Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matej Klančar

Algoritmi za reševanje problema matričnih napolnitev

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Aljaž Zalar

Ljubljana, 2023

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Kandidat: Matej Klančar
Naslov: Naslov diplomskega dela
Vrsta naloge: Diplomska naloga na univerzitetnem programu prve stopnje Računalništvo in informatika
Mentor: doc. dr. Aljaž Zalar

Opis:
Besedilo teme diplomskega dela študent prepiše iz študijskega informacijskega sistema, kamor ga je vnesel mentor. V nekaj stavkih bo opisal, kaj pričakuje od kandidatovega diplomskega dela. Kaj so cilji, kakšne metode naj uporabi, morda bo zapisal tudi ključno literaturo.

Title: Algorithms for solving matrix completion problem

Description:
opis diplome v angleščini

Na tem mestu zapišite, komu se zahvaljujete za pomoč pri izdelavi diplomske naloge oziroma pri vašem študiju nasploh. Pazite, da ne boste koga pozabili. Utegnil vam bo zameriti. Temu se da izogniti tako, da celotno zahvalo izpustite.

Svoji dragi Alenčici.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CA	classification accuracy	klasifikacijska točnost
DBMS	database management system	sistem za upravljanje podatkovnih baz
SVM	support vector machine	metoda podpornih vektorjev

Povzetek

Naslov: Algoritmi za reševanje problema matričnih napolnitev

Avtor: Matej Klančar

V vzorcu je predstavljen postopek priprave diplomskega dela z uporabo okolja LaTeX. Vaš povzetek mora sicer vsebovati približno 100 besed, ta tukaj je odločno prekratek. Dober povzetek vključuje: (1) kratek opis obravnavanega problema, (2) kratek opis vašega pristopa za reševanje tega problema in (3) (najbolj uspešen) rezultat ali prispevek diplomske naloge.

Ključne besede: računalnik, računalnik, računalnik.

Abstract

Title: Diploma thesis template

Author: Matej Klančar

This sample document presents an approach to typesetting your BSc thesis using LaTeX. A proper abstract should contain around 100 words which makes this one way too short.

Keywords: computer, computer, computer.

Poglavje 1

Uvod

1.1 Motivacija

Problem matričnih napolnitev sprejme matriko, največkat označeno z M, pri kateri so nekateri elementi označeni kot neznani. Problem nato sprašuje po vrednostih, ki jih lahko vstavimo v neznane vrednosti, tako da bo rang matrike najmanjši možen. Gre za NP-poln problem, zato ga poskušamo poenostaviti, ter reševati lažje probleme, ki vrnejo dovolj dobre, a ne optimalne rešitve.

Problem je v zadnjih letih zelo popularen, z njim pa se ukvarjajo tako številni matematiki kot računalničarji. Njegova splošnost naredi reševanje problema na številnih področjih, sam pa se v diplomski nalogi osredotočim na razreševanje neznanih pikslov v slikah. Prav tako omenjam in preizkusim algoritem na priporočilnih sistemih. Rezultate teh predstavim v poglavju X.

V tej diplomski nalogi bom predstavil par algoritmov, ki rešujejo omenjen problem. Algoritmi so bili izbrani glede na njihovo popularnost in priznanost v literaturi. Prav tako sem poskrbel, da so algoritmi primerno različni in temeljijo na drugačnih principih. Algoritme sem tudi implementiral, nato pa še napravil analizo ter opisal ugotovitve v poglavju X.

1.2 Cilji

Nekaj o tem, kaj želimo narediti.

Glavni prispevki tega diplomskega dela so ...

1.3 Struktura diplomskega dela

V poglavju 2 ,,,, v poglavju 3 ....

Poglavje 2

Pregled področja

Področje je trenutno zelo aktivno, z številnimi raziskovalci, ki se specializirajo na dano področje. Eden vodilnih raziskovalcev je Emmanuel J. Candès, na čigar dela se tekom diplomske naloge večkrat sklicujem.

Vodilna literatura tekom pisanja diplomske naloge je bil članek [**survey**], ki opisuje problem, ter mnoge algoritme. Medtem ko opisi pogosto niso bili dovolj podrobni, da bi lahko začel algoritme implementirati, je članek ponujal dobro razumljive opise algoritmov, kot tudi navedel vire, ki so pomagali pri implementaciji. V kasnejših poglavjih se tam, kjer sem članke potreboval, nanje sklicujem.

Poglavje 3

Algoritmi

Na koncu napisati kratek odstavek o tem, kaj je vsebina poglavja.

3.1 Pomembe definicije

Nekatere definicije so uporabljene čez več algoritmov. Z namenom preglednosti, te opisujem v tem poglavju

1. Ω je definirana množica znanih vrednosti
2. $$[\mathcal{P}_\Omega]_{i,j} = \begin{cases} a_{ij} & (i,j) \in \Omega \\ 0 & \text{drugače} \end{cases}$$
3. Operator $\mathcal{D}_\tau$ kot

   $$\mathcal{D}_\tau(A) := U\mathcal{D}_\tau(\Sigma)V^T, \quad \mathcal{D}_\tau(\Sigma) = diag(max(\sigma_i - \tau, 0))$$

   [1]
4. Z oznako $M \in \mathbb{R}^{n_1 \times n_2}$ označujemo vhodno matriko, torej tisto, ki ima nekatere podatke neznane.

