Lasso s ograničenjima

Orešić, Filip

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Filip Orešić

LASSO S OGRANIčENJIMA

Diplomski rad

Voditelj rada:
prof.dr.sc. Ivica Nakić

Zagreb, veljača, 2025.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

U današnjem svijetu, uz svakodnevno korištenje računala te brojne elektroničke opreme, generiraju se velike količine podataka. Uz korištenje matematičkih procesa, moguće je iz određenih grupa podataka primjetiti uzorke koji mogu usmjeriti na daljnje ponašanje i postupke. Na primjer, ukoliko se po podacima primjeti da određena dobna grupa posjećuje web stranicu više od ostalih, tada je bolje prikazivati veću količinu sadržaja kojeg ta grupa konzumira. Drugi primjer je povezivanje uzroka bolesti s bolestima, te na taj način liječenje bolesti svesti na moguću prevenciju pojavljivanja bolesti u rizičnim grupa. Nadalje, moguće je iz postojećih podataka pretpostaviti buduća ponašanja događaja koji su opisani tim podacima. Zbog toga, u današnje doba raste zainteresiranost za područja poput statistike, strojnog učenje te podatkovne znanosti.

Jedan primjer takvog matematičkog procesa je regresijska analiza. U statistici, regresijska analiza je proces za procjenjivanje odnosa između varijabli koje predstavljaju rezultate, te nezavnisnih varijabli. U ovom radu ćemo promatrati posebnu vrstu jedne od metoda regresijske analize, a ta metoda je Least absolute shrinkage and selection operator (LASSO).

LASSO se prvi put spominje u članku Fadila Santose i Williama Symesa 1986. godine. LASSO je uveden za potrebe geofizike, točnije za linearne inverzije seizmograma. Deset godina kasnije, 1996., statističar Robert Tibshirani objavljuje članak koji dovodi do popularizacije LASSOa. Originalno je LASSO korišten za linearnu regresiju, ali je kasnije proširen na druge statističke modele. Danas LASSO nalazi svoje primjene u statistici, konveksnoj analizi, obradi signala, strojnom učenju te u drugim granama matematike. U ovom radu ćemo promatrati verziju LASSOa s dodatnim ograničenjima na uvjete regresije.

U prvom poglavlju krećemo od motivacije za korištenje LASSOa, te dajemo definicije operatora koja se koriste za definiciju problema LASSO, te definiciju ograničenog LASSOa uz primjere gdje bi se ograničeni LASSO mogao koristiti te dokazujemo ekvivalentnost s još jednim tipom LASSOa.

U drugom poglavlju ćemo opisati neke od algoritama za rješavanje problema ograničenog LASSOa te ćemo u trećem poglavlju opisati implementaciju jednog od algoritama iz drugog poglavlja u programskom jeziku C++.

U četvrtom, ujedno i zadnjem poglavlju, opisujemo primjenu implementacije algoritma opisanu u trećem poglavlju na neke od problema u stvarnom životu, iako je jedan

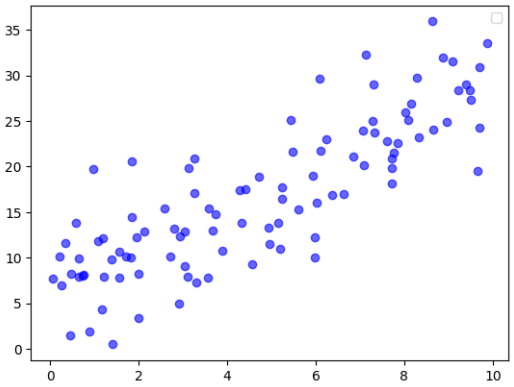
od problema implementiran na simuliranim podacima. Problemi koriste različite pristupe implementaciji ograničenja, obzirom da sama ograničenja na regresiju nisu ista ovisno o tipu problema.

U dodatku se nalazi implementacija algoritma koji se koristio u poglavljima 3 i 4.

Poglavlje 1

Matematički opis

Često iz prethodnih događaja želimo zaključiti ili pretpostaviti daljnje ponašanje. U tome nam može pomoći regresijska analiza. Najjednostavniji oblik regresijske je linearna regresija, gdje podatke pokušavamo modelirati preko linearne funkcije $y = kx + l$. Uzmimo primjer skupa podataka prikazanih kao na sljećoj slici:



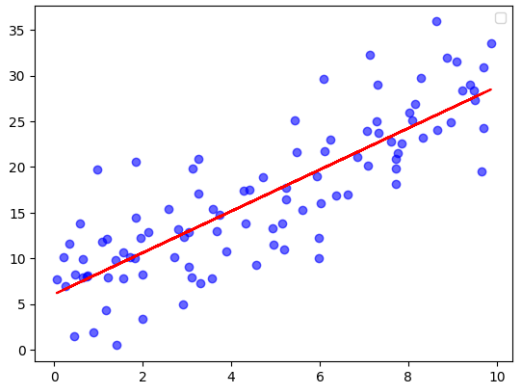
Slika 1.1: Nasumično generirani podaci

Vidimo da bi podaci mogli biti aproksimirani pravcem, pa tako nakon provođenja linearne regresije dobivamo situaciju prikazanu na slici 1.2.

LASSO je oblik linearne regresije koji još dodaje regularizaciju. Regularizacija je pojednostavljenje rješenja, te u ovom kontekstu označava pretvorbu vektora rješenja u rjeđi vektor. Za vektor kažemo da je rijedak ako su mu većina koeficijenata jednaki nuli. Neke od prednosti rijetkih vektora su:

- pohrana: nije potrebno pohranjivati cijeli vektor, nego samo ne-nul elemente

- efikasnije operacije: moguće je izbjeći korištenje nul elemenata u operacijama
- pohranjivanje velikih skupova podataka za korištenje u strojnom učenju
- prikazivanje odnosa u velikim skupovima podataka



Slika 1.2: Nasumično generirani podaci s pravcem regresije

1.1 LASSO

Definicija 1.1.1. *Neka je* $\mathbf{x} = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n$. *Definiramo* L_1*-normu vektora* $\mathbf{x}$ *kao*

$$\|\mathbf{x}\|_1 = \sum_{k=1}^{n} |x_k|$$

Neka je $\mathbf{x} = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n$. *Definiramo* L_2*-normu vektora* $\mathbf{x}$ *kao*

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{k=1}^{n} |x_k|^2}$$

Definicija 1.1.2 (Problem LASSO)**.** *Za dani vektor mjerenja* $y \in \mathbb{R}^n$ *koji predstavlja izlazne podatke, matricu prediktora* $X \in \mathbb{R}^{n \times m}$ *koja sadrži nezavisne podatke, i parametar* $\lambda \geq 0$ *koji određuje regularizaciju u regresiji želimo procjeniti vektor* $\beta \in \mathbb{R}^m$ *koji najbolje određuje povezanost nezavisne varijable iz matrice* X *s izlazima iz vektora* y. *Definiramo LASSO problem (Least absolute shrinkage and selection operator) kao*

$$\min_{\beta} \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\|\beta\|_1 \tag{1.1}$$

Zbog prirode L_1-norme te minimizacije u LASSO problemu, veće vrijednosti parametra λ dovode do rjeđeg vektora β, odnosno više koeficijenata vektora β su jednaki 0.

Lema 1.1.3. *Za svaki $y, X, \lambda \geq 0$, LASSO problem ima sljedeća svojstva:*
(*i*) *Postoji jedinstveno rješenje LASSO problema ili neprebrojivo mnogo rješenja*
(*ii*) *Za svaka dva različita rješenja $\hat{\beta}^{(1)}$, $\hat{\beta}^{(2)}$ vrijedi $X\hat{\beta}^{(1)} = X\hat{\beta}^{(2)}$*
(*iii*) *Ako je $\lambda > 0$, svako rješenje $\hat{\beta}$ ima jednaku L_1-normu*

Dokaz. (i) LASSO funkcija $\mathcal{L}(\beta) = \min_\beta \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$ je konveksna. Zbog svojstva da za konveksnu funkciju svaki lokalni minimum je ujedno i globalni minimum, funkcija $\mathcal{L}$ postiže minimalnu vrijednost na $\mathbb{R}^m$, tj. LASSO problem ima barem jedno rješenje. Pretpostavimo sada da imamo dva rješenja $\hat{\beta}^{(1)}$ i $\hat{\beta}^{(2)}$, pri čemu $\hat{\beta}^{(1)} \neq \hat{\beta}^{(2)}$. Budući da je skup rješenja konveksnog minimizacijskog problema konveksan, znamo da je $\alpha\hat{\beta}^{(1)} + (1-\alpha)\hat{\beta}^{(2)}$ također rješenje za bilo koji $0 < \alpha < 1$, što daje beskonačno mnogo LASSO rješenja kako α varira unutar $(0, 1)$.

