Tenzorske dekompozicije u prepoznavanju akcije

Šantek, Matija

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Matija Šantek

TENZORSKE DEKOMPOZICIJE U PREPOZNAVANJU AKCIJE

Diplomski rad

Voditelj rada:
Prof. dr. sc. Zlatko Drmač

Zagreb, srpanj 2023.

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. _____

Mojoj Mari, bratu Marinu i cijeloj obitelji

Sadržaj

Uvod

U posljednjim godinama dolazi do velikog interesa i razvoja umjetne inteligencije. Porast se može pripisati brojnim čimbenicima, uključujući otkrića u arhitekturama dubokog učenja, dostupnost golemih količina podataka i do jačanja računalnih resursa. Nadalje, pojava naprednog jezičnog modela, specifično ChatGPT-a, značajno doprinosi popularizaciji samog područja.

Jedno od područja unutar umjetne inteligencije je računalni vid, koji pokušava omogućiti strojevima da razumiju i interpretiraju vizualne informacije, poput ljudi.

Računalni vid je uveo velike napretke u različita područja, uključujući autonomna vozila, nadzorne sustave, robotiku i medicinsko snimanje. Prepoznavanje radnji, jedan od zadataka računalnog vida, ima za cilj automatsku analizu i razumijevanje ljudskih radnji iz slika ili videa. Jedan od izazova leži u viskodimenzionalnoj prirodi podataka. Videozapisi i slike prirodno se mogu predstaviti kao tenzori, koji su višedimenzionalni nizovi sposobni prikazati prostorno-vremenske informacije. Međutim, iskorištavanje tenzorskih struktura i odnosa među njima nije trivijalno. Kako bismo lakše radili s tenzorima velikih dimenzija koristimo tenzorske dekompozicije. Kao što ćemo vidjeti, pomoću tenzorskih dekompozicija možemo izvući bitne informacije te tako znatno smanjiti dimenziju, uz minimalan gubitak informacija.

U prvom poglavlju prvo definiramo neka matrična množenja koja se prirodno javljaju u radu s tenzorima, njihovim dekompozicijama te u obradi slika. Nakon toga uvodimo definiciju tenzora, niti i odsječke tenzora te definiramo osnovne operacije s tenzorima. Na kraju poglavlja je kratki opis SVD-a koji se koristi za dekompoziciju matrica.

U drugom poglavlju opisujemo četiri dekompozicije tenzora. Prva je CP dekompozicija koja tenzor rastavlja na sumu tenzora ranga jedan. Nakon toga opisujemo Tuckerovu dekompoziciju i njezinu specijalnu verziju, a to je HOSVD za koji zapravo možemo reći da je generalizirana verzija SVD na više dimenzija. Na kraju, opisujemo Tenzorski vlak dekompoziciju koja je najmlađa od svih dekompozcija i nastala je kako bi riješila neke od glavnih problema ostalih dekompozicija.

U trećem poglavlju implementiramo Tuckerovu dekompoziciju s *ridge* regesijom te navodimo rezultate testiranja algoritma na *People playing musical instruments (PPMI)* skupu podataka.

Poglavlje 1

Osnovni pojmovi i definicije

U ovom ćemo poglavlju predstaviti temeljne koncepte, definicije i operacije vezane za tenzore koji će nam pomoći za bolje razumijevanje tenzorskih dekompozicija i konačno za razumijevanje kako tenzori prepoznaju akcije.

Uvedimo prvo notaciju:

a	vektor
A	matrica dimenzija $m \times n$
A^T	transponirana matrica A
$A^\dagger$	Moore-Penrose pseudoinverz od A
$\mathcal{A}$	tenzor višeg reda
$\|\cdot\|_F$	Frobenijusova norma
$\|\cdot\|_2$	Euklidska norma
$\langle\cdot,\cdot\rangle$	Skalarni produkt

Prije nego što krenemo navoditi rezultate vezane uz tenzore, navedimo tri definicije različitih matričnih operacija koje će nam kasnije biti od koristi.

1.1 Matrična množenja

U linearnoj algebri, matrica predstavlja neku linearnu transformaciju, pa definicija matričnog množenja zapravo prati tu intuiciju i predstavlja kompoziciju dvije linearne transformacije.

Međutim, zbog raznih drukčijih interpretacija matrica i njihovih kombinacija s tenzorima prirodno se uvodi nova definicija matričnog množenja koji se zove Kronekerov produkt.

