Lineárne najmenšie štvorce(binárny klasifikátor)

Maximilián Zivák(0.5), Dávid Daniš(0.5)

April 2023

1 Matematické úpravy

1.1 Transformácia sumy do maticovej formy

V zadaní je úloha o najmenších štvorcoch zadaná nasledovne

$$\text{Min} \sum_{i=1}^{N} \left(y^i - \tilde{f}\left(x^i\right) \right)^2 \tag{1}$$

Pričom $\tilde{f}\left(x^i\right) = \left(x^i\right)^T \beta + v$, pre jednoduchosť bude lepšie prepísať samotnú sumu do maticovej formy ktorá vráti $z@1^T$, v ktorom $z^i = \left(y^i - \tilde{f}\left(x^i\right)\right)$. Prvky vo vektore z budú skaláre, keďže y^i je skalar aj $\left(x^i\right)^T \beta$ je skalár. Teda ich môžeme sčítať tak že z vynásobime transponovaným jednotkovým vektorom. Ak zadefinujeme $z = y - z'$ potom vieme zapísať $z' = \tilde{f}$ nasledovne

$$X\beta + v$$

Pričom

$$X = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & x_1^4 \\ x_2^1 & x_2^2 & x_2^3 & x_2^4 \\ . & . & . & . \\ x_N^1 & x_N^2 & x_N^3 & x_N^4 \end{bmatrix}$$

Výsledkom násobenie matice X s vektorom β bude vektor ktorého prvky sú riešenia jednotlivých iterácií $\tilde{f}\left(x^i\right) = \left(x^i\right)^T \beta + v$, do ktorého pripočítame skalár v čím dostaneme z'. Pre potreby projektu bude ale lepšie ak v jednom vektore budú obidve premenné (β, v). Ak zadefinujeme nový vektor β' ako:

$$\beta' = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ v \end{bmatrix}$$

Musíme upraviť aj maticu X, keďže výsledok $X\beta + v$ je tvaru

$$\begin{bmatrix} \beta_1 x_1^1 + \beta_2 x_1^2 + \beta_3 x_1^3 + \beta_4 x_1^4 + v \\ . \\ . \\ \beta_1 x_N^1 + \beta_2 x_N^2 + \beta_3 x_N^3 + \beta_4 x_N^4 + v \end{bmatrix}$$

Stačí do matice X pridať stĺpec jednotiek čím dostaneme maticu

$$X' = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & x_1^4 & 1 \\ x_2^1 & x_2^2 & x_2^3 & x_2^4 & 1 \\ . & . & . & . & \\ x_N^1 & x_N^2 & x_N^3 & x_N^4 & 1 \end{bmatrix}$$

Ak túto maticu teraz vynásobíme β' získame

$$X'\beta' = \begin{bmatrix} \beta_1 x_1^1 + \beta_2 x_1^2 + \beta_3 x_1^3 + \beta_4 x_1^4 + 1 * v \\ . \\ . \\ \beta_1 x_N^1 + \beta_2 x_N^2 + \beta_3 x_N^3 + \beta_4 x_N^4 + 1 * v \end{bmatrix}$$

Teda $z' = X'\beta' = X\beta + v$. Teraz len dosadíme a získame funkciu

$$\mathrm{Min}((y - X'\beta')\, 1^T)^2$$

1.2 Gradient

Po transformácií úlohy teraz potrebujeme získať gradient na aplikáciu cauchyho metódy. Konkrétne hľadáme gradient funkcie

$$f(\beta') = (y - X'\beta')1^T)^2$$

Aplikovaním chain rule získame

$$2 * (y - X'\beta')1^T) * (y - X'\beta')1^T)'$$

Pred derivovaním druhej časti si ale musím dať pozor lebo vektor $\beta' = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ v \end{bmatrix}$

Teda gradient bude 5 prvkový. Druhú časť si vieme rozpísať pre potreby gradientu ako:

$$-(x_1^1 * \beta_1 + x_1^2 * \beta_2 + x_1^3 * \beta_3 + x_1^4 * \beta_4 + v + ... x_N^1 * \beta_1 + x_N^2 * \beta_2 + x_N^3 * \beta_3 + x_N^4 * \beta_4 + v)$$

(y vynecháme keďže to je konštanta a stále by sa zderivovala na nulu) Je celkom jasne vidieť že pri derivovaní prvkom β_n dostaneme

$$-\sum_{i=1}^{N} x_i^n$$

A pri derivovaní podľa v dostaneme $-N$, teda výsledný gradient bude

$$\nabla f(\beta') = 2 * (y - X'\beta')1^T) * \begin{bmatrix} -\sum_{i=1}^N x_i^1 \\ -\sum_{i=1}^N x_i^2 \\ -\sum_{i=1}^N x_i^3 \\ -\sum_{i=1}^N x_i^4 \\ -N \end{bmatrix}$$

2 Úloha a

Skúsili sme stepsize medzi $1 * 10^0$ a $1 * 10^-8$
Ak graf neobsahuje červenú bodku, znamená to že funkcia odišla do nekonečna.