(ii) Pretpostavimo da imamo dva rješenja $\hat{\beta}^{(1)}$ i $\hat{\beta}^{(2)}$ za koje vrijedi $X\hat{\beta}^{(1)} \neq X\hat{\beta}^{(2)}$. Neka c^* označava minimalnu vrijednost LASSO funkcije dobivenu za $\hat{\beta}^{(1)}$ i $\hat{\beta}^{(2)}$. Za bilo koji $0 < \alpha < 1$, imamo

$$\frac{1}{2}\|y - X(\alpha\hat{\beta}^{(1)} + (1-\alpha)\hat{\beta}^{(2)})\|_2^2 + \lambda\|\alpha\hat{\beta}^{(1)} + (1-\alpha)\hat{\beta}^{(2)}\|_1 < \alpha c^* + (1-\alpha)c^* = c^*,$$

gdje je stroga nejednakost zbog stroge konveksnosti funkcije $f(x) = \|y - x\|_2^2$ te konveksnosti L_1- norme. To znači da $\alpha\hat{\beta}^{(1)} + (1-\alpha)\hat{\beta}^{(2)}$ postiže nižu vrijednost LASSO funkcije $\mathcal{L}$ nego c^*, što je kontradikcija.

(iii) Prema (ii), bilo koja dva rješenja $\hat{\beta}^{(1)}$ i $\hat{\beta}^{(2)}$ moraju imati iste vrijednosti $X\hat{\beta}^{(1)}$ i $X\hat{\beta}^{(2)}$, a time i istu kvadratnu pogrešku. Ali rješenja također postižu istu vrijednost LASSO funkcije $\mathcal{L}$, i kako je $\lambda \geq 0$, tada moraju imati istu L_1 normu.

□

1.2 Ograničeni LASSO

U mnogim slučajevima u stvarnome životu možemo očekivati neka uvjete i ograničenja. Na primjer, ukoliko modeliramo očekivanu vrijednost novčane valute, zbog inflacije možemo očekivati da je vrijednost svake sljedeće godine manja od prethodne. Zato na LASSO možemo uvesti ograničenja.

Definicija 1.2.1 (Problem ograničeni LASSO)**.** *Za dani vektor mjerenja $y \in \mathbb{R}^n$ koji predstavlja izlazne podatke, matricu prediktora $X \in \mathbb{R}^{n\times m}$ koja sadrži nezavisne podatke, i parametar $\lambda \geq 0$ koji određuje regularizaciju u regresiji želimo procjeniti vektor $\beta \in \mathbb{R}^m$*

koji najbolje određuje povezanost nezavisne varijable iz matrice X s izlazima iz vektora y. Uz predefinirane matrice $A \in \mathbb{R}^{k\times m}$ i $C \in \mathbb{R}^{l\times m}$, te predefinirane vektore $b \in \mathbb{R}^k$ i $d \in \mathbb{R}^l$, definiramo ograničeni LASSO problem kao

$$\begin{aligned} &\min_{\beta} \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \\ &\text{uz uvjete } A\beta = b \\ &\qquad\qquad C\beta \leq d \end{aligned} \tag{1.2}$$

Pretpostavka je da su matrice ograničenja A i C punog ranga. Ovaj problem se naziva i penalizirana i ograničena optimizacija, ili PaC, od engleskog naziva penalized and constrained.

Primjer 1.2.2 (Monotono prilagođavanje krivulje)**.** *Razmotrimo problem prilagođavanja glatke funkcije $h(x)$ skupu opažanja $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, uz uvjet da funkcija h mora biti monotona. Modeliramo $h(x)$ kao $B(x_i)^T\beta$, gdje je $B(x_i)$ visoko-dimenzionalna fleksibilna bazna funkcija, poput bazne funkcije splajnova. Ovaj problem možemo riješiti korištenjem PaC metodologije minimiziranjem*

$$g(\beta) = \sum_{i=1}^{n} (y_i - B(x_i)^T\beta)^2$$

uz uvjet $C\beta \leq 0$, gdje je l-ti redak matrice C derivacija $B'(u_l)$ baznih funkcija evaluiranih u u_l za fino raspoređene točke $u_1, \ldots, u_m$ u rasponu x. Nametanje ovog uvjeta osigurava da je derivacija h nenegativna, pa će h biti monotono opadajuća.

Primjer 1.2.3 (Optimizacija portfelja)**.** *Optimizacija portfelja je još jedan poznat problem koji se uklapa u PaC okruženje. Pretpostavimo da imamo p financijskih instrumenata označenih s $1, 2, \ldots, p$, čija je matrica kovarijance označena sa Σ. H.M. Markowitz je razvio radni okvir za analizu varijance i prinosa portelja. Konkretno, njegov pristup uključivao je davanje težina financijskim instrumentima β kako bi se minimizirao rizik portfelja $R(\beta) = \beta^T\Sigma\beta$ uz uvjet $\beta^T 1 = 1$. Često se također želi nametnuti dodatne uvjete na β kako bi se kontrolirao očekivani prinos portfelja, alokacije među sektorima ili industrijama ili izloženost određenim poznatim faktorima rizika.*

U praksi je Σ nepoznata pa se mora procijeniti koristeći uzorkovnu matricu kovarijance, $\hat{\Sigma}$. Međutim, u financijskoj literaturi je dobro dokumentirano da kada je p velik, što je norma u stvarnim aplikacijama, minimiziranje $\hat{R}(\beta) = \beta^T\hat{\Sigma}\beta$ daje loše procjene za β. Jedno moguće rješenje uključuje regularizaciju $\hat{\Sigma}$, ali nedavno se pozornost usmjerila na izravno penaliziranje ili ograničavanje težina, analogno pristupu penaliziranja koeficijenata u regresijskom okruženju. Fan i sur. (2012) usvojili su ovaj pristup minimiziranjem $\hat{R}(\beta)$ uz uvjet $\beta^T 1 = 1$ i $\|\beta\|_1 \leq c$, gdje je c regularizacijski parametar. Nije teško provjeriti

da se ovaj optimizacijski problem može izraziti u obliku ograničenog LASSO problema, gdje C ima barem jedan redak (za ograničenje β na zbroj jedan), ali može imati i dodatne retke ako postavimo ograničenja na očekivani prinos, udjele industrija, itd. Stoga, implementacija PaC s $g(\beta) = \hat{R}(\beta)$ omogućava nam rješavanje ograničenog i regulariziranog problema optimizacije portfelja.

Primjer 1.2.4 (Problem generalizirani LASSO)**.** *Za dani vektor mjerenja $Y \in \mathbb{R}^n$, matricu prediktora $\tilde{X} \in \mathbb{R}^{n\times m}$, nepoznati vektor $\theta \in \mathbb{R}^m$, parametar $\lambda \geq 0$ koji određuje regularizaciju u regresiji, te matricu $D \in \mathbb{R}^{n\times m}$, definiramo generalizirani LASSO problem kao*

$$\min_{\theta} \frac{1}{2} \left\| Y - \tilde{X}\theta \right\|_2^2 + \lambda \|D\theta\|_1$$

Kada je rang matrice $D = n$, i Kada vrijedi $n \leq m$, problem se generaliziranog LASSO-a može svesti na problem LASSO. U situaciji kada je $n > m$, tada nije moguće svesti problem generaliziranog LASSO-a na problem LASSO.

Lema 1.2.5. *Neka su oznake kao u primjeru 1.2.4. Kada je rang matrice $D = m$ i $n > m$, tada postoje matrice A, C i X, tako da za svaku vrijednost parametra λ, rješenje problema generaliziranog LASSO-a je jednako $\theta = A\beta$, gdje je β dana s*

$$\min_{\beta} \frac{1}{2} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

uz uvjet $C\beta = 0$.

Dokaz. Obzirom da matrica D ima puni rang po stupcima, možemo pisati D kao

$$D = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix},$$

gdje je $D_1 \in \mathbb{R}^{m\times m}$ invertibilna matrica, a $D_2 \in \mathbb{R}^{n-m\times m}$. Tada vrijedi

$$\begin{aligned}
&\frac{1}{2} \left\| Y - \tilde{X}\theta \right\|_2^2 + \lambda \|D\theta\|_1 \\
&= \frac{1}{2} \left\| Y - \tilde{X}D_1 D_1^{-1}\theta \right\|_2^2 + \lambda \|D_1\theta\|_1 + \lambda \|D_2\theta\|_1 \\
&= \frac{1}{2} \left\| Y - \left(\tilde{X}D_1^{-1}\right) D_1\theta \right\|_2^2 + \lambda \|D_1\theta\|_1 + \lambda \left\| D_2 D_1^{-1} D_1\theta \right\|_1 .
\end{aligned}$$

Koristimo sljedeću zamjenu varijabli:

$$\beta_1 = D_1\theta, \beta_2 = D_2 D_1^{-1} D_1\theta = D_2 D_1^{-1}\beta_1$$

te stavimo

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}.$$

Uz danu zamjenu varijabli generalizirani LASSO problem sada glasi:

$$\begin{aligned} &\min_{\theta \in \mathbb{R}^m} \frac{1}{2} \left\| Y - \tilde{X}\theta \right\|_2^2 + \lambda \left\| D\theta \right\|_1 \\ &= \min_{\beta \in \mathbb{R}^n} \left\{ \frac{1}{2} \left\| Y - \tilde{X} D_1^{-1} \beta_1 \right\|_2^2 + \lambda \left\| \beta \right\|_1 \middle| D_2 D_1^{-1} \beta_1 - \beta_2 = 0 \right\} \\ &= \min_{\beta \in \mathbb{R}^n} \left\{ \frac{1}{2} \left\| Y - X\beta \right\|_2^2 + \lambda \left\| \beta \right\|_1 \middle| C\beta = 0 \right\}, \end{aligned}$$

gdje je $X = \begin{bmatrix} \tilde{X} D_1 - 1 & 0 \end{bmatrix}$, $C = \begin{bmatrix} D_2 D_1^{-1} & -I \end{bmatrix}$, te $\theta = \begin{bmatrix} D^{-1} & 0 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$. S time dobivamo da je generalizirani LASSO specijalan slučaj ograničenog LASSO-a.

