

**Fakulta matematiky, fyziky a informatiky Univerzity Komenského,
Bratislava**

**Projekt z lineárneho programovania
A04 - Predikcia kvality vín**

Piati proti optimalizácii

Tomáš Antal, 2DAV, 0.2

Erik Božík, 2DAV, 0.2

Róbert Kendereš, 2DAV, 0.2

Teo Pazera, 2DAV, 0.2

Andrej Špitalský, 2DAV, 0.2

Obsah

1 Úloha A	1
1.1 Minimalizovanie L^1 normy	1
1.1.1 Prípustnosť a optimalita	3
1.2 Minimalizovanie L^∞ normy	3
1.2.1 Prípustnosť a optimalita	4

1 Úloha A

Máme dané vektory y, x_1, x_2, \dots, x_k . Chceme nájsť parametre $\beta_0, \beta_1, \dots, \beta_k$ také, aby pre vektor $\hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$, boli normy $\|y - \hat{y}\|_1$ a $\|y - \hat{y}\|_\infty$ minimálne.

Vyjadriť vektor \hat{y} ako súčin matice a vektora $\beta = (\beta_0, \beta_1, \dots, \beta_k)^T$.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k = \begin{pmatrix} | & | & | & \dots & | \\ \mathbf{1}_n & x_1 & x_2 & \dots & x_k \\ | & | & | & \dots & | \end{pmatrix} \beta =: \mathbf{A}\beta$$

1.1 Minimalizovanie L^1 normy

Prevedieme problém zo zadania do tvaru:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \end{aligned}$$

Zavedme si nový vektor premenných $t \in \mathbb{R}^n$, ktorým ohraničíme normu $\|y - \mathbf{A}\beta\|_1$.

$$-t \leq y - \mathbf{A}\beta \leq t$$

Pre obe ohraničenia, odseparujme premenné od konštánt a prevedme do maticového tvaru.

$$\begin{aligned} (\mathbf{A} \mid \mathbb{I}_n) \begin{pmatrix} \beta \\ t \end{pmatrix} &\geq y \\ (-\mathbf{A} \mid \mathbb{I}_n) \begin{pmatrix} \beta \\ t \end{pmatrix} &\geq -y \end{aligned}$$

Minimalizovanie L^1 normy ako úloha lineárneho programovania vyzerá teda nasledovne.

$$\begin{aligned} \min \quad & (\mathbf{0}_{k+1}^T \mid \mathbf{1}_n^T) \begin{pmatrix} \beta \\ t \end{pmatrix} \\ \text{s.t.} \quad & \begin{pmatrix} \mathbf{A} \mid \mathbb{I}_n \\ -\mathbf{A} \mid \mathbb{I}_n \end{pmatrix} \begin{pmatrix} \beta \\ t \end{pmatrix} \geq \begin{pmatrix} y \\ -y \end{pmatrix} \\ & \beta \in \mathbb{R}^{k+1}, t \geq \mathbf{0}_n \end{aligned} \tag{1}$$

1.1.1 Prípustnosť a optimalita

Dokážme, že (1) je úloha, ktorá nadobúda optimálne riešenie pre ľubovoľné vektory y, x_1, x_2, \dots, x_k . Nech $|y| := (|y_1|, |y_2|, \dots, |y_n|)^T$ pre $y = (y_1, y_2, \dots, y_n)^T$. Ukážme prípustnosť zvolením $\beta = \mathbf{0}_{k+1}$ a $t = |y|$:

$$\left(\begin{array}{c|c} \mathbf{A} & \mathbb{I}_n \\ \hline -\mathbf{A} & \mathbb{I}_n \end{array} \right) \left(\begin{array}{c} \mathbf{0}_{k+1} \\ |y| \end{array} \right) = \left(\begin{array}{c} |y| \\ |y| \end{array} \right) \geq \left(\begin{array}{c} y \\ -y \end{array} \right)$$

$$|y| \geq \mathbf{0}_n$$

Vidíme, že obe ohraňovania platia, čiže $(\mathbf{0}_{k+1}^T, |y|^T)^T$ je prípustné riešenie.

Optimalitu ukážeme zo silnej duality. Sformulujme duálnu úlohu pre duálne premenné $\alpha_1, \alpha_2 \in \mathbb{R}^n$:

$$\begin{aligned} \max \quad & (y^T \mid -y^T) \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \end{array} \right) \\ & (\mathbf{A}^T \mid -\mathbf{A}^T) \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \end{array} \right) = \mathbf{0}_{k+1} \\ & (\mathbb{I}_n \mid \mathbb{I}_n) \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \end{array} \right) \leq \mathbf{1}_n \\ & \alpha_1, \alpha_2 \geq \mathbf{0}_n \end{aligned}$$

Vidíme, že táto úloha je prípustná pre $\alpha_1 = \alpha_2 = \mathbf{0}_n$. Z prípustnosti primárnej a duálnej úlohy teda vyplýva, že úloha (1) nadobúda optimálne riešenie pre ľubovoľnú voľbu počiatočných vektorov.