Definicija 1.1.1. *Neka su dane dvije matrice $A \in \mathbb{R}^{m_1 \times n_1}$ i $B \in \mathbb{R}^{m_2 \times n_2}$. Kronekerov produkt se označava s $A \otimes B \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}$, a definira se:*

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1m_2}B \\ a_{21}B & a_{22}B & \dots & a_{2m_2}B \\ \vdots & & & \vdots \\ a_{m_1 1}B & a_{m_1 2}B & \dots & a_{m_1 m_2}B \end{bmatrix}.$$

Još jedno množenje matrica je posebno zanimljivo u kontekstu tenzorskih dekompozicija, a to je Khatri-Raov produkt. U idućem poglavlju ćemo imati prilike vidjeti kako se prirodno taj produkt pojavljuje kod CP dekompozicije i u konstrukciji matrica umnožaka Tuckerove dekompozicije.

Definicija 1.1.2. *([8]) Neka su dane dvije matrice $A \in \mathbb{R}^{m_1 \times n}$ i $B \in \mathbb{R}^{m_2 \times n}$. Khatri-Raov produkt matrica A i B je realna matrica, označena s $A \odot B \in \mathbb{R}^{m_1 m_2 \times n}$, a njezini elementi su:*

$$A \odot B = (a_1 \otimes b_1, \dots, a_n \otimes b_n),$$

gdje su vektori $a_k, b_k, k = 1, \dots, n$ stupci matrica A i B redom.

Za kraj još spomenimo Hadamardov produkt koji se dosta često pojavljuje u procesiranju slika gdje želimo primjeniti neki efekt za svaki piksel u slici (npr. smanjiti intezitet, primjeniti neku masku na cijelu sliku, itd.).

Definicija 1.1.3. *([8]) Neka su dane dvije matrice $A, B \in \mathbb{R}^{m \times n}$. Hadamardov produkt se označava kao $A * B$, a definira se:*

$$A * B = \begin{pmatrix} a_{11}b_{11} & \dots & a_{1n}b_{1n} \\ \vdots & & \vdots \\ a_{m1}b_{m1} & \dots & a_{mn}b_{mn} \end{pmatrix}.$$

Navedimo neka svojstva([22], [1]) ovih matričnih umnožaka koja će nam biti od koristi kasnije:

$$(A \otimes B)(C \otimes D) = AC \otimes BD, \tag{1.1}$$

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger, \tag{1.2}$$

$$A \odot B \odot C = (A \odot B) \odot C = A \odot (B \odot C), \tag{1.3}$$

$$(A \odot B)^T (A \odot B) = A^T A * B^T B, \tag{1.4}$$

$$(A \odot B)^\dagger = ((A^T A) * (B^T B))^\dagger (A \odot B)^T. \tag{1.5}$$

1.2 Definicija tenzora i osnovna svojstva

Definicija 1.2.1. *([8]) Tenzor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ je element tenzorskog produkta N vektorskih prostora, gdje svaki ima svoj koordinatni sustav. Red tenzora je broj njegovih dimenzija, također poznatih kao modovi.*

Ukratko, tenzor je višedimenzionalno polje podataka. Tenzor reda jedan je vektor, tenzor reda dva je matrica, itd. Na primjer, videozapis je tenzor reda tri gdje je svaki frame videa jedna slika neke visine i širine.

Niti tenzora

Da bi lakše objasnili nit tenzora u modu k, uvedimo pojam indeksa u tenzorima. Za motivaciju pogledajmo indekse neke proizvoljne matrice

$$A = (a_{ij}),$$

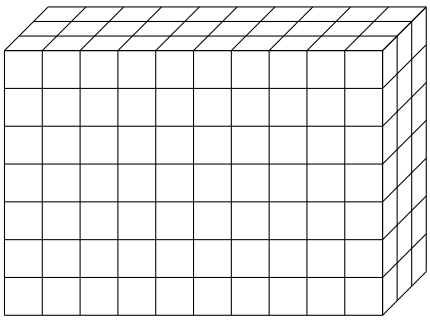
gdje je i indeks retka, a j indeks stupca, još ih nazivamo i 1. i 2. indeks redom. Sada s

$$A(i, :) = (a_{i1}, a_{i2}, a_{i3}, ..., a_{in})$$

označavamo i-ti redak matrice A, dok je

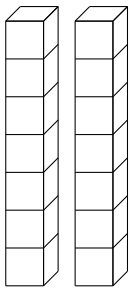
$$A(:, j) = (a_{1j}, a_{2j}, a_{3j}, ..., a_{nj})$$

j-stupac. Ako sada pogledamo tenzore reda tri (radi lakše vizualizacije)