□

Poglavlje 2

Algoritmi

2.1 Kvadratični program

Generalni kvadratnični program (quadratic program ili QP) je oblika

$$\begin{aligned} \min_x q(x) &= \frac{1}{2}x^T G x + c^T x \\ \text{uz uvjete } a_i^T x &= b_i, i \in \mathcal{E} \\ a_i^T x &\geq b_i, i \in \mathcal{I} \end{aligned}$$

gdje je G simetrična $n \times n$ matrica, $\mathcal{E}$ i $\mathcal{I}$ su skupovi indeksa, dok su c, x i $\{a_i\}, i \in \mathcal{E} \cup \mathcal{I}$ vektori u $\mathbb{R}^n$.

Kvadratnični programi nalaze primjenu u različitim područjima, uključujući optimizaciju portfelja, računalni vid, obradu signala i slika, te u mnogim drugim optimizacijskim problemima.

Za rješavanje kvadratničnog programiranja koriste se najčešće dvije skupine algoritama: metoda aktivnog skupa(ASM) i metoda unutrašnje točke(IPM).

Neka su $f, g_i, h_j : \mathbb{R}^n \to \mathbb{R}$ diferencijabilne funkcije te neka imamo ne nužno linearan problem

$$\begin{aligned} &\min f(x) \\ \text{uz uvjete } &g_i(x) \leq 0, i = 1, ..., m \\ &h_j(x) = 0, j = 1, ..., l \end{aligned}$$

Definiramo Lagrangeovu funkciju s

$$L(x, u, v) = f(x) + \sum_{i=1}^{m} u_i g_i(x) + \sum_{j=1}^{l} v_j h_j(x), u_i \geq 0$$

Tada su Karush-Kuhn-Tucker(KKT) uvjeti optimalnosti dani s:

$$
\begin{aligned}
&i)\ g_i(x) \leq 0, i = 1, ..., m, h_j(x) = 0, j = 1, ..., l \\
&ii)\ u_i g_i(x) = 0, u_i \geq 0, i = 1, ..., m \\
&iii)\ \nabla_x L(x, u, v) = 0
\end{aligned}
$$

Točka (x, u, v) koja zadovoljava KKT uvjete se naziva KKT točka.
Metoda aktivnog skupa je opisana sljedećim pseudokodom:

Algorithm 1 Metoda aktivnog skupa

Odaberi početni vektor x^0
Inicijaliziraj aktivni skup $\mathcal{A}^0 = \left\{ j \middle| b_j^T x^0 - b_j = 0, j = 1, ..., m \right\}$
Postavi $k = 0$
while nema konvergencije **do**
    Izračunaj $g^k = Gx^k + c$
    Izračunaj d^k, u^k, v^k rješavajući KKT uvjete za

$$
\begin{aligned}
&\min_d \left\{ \frac{1}{2} d^T G d + \left[g^k \right]^T d \right\} \\
\text{uz uvjete } &a_i^T d = 0 \\
&b_j^T d = 0, j \in \mathcal{A}^k
\end{aligned}
$$

    if $d^k = 0$ **then**
        if $v^k \geq 0$ **then**
            x^k je KKT točka, izađi iz programa
        else
            $v_{j_0} = \min \left\{ v_j \middle| v_j < 0, j \in \mathcal{A}^k \right\}$
            Postavi $\mathcal{A}^k = \mathcal{A}^k \setminus \{j_0\}$ i GO TO korak 6
    if $d^k \neq 0$ **then**
        $\alpha_k = \min \left\{ 1, \frac{b_j - b_j^T d_k}{b_j^T d^k} \middle| j \notin \mathcal{A}^k i b_j^T d^k > 0 \right\}$
    $x^{k+1} = x^k + \alpha_k d^k$
    if $\alpha_k = 1$ **then**
        $\mathcal{A}^{k+1} = \mathcal{A}$
    else
        $\mathcal{A}^{k+1} = \mathcal{A}^k \cup \{j_0\}$
    $k = k + 1$

Rješenje problema je vektor x^k za koji algoritam stane. Problemi kod metode aktivnog skupa su kako izabrati dobar početni vektor x^0 te efikasna strategija za određivanje aktivnog skupa $\mathcal{A}^k$ za svaki x^k.

Metoda unutrašnje točke je dana sa sljedećim algoritmom:

Algorithm 2 Metoda aktivnog skupa

1: Odaberi σ_{min}
2: Odaberi početne vrijednosti (x^0, λ^0, s^0) tako da $(x^0, s^0) > 0$
3: Postavi $k = 0$
4: **while** $\frac{\left(x^k\right)^T s^k}{n} > \epsilon$ **do**
5: Odaberi $\sigma_k \in [\sigma_{min}, \sigma_{max}]$
6: Rješi sustav i odredi $d^k = (\Delta x_k, \Delta\lambda_k, \Delta s_k)$

$$\begin{bmatrix} G & -A^T & -I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} d = -\begin{bmatrix} Gx - A^T\lambda - s - q \\ Ax - b \\ XS - \sigma_k\mu_k e \end{bmatrix}$$

▷ A predstavlja matricu vektora za uvjet jednakosti
7: Odaberi najveći α_{max} tako da $(x^k, s^k) + \alpha\,(\Delta x_k, \Delta s_k)$
8: Postavi $\alpha_k = \min\{1, \nu_k\alpha_{max}\}$ za neki $\nu_k \in \langle 0, 1\rangle$ i $\mu_k = \frac{\left(x^k\right)^T s^k}{n}$
9: $\left(x^{k+1}, \lambda^{k+1}, s^{k+1}\right) + \alpha_k \left(\Delta x_k, \Delta\lambda^k, \Delta s_k\right)$
10: $k = k + 1$

Za uvjet zaustavljanja se bira pogodna vrijednost parametra ϵ.

Kako bi sveli problem ograničenog LASSO-a na kvadratnično programiranje, razdvajamo β u pozitivni i negativni dio, $\beta = \beta_+ - \beta_-$, tako da kada to uvrstimo u (1.2) u kombinaciji s dodavanje uvjeta nenegativnosti na β_+ i β_-, dobivamo kvadratnični program od $2m$ varijabli:

$$\begin{aligned} \text{minimiziraj } & \frac{1}{2}\begin{pmatrix}\beta_+\\ \beta_-\end{pmatrix}^T \begin{pmatrix} X^TX & -X^TX \\ -X^TX & X^TX \end{pmatrix}\begin{pmatrix}\beta_+\\ \beta_-\end{pmatrix} \\ & + \left(\lambda I_{2m} - \begin{pmatrix} X^Ty \\ -X^Ty \end{pmatrix}\right)^T \begin{pmatrix}\beta_+\\ \beta_-\end{pmatrix} \\ \text{uz uvjet } & \begin{pmatrix} A & -A \end{pmatrix}\begin{pmatrix}\beta_+\\ \beta_-\end{pmatrix} = b,\ \beta_+ \geq 0 \\ & \begin{pmatrix} C & -C \end{pmatrix}\begin{pmatrix}\beta_+\\ \beta_-\end{pmatrix} \leq d,\ \beta_- \geq 0 \end{aligned}$$

2.2 ADMM

Drugi algoritam za rješavanje problema ograničenog LASSOa je Alternative direction method of multipliers (ADMM). ADMM se koristi za rješavanje problema gdje se funkcija koja se minimizira može separirati, te je oblika

$$\begin{aligned} &\min f(x) + g(z), \\ &\text{uz uvjet } Mx + Fz = c, \end{aligned}$$

gdje su $f, g : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ zatvorene prave konveksne funkcije. To su konveksne funkcije koje nikad ne poprimaju vrijednost $-\infty$, te za koje postoji bar jedna točka iz domene za koju je vrijednost funkcije različita od $+\infty$, i funkcije za koje je skup točaka koje leže na ili iznad njenog grafa zatvoren skup. Ideja je koristiti koordinatni spust Lagrangeove funkcije uz promjenu dualne varijable v:

$$\begin{aligned} x^{(t+1)} &= \min_x \mathcal{L}_\tau \left(x, z^{(t)}, v^{(t)}\right), \\ z^{(t+1)} &= \min_z \mathcal{L}_\tau \left(x^{(t+1)}, z, v^{(t)}\right), \\ v^{(t+1)} &= v^{(t)} + \tau \left(Mx^{(t+1)} + Fz^{(t+1)} - c\right), \end{aligned}$$

gdje je t brojač iteracija, a Lagrangeova funkcija je

$$\mathcal{L}_\tau(x, z, v) = f(x) + g(x) + v^T (Mx + Fz - c) + \frac{\tau}{2} \|Mx + Fz - c\|_2^2 .$$