3.2 Minimizacija nuklearne norme

Ker je minimizacija ranga matrike NP-poln problem, so se razvile druge metode, ki samo kompleksnost problema zmanjšujejo. Minimizacija nuklearne norme (Nuclear Norm Minimization) se zanaša na dejstvo, da je rang matrike povezan z nuklearno normo matrike. Ta je definirana kot

$$||X||_* = \sum_{i=i}^{n} \sigma_i(X)$$

se lahko sklicujem?

Dokazano je bilo, da nuklearna norma predstavlja konveksno ovojnico ranga. [2] Konveksna ovojnica funkcije $f : \mathcal{C} \to \mathbb{R}$ je največja konveksna funkcija g tako da velja $f(x) \geq g(x)$ za vse $x \in \mathcal{C}$. [4]

Minimizacijo nuklearne norme je možno pretvoriti v semidefinitni problem, ki ga lahko rešujemo z različnimi pripomočki, na primer SeDuMi [5].

Standardna oblika semidefinitnega programa je

$$\begin{aligned} \min_{Y} \quad & \langle C, Y \rangle \\ \text{tako da} \quad & \langle A_k, Y \rangle = b_k, \quad k = 1, \ldots, l \\ & Y \succeq 0 \end{aligned}$$

Kjer je C dana matrika, $\{A_k\}$ in $\{b_k\}$ pa množici matrik in vektorjev.

Problem minimizacije nuklearne norme lahko zapišemo kot

$$\begin{aligned} \min_{X,t} \quad & t \\ \text{tako da} \quad & ||X||_* \leq t, \\ & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned}$$

se lahko sklicujem?

Trdimo, da za matriko $X \in \mathbb{R}^{n_1 \times n_2}$ in $t \in \mathbb{R}$ velja $||X||_* \leq t \iff \exists W_1 \in \mathbb{R}^{n_1 \times n_1}, W_2 \in \mathbb{R}^{n_2 \times n_2}$ tako da [2]

$$Y = \begin{bmatrix} W_1 & X \\ X^T & W_2 \end{bmatrix},$$
$$Y \succeq 0, \ \ \mathrm{Tr}(Y) \leq 2t$$

Minimizacijski problem lahko tako redefiniramo kot

$$\begin{aligned} \min_{Y,t} \quad & 2t \\ \text{tako da} \quad & Y \succeq 0, \\ & \langle Y, A_{a,b} \rangle = b_{a,b} \end{aligned}$$

kjer je $A_{a,b} \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$ matrika v množici A. Velja $A_{a,b} \in A \iff (a,b) \in \Omega$. Matrika $A_{a,b}$ ima vse elemente ničelne, razen na mestu (a, n_1+b), kjer ima vrednost 1. Podobno je definiran tudi $b_{a,b} \in \mathbb{R}$, ki ima vrednost $M_{a,b}$. Ker velja

$$\langle A, B \rangle = \sum_{i}^{n_1} \sum_{j}^{n_2} a_{i,j} b_{i,j}$$

je lahko videti, da je tak pogoj smiselen. Programi za reševanje SDP pa lahko tako obliko že sprejmejo. [4]

3.3 Prag singularnih vrednosti

Algoritem praga singularnih vrednosti, oziroma v nadaljevanju SVT (Singular Value Thresholding) sloni na dejstvu, da imamo pri matrikah z majhnim rangom nekaj velikih singularnih vrednosti, ostale pa blizu ničli. Za svoje delovanje uvede dva nova pomembna koncepta, prvi je premik, drugi pa prag, potreben za uporabo operatorja $\mathcal{D}_\tau$. Algoritem lahko na kratko povzamemo z zapisom

$$\begin{cases} X^k = \mathcal{D}_\tau(Y^{k-1}) \\ Y^k = Y^{k-1} + \delta_k \mathcal{P}_\Omega(M - X^k) \end{cases}$$

kjer $\tau > 0$ predstavlja prag, δ_k predstavlja k-ti premik, $X \in \mathbb{R}^{n_1 \times n_2}$ ter $Y^0 = 0 \in \mathbb{R}^{n_1 \times n_2}$. [1]

Smiselnost algoritma lahko pokažemo s pomočjo Lagrangeovega multiplikatorja. Ponovno rešujemo minimizacijski problem, le da dodamo dodatne parametre in s tem minimizacijo omilimo.

