Úloha B

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

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

$$\begin{aligned} \min\ & c^T x \\ & A_{ub} x \leq b_{ub} \\ & A_{eq} x \leq b_{eq} \\ & l \leq x \leq u \qquad\qquad l, u \in (\mathbb{R} \cup \{\texttt{None}\})^n \end{aligned}$$

Hodnota `None` vo vektoroch $l,\ u$ značí neohraničenosť v danom smere. Upravme teda úlohy vyjadrené vyššie do predpísaného tvaru.

Pre L^1 regresiu:

$$\begin{aligned} \min\ & \left(\mathbf{0}_{k+1}^T \mid \mathbf{1}_n^T \right) \begin{pmatrix} \beta \\ \hline t \end{pmatrix} \\ & \left(\begin{array}{c|c} -\mathbf{A} & -\mathbb{I}_n \\ \hline \mathbf{A} & -\mathbb{I}_n \end{array}\right) \begin{pmatrix} \beta \\ \hline t \end{pmatrix} \leq \begin{pmatrix} -y \\ \hline y \end{pmatrix} \\ & \texttt{None} \leq \beta_i \leq \texttt{None} && i = 0, 1, \ldots, k \\ & 0 \leq t_j \leq \texttt{None} && j = 1, \ldots, n \end{aligned}$$

Pre L^∞ regresiu:

$$\begin{aligned} \min\ & \left(\mathbf{0}_{k+1}^T \mid 1 \right) \begin{pmatrix} \beta \\ \hline \gamma \end{pmatrix} \\ & \left(\begin{array}{c|c} -\mathbf{A} & -\mathbf{1}_n \\ \hline \mathbf{A} & -\mathbf{1}_n \end{array}\right) \begin{pmatrix} \beta \\ \hline \gamma \end{pmatrix} \leq \begin{pmatrix} -y \\ \hline y \end{pmatrix} \\ & \texttt{None} \leq \beta_i \leq \texttt{None} && i = 0, 1, \ldots, k \\ & 0 \leq \gamma \leq \texttt{None} \end{aligned}$$

Úlohy v zdrojovom kóde sú implementované práve v tomto tvare.

Implementovanie regresných LP úloh

Naimportovali sme knižnice; `numpy` na tvorenie matíc a vektorov, `scipy.optimize` na riešenie danej LP úlohy a `matplotlib` na vykresľovanie grafov.

```
import numpy as np
from scipy.optimize import linprog
import pandas as pd
import matplotlib.pyplot as plt
```

Načítali dáta a rozdelili ich do premenných.

```
data = np.load('data/A04plotregres.npz')
x = data['x']
y = data['y']
```

Začali sme s L^1 $(Manhattan)$ normou

```
# l1 norm
c = np.array([0,0] + [1] * len(x))
A = np.matrix([[1] * len(x), x]).transpose()
I = np.identity(len(x))
A_ub = np.block([[-A,-I], [A,-I]])
b_ub = np.concatenate([-y, y])
solve = linprog(c, A_ub, b_ub)
beta0 = solve.x[0]
beta1 = solve.x[1]
values = [i*beta1 + beta0 for i in x]
```

```
fig, ax = plt.subplots()
ax.plot(x, values, label = 'Manhattan')
ax.set_xlim([0, max(x) + 5])
ax.set_ylim([0, max(y) + 5])
```

Potom sme spočítali L^{∞} $(maximum)$ normu, a nasledovne ju tiež vykreslili do grafu.

```
# l inf norm
c_inf = np.array([0,0,1])
A_inf = np.matrix([[1] * len(x), x]).transpose()
i_inf = np.array([[1] * len(x)]).transpose()
A_ub_inf = np.block([[-A_inf, -i_inf], [A_inf, -i_inf]])
b_ub_inf = np.concatenate([-y, y])
solve_inf = linprog(c_inf, A_ub_inf, b_ub_inf)
beta0_inf = solve_inf.x[0]
beta1_inf = solve_inf.x[1]
values_inf = [i * beta1_inf + beta0_inf for i in x]
```

```
ax.plot(x, values_inf, label = 'infinity')
ax.plot(x,y, 'o')
ax.legend()
plt.show()
```