Često je jednostavnije raditi s ekvivalentnom verzijom ADMM-a, koja kombinira linearne i kvadratne varijable, pa tako dobivamo:

$$\begin{aligned} x^{(t+1)} &= \min_x f(x) + \frac{\tau}{2} \left\|Mx + Fz^{(t)} - c + u^{(t)}\right\|_2^2, \\ z^{(t+1)} &= \min_z g(z) + \frac{\tau}{2} \left\|Mx^{(t+1)} + Fz - c + u^{(t)}\right\|_2^2, \\ u^{(t+1)} &= u^{(t)} + Mx^{(t+1)} + Fz^{(t+1)} - c, \end{aligned}$$

gdje je $u = \dfrac{v}{\tau}$. Ova verzija je izrazito korisna u situaciji kada je $M = F = I$, jer se promjene varijabli mogu zapisati kao:

$$\begin{aligned} x^{(t+1)} &= \text{prox}_{\frac{1}{\tau} f} \left(c - z^{(t)} - u^{(t)}\right), \\ z^{(t+1)} &= \text{prox}_{\frac{1}{\tau} g} \left(c - x^{(t+1)} - u^{(t)}\right), \\ u^{(t+1)} &= u^{(t)} + x^{(t+1)} + z^{(t+1)} - c. \end{aligned}$$

Funkciju prox definiramo kao

$$\text{prox}_{\frac{1}{\tau}f}(v) = \min_x \left(f(x) + \frac{\tau}{2} \|x - v\|_2^2 \right)$$

Kako bismo koristili ADMM za rješavanje problema ograničenog LASSOa, funkciju f definiramo kao funkciju minimizacije u definiciji LASSOa, dok funkciju g definiramo za skup $C = \{\beta \in \mathbb{R}^m : A\beta = b, C\beta \leq d\}$ kao

$$g(\beta) = \chi_C = \begin{cases} \infty, & \beta \in C \\ 0, & \beta \notin C \end{cases}$$

Za ažuriranja varijabli, $\text{prox}_{\frac{1}{\tau}f}$ predstavlja regularni LASSO problem, dok $\text{prox}_{\frac{1}{\tau}g}$ predstavlja projekcija na afin prostor C. Algoritam za rješavanje problema ograničenog LASSOa koristeći ADMM je:

Algorithm 3 ADMM

1: $\beta^(0) = z^{(0)} = \beta^0, u^{(0)} = 0, \tau > 0$
2: **while** kriterij konvergencije nije zadovoljen **do**
3: $\beta^{(t+1)} = \min \frac{1}{2} \|y - X\beta\|_2^2 + \frac{\tau}{2} \left\|\beta + z^{(t)} + u^{(t)}\right\|_2^2 + \rho \|\beta\|_1$
4: $z^{(t+1)} = \text{proj}_C \left(\beta^{(t+1)} + u^{(t)}\right)$
5: $u^{(t+1)} = u^{(t)} + \beta^{(t+1)} + z^{(t+1)}$

2.3 Algoritam puta

Treći algoritam za rješavanje problema ograničenog LASSOa je algoritam puta. Prema KKT uvjetima optimalna točka $\beta(\rho)$ je dana s stacionarnim uvjetom

$$-X^T \left[y - X\beta(\rho)\right] + \rho s(\rho) + A^T \lambda(\rho) + C^T \mu(\rho) = 0_m$$

u kombinaciji s linearnim uvjetima. Funkcija $s(\rho)$ je dana s

$$s_j(\rho) = \begin{cases} 1, & \beta_j(\rho) > 0 \\ [-1, 1], & \beta_j(\rho) = 0 \\ -1, & \beta_j(\rho) < 0 \end{cases}$$

te vrijedi $\mu_l = 0$ ako je $c_l^T \beta < d_l$ i $\mu_l \geq 0$ ako je $c_l^T \beta = d_l$. Također, imamo i skupove indeksa

$$\mathcal{A} = \left\{j : \beta_j \neq 0\right\}, \mathcal{Z}_{\mathcal{I}} = \left\{l : c_l^T \beta = d_l\right\}.$$

Prvi skup prati aktivne koeficijente, dok drugi prati uvjet nejednakosti. Gledajući vektorske jednadžbe

$$0_{|\mathcal{A}|} = -X_{:,\mathcal{A}}^T (y - X_{:,\mathcal{A}}\beta_{\mathcal{A}}) + \rho s_{\mathcal{A}} + A_{:,\mathcal{A}}^T \lambda + C_{\mathcal{Z}_I,\mathcal{A}}^T \mu_{\mathcal{Z}_I}$$
$$\begin{pmatrix} b \\ d_{\mathcal{Z}_I} \end{pmatrix} = \begin{pmatrix} A_{:,\mathcal{A}} \\ C_{\mathcal{Z}_I,\mathcal{A}} \end{pmatrix} \beta_{\mathcal{A}}$$

koje sadrže nepoznate varijable $\beta_{\mathcal{A}}, \lambda, \rho$ i $\mu_{\mathcal{Z}_I}$ i primjenjujući na njih teorem o implicitnoj funkciji dobivamo smjer za praćenje puta:

$$\frac{d}{d\rho}\begin{pmatrix} \beta_{\mathcal{A}} \\ \lambda \\ \mu_{\mathcal{Z}_I} \end{pmatrix} = -\begin{pmatrix} X_{:,\mathcal{A}}^T X_{:,\mathcal{A}} & A_{:,\mathcal{A}}^T & C_{\mathcal{Z}_I}^T \\ A_{:,\mathcal{A}} & 0 & 0 \\ C_{\mathcal{Z}_I,\mathcal{A}} & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} s_{\mathcal{A}} \\ 0 \\ 0 \end{pmatrix}. \tag{1}$$

Desna strane je konstantna na segmentu puta ukoliko se skupovi $\mathcal{A}$ i $\mathcal{Z}_I$ te predznaci aktivnih koeficijenata $s_{\mathcal{A}}$ ne mijenjaju te je zato put rješenja ograničenog LASSOa po dijelovima linearan. Uvjet stacionarnosti restringiran na neaktivne koeficijente

$$-X_{:,\mathcal{A}^C}^T (y - X_{:,\mathcal{A}}\beta_{\mathcal{A}}(\rho)) + \rho s_{\mathcal{A}^C}(\rho) + A_{:,\mathcal{A}^C}^T \lambda(\rho) + C_{\mathcal{Z}_I,\mathcal{A}^C}\mu_{\mathcal{Z}_I}(\rho) = 0_{|\mathcal{A}^C|}$$

nam daje

$$\rho s_{\mathcal{A}^C}(\rho) = X_{:,\mathcal{A}^C}^T (y - X_{:,\mathcal{A}}\beta_{\mathcal{A}}(\rho)) - A_{:,\mathcal{A}^C}^T \lambda(\rho) - C_{\mathcal{Z}_I,\mathcal{A}^C}\mu_{\mathcal{Z}_I}(\rho).$$

Iz toga slijedi da se $\rho s_{\mathcal{A}^C}$ kreće linearno na putu po

$$\frac{d}{d\rho}\left[\rho s_{\mathcal{A}^C}\right] = -\begin{pmatrix} X_{:,\mathcal{A}}^T X_{\mathcal{A}^C} \\ A_{:,\mathcal{A}^C} \\ C_{\mathcal{Z}_I,\mathcal{A}^C}^T. \end{pmatrix} \frac{d}{d\rho}\begin{pmatrix} \beta_{\mathcal{A}} \\ \lambda \\ \mu_{\mathcal{Z}_I} \end{pmatrix} \tag{2}$$

Rezidual nejednakosti $r_{\mathcal{Z}_I^C} = C_{\mathcal{Z}_I^C,\mathcal{A}}\beta_{\mathcal{A}} - d_{\mathcal{Z}_I^C}$ se također kreće linearno s gradijentom

$$\frac{d}{d\rho} r_{\mathcal{Z}_I^C} = C_{\mathcal{Z}_I^C,\mathcal{A}} \frac{d}{d\rho}\beta_{\mathcal{A}}. \tag{3}$$

Jednadžbe (1), (2) i (3) se koriste za praćenje promjena skupova $\mathcal{A}$ i $\mathcal{Z}_I$ koji mogu rezultirati s točkama loma na putu rješenja. Obzirom da je put rješenja po dijelovima linearan, samo moramo odrediti točke loma na putu, dok se ostatak može interpolirati. Put se prati u smjeru od ρ_{max} do $\rho = 0$. Neka je $\beta^{(t)}$ rješenje kod točke loma t, tada je sljedeća točka loma $t + 1$ dobivena uz najmanji $\Delta\rho > 0$, koji je dobiven uz pomoć odgovarajućih uvjeta za praćenje puta. Također, potrebno je i paziti da subgradijent $s_j(\rho)$ bude zadovoljen duž puta kako bi put rješenja bio dobro definiran. Pojavljuje se problem kada neaktivni koeficijenti na granici intervala subgradijenta napreduju presporo duž puta, tako da bi njihov subgradijent mogao izaći iz intervala $[-1, 1]$ na sljedećoj točki loma $t + 1$. Kako bi se taj problem riješio, ako neaktivni koeficijent β_j, $j \in \mathcal{A}^c$, sa subgradijentom $s_j = \pm 1$ napreduje presporo, koeficijent se premješta u aktivni skup $\mathcal{A}$ i jednadžba (1) se ponovno izračunava prije nego što se algoritam nastavi.