Uvedimo funkcijo $f_\tau(X) = \tau||X||_* + \frac{1}{2}||X||_F^2$. Problem lahko tako zapišemo kot

$$\begin{aligned} \min \quad & f_\tau(X) \\ \text{tako da} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned} \tag{3.1}$$

Lahko je videti, da za velike vrednosti τ velja $f_\tau(X) \approx \tau||X||_*$, kar pokaže, da bo s primerno velikim τ, algoritem res skoraj minimiziral nuklearno normo.

Lagrangeov multiplikator je definiran kot $\mathcal{L}(x, \lambda) = f(x) + \lambda \cdot g(x)$, kjer je $f(x)$ funkcija, ki jo minimiziramo, pod pogojem da velja $g(x) = 0$. Naš problem tako prevedemo v

$$\mathcal{L}(X, Y) = f_\tau(X) + \langle Y, \mathcal{P}_\Omega(M - X)\rangle$$

S pomočjo tako imenovanega Uzawoega algoritma, pa lahko problem pretvorimo v iterativni algoritem. [1]

$$\begin{cases} X^k = \arg\min_X \mathcal{L}(X^k, Y^{k-1}) \\ Y^k = Y^{k-1} + \delta_k \mathcal{P}_\Omega(M - X^k) \end{cases}$$

Za reševanje minimizacijskega problema dokažimo teorem

$$\mathcal{D}_\tau(Y) = \arg\min_X \{\frac{1}{2}||X - Y||_F^2 + \tau||X||_*\} \tag{3.2}$$

Ker je $h(X) := \frac{1}{2}||X - Y||_F^2 + \tau||X||_*$ strogo konveksna funkcija, lahko za subgradient Z v točki X_0 rečemo, da velja $\forall X : f(X) \geq f(X_0) + \langle Z, X - X_0\rangle$. Ali drugače povedano, vse točke na vseh tangentah na funckijo $h(X)$ bodo pod ali na funkciji $h(X)$. To velja po sami definiciji, saj je to zahtevan pogoj subgradienta v neki točki.

Pri iskanju minimuma torej iščemo tako točko X', da bo subgradient v točki X' enak 0. Problem sedaj zapišemo kot $0 \in X' - Y + \tau\partial||X'||_*$. Izkaže se, da je množica subgradientov nuklearne norme definirana kot

$$\partial||X||_* = \{UV^* + W : W \in \mathbb{R}^{n_1 \times n_2}, U^*W = 0, WV = 0, ||W||_2 \leq 1\}.$$

kjer $U\Sigma V^T$ predstavlja SVD razcep matrike X. [1]

Cilj dokaza je pokazati, da velja $X' = \mathcal{D}_\tau(Y)$ Najprej razčlenimo SVD razcep matrike Y kot

$$Y = U_0\Sigma_0V_0^T + U_1\Sigma_1V_1^T$$

, kjer U_0, Σ_0 in V_0 predstavljajo lastne vrednosti ter njihove lastne vektorje večje od τ, U_1, Σ_1 in V_1 pa tiste manjše od τ. Ustrezno je torej pokazati, da velja

$$X' = U_0(\Sigma_0 - \tau I)V_0^T$$

Gre preprosto za drugačen zapis operatorja $\mathcal{D}_\tau(Y)$. Če zapis vstavimo v prejšnji podan pogoj dobimo

$$\begin{aligned} 0 &= X' - Y + \tau\partial||X||_* \\ Y - X' &= \tau(U_0V_0^T + W) \end{aligned}$$

primerna izbira za $W = \tau^{-1}U_1\Sigma_1V_1^T$, saj

$$\begin{aligned} Y - X' &= U_0\Sigma_0V_0^T + U_1\Sigma_1V_1^T - U_0(\Sigma_0 - \tau I)V_0^T = \\ &= U_0V_0^T(\Sigma_0 - \Sigma_0 + \tau I) + U_1\Sigma_1V_1^T = \\ &= \tau U_0V_0^T + U_1\Sigma_1V_1^T \end{aligned}$$

in

$$\begin{aligned} \tau(U_0V_0^T + W) &= \tau(U_0V_0^T + \tau^{-1}U_1\Sigma_1V_1^T) \\ &= \tau U_0V_0^T + U_1\Sigma_1V_1^T \end{aligned}$$

Sedaj je zgolj potrebno pokazati, da veljajo potrebne lastnosti matrike W. Po sami definiciji SVD vemo, da so vsi stolpci matrik U in V ortogonalni. Torej velja $U_0^TW = 0$ in $WV_0 = 0$. Ker pa ima matrika Σ_1 vse elemente manjše od τ velja tudi $||W||_2 \leq 1$. S tem smo pokazali, da $Y - X' \in \tau\partial||X'||_*$.

Prav tako lahko sedaj zapišemo algoritem kot [1]

$$\begin{cases} X^k = \mathcal{D}_\tau(Y^{k-1}) \\ Y^k = Y^{k-1} + \delta_k\mathcal{P}_\Omega(M - X^k) \end{cases}$$

3.3.1 Nastavljanje parametrov

Medtem, ko so koraki v samem algoritmu definirani kot množica korakov, sem sam za premik uporabljal konstanto, ter korak nastavil na

$$\delta = 1.2\,\frac{n_1 n_2}{m}$$

po priporočilih [1].