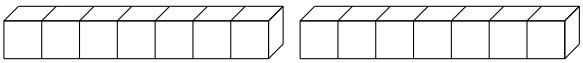


i fiksiramo sve indekse osim prvog:

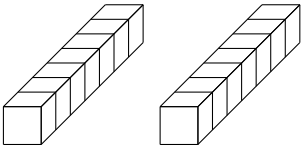
$$A(:, j, k) = (a_{1jk}, a_{2jk}, a_{3jk}, ..., a_{njk}).$$



Kažemo da su to **niti u modu 1** i vidimo da su kod tenzora reda 2 (tj. kod matrica) upravo to stupci. Vidimo da su niti u modu 2 redci:



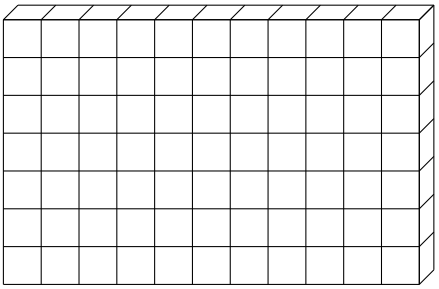
Dok su niti u modu 3:



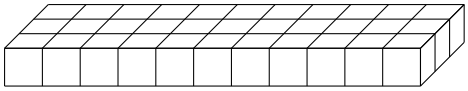
Analogno sad to možemo poopćiti: Fiksiranjem svih indeksa osim k-tog dobivamo nit u modu k.

Odsječak tenzora

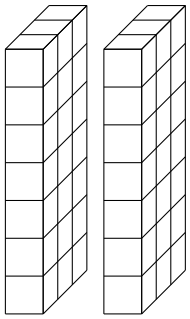
Da bismo dobili odsječak tenzora fiksiramo sve indekse osim dva pa dobivamo odsječke ili *sliceove*.
Frontalni odsječak:



Horizontalni odsječak:



Bočni odsječak:



Osnovne operacije

Često neke objekte iz stvarnog svijeta možemo opisati kao tenzore, kao što je to na primjer video. Tada se prirodno postavlja pitanje kako da uspoređujemo te novonastale tenzore. Tu onda uvodimo definicije norme odnosno skalarnog produkta.

Definicija 1.2.2. *Neka su dani dva tenzora reda N, $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$. Njihov skalarni produkt definiramo kao sumu produkata njihovih elementa, tj:*

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=0}^{d_1} \sum_{i_2=0}^{d_2} \cdots \sum_{i_N=0}^{d_N} a_{i_1,i_2,\ldots,i_N} b_{i_1,i_2,\ldots,i_N}.$$

Normu definiranu tim skalarnim produktom nazivamo Frobenijusova norma. Ona je dana s:

$$\|\mathcal{X}\|_F = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle} = \sqrt{\sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x^2_{i_1 i_2 i_3 \ldots i_N}}.$$

Definicija 1.2.3. *([8]) Neka je dan tenzor N-tog reda, označimo ga s $\mathcal{T} \in \mathbb{R}^{J_1 \times \cdots \times J_{n-1} \times J_n \times J_{n+1} \times \cdots \times J_N}$ i matrica $U \in \mathbb{R}^{I \times J_n}$. Umnožak u modu n tenzora $\mathcal{T}$ i matrice U označujemo s $\mathcal{T} \times_n U \in \mathbb{R}^{J_1 \times \cdots \times J_{n-1} \times I \times J_{n+1} \times \cdots \times J_N}$, a njezini elementi su definirani:*

$$(\mathcal{T} \times_n U)_{j_1,\ldots,j_{n-1},i,j_{n+1},\ldots,j_N} = \sum_{k=1}^{J_n} t_{j_1,j_2,\ldots,j_{n-1},k,j_{n+1},\ldots,j_N} u_{i,k}.$$