Inicijalizacija

Obzirom da se put prati prema smjeru spuštanja, za početni parametar je potrebno uzeti maksimalnu vrijednost ρ_{max}. Kad $\rho \to \infty$, problem ograničeni LASSO postaje

$$\begin{aligned} &\min_{\beta} \|\beta\|_1 , \\ &\text{uz uvjete } A\beta = b, \\ &\qquad C\beta \leq d. \end{aligned} \tag{4}$$

Prvo rješimo taj problem kako bi dobili inicijalno rješenje β^0 i odgovarajuće skupove $\mathcal{A}$ i $\mathcal{Z}_I$, kao i početne vrijednosti Lagrangeovih multiplikatora λ^0 i μ^0. Put počinje s vrijednosti

$$\rho_{max} = \max \left| x_j^T \left(y - X\beta^0 \right) - A_{:,j}^T \lambda^0 - C_{\mathcal{Z}_I,j}^T \mu_{\mathcal{Z}_I}^T \right|,$$

dok je subgradijent postavljen pomoću vrijednosti $s(\rho)$ i $\rho s_{\mathcal{A}^C}(\rho)$. Ovaj pristup može ne uspjeti u slučaju kada imamo nejedinstveno rješenje za (4). Npr., za ograničeni LASSO gdje vrijedi $\sum_j \beta_j = 1$, bilo koji elementarni vektor e_j, zadovoljava ograničenje te ima najmanju L_1 normu, te dobivamo više rješenja.

Završetak

Definiramo stupanj slobode funkcije kao

$$df(g) = E\left[\sum_{i=1}^{n} \frac{\partial g_i}{\partial y_i} \right], \tag{5}$$

gdje je E očekivanje, a g neprekidna i skoro derivabilna funkcija, u našem slučaju za problem LASSO koristimo $g(y) = \hat{y} = X\hat{\beta}$. Kako bi koristili (5), moramo pretpostaviti da je y $N(\mu, \sigma^2 I)$, tj. da je y normalno distribuirana. Kao i ranije, također pretpostavljamo da obje matrice ograničenja, A i C, imaju pun rang po redcima, a X ima pun rang po stupcima. Zatim, uz $D = I$, za fiksnu vrijednost $\rho \geq 0$, stupnjevi slobode dani su sa:

$$df(X\hat{\beta}(\rho)) = E[|\mathcal{A}| - (q + |\mathcal{Z}_I|)],$$

gdje je E očekivanje, $|\mathcal{A}|$ broj aktivnih ograničenja, q broj ograničenja jednakosti, a $|\mathcal{Z}_I|$ broj nejednakosnih ograničenja na granici.

Nepristrani procjenitelj za stupnjeve slobode tada je $|\mathcal{A}| - (q + |\mathcal{Z}_I|)$. Stupnjevi slobode počinju kao broj aktivnih prediktora, a zatim se gubi jedan stupanj slobode za svako ograničenje jednakosti i svako ograničenje nejednakosti. Kada nema ograničenja, jednadžba (4) postaje klasični LASSO problem sa stupnjevima slobode jednakim $|\mathcal{A}|$.

Stupnjevi slobode koriste se prilikom implementacije algoritma puta za prekidanje puta kada stupnjevi slobode dosegnu n.

Poglavlje 3

Implementacija

U ovom odjeljku ćemo objasniti implementaciju ADMM algoritma za rješavanje problema ograničenog LASSOa. Za implentaciju algoritma je korišten programski jezik C++, te opisujemo strukturu funkcija koje implementiraju algoritam, korištenje biblioteke Eigen za rad s elementima linerne algebre, te programsku implementaciju pojedinih dijelova algoritma.

Glavna funkcija za računanje je funkcija *constrainedADMM*. Funkcija je tipa *void*, to jest nema povratnu vrijednost, nego se parametar beta koji predstavlja optimizirano rješenje prenosi po referenci u funkciju. Deklaracija te funkcije je:

```
1 void constrainedADMM(const Eigen::MatrixXd& X, const Eigen::VectorXd&
    y, double rho, const Eigen::MatrixXd& A, const Eigen::VectorXd& b,
    const Eigen::MatrixXd& C, const Eigen::VectorXd& d,
    Eigen::VectorXd& beta, double tau, int maxIter, double tol);
```

Funkcija prima sljedeće parametre:

- matricu podataka X,
- vektor y, koji sadrži vrijednosti za promatranja, iz matrice X
- parametar ρ, koji određuje regularizaciju regresije, te je istovjetan s parametrom λ u definiciji problema ograničenog LASSOa,
- matricu za uvjet jednakosti A,
- vektor za uvjet jednakosti b,
- matricu za uvjet nejednakosti C,
- vektor za uvjet nejednakosti d,

- parametar τ za snagu regularizacije, veći τ pojačava regularizaciju i čini rješenje otpornim na promjene u podacima,
- varijablu *maxIter* koja predstavlja maksimalan broj iteracija koje će funkcija provesti,
- varijablu *tol* koja predstavlja toleranciju za provjeru konvergencije.

Inicijaliziramo varijable: m koja predstavlja broj stupaca matrice X, z koja se koristi kao kopija vektora β te se koristi kao pomoćna varijabla, te u koja se koristi za provjere konvergencija rješenja:

```
1     int m = X.cols();
2     Eigen::VectorXd z = beta;
3     Eigen::VectorXd u = Eigen::VectorXd::Zero(m);
```

Gornji kod odgovara sljedećem dijelu pseudokoda:

1: $\beta^{(0)} = z^{(0)} = \beta^0, u^{(0)} = 0, \tau > 0$

Nakon toga ulazimo u petlju čiji je broj iteracija ograničen s varijablom *maxIter*. Prvi dio koda u petlji predstavlja sljedeći dio pseudokoda algoritma:

1: $\beta^{(t+1)} = \min \frac{1}{2} \|y - X\beta\|_2^2 + \frac{\tau}{2} \left\|\beta + z^{(t)} + u^{(t)}\right\|_2^2 + \rho \|\beta\|_1$

U C++-u je taj dio realiziran kao:

```
1     Eigen::VectorXd q = z - u;
2     Eigen::MatrixXd I = Eigen::MatrixXd::Identity(m, m);
3     Eigen::VectorXd nextBeta = (X.transpose() * X + tau *
      I).ldlt().solve(X.transpose() * y + tau * q);
4     nextBeta = softThresholding(nextBeta, rho / tau);
```

Vektor q predstavlja korigirane vrijednosti za trenutačnu iteraciju, koje će se koristiti za ažuriranje β. Vektor *nextBeta* dobivamo koristeći biblioteku Eigen, u ovom slučaju koristimo metodu najmanjih kvadrata s regularizacijom.

Nakon izračuna *nextBeta* vrijednosti, koristimo funkciju $softThresholding$ definiranu kao

```
1     Eigen::VectorXd softThresholding(const Eigen::VectorXd& x, double
      lambda) {
2     return x.array().sign() * (x.array().abs() - lambda).max(0.0);
3 }
```

Razlog korištenja te funkcije je postavljanje nekih od koeficijenata vektora *nextBeta* na vrijednost 0 kako bi dobili rijedak vektor. Razlog zašto želimo rijedak vektor je to što L_1 regularizacija koja se koristi u problemu LASSO preferira za rješenje rjeđi vektor.

Prethodni koraci algoritma mogu proizvesti rješenje koje ne zadovoljava uvjete jednakosti i/ili nejednakosti te zato projiciramo vektora β na skup definiran u pomoć tih uvjeta. Cilj projekcije na skup je vrijednost vektora rješenja β dobivenog u trenutnoj iteraciji svesti na vektor koji zadovoljava dane uvjete.