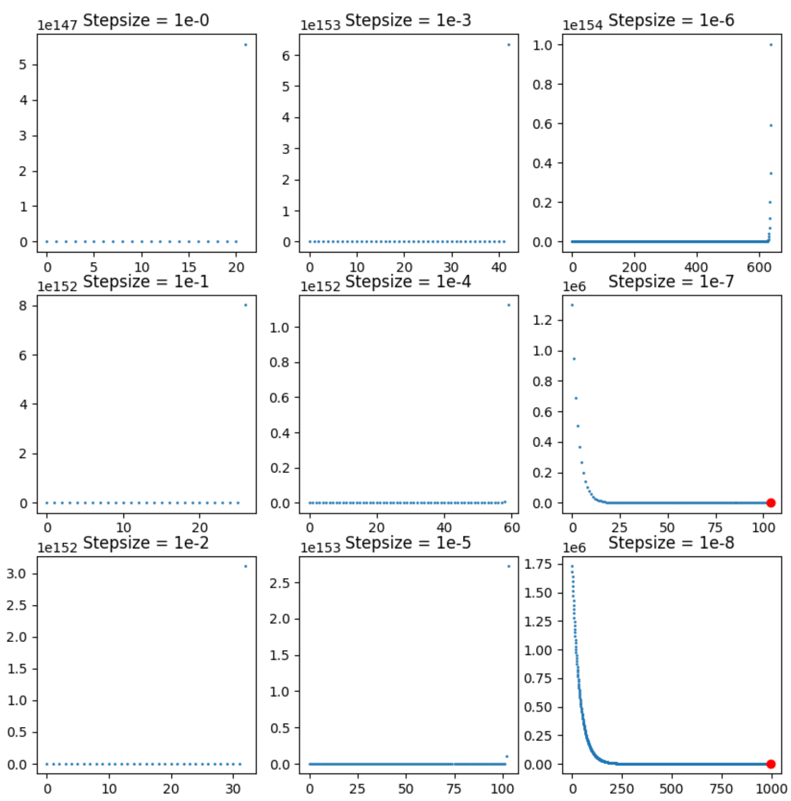

Figure 1: Konvergencia

3 Uloha b

Pre cauchyho metódu sme zvolili bisekciu ako metódu na nájdenie optimálneho kroku, backtracking aj cauchy našli rovnaké riešenie ale cauchy za menej iterácií, ďalej budeme pokračovať s cauchy metódou.

```
#backtrack

print(backtrack(dfun,np.array(x0),args=[fun],options={"tol":1e-8},vec=(X,y,c)))

✓ 0.0s

     fun: array([-1.65662328e-09, -8.65829008e-10, -1.06560663e-09, -3.39829853e-10,
       -2.83506552e-10])
     nit: 7
 success: True
       x: array([  5.34677348,   2.79447454,   3.43925943,   1.09680533,
       -53.68570534])
```

```
#cauchy

print(cauchy(dfun,np.array(x0),args=[fun],options={"tol":1e-8},vec=(X,y,c)))

✓ 0.0s

     fun: array([3.28833494e-09, 1.71863803e-09, 2.11518909e-09, 6.74549483e-10,
       5.62749847e-10])
     nit: 6
 success: True
       x: array([  5.34677348,   2.79447454,   3.43925943,   1.09680533,
       -53.68570534])
```

Figure 2: Konvergencia

4 Uloha c

Klasifikátor sa dá jednoduchu implementovať cez if, else a má chybovosť 0.226667, čo je celkom decentné keďže keby všade dáme nuly tak chybovosť by bola 0.333334.

5 Uloha d

Rozdelenie prebieha celkom naivne, Najprv sa náhodne vyberie 5 kosatcov Virginica potom náhodne 20 ostatných, na koniec sa týchto 25 kosatcov odstráni.

```
[  4.49760976   2.4768527    2.70173319   0.84251    -43.20876294]
Chybovosť klasifikatoru:  0.24
[  4.45935481   2.32539192   2.69883542   0.85358516 -42.25212889]
Chybovosť klasifikatoru:  0.24
[  4.32390165   2.39567524   2.57096855   0.80265884 -41.0178416 ]
Chybovosť klasifikatoru:  0.28
[  5.05746697   2.58659145   3.15155748   0.95976156 -50.48080161]
Chybovosť klasifikatoru:  0.2
```

Figure 3: Klasifikatory

6 Uloha e

Ak sa pozrieme na koeficienty β v modeloch použitých na cross validáciu a spočítame štandardnú odchylku, uvidíme že β_4, ju má nejmenšiu. Teda môžeme tvrdiť že práve tento prvok je najsignifikantnejší.

```
SDs = [0.7288131249525166, 0.35827272513545455, 0.5833285343890893, 0.20438254399317443]
```

Figure 4: Odchylky

6.1 Zmena formulacie

Premenná β bude len číslo, čím vlastne dostaneme

$$\text{Min} \sum_{i=1}^{N} \left(y^i - \left(x_1^i * \beta + v\right)\right)^2 \tag{2}$$

Podobnými úpravami ako pri ulohe 1 dostaneme

$$(\left(y - (x_1 * \beta + v)\right)1^T\right)^2$$

$$((y - \begin{bmatrix} x_1^1 & 1 \\ . & \\ . & \\ . & \\ x_1^N & 1 \end{bmatrix} \begin{bmatrix} \beta \\ v \end{bmatrix})1^T)^2$$

$$\nabla f(\beta') = 2 * (\left(y - (x_1 * \beta + v)\right)1^T\right) * \begin{bmatrix} -\sum_{i=1}^N x_i^1 \\ -N \end{bmatrix}$$

Klasifikator mozeme pouzit rovnaky, vidíme že chybovosť je 0.286667, čo je skoro tak dobre ako originalny klasifikator.