Pogledajmo primjer gdje je $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2}$ matrica i imamo matrice $U \in \mathbb{R}^{j \times d_1}$, $V \in \mathbb{R}^{j \times d_2}$, tada je množenje u modu 1 matrice $\mathcal{T}$ s U zapravo definirano kao množenje dvije matrice, dok je množenje u modu 2 matrice $\mathcal{T}$ s V zapravo množenje s matricom s V^T s desna:

$$\mathcal{T} \times_1 U = U\mathcal{T} \quad (UT)(i,j) = \sum_{k=1}^{d_1} u_{j,k} t_{k,i},$$

$$\mathcal{T} \times_2 V = \mathcal{T} V^T \quad \left(TV^T\right)(i,j) = \sum_{i_2=1}^{d_2} t_{i,i_2} v_{j,i_2}.$$

S tako definiranim množenjem u modu n, pokaže se da zadovoljava svojstvo komutativnosti ([8]). Neka je $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_n \times \cdots \times d_m \times \cdots \times d_N}$, $U \in \mathbb{R}^{j \times d_n}$ i $V \in \mathbb{R}^{j \times d_m}$

$$\mathcal{T} \times_n U \times_m V = \mathcal{T} \times_m V \times_n U, \quad \forall m \neq n.$$

Definicija 1.2.4. *([8]) Neka imamo N vektora: $a^{(1)} \in \mathbb{R}^{d_1}, a^{(2)} \in \mathbb{R}^{d_2}, \ldots, a^{(N)} \in \mathbb{R}^{d_N}$. Sada definirajmo njihov vanjski produkt , označavamo ga s $a^{(1)} \circ a^{(2)} \circ \cdots \circ a^{(N)}$, kao tenzor N-tog reda $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ čiji su elementi dani s:*

$$\mathcal{T}(i_1, \ldots, i_N) = a^{(1)}_{i_1} a^{(2)}_{i_2} \ldots a^{(N)}_{i_N}, \quad 1 \leq i_n \leq d_n, \quad za \quad n = 1, \ldots N.$$

Rang tenzora

Definicija 1.2.5. *([19]) Neka je $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$. Tada definiramo rang tenzora $\mathcal{X}$ kao minimalni broj tenzora ranga jedan koji, u linearnoj kombinaciji, daju $\mathcal{X}$ i označavamo ga s $\eta = rang(\mathcal{X})$.*

Rang tenzora predstavlja broj indeksa potrebnih za opisivanje tenzora u određenom koordinatnom sustavu. To je zapravo broj dimenzija tenzorskog prostora kojem tenzor pripada.

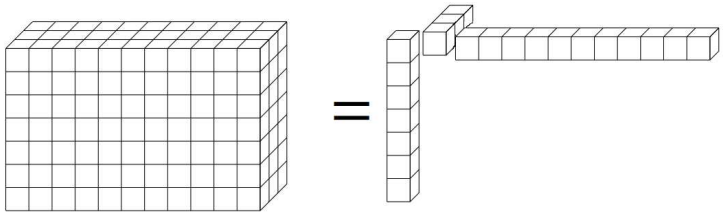
Za primjer pogledajmo tenzor reda tri i ranga jedan (iz 1.2.4):

$$\mathcal{A} = a^{(1)} \circ a^{(2)} \circ a^{(3)}.$$

Određivanje ranga tenzora nije trivijalan problem i čak spada u klasu NP-teških problema, ali iduća napomena nam daje gornju granicu ranga tenzora reda 3.

Napomena 1.2.6. *([8]) Neka je $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ tenzor trećeg reda. Tada vrijedi:*

$$rang(\mathcal{T}) \leq \min\{d_1 d_2, d_1 d_3, d_2, d_3\}.$$



Slika 1.1: Vizualni prikaz tenzora reda tri i ranga jedan

Matrizacija tenzora (*unfold*)

Matrizacija tenzora u n-tom modu je postupak u kojem se niti u n-tom modu poslože u stupce matrice u nekom prije određenom poretku.
Da bi lakše opisali o čemu je riječ pogledajmo primjer tenzora reda tri.