Stoga, implementiramo dio algoritma koji predstavlja projekciju na skup C dan kao $C = \{\beta \in \mathbb{R}^m : A\beta = b, C\beta \leq d\}$:

```
1     Eigen::VectorXd projectToSet(const Eigen::VectorXd& beta, const
    Eigen::MatrixXd& A, const Eigen::VectorXd& b, const
    Eigen::MatrixXd& C, const Eigen::VectorXd& d)
```

U funkciji *projectToSet* se na početku namještaju parametri poput tolerancije na greške, vektora koji predstavlja projekciju i maksimalnog broja iteracija. Većina radnje funkcije se obavlja unutar petlje čiji broj iteracija je ograničen s lokalnom varijablom *maxIter*. Projekcija na $A\beta = b$ je realizirana na sljedeći način:

```
1     nextBeta -= A.transpose() * (A * nextBeta - b);
```

Oduzimanjem vrijednosti $A^T(A\beta - b)$ od trenutne vrijednosti β se korigira β tako da bolje zadovoljava ograničenje jednakosti.

Projekcija na $C\beta \leq d$ je složenija:

```
1     for (int i = 0; i < C.rows(); ++i) {
2             double scalingFactor = (C.row(i) * nextBeta - d(i)) /
    C.row(i).squaredNorm();
3             if (scalingFactor > 0.0) {
4                 nextBeta -= scalingFactor * C.row(i).transpose();
5             }
6         }
```

Prolaskom petlje kroz svaki redak matrice C, provjerava se vrijedi li ograničenje nejednakosti. Varijabla *scalingFactor* služi sa provjeru ukoliko $C_i\beta > d_i$. Ukoliko to ograničenje nije zadovoljeno, β se korigira kako bi se taj uvjet zadovoljio.

Na kraju svake iteracije imamo provjeru

```
1     (nextBeta - projection).norm() < tolerance
```

Ukoliko je taj uvjet zadovoljen, projekcija je dovoljno konvergirala unutar dane tolerancije, vektor *projection* tražanja projekcija te funkcija vraća vrijednost vektora *projection*.

Nakon računanja projekcije, ostaje dio algoritma dan s:

gdje se staroj vrijednosti varijable u pridodaju vrijednosti varijabli *nextBeta* i *nextZ*.

Konačno, preostaje provjera konvergencije:

1: $u^{(t+1)} = u^{(t)} + \beta^{(t+1)} + z^{(t+1)}$

```
1     if (primalResidualNorm < tol * betaNorm && dualResidualNorm < tol *
      rho * z.norm()) {
2             beta = nextBeta;
3             z = nextZ;
4             break;
5         }
```

U vektor *r* spremamo primarni, a u vektor *s* dualni rezidual te računamo njihove norme koje spremamo u varijable *primalResidualNorm* i *dualResidualNorm*, te normu vektora *beta*. Konačno, ukoliko je provjera uspješna, *beta* je rješenje problema ograničenog LASSOa koje zadovoljava danu toleranciju te je s time provođenje algoritma završeno.

3.1 Provjera točnosti algoritma

Nakon implementacije, potrebno je provjeriti točnost algoritma. Generiramo rijetki vektor β^*, i postavimo neke od elemenata na ne-nul vrijednosti. Nadalje, kreiramo matricu X s nasumničnim elementima, te vektor $y = X \cdot \beta^*$. Matricu A postavljamo matricu jedinica s jednim redom, a vektor b na broj 0, te tako dobivamo uvjet da je suma koeficijenata u vektoru β jednaka 0. Nadalje, matricu C kreiramo kao negativnu jediničnu matricu, a vektor d kao vektor kojem je svaki element jednak -0.5, te tako dobivamo uvjet $\beta_i \geq -0.5$. Nakon namještanja parametara $\tau, \rho, maxIter$ i tol, pozivamo funkciju *constrainedADMM* te dobivamo

```
Originalni betaStar:
 0.5    0 -0.5    0    0    0    0    0    0    0
Rekonstruirani beta:
 0.496375         0 -0.496381         0         0         0         0         0         0         0
```

Slika 3.1: Testiranje algoritma

te je s time potvrđeno da algoritam radi jer smo rekonstruirali vektor te $\beta_i \geq -0,5$ i $\sum \beta_i = -0.000006$.

Poglavlje 4

Testiranje na podacima

U ovom poglavlju ćemo iskoristiti implementaciju ADMM algoritma u C++-u kako bi rješili neke od problema iz stvarnog svijeta.

4.1 Simulirani podaci

Za prvo testiranje implementacije ADMM algoritma ćemo simulirati podatke, te ćemo koristi ograničenje dano s $\sum_{j=1}^{m} \beta_j = 0$. Iz toga slijedi da su ograničenja samo tipa jednakosti, tj.

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix} \beta = 0,$$

pa je $A = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}, b = 0, d = 0, C = 0.$

Implementacija

Kreiramo vektor *beta* tako da je prva četvrtina vektora nule, druga jedinice, treća nule, te četvrta -1. Zatim nasumično stvorimo vrijednosti za matricu X i vektor y. Inicijaliziramo matrice A i C, te vektore b i d kako je objašnjeno:

```
1    Eigen::MatrixXd A = Eigen::MatrixXd::Ones(1, m);
2
3    Eigen::VectorXd b = Eigen::VectorXd::Zero(1);
4    Eigen::MatrixXd C = Eigen::MatrixXd::Zero(n, n);
5    Eigen::VectorXd d = Eigen::VectorXd::Zero(n);
```

Nakon postavljanja parametara te poziva funkcije *constrainedADMM*, dobivamo za nasumične podatke

$$\beta = (-0.912781, 0, ..., 0, -0.0496585)^T,$$

te ispis ”Suma koeficijenata od beta: 2.0029e-05”, s čime je uvjet $\sum_{j=1}^{m} \beta_j = 0$ zadovoljen.

4.2 Globalno zatopljenje

Podaci koji se koriste prikazuju promjene prosječnih godišnjih temperatura od od 1850. do 2015. godine, u odnosu na prosječnu godišnju temperaturu od 1961. do 1990. godine. Ozbirom da se iz podataka može vidjeti trend rasta, nakon što to zapažanje inkorporiramo u regresijski problem, dobivamo

$$\min_{\beta} \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|\beta\|_1 , \tag{1.2}$$
$$\text{uz uvjete } \beta_1 \leq ... \leq \beta_n.$$

Stoga je matrica C dana kao

$$C = \begin{pmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & & \ddots & \ddots & \\ & & & & 1 & -1 \\ & & & & & 1 \end{pmatrix},$$

te vektor $d = \mathbf{0}$.

Implementacija

Matricu X i vektor y pripremamo uz funkciju *prepareMatrices*:

```
    void prepareMatrices(const std::vector<WarmingData>& data,
   Eigen::MatrixXd& X, Eigen::VectorXd& y) {
    int numSamples = data.size();
    X.resize(numSamples, 2);
    y.resize(numSamples);

    for (int i = 0; i < numSamples; ++i) {
        X(i, 0) = 1.0;
        X(i, 1) = data[i].year;
        y(i) = data[i].annual;
    }
}
```

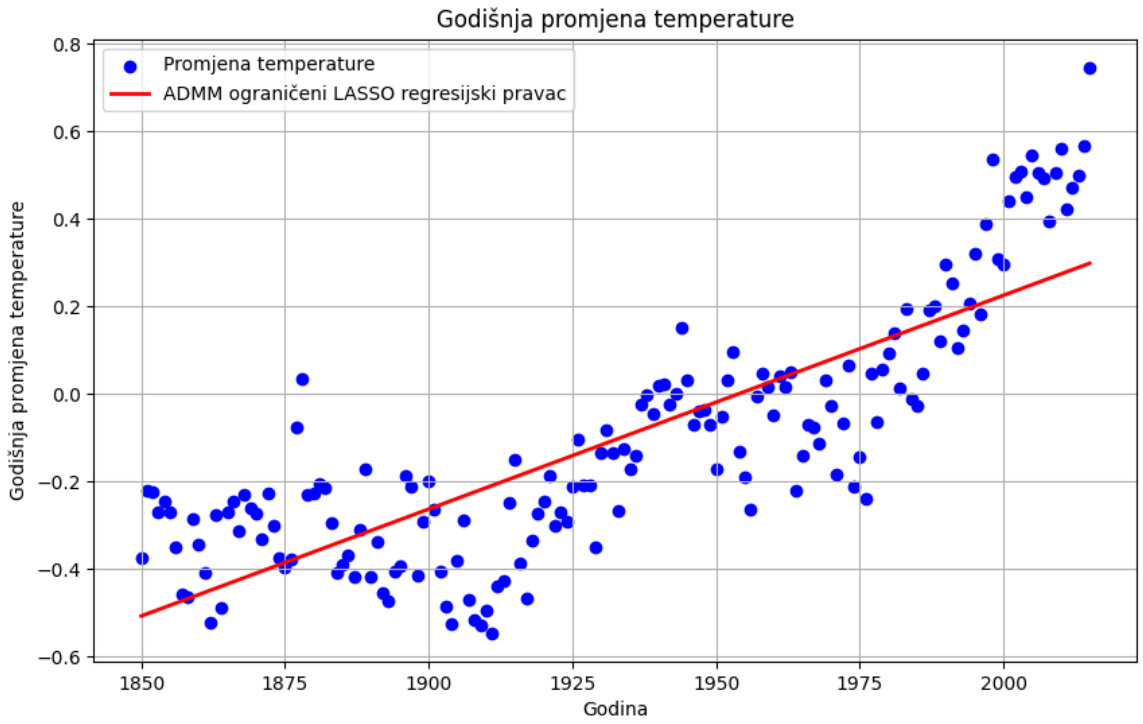
Matricu C i vektor d inicijaliziramo kako je objašnjeno ranije u tekstu:

```
    C.setZero();
    for (int i = 0; i < n-1; ++i) {
        C(i, i) = 1;
        C(i, i+1) = -1;
    }
    d.setZero();
```

Obzirom da nema ograničenja jednakosti, nije potrebno inicijalizirati te uvjete. Nakon zvanja funkcije zbog prijenosa po referenci u varijabli *beta* je spremljeno rješenje problema ograničenog LASSOa.

```
1    constrainedADMM(X, y, rho, Eigen::MatrixXd::Zero(n, n),
    Eigen::VectorXd::Zero(n), C, d, beta, tau, maxIter, tol);
```

Nakon eksportiranja rezultata u csv datoteku, koristeći vizualizacijski softver prikazujemo rezultat izvedbe algoritma. Rezultati su prikazani na slici 4.1. Vidimo da regresijski pravac poprilično dobro aproksimira podatke.