Prav tako članek navaja, da je za matrike velikosti $\mathbb{R}^{n\times n}$ smiselno nastaviti $\tau = 5n$, vendar sem v moji implementaciji zaradi posploševanja na nekvadratne matrike, za matrike velikosti $\mathbb{R}^{n_1\times n_2}$ parameter nastavil na

$$\tau = 5\,\frac{n_1 + n_2}{2}$$

Medtem ko so taki parametri dobri za večje matrike, jih ne smemo uporabljati kot definitivno najboljše vrednosti. Med mojimi testiranji sem ugotovil, da je vrednost premika velikokrat treba zmanjšati, posebno za manjše matrike z več neznanimi vrednostmi, saj drugače program ni konvergiral. Prav tako, se je večkrat zgodilo, da je bil pridobljen rezultat še vedno zelo zašumljen. Takrat je bilo smiselno prag τ povečati. To je sicer upočasnilo program, vendar izboljšalo rezultat.

3.4 Minimizacija prirezane nuklearne norme

Že samo ime nam pove, da bo algoritem minimizacije prirezane nuklearne norme, oziroma TNNM (Truncated Nuclear Norm Minimization) podoben algoritmu NNM. Vendar tu privzamemo, da imamo o samem algoritmu še neko dodatno informacijo $r \in \mathbb{N}$, ki je povezana z rangom originalne, nezašumljene matrike.

Sam algoritem uvede tako imenovano skrajšano nuklearno normo, ki je za matriko $X \in \mathbb{R}^{n_1\times n_2}$ definirana kot

$$||X||_r = \sum_{i=r+1}^{\min(n_1,n_2)} \delta_i(X)$$

torej vsoto $\min(n_1, n_2) - r$ najmanjših singularnih vrednosti, ter z njeno pomočjo definira problem [3]

$$\begin{aligned} \min_X \quad & ||X||_r \\ \text{tako da} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned}$$

Cilj algoritma je torej čim bolj zmanjšati najmanjše singularne vrednosti, medtem ko velikih ne omejujemo. S tem problem minimizacije omilimo.

Za reševanje minimizacije bomo uporabljali algoritem ADMM, vendar moramo prej sam problem nekoliko spremeniti. Razpišimo problem kot

$$\begin{aligned} \min_X \quad & ||X||_* - \sum_{i=1}^{r} \sigma_i \\ \text{tako da} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned}$$

.

Za nadaljne korake bomo potrebovali teorem

$$\mathrm{Tr}(AXB^T) \leq \sum_{i=1}^{r}$$

kjer velja $X \in \mathbb{R}^{n_1 \times n_2}$, $A \in \mathbb{R}^{r \times n_1}$, $B \in \mathbb{R}^{r \times n_2}$ in $r \in \mathbb{N}, r \leq min(n_1, n_2)$, kot tudi $AA^T = I_r, BB^T = I_r$.

Za dokaz uporabljamo Von Neumannovo neenakost sledi, s katero lahko zapišemo

$$\mathrm{Tr}(AXB^T) = \mathrm{Tr}(XB^TA) \leq \sum_{i=1}^{\min(n_1,n_2)} \sigma_i(X)\sigma_i(B^TA)$$

Enakost $\mathrm{Tr}(AXB^T) = \mathrm{Tr}(XB^TA)$ sledi iz dejstva, da je sled produkta matrik invariantna pod cikličnimi permutacijami.

Po definiciji lahko singularne vrednosti matrike Y najdemo tako, da najdemo korene nenegativnih lastnih vrednosti matrike Y^TY. Tako lahko povemo, da so singularne vrednosti matrike B^TA enake lastnim vrednostim matrike A^TBB^TA. Izraz razpišemo v

$$A^TBB^TA = A^TI_rA = A^TA$$

Ker pa velja, da imata matriki XY in YX enake neničelne lastne vrednosti, ter vemo da $AA^T = I_r$, lahko povemo, da ima produkt B^TA r singularnih vrednosti enakih 1, saj ima I_n n lastnih vrednosti enakih 1. Tako lahko sedaj razpišemo izraz

$$\sum_{i=1}^{\min(n_1,n_2)} \sigma_i(X)\sigma_i(B^TA) = \sum_{i=1}^{r} \sigma_i(X)$$

Ugotovili smo, da velja

$$\mathrm{Tr}(AXB^T) \leq \sum_{i=1}^{r} \sigma_i(Xs) \tag{3.3}$$

Če definiramo SVD razcep $X = U\Sigma V^T$, ter matriki $A = (u_1, \ldots, u_r)^T$ in $B = (v_1, \ldots, v_r)^T$, kjer je u_i i-ti stolpec matrike U ter v_i i-ti stolpec matrike V, lahko pokažemo da velja $\mathrm{Tr}(AXB^T) = \sum_{i=1}^{r} \sigma_i(X)$.