1.2 Minimalizovanie L^∞ normy

Budeme používať podobné značenie ako pri formulácii L^1 normy. Zavedme si skalár $\gamma \in \mathbb{R}$, ktorým ohraňujeme normu $\|y - \mathbf{A}\beta\|_\infty$.

$$-\gamma \mathbf{1}_n \leq y - \mathbf{A}\beta \leq \gamma \mathbf{1}_n$$

Pre jednotlivé ohraňovania odseparujeme premenné od konštánt a zapíšeme v maticovom tvare.

$$\begin{aligned} (\mathbf{A} \mid \mathbf{1}_n) \left(\begin{array}{c} \beta \\ \gamma \end{array} \right) &\geq y \\ (-\mathbf{A} \mid \mathbf{1}_n) \left(\begin{array}{c} \beta \\ \gamma \end{array} \right) &\geq -y \end{aligned}$$

Minimalizovanie L^∞ normy ako úloha lineárneho programovania vyzerá teda nasledovne.

$$\begin{aligned} \min \quad & \left(\begin{array}{c|c} \mathbf{0}_{k+1}^T & 1 \end{array} \right) \left(\begin{array}{c} \beta \\ \gamma \end{array} \right) \\ \left(\begin{array}{c|c} \mathbf{A} & \mathbf{1}_n \\ -\mathbf{A} & \mathbf{1}_n \end{array} \right) \left(\begin{array}{c} \beta \\ \gamma \end{array} \right) \geq \left(\begin{array}{c} y \\ -y \end{array} \right) \\ \beta \in \mathbb{R}^{k+1}, \gamma \geq 0 \end{aligned} \quad (2)$$

1.2.1 Prípustnosť a optimalita

Podobný spôsobom ako vyššie ukážeme optimalitu (2). Nech $\beta = \mathbf{0}_{k+1}$ a $\gamma = |\tilde{y}|$, kde $|\tilde{y}| := |\max(y_1, y_2, \dots, y_n)|$ pre $y = (y_1, y_2, \dots, y_n)^T$:

$$\begin{aligned} \left(\begin{array}{c|c} \mathbf{A} & \mathbf{1}_n \\ -\mathbf{A} & \mathbf{1}_n \end{array} \right) \left(\begin{array}{c} \mathbf{0}_{k+1} \\ |\tilde{y}| \end{array} \right) &= \left(\begin{array}{c} |\tilde{y}| \mathbf{1}_n \\ |\tilde{y}| \mathbf{1}_n \end{array} \right) \geq \left(\begin{array}{c} y \\ -y \end{array} \right) \\ |\tilde{y}| &\geq 0 \end{aligned}$$

Obe ohraničenia platia, čiže $(\mathbf{0}_{k+1}^T, |\tilde{y}|)^T$ je prípustné riešenie. Sformulujme duálnu úlohu s duálnymi premennými $\alpha_1, \alpha_2 \in \mathbb{R}^n$:

$$\begin{aligned} \max \quad & \left(\begin{array}{c|c} y^T & -y^T \end{array} \right) \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \end{array} \right) \\ \left(\begin{array}{c|c} \mathbf{A}^T & -\mathbf{A}^T \end{array} \right) \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \end{array} \right) &= \mathbf{0}_{k+1} \\ \left(\begin{array}{c|c} \mathbf{1}_n^T & \mathbf{1}_n^T \end{array} \right) \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \end{array} \right) &\leq 1 \\ \alpha_1, \alpha_2 &\geq \mathbf{0}_n \end{aligned}$$

Rovnako vidíme, že táto úloha je prípustná pre $\alpha_1 = \alpha_2 = \mathbf{0}_n$. Teda, zo silnej duality, úloha (2) nadobúda optimálne riešenie pre ľubovoľnú voľbu počiatkových vektorov.

2 Úloha B

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

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

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

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

Pre L^1 regresiu:

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

Pre L^∞ regresiu:

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

Úlohy v zdrojovom kóde sú implementované práve v tomto tvare.