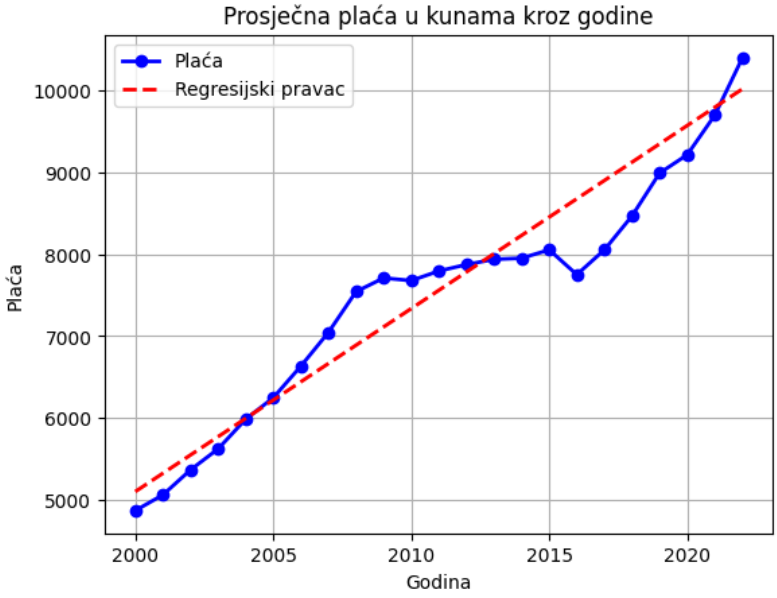
Slika 4.1: ADMM globalno zatopljenje

4.3 Prosječna plaća kroz godine - LASSO bez ograničenja

Podaci koje koristimo predstavljaju kretanje prosječne hrvatske bruto plaće u kunama od 2000. godine do 2022. godine. Obzirom da je u tom razdoblju bila recesija, nije korektno kao u slučaju globalnog zatopljenja pretpostaviti trend rasta. Zato će se u ovom slučaju koristiti LASSO bez ograničenja. Priprema matrica je identična kao i u slučaju testiranja podataka globalnog zatopljenja, ali se funkcija *constrainedADMM* poziva na sljedeći način:

```
1    constrainedADMM(X, y, rho, Eigen::MatrixXd::Zero(n, n),
   Eigen::VectorXd::Zero(n), C, d, beta, tau, maxIter, tol);
```

Nakon eksportiranja rezultata u csv datoteku, koristeći vizualizacijski softver prikazujemo rezultat izvedbe algoritma. Rezultati su prikazani na slici 4.2. Vidimo da regresijski pravac poprilično dobro aproksimira podatke.



Slika 4.2: ADMM prosječna plaća

4.4 Tumor na mozgu

CGH, comparative genomic hybridization, je tehnika koja se koristi za analizu i kvantifikaciju genetskih promjena u genomu. Koristi se za otkrivanje amplifikacija (povećanje broja kopija) i deleta (gubitak kopija) gena u tumorima, što može pomoći pri promatranju mutacija koje doprinose razvoju tumora. CGH se koristi za usporedbu količine DNK u referentnim stanicama, te se na taj način prepoznaju mutacije koji uzrokuju promjene u tumorskim stanicama.

Skup podataka sadrži CGH mjerenja multiformnog glioblastoma (GBM), jednog od najagresivnijih tipova tumora na mozgu. Tibshirani i Wang (2008) su predložili korištenje jedne verzije LASSO problema za aproksimaciju CGH vrijednosti. Spomenuti LASSO problem, zvan sparse fused LASSO, je definiran kao

$$\min_{\beta} \frac{1}{2}\|y-\beta\|_2^2 + \rho_1\|\beta\|_1 + \rho_2 \sum_{j=2}^{p} |\beta_j - \beta_{j-1}|$$

Sparse fused LASSO je specijalan tip generaliziranog LASSOa, gdje je matrica D dana kao

$$D = \begin{pmatrix} 1 & -1 & & & & & \\ & 1 & -1 & & & & \\ & & 1 & -1 & & & \\ & & & \ddots & \ddots & & \\ & & & & & 1 & -1 \\ 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ & & & & & & 1 \end{pmatrix} \in \mathbb{R}^{(2m-1)\times m}.$$

Prema lemi 1.2.5 možemo generalizirani LASSO svesti na ograničeni LASSO, te ćemo za određivanje vektora β koristiti implementirani ADMM algoritam.

Implementacija

Inicijaliziramo matrice X i D, te učitavamo podatke u vektor y:

```
1    Eigen::VectorXd y = readData(filePath);
2    int n = y.size();
3    int m = n;
4
```

```
    Eigen::MatrixXd X = Eigen::MatrixXd::Identity(n, n);

    Eigen::MatrixXd D(2*(m - 1), m);
    D.setZero();

    for (int i = 0; i < m - 1; i++) {
        D(i, i) = 1;
        D(i, i + 1) = -1;
        D(i + (m - 1), i) = 1;
    }
```

Koristeći postupak iz leme 1.2.5, pripremamo podatke za oblik pogodan za implementirani ADMM algoritam, postupak se obavlja u sljedećoj funkciji:

```
Eigen::VectorXd solveGeneralizedLASSO(const Eigen::MatrixXd& X_tilde,
    const Eigen::VectorXd& Y, double lambda, const Eigen::MatrixXd& D)
```

Prvo dekomponiramo matricu D:

```
    int n = Y.size();
    int m = X_tilde.cols();

    Eigen::MatrixXd D1 = D.topRows(m);
    Eigen::MatrixXd D2 = D.bottomRows(n - m);
```

Zatim pripremimo matrice A, C i X, te vektore β, b, d:

```
    Eigen::MatrixXd A = D2 * D1.inverse();
    Eigen::MatrixXd C = Eigen::MatrixXd::Identity(D2.rows(), D2.rows());

    Eigen::MatrixXd X = Eigen::MatrixXd::Zero(n, m);
    X.block(0, 0, n, m) = X_tilde * D1.inverse();

    Eigen::VectorXd beta = Eigen::VectorXd::Zero(m);

    Eigen::VectorXd b = Eigen::VectorXd::Zero(D2.rows());
    Eigen::VectorXd d = Eigen::VectorXd::Zero(D2.rows());
```

Nakon prilagodbe parametara te poziva funkcije *constrainedADMM*, dobivamo vektor rješenja

$$\beta = (0,0,0,0,0,0,0,0,0,0,-0.547671,0,...,0,0,0.144189,0,0,0,0,0,0)^T$$

Vektor β je dimenzije 1×1000, a sadrži 96 ne-nul vrijednosti. Nul vrijednosti predstavljaju gene u kojima nije primjećena značajnija promjena u broju DNK, pozitivne vrijednosti predstavljaju amplifikacije gena, dok negativne vrijednosti predstavljaju delecije gena.

Bibliografija

[1] PICARD, Franck, et al. A statistical approach for array CGH data analysis. BMC bioinformatics, 2005, 6: 1-14.

[2] GAINES, Brian R.; KIM, Juhyun; ZHOU, Hua. Algorithms for fitting the constrained lasso. Journal of Computational and Graphical Statistics, 2018, 27.4: 861-871.

[3] GAINES, Brian R.; KIM, Juhyun; ZHOU, Hua. Algorithms for fitting the constrained lasso. Supplemental material `https://www.tandfonline.com/doi/suppl/10.1080/10618600.2018.1473777`

[4] Državni zavod za statistiku, `https://dzs.gov.hr/` (prosinac 2024.)

[5] Eigen, `https://eigen.tuxfamily.org/` (rujan 2024.)

[6] BUNEA, Florentina. Honest variable selection in linear and logistic regression models via L_1 and $L_1 + L_2$ penalization. 2008.