$$\begin{aligned}
\mathrm{Tr}(AXB^T) &= \mathrm{Tr}((u_1, \ldots, u_r)^T X (u_1, \ldots, u_r)^T) = \\
&= \mathrm{Tr}((u_1, \ldots, u_r)^T U\Sigma V^T (u_1, \ldots, u_r)^T) = \\
&= \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \Sigma \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} = \\
&= \mathrm{Tr}\left(\begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \right) = \sum_{i=1}^{r} \sigma_i(X)
\end{aligned} \tag{3.4}$$

Z združitvijo dokazov 3.3 in 3.4 zapišemo

$$\max_{AA^T=I, BB^T=I} \mathrm{Tr}(AXB^T) = \sum_{i=1}^{r} \sigma_i(X)$$

Torej je problem, ki ga minimizarmo lahko podan kot

$$\begin{aligned}
\min_{X} \quad & ||X||_* - \max_{AA^T=I, BB^T=I} \mathrm{Tr}(AXB^T) \\
\text{tako da} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M)
\end{aligned}$$

Glede na vse ugotovitve, nato nastavimo iterativni algoritem, tako da, izračunamo $X^0 = \mathcal{P}_\Omega(M)$. V i-ti iteraciji izračunamo A^i in B^i, tako da izračunamo SVD razcep $X^i = U\Sigma V^T$, ter A nastavimo kot prvih r stolpcev matrike U, B pa kot prvih r stolpcev matrike V. X^{i+1} lahko sedaj izračunamo kot [3]

$$\begin{aligned} \min_X \quad & ||X||_* - \mathrm{Tr}(A^i X (B^i)^T) \\ \text{tako da} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned}$$

Če minimizacijo še malo obrnemo, pa lahko problem rešujemo z uporabo algoritma ADMM.

iz kje pride izracun Y^{i+1}

$$\begin{aligned} \min_X \quad & ||X||_* - \mathrm{Tr}(A^i W (B^i)^T) \\ \text{tako da} \quad & W = X, \;\; \mathcal{P}_\Omega(W) = \mathcal{P}_\Omega(M) \end{aligned}$$

Ta problem ponovno zapišemo s pomočjo Lagrangeove funkcije, le da algoritem ADMM definira še kazenski parameter β. [3]

kazenski parameter?

$$\mathcal{L}(X, Y, W, \beta) = ||X||_* - Tr(A_l W B_l^T) + \frac{\beta}{2}||X - W||_F^2 + Tr(Y^T(X - W))$$

Opazimo lahko, da funkcija definira zgolj pogoj $X = W$. Videli bomo, da algoritem pogoj $\mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M)$ definira posredno, s popravljanjem znanih vrednosti. 3.5

Matriko X^{k+1} ponovno definiramo kot

$$X^{k+1} = \arg\min_X \mathcal{L}(X, Y^k, W^k, \beta)$$

Z ignoriranjem konstantnih členov, pa lahko tako zapišemo

$$\begin{aligned} X^{k+1} &= \arg\min_X ||X||_* + \frac{\beta}{2}\langle X - W^k, X - W^k\rangle + \frac{\beta}{2}\langle\frac{2}{\beta}Y, X\rangle = \\ &= \arg\min_X ||X||_* + \frac{\beta}{2}\mathrm{Tr}(X^T X - X^T W^k - W^{kT} X - W^{kT} W^k + \frac{2}{\beta}Y^{kT} X) \end{aligned}$$

Z namenom faktorizacije dodamo konstantne člene.

$$\begin{aligned}
&\arg\min_X ||X||_* + \frac{\beta}{2}\mathrm{Tr}(X^T(X - W^k + \frac{1}{\beta}Y^k) - W^{k^T}(X - W^k + \frac{1}{\beta}Y^k) \\
&+ \frac{1}{\beta}Y^{k^T}(X - W^k + \frac{1}{\beta}Y^k)) = \\
&= \arg\min_X ||X||_* + \frac{\beta}{2}||X - W^k + \frac{1}{\beta}Y^k||_F^2 = \\
&= \arg\min_X \frac{1}{\beta}||X||_* + \frac{1}{2}||X - (W^k - \frac{1}{\beta}Y^k)||_F^2
\end{aligned}$$

Po teoremu 3.2 pa tako lahko zapišemo

$$X^{k+1} = \mathcal{D}_{\frac{1}{\beta}}(W_k - \frac{1}{\beta}Y_k)$$

Matriko W podobno izračunamo kot

$$\begin{aligned}
W^{k+1} &= \arg\min_W \mathcal{L}(X^{k+1}, Y^k, W, \beta) \\
&= \arg\min_W \frac{\beta}{2}||W - (X^{k+1} + \frac{1}{\beta}(A_l^T B_l + Y_k))||_F^2
\end{aligned}$$

Lahko je videti, da velja

$$W^{k+1} = X^{k+1} + \frac{1}{\beta}(A_l^T B_l + Y_k)$$

zaradi pogoja $\mathcal{P}_\Omega(W) = \mathcal{P}_\Omega(M)$ pa tiste elemente, ki poznamo, popravimo.