2.0.2 Implementovanie regresných LP úloh

Na implementáciu formulovaných LP úloh využívame tri knžnice:

- `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:

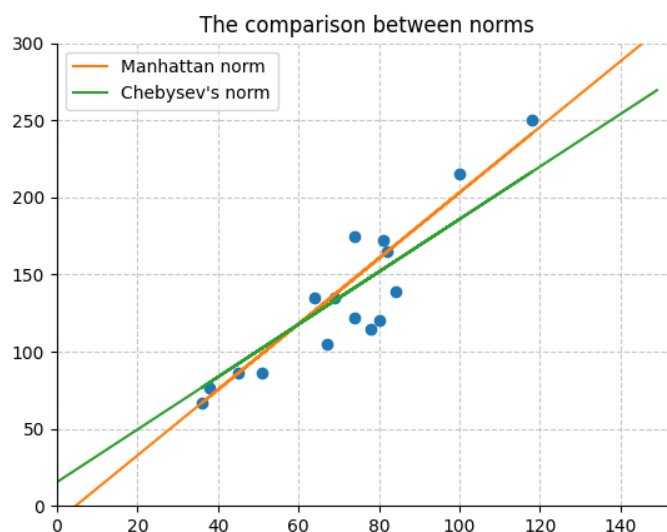
```
c_inf = np.array([0,0,1]) #objective function vector
A_inf = np.matrix([[1] * len(x), x]).transpose()
i_inf = np.array([[1] * len(x)]).transpose() # vector of ones

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.



3 Úloha C

V tejto úlohe sa snažíme predikovať kvalitu vína, inšpirovaní prístupom Orleya Ashenfeltera k predikcii cien vína z Bordeaux.

Využívame dáta zo súboru `A04wine.csv` a aplikujeme modely L^1 a L^∞ z úlohy A. Budeme využívať podobný postup ako v úlohe B. Na implementáciu formulovaných LP úloh využívame opäť:

- `pandas` - načítanie dát z `csv` súboru
- `numpy` - tvorenie matíc a vektorov
- `scipy.optimize` - implementovaný LP solver

Vyberieme z dát dané nezávislé premenné x a závislú premennú y

```
y = data['Price']
x = data[['WinterRain', 'AGST', 'HarvestRain', 'Age', 'FrancePop']]
```

Zistíme počet premenných (plus 1 pre intercept)

```
numberOfVariablesBeta = x.shape[1] + 1
```

Vytvoríme potrebné štruktúry pre zostavenie modelu normy l_1

```
c = np.array([0]*numberOfVariablesBeta + [1] * len(x.values)) #
                                     Objective function coefficients

ALeft = np.matrix([ [1] * len(x.values)]).transpose() # Coefficients for
                                                         beta0
ARigth = np.matrix(x.values) # Coefficients for other independent
                               variables beta
A = np.block([ALeft, ARigth]) # Concatenate coefficients of variables into
                               one matrix

I = np.identity(len(x.values)) # Identity matrix for intercept term
```

Naformulujeme problém a vyriešime pomocou `scipy.optimize.linprog`

```
A_ub = np.block([[-A, -I], [A, -I]])
b_ub = np.concatenate([-y, y])

solve = linprog(c, A_ub, b_ub, bounds = [(None, None)] *
                                         numberOfVariablesBeta + [(0, None)])
```

Po vyriešení vyberieme z riešenia koeficienty

```
betas = solve.x[:numberOfVariablesBeta]
```

Čo nám dá:

$$\begin{aligned}\beta_0^{(1)} &\approx -8.8801 \cdot 10^{-1}, \beta_1^{(1)} \approx 1.5793 \cdot 10^{-3}, \beta_2^{(1)} \approx 5.2130 \cdot 10^{-1} \\ \beta_3^{(1)} &\approx -4.5137 \cdot 10^{-3}, \beta_4^{(1)} \approx 1.1300 \cdot 10^{-2}, \beta_5^{(1)} \approx -2.2111 \cdot 10^{-5}\end{aligned}$$

Ďalej zostrojíme relevantné štruktúry a naformulujeme LP pre L^∞ normu:

```
c_inf = np.array([0]*numberOfVariablesBeta + [1]) # Koeficienty cielovej
                                                # funkcie
A_inf = np.block([ALeft, ARigth]) # Koeficienty pre nezavisle premenne
                                   # pre l_inf normu
i_inf = np.array([ [1] * len(x.values)]).transpose() # Koeficienty pre
                                                        # intercept term pre l_inf normu

A_ub_inf = np.block([-A_inf, -i_inf], [A_inf, -i_inf])
b_ub_inf = np.concatenate([-y, y])
```

Vyriešime aj tento problém pomocou `scipy.optimize.linprog()` pre L^∞ normu a vyberieme β koeficienty:

```
solve_inf = linprog(c_inf, A_ub_inf, b_ub_inf, bounds=[(None, None)]*
                                                        numberOfVariablesBeta + [(0, None)])
betas_inf = solve_inf.x[:numberOfVariablesBeta]
```

Po čom dostaneme:

$$\beta_0^{(\infty)} \approx 3.4841, \beta_1^{(\infty)} \approx 8.3399 \cdot 10^{-4}, \beta_2^{(\infty)} \approx 6.0027 \cdot 10^{-1}$$
$$\beta_3^{(\infty)} \approx -3.3416 \cdot 10^{-3}, \beta_4^{(\infty)} \approx -2.3036 \cdot 10^{-2}, \beta_5^{(\infty)} \approx -1.1958 \cdot 10^{-4}$$

4 Úloha D

Vytvorme funkciu `r_squared(x, y, beta)` - kde `x` je matica vektorov nezávislých premenných, `y` je vektor závislej premennej, `beta` je vektor optimálnych β koeficientov získaných regresiou - ktorá bude počítat R^2 koeficient podľa definície:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad \hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

```
def r_squared(x: np.ndarray, y: np.ndarray, beta: np.ndarray) -> float:
    # calculate y-hat and mean of y vector
    y_hat = beta[0] + np.dot(x, beta[1:])
    y_mean = np.mean(y)

    res1 = 0      # partial result for the numerator in the formula
    res2 = 0      # partial result for the denominator in the formula

    # calculate the sums
    for i in range(len(y)):
        res1 += (y[i] - y_hat[i]) ** 2
        res2 += (y[i] - y_mean) ** 2

    # calculate the R^2 coefficient
    result = 1 - (res1 / res2)
    return result
```

Implementujeme metódu na dátach `A04wine.csv`. Načítame dáta pomocou `pandas.read_csv()`, rozdelíme ich do premenných (rovnako ako v predošlých úlohách):

```
data = pd.read_csv('data/A04wine.csv')
y = data['Price']
x = data[['WinterRain', 'AGST', 'HarvestRain', 'Age', 'FrancePop']]
```

Podobne ako vyššie, vyriešime potrebné LP problémy pre načítané dáta a vypočítame R^2 koeficient:

```
betas = solve.x[:numberOfVariablesBeta]
betas_inf = solve_inf.x[:numberOfVariablesBeta]

r_squared(x, y, betas)
r_squared(x, y, betas_inf)
```

Vypočítané príslušné R-kvadráty teda sú:

$$R_{(1)}^2 \approx 0.78813$$
$$R_{(\infty)}^2 \approx 0.80649$$

Z toho môžeme usúdiť, že pre dané dáta regresia pomocou Chebyshevovej normy lepšie zachytáva rozptyl dát.

5 Úloha E

5.1 Spracovanie všeobecnej triedy pre L^1 a L^∞ regresiu

Vypracovali sme modul `Model` pre počítanie L^1 a L^∞ regresie z ľubovoľných číselných dát, ktorý využíva LP formulácie popísané vyššie. Konkrétne `L1Model` využíva formuláciu na minimalizovanie L^1 normy a `LInfModel` minimalizuje L^∞ normu. Príklad použitia tohto modelu sa nachádza v `model_demonstration.ipynb`. Následne opíšeme jednotlivé metódy jednotlivých modelov.

`Model.__init__(dependent_vect, independent_vect)`

Konštruktor triedy, spoločný pre oba modely, vytvorí inštanciu, ktorá si drží dáta a vie na nich vykonávať operácie popísané nižšie.

Argumenty:

- `dependent_vect: np.array` - vektor závislých premenných
- `independent_vect: np.array` - matica, ktorej stĺpce sú vektory nezávislých premenných

`Model.solve()`

Metóda, ktorá vyrieši regresnú LP úlohu na daných dátach. `L1Model.solve()` rieši minimalizáciou L^1 normy a `LInfModel.solve()`, rieši minimalizáciou L^∞ normy.

Vracia:

- `np.array` - vektor optimálnych β premenných

Po zavolaní tejto metódy si inštancia uloží vektor optimálnych β premenných do atribútu `self._beta`, potrebné pre metódy popísané nižšie.

`Model.r2()`

Vypočíta R^2 koeficient pre dané dáta a vypočítaný vektor β .

Vracia:

- `float` - výsledný R^2 koeficient

`Model.visualize()`

Ak je počet nezávislých premenných 1 alebo 2, táto metóda vykreslí graf dát spolu s vypočítanou regresnou priamkou, resp. rovinou.

Vracia:

- `bool` - úspešnosť vizualizácie, kde `False` označuje, že nezávislých premenných je viac ako 2, čiže nie je možné vykresliť graf