktúry pre využitie metódy `scipy.optimize.linprog` pre LP formuláciu s L^1 normou:

```
c = np.array([0,0] + [1] * len(x)) #objective function vector, two zeros
                                   #stand for betas
A = np.matrix([[1] * len(x), x]).transpose()
I = np.identity(len(x)) # Identity matrix

A_ub = np.block([[-A,-I], [A,-I]]) #creating a block matrix
b_ub = np.concatenate([-y, y]) #right side vector
bounds = [(None, None), (None, None)] + [(0, None) for _ in range(len(x))
                                           ] # bounds for variables
```

Pomocou solvera získame vektor optimálnych β koeficientov:

$$\beta_0^{(1)} \approx -9.8378, \beta_1^{(1)} \approx 2.1297$$

Podobne implementujeme L^∞ formuláciu:

```
c_inf = np.array([0,0,1]) #objective function vector
A_inf = np.matrix([[1] * len(x), x]).transpose()
i_inf = np.array([[1] * len(x)]).transpose() # vector of ones

A_ub_inf = np.block([[-A_inf, -i_inf], [A_inf, -i_inf]]) # creating a
                                                         #block matrix
b_ub_inf = np.concatenate([-y, y]) #right side vector
bounds = [(None, None), (None, None), (0, None)]
```

Znovu, pomocou solvera získame vektor optimálnych β koeficientov:

$$\beta_0^{(\infty)} \approx 15.4545, \beta_1^{(\infty)} \approx 1.7045$$

Pomocou získaných β koeficientov vykreslíme regresné priamky spolu s pôvodnými dátami.