malo spremenjeno iz clanka [3]

$$W_{k+1} = W_{k+1} + \mathcal{P}_\Omega(M - W_{k+1}) \tag{3.5}$$

3.5 Izmenjajoč gradientni spust

Ker je računanje SVD razcepa zahtevna operacija, saj ima časovno zahtevnost $O(n^3)$, je bilo predlaganih nekaj algoritmov, ki za svoje delovanje ne potrebujejo SVD-ja. Algoritem Izmenjajočega gradientnega spusta, oziroma v nadaljevnu ASD (Alternating Steepest Descent) sloni na računanju gradienta in premikanju po njem. Glavni cilj algoritma je, najti dve matriki $X \in \mathbb{R}^{n_1 \times r}$ ter $Y \in \mathbb{R}^{r \times n_2}$, tako da bo veljalo $M = XY$. Vidimo lahko,

da ponovno potrebujemo informacijo o rangu matrike, ki jo rekonstruiramo. Ker imata tako X in Y kvečjemu rang r, vemo, da tudi njun produkt XY ne bo imel ranga večjega od r.

Cilj algoritma je minimizirati

$$\min_{X,Y} \quad \frac{1}{2} ||\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY)||_F^2$$

Algoritem minimizacijo razdeli na dve, nato pa izmenično rešuje eno in nato drugo kot

$$X_{i+1} = \arg\min_X \quad ||\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY_i)||_F^2$$
$$Y_{i+1} = \arg\min_Y \quad ||\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(X_{i+1}Y)||_F^2$$

[6]

Za uporabo algoritma ASD potrebujemo odvoda funkcije $f(X,Y) = \frac{1}{2} ||\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY)||_F^2$, ki ga izračunamo za vsak element posebej.

$$\begin{aligned}
\frac{\partial}{\partial x_{a,b}} f &= \frac{\partial}{\partial x_{a,b}} \frac{1}{2} ||\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY)||_F^2 = \\
&= \frac{\partial}{\partial x_{a,b}} \frac{1}{2} \sum_i^{n_1} \sum_j^{n_2} (\delta_{i,j} m_{i,j} - \delta_{i,j} \sum_k^r (x_{i,k} y_{k,j}))^2 = \\
&= \sum_j^{n_2} (\delta_{a,j} m_{a,j} - \delta_{i,j} \sum_k^r (x_{a,k} y_{k,j}))(-y_{b,j}) \implies \\
\implies \frac{\partial}{\partial X} f &= -(\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY)) Y^T
\end{aligned}$$

kjer

$$\delta_{i,j} = \begin{cases} 1, (i,j) \in \Omega \\ 0, (i,j) \notin \Omega \end{cases}$$

Na podoben način bi lahko pokazali tudi

$$\frac{\partial}{\partial Y} = -X^T(\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(XY))$$

Poiščimo še najboljši korak gradientnega spusta. Cilj je pokazati, da je premik po gradientu s korakom t_x najboljši. Najprej definirajmo sam premik, kot

$$X^{k+1} = X^k - t_X \nabla f_Y(X)$$

Ker želimo, da bodo znane vrednosti produkta $X^{k+1}Y^k$ kar se da podobne znanim vrednostim matrike M, nastavimo t_x kot

$$t_x = \arg\min_t \quad g(t)$$

kjer

$$g(t) = \frac{1}{2}||\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega((X - t\nabla f_Y(X))Y)||_F^2$$

$$g(t) = \frac{1}{2}||\mathcal{P}_\Omega(M - XY)||_F^2 - t\mathrm{Tr}(\mathcal{P}_\Omega(M - XY)\mathcal{P}_\Omega(\nabla f_Y(X)Y)^T) + \frac{t^2}{2}||\mathcal{P}_\Omega(\nabla f_Y(X)Y)||_F^2$$

ali lahko tako?

Ker je [6]

$$g'(t) = -||\nabla f_Y(X)||_F^2 + t||\mathcal{P}_\Omega(f_Y(X)Y)||_F^2$$

vidimo, da funkcija $g(t)$ doseže minimum pri

$$t_x = \frac{||\nabla f_Y(X)||_F^2}{||\mathcal{P}_\Omega(\nabla f_Y(X)Y)||_F^2}$$

Podobno velja za korak v smeri gradientnega spusta matrike Y, kjer

$$t_y = \frac{||\nabla f_X(Y)||_F^2}{||\mathcal{P}_\Omega(X\nabla f_X(Y))||_F^2}$$

3.6 LMaFit

LMaFit je algoritem, ki podobno kot ASD, rešuje problem matričnih napolnitev z uporabo dveh manjših matrik $X^{n_1 \times r}$ in $Y^{n_1 \times r}$. Podobno kot v prejšnjih algoritmih rešujemo problem

$$\begin{aligned} \min_{X,Y,Z} \quad & \frac{1}{2}||XY - Z||_F^2 \\ \text{tako da} \quad & \mathcal{P}_\Omega(M) = \mathcal{P}_\Omega(Z) \end{aligned}$$

le da sam problem rešujemo s pomočjo Moore-Penrose inverza (Označen z †).