Primjer 1.2.7. *Neka je dan tenzor* $\mathcal{A} \in \mathbb{R}^{2\times3\times3}$. *Njegovi frontalni odsječci su dani s:*

$$A_1 = \begin{pmatrix} 3 & 1 & 4 \\ 1 & 5 & 9 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 2 & 6 & 5 \\ 3 & 5 & 8 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 9 & 7 & 9 \\ 3 & 2 & 4 \end{pmatrix}$$

i njegove matrizacije u sva tri moda:

$$A_{(1)} = \begin{pmatrix} 3 & 1 & 4 & 2 & 6 & 5 & 9 & 7 & 9 \\ 1 & 5 & 9 & 3 & 5 & 8 & 3 & 2 & 4 \end{pmatrix},$$

$$A_{(2)} = \begin{pmatrix} 3 & 1 & 2 & 3 & 9 & 3 \\ 1 & 5 & 6 & 5 & 7 & 2 \\ 4 & 9 & 5 & 8 & 9 & 4 \end{pmatrix},$$

$$A_{(3)} = \begin{pmatrix} 3 & 1 & 4 & 1 & 5 & 9 \\ 2 & 6 & 5 & 3 & 5 & 8 \\ 9 & 7 & 9 & 3 & 2 & 4 \end{pmatrix}.$$

Definirajmo odmah i operaciju *fold* kao inverz matrizaciji tenzora.

Preoblikovanje tenzora

Preoblikovanje tenzora (eng. *reshape*) $\mathcal{A} \in \mathbb{R}^{d_1\times d_2\times\cdots\times d_N}$ je matrica, označavati ćemo je s $A_{[k]}$, $A_{[k]} \in \mathbb{R}^{d_1 d_2 \ldots d_k \times d_{k+1}\ldots d_N}$, gdje je $k \in \{1, ...N\}$, čiji se elementi uzimaju po stupcima tenzora $\mathcal{A}$:

$$A_{[k]}(\overline{i_1 \dots i_k}, \overline{i_{k+1} \dots i_N}) = \mathcal{A}(i_1, i_2, \dots, i_k, i_{k+1}, \dots, i_N), \tag{1.6}$$

gdje je multi indeks $\overline{i_1 \dots i_N}$ definiramo kao:

$$\overline{i_1 \dots i_N} = i_1 + (i_2 - 1)d_1 + \cdots + (i_N - 1)d_1 \dots d_{N-1}.$$

Primjer 1.2.8. *Neka je dan tenzor $\mathcal{A} \in \mathbb{R}^{2\times3\times3}$ kao i u primjeru 1.2.7. Tada su preoblikovanja tenzora po modovima dana s:*

$$A_{[1]} = \begin{pmatrix} 3 & 1 & 4 & 2 & 6 & 5 & 9 & 7 & 9 \\ 1 & 5 & 9 & 3 & 5 & 8 & 3 & 2 & 4 \end{pmatrix},$$

$$A_{[2]} = \begin{pmatrix} 3 & 2 & 9 \\ 1 & 6 & 7 \\ 4 & 5 & 9 \\ 1 & 3 & 3 \\ 5 & 5 & 2 \\ 9 & 8 & 4 \end{pmatrix}.$$

Navedimo još jednu bitnu propoziciju koja će nam kasnije biti od važnosti, a ujedno pokazuje i korisnost Kroneckerovog produkta.

Propozicija 1.2.9. *([8]) Neka je $\mathcal{T} \in \mathbb{R}^{d_1\times\cdots\times d_N}$ tenzor i definirajmo matrice $U_n \in \mathbb{R}^{j_n\times d_n}$, za $n = 1, \dots, N$. Tada definirajmo:*

$$\mathcal{B} = \mathcal{T} \times_1 U_1 \times_2 \cdots \times_N U_N.$$

Sada je matrizacija u modu n tenzora $\mathcal{B}$ dana s

$$\mathcal{B}_{(n)} = U_n \mathcal{T}_{(n)} \left(U_N \otimes \cdots \otimes U_{n+1} \otimes U_{n-1} \otimes \cdots \otimes U_1\right)^T.$$

1.3 SVD

Singular value decomposition (SVD) je faktorizacija realne ili kompleksne matrice. Koristi se u linearnoj algebri, statistici i strojnom učenju. SVD je snažan alat za smanjivanje dimenzije, smanjivanje šuma u podacima, izvlačenja bitnih značajki ili opisivanju podataka općenito.

Neka je dana matrica A $\in \mathbb{R}^{m\times n}$ ranga r. Tada postoje matrice $U \in \mathbb{R}^{m\times n}$,$\Sigma \in \mathbb{R}^{m\times n}$, $V \in \mathbb{R}^{n\times n}$, takve da vrijedi da su U i V ortogonalne matrice, a Σ dijagonalna matrica s nenegativnim realnim vrijednostima na dijagonali (singularne vrijednosti matrice A):

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T,$$

gdje je σ_i singularna vrijednost, u_i lijevi singularni vektor, a v_i desni singularni vektor za $i = 1, \ldots, r$.

