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:

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

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:

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

Pre L^∞ lineárnu regresiu:

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

Úlohy v zdrojovom kóde sú implementované práve v tomto tvare.

1.2 Implementovanie LP úloh

Na implementáciu formulovaných LP úloh využívame tri knižnice:

- `numpy` - tvorenie matíc a vektorov, načítanie dát
- `scipy.optimize` - implementovaný LP solver
- `matplotlib.pyplot` - 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.concatenate(([0, 0], np.ones(len(x))))    # objective function
                                                 vector, zeros stand for betas
A = np.block([np.ones((len(x),1)), x[:,np.newaxis]]) # creating matrix A
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.block([np.ones((len(x), 1)), x[:, np.newaxis]])
i_inf = np.ones((len(x), 1))     # 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.