Lahko je videti, da bo funkcija $f(X, Y, Z) = frac12||XY - Z||_F^2$ imela najmanjšo vrednost, ko bo $XY = Z$. Zato uvedemo iterativni algoritem, ki

izmenično posodablja X, Y in Z tako, da fiksira dve izmed matrik. Minimizacijo matrik X in Y torej dosežemo na naslednji način.

$$X^{i+1}Y^i = Z^i$$
$$X^{i+1} = Z^i(Y^i)^\dagger$$

Kjer se zanašamo na dejstvo, da nam množenje z Moore-Penrose inverzom z desne da najboljši možen rezultat.

Podobno posodobimo matriko Y.

$$X^{i+1}Y^{i+1} = Z^i$$
$$Y^{i+1} = (X^{i+1})^\dagger Z^i$$

Sedaj le še posodobimo matriko Z, kot

$$Z^{i+1} = X^{i+1}Y^{i+1} + \mathcal{P}_\Omega(M - X^{i+1}Y^{i+1})$$

Torej podobno kot prej poskušamo doseči $XY = Z$, vendar zaradi omejitve znanih vrednosti matrike M, na tistem mestu vrednosti popravimo. [7]

Poglavje 4

Rezultati

V tem poglavju bom opisal rezultate, ki sem jih pridobil. Kot sem že omenil, bo večji del preizkušanja programa opravljen na slikah, kjer bodo nekateri piksli manjkali. Gre za problem, ki ga je moč lepo vizualizirati, saj pogosto pri surovih podatkih ni lahko definirati njihovo točnost, zaradi česar težko interpretiramo, kako koristen je sam algoritem.

Prav tako bom opisal točnost rezulatov različnih metod kot tudi čas izvajanja posameznih metod. Probleme bom zagnal tudi na različnih vrst podatkov, npr. podatkih ki so generirani normalno kot tudi enakomerno porazdeljeno.

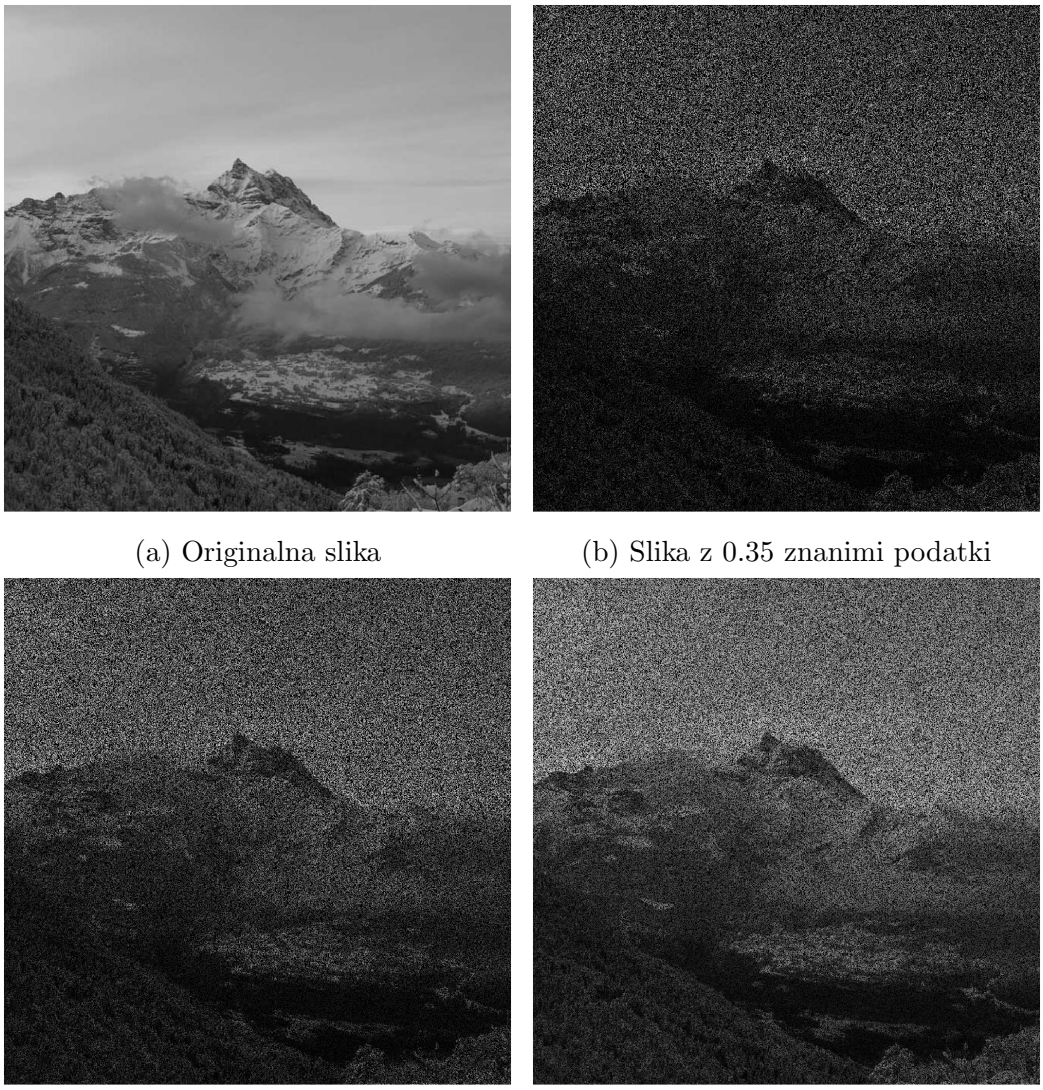
Poglavje bom začel pisati, ko končam z vsemi implementacijami, pri delu pa si bom pomagal z orodjem Microsoft Excel, kjer bom lahko napake in čas izvajanja tudi grafično vizualiziral.