[7] GORDON, Geoff; TIBSHIRANI, Ryan. Karush-kuhn-tucker conditions. Optimization, 2012, 10.725/36: 725.

[8] RANSTAM, Jonas; COOK, Jonathan A. LASSO regression. Journal of British Surgery, 2018, 105.10: 1348-1348.

[9] SANTOSA, Fadil; SYMES, William W. Linear inversion of band-limited reflection seismograms. SIAM journal on scientific and statistical computing, 1986, 7.4: 1307-1330.

[10] JAMES, Gareth M.; PAULSON, Courtney; RUSMEVICHIENTONG, Paat. Penalized and constrained optimization: an application to high-dimensional website advertising. Journal of the American Statistical Association, 2020.

[11] NOCEDAL, Jorge; WRIGHT, Stephen J. Quadratic programming. Numerical optimization, 2006, 448-492.

[12] GELETU, Abebe. Quadratic programming problems-a review on algorithms and applications (Active-set and interior point methods). Ilmenau University of Technology, 2015.

[13] TIBSHIRANI, Robert. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B: Statistical Methodology, 1996, 58.1: 267-288.

[14] SHINAWI, Marwan; CHEUNG, Sau Wai. The array CGH and its clinical applications. Drug discovery today, 2008, 13.17-18: 760-770.

[15] JAMES, Gareth M.; PAULSON, Courtney; RUSMEVICHIENTONG, Paat. The constrained lasso. In: Refereed Conference Proceedings. 2012. p. 4945-4950.

[16] ROTH, Volker. The generalized LASSO. IEEE transactions on neural networks, 2004, 15.1: 16-28.

[17] TIBSHIRANI, Ryan J. The lasso problem and uniqueness. 2013.

Dodatak A

Kod implementacije ADMM algoritma

U ovom dodatku dajemo kod implementacije ADMM algoritma za rješavanje problema ograničenog LASSOa. Datoteka ”admm.h” sadrži definicije funkcija:

```
#include <Eigen/Dense>
#include <cmath>
#include <iostream>

Eigen::VectorXd softThresholding(const Eigen::VectorXd& x, double
    lambda);

Eigen::VectorXd projectToSet(const Eigen::VectorXd& beta, const
    Eigen::MatrixXd& A, const Eigen::VectorXd& b, const
    Eigen::MatrixXd& C, const Eigen::VectorXd& d);

void constrainedADMM(const Eigen::MatrixXd& X, const Eigen::VectorXd&
    y, double rho, const Eigen::MatrixXd& A, const Eigen::VectorXd& b,
    const Eigen::MatrixXd& C, const Eigen::VectorXd& d,
    Eigen::VectorXd& beta, double tau, int maxIter, double tol);
```

Algoritam je u potpunosti implementiran u datoteci *admm.cpp*:

```
#include "admm.h"

Eigen::VectorXd softThresholding(const Eigen::VectorXd& x, double
    lambda) {
    return x.array().sign() * (x.array().abs() - lambda).max(0.0);
}

// Projekcija na skupove ogranicenja
Eigen::VectorXd projectToSet(const Eigen::VectorXd& beta, const
    Eigen::MatrixXd& A, const Eigen::VectorXd& b, const
    Eigen::MatrixXd& C, const Eigen::VectorXd& d) {

```

```
10      Eigen::VectorXd projection = beta;
11
12      bool converged = false;
13      int maxIter = 10000;
14      double tolerance = 1e-6;
15      double stepSize = 1; // Za potencijalno skaliranje koraka, potrebno
    za zero sum probleme
16
17      for (int iter = 0; iter < maxIter; ++iter) {
18          Eigen::VectorXd nextBeta = projection;
19
20          // Projekcija za uvjet jednakosti
21          nextBeta -= stepSize * A.transpose() * (A * nextBeta - b);
22
23          // Projekcija za uvjet nejednakosti
24          for (int i = 0; i < C.rows(); ++i) {
25              double scalingFactor;
26              if(C.row(i).squaredNorm() != 0)
27                  scalingFactor = (C.row(i) * nextBeta - d(i)) /
    C.row(i).squaredNorm();
28              else
29                  scalingFactor = 0;
30              if (scalingFactor > 0.0) {
31                  nextBeta -= scalingFactor * C.row(i).transpose();
32              }
33          }
34
35          if ((nextBeta - projection).norm() < tolerance) {
36              converged = true;
37              break;
38          }
39
40          projection = nextBeta;
41
42          if (projection.norm() > 1e6) {
43              stepSize *= 0.5;
44          }
45      }
46
47      if (!converged) {
48          std::cerr << "Neuspjesna projekcija na skupove ogranicenja!" <<
    std::endl;
49      }
50
51      return projection;
52  }
53
```

```
54
55
56 void constrainedADMM(const Eigen::MatrixXd& X, const Eigen::VectorXd&
       y, double rho, const Eigen::MatrixXd& A, const Eigen::VectorXd& b,
       const Eigen::MatrixXd& C, const Eigen::VectorXd& d,
       Eigen::VectorXd& beta, double tau, int maxIter, double tol) {
57     int m = X.cols();
58
59     //(1)
60     Eigen::VectorXd z = beta;
61     Eigen::VectorXd u = Eigen::VectorXd::Zero(m);
62
63     for (int t = 0; t < maxIter; ++t) {
64         Eigen::VectorXd q = z - u;
65         Eigen::MatrixXd I = Eigen::MatrixXd::Identity(m, m);
66
67         //(2)
68         Eigen::VectorXd nextBeta = (X.transpose() * X + tau *
       I).ldlt().solve(X.transpose() * y + tau * q);
69         nextBeta = softThresholding(nextBeta, rho / tau);
70
71         //(3)
72         Eigen::VectorXd nextZ = projectToSet(nextBeta + u, A, b, C, d);
73
74         //(4)
75         u += nextBeta + nextZ;
76
77         //Provjere konvergencije
78         Eigen::VectorXd r = nextBeta - nextZ;  // Primalni rezidual
79         Eigen::VectorXd s = rho * (nextZ - z);  // Dualni rezidual
80
81         double primalResidualNorm = r.norm();
82         double dualResidualNorm = s.norm();
83         double betaNorm = beta.norm();
84
85         //Provjera zadovoljenosti dane tolerancije
86         if (primalResidualNorm < tol * betaNorm && dualResidualNorm <
       tol * rho * z.norm()) {
87             beta = nextBeta;
88             z = nextZ;
89             break;
90         }
91
92         beta = nextBeta;
93         z = nextZ;
94     }
95 }
```

U kodu komentari predstavljaju sljedeće dijelove algoritma:

- (1): $\beta(0) = z^{(0)} = \beta^0, u^{(0)} = 0, \tau > 0$
- (2): $\beta^{(t+1)} = \min \frac{1}{2} ||y - X\beta||_2^2 + \frac{\tau}{2} \left\|\beta + z^{(t)} + u^{(t)}\right\|_2^2 + \rho ||\beta||_1$
- (3): $z^{(t+1)} = \text{proj}_C \left(\beta^{(t+1)} + u^{(t)}\right)$
- (4): $u^{(t+1)} = u^{(t)} + \beta^{(t+1)} + z^{(t+1)}$

Sažetak

Glavni cilj ovog rada je bio dati uvod u LASSO metodu regresijske analize, preciznije LASSO s ograničenjima, objasniti matematičku pozadinu metode te implementirati algoritam za programsko rješavanje problema koji koriste LASSO s ograničenjima. U prvom poglavlju smo iznijeli osnovne matematičke pojmove i pozadinu, dok smo u drugom poglavlju objasnili neke od algoritama za rješavanje problema ograničenog LASSOa. U trećem poglavlju objašnjavamo programsku implementaciju algoritma. Konačno, u četvrtom poglavlju prikazujemo primjenu algoritma na podacima iz stvarnoga svijeta.

Summary

The main goal of this thesis was to provide an introduction to the LASSO method of regression analysis, more precisely LASSO with constraints, to explain the mathematical background of the method, and to implement an algorithm that will solve problems that use LASSO with constraints. In the first chapter we presented the basic mathematical concepts and background, while in the second chapter, we explained some of the algorithms for solving the constrained LASSO problem. In the third chapter, we explained the software implementation of the algorithm. Finally, in the fourth chapter, we present the application of the algorithm to real-world data.

Životopis

Rođen sam 27.10.1998. u Zagrebu. Završio sam Osnovnu školu Ivan Benković u Dugom Selu, Osnovnu glazbenu školu Dugo Selo za instrumente tamburu i kontrabas, te Srednju školu Dugo Selo. 2017. godine upisujem Prirodoslovno-matematički fakultet, prediplomski studij, smjer matematika. 2022. godine upisujem diplomski studij Računarstvo i matematika na istom fakultetu. Tokom studija sam radio kao software developer u Ericsson Nikola Tesli, priravnik u upravljanju rizicima u Croatia Osiguranju, dok se u jesen 2024. godine zapošljavam kao software comissioning engineer u tvrtci Knapp Hrvatska d.o.o.