Zaradi interpretacije, sem slike razdelil v več skupin, za katere opišemo ugotovitve.

4.1 Velika črno-bela slika

Same teste algoritmov najprej poženemo na veliki, črno-beli sliki, velikosti 1000×1000 pikslov. Taka izbira je smiselna, iz vidika, da imamo dovolj podatkov, potrebnih za rekonstrukcijo. Ker je časovna zahtevnost algoritmov

pri večjih slikah, kot bomo videli v nadaljevanju že precej velika, nam ta faza testiranja služi kot preverjanje delovanja samih algoritmov. Same podrobnosti razlik med rezultati si bomo zato podrobneje pogledali na manjših slikah v nadaljevanju. Algoritme preizkušamo trikrat, na podatkih z deleži znanih vrednosti $0.35, 0.45$ in 0.6.



(a) Originalna slika (b) Slika z 0.35 znanimi podatki

(c) Slika z 0.45 znanimi podatki (d) Slika z 0.60 znanimi podatki

Slika 4.1: Vir slike: Unsplash

Kot vidimo, je med rezultati velika razlika. Očitno je, da algoritma

TNNM in SVT delujeta najbolje, medtem ko imata algoritma ADDM in LMaFit vprašljive rezultate. Te si lahko interpretiramo kot posledica slabosti, da lahko algoritma končata v lokalnem optimumu. Zato je ta algoritma smiselno uporabljati, kadar ima dober začeten približek matrik X in Y. Algoritem NNM med rezultati manjka, saj je zaradi velikega števila matrik, potrebnih za definicijo omejitev, algoritem preveč prostorsko kompleksen. Ta algoritem bomo zato obravnavali posebej.

Poglavje 5

Zaključek

Ko bodo vsa prejšnja poglavja končana, se bom lahko lotil pisanja zaključka. V njem bom opisal kaj vse sem tekom pisanja diplomske naloge ugotovil ter opravil, ter povedal če sem z rezultati zadovoljen. Omenil bom tudi kaj bi lahko spremenil in kako bi algoritme nadgradil, ter omenil par idej, na katerih raziskovalci trenutno delajo, ter opisal kako se področje razvija.

Literatura

[1] Jian-Feng Cai, Emmanuel J. Candes in Zuowei Shen. *A Singular Value Thresholding Algorithm for Matrix Completion*. 2008. arXiv: 0810.3286 [math.OC].

[2] Maryam Fazel. "Matrix rank minimization with applications". PhD dezertacija. Stanford, CA: Stanford University, mar. 2002.

[3] Yao Hu in sod. "Fast and Accurate Matrix Completion via Truncated Nuclear Norm Regularization". V: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.9 (2013), str. 2117–2130. DOI: 10.1109/TPAMI.2012.271.

[4] Luong Trung Nguyen, Junhan Kim in Byonghyo Shim. "Low-Rank Matrix Completion: A Contemporary Survey". V: *IEEE Access* 7 (2019), str. 94215–94237. DOI: 10.1109/ACCESS.2019.2928130.

[5] Jos F. Sturm. "Using SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones". V: *Optimization Methods and Software* 11.1-4 (1999), str. 625–653. DOI: 10.1080/10556789908805766. URL: https://doi.org/10.1080/10556789908805766.

[6] Jared Tanner in Ke Wei. "Low rank matrix completion by alternating steepest descent methods". V: *Applied and Computational Harmonic Analysis* 40.2 (2016), str. 417–429. ISSN: 1063-5203. DOI: https://doi.org/10.1016/j.acha.2015.08.003. URL: https://www.sciencedirect.com/science/article/pii/S1063520315001062.

[7] Zaiwen Wen, Wotao Yin in Yin Zhang. "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm". V: *Mathematical Programming Computation* 4 (dec. 2012). DOI: 10.1007/s12532-012-0044-1.