Računaje

Za izračunavanje SVD-a proizvoljne matrice A, prvo je potrebno izračunati svojstvene vrijednosti i svojstvene vektore matrica AA^T i A^TA. Tada su singularne vrijednosti matrice A kvadratni korijeni svojstvenih vrijednosti, a stupci matrice U su svojstveni vektori matrice AA^T, dok su stupci matrice V svojstveni vektori matrice A^TA. Naravno ovakav način računanja se primjenjuje samo u teoriji dok za praktično računanje imamo razvijene mnoge metode poput QR metode, Jakobijeve metode, Podijeli pa vladaj metode, itd. [2]. Lapack ima mnoge rutine za računanje SVD dekompozicije [21].

Napomena 1.3.1. *Ako je A kompleksna tada su i U i V unitarne kompleksne matrice te vrijedi* $A = U\Sigma V^*$.

Smanjivanje dimenzije

Jednom kada smo dobili matrice U, Σ, V, odaberemo k (takav da $k < r$) najvećih singularnih vrijednosti i stavimo ostale vrijednosti na nulu. Tako dobivamo novu dijagonalnu matricu, označimo ju s Σ_k. Sada možemo dobiti matricu A_k, aproksimacija matrice A s rangom k:

$$A_k = U_k\Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

Sada pogledajmo primjer u kojem imamo neku proizvoljnu sliku (1.2a) i napravimo SVD te dobijemo rekonstruiranu sliku, ali fiksirajmo k na neku vrijednost iz skupa $\{1, 2, ..., r\}$. Ako sada pogledamo kako k ovisi o količini informacija koja je sačuvana u slici dobivamo graf 1.3 gdje vidimo da je velika količina informacija o slici zapravo sačuvana u par prvih svojstvenih vektora i vrijednosti, pa nas to motivira da pogledamo kako izgledaju slike s različitim vrijednostima k 1.2 gdje smo za vrijednosti k uzeli 1, 5, 10, 20 i 200.

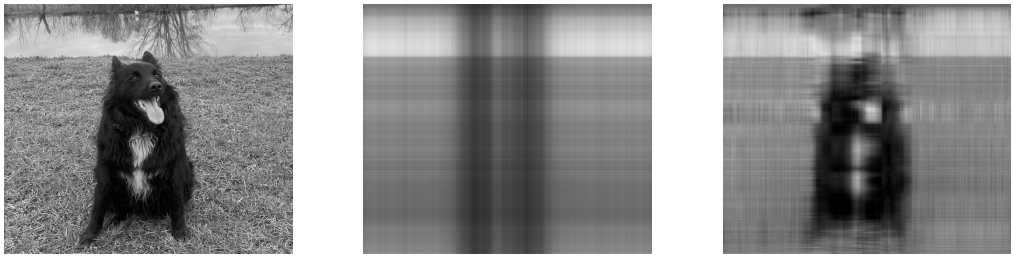
Kao što vidimo na slici (1130×1301 piksela) (1.2b), gdje smo uzeli $k = 1$, slika je mutna i na njoj se ništa ne raspoznaje. Kako polako povećavamo k kroz iduće slike c), d), e) vidimo da slike postaju sve jasnije i jasnije te za $k = 200$ (1.2f) jedva raspoznajemo razliku. Sad vidimo da slika može biti prikazana korištenjem puno manje memorije, što može biti od velike koristi u situacijama gdje nam je ona ograničena.

Za lakši odabir vrijednosti k može nam pomoći sljedeći teorem.

Teorem 1.3.2. *(Eckart–Young–Mirsky–Schmidt) Neka je $l, r \in \mathbb{N}$, $l < r$ i $A_l = U_l \Sigma_l V_l^T = \sum_{i=1}^{l} \sigma_i u_i v_i^T$. Tada je*

$$\min_{\text{rank}(X)\leq l} ||A - X||_F = ||A - A_l||_F = \sqrt{\sum_{i=l+1}^{r} \sigma_i^2},$$

$$\min_{\text{rank}(X)\leq l} ||A - X||_2 = ||A - A_l||_2 = \sigma_{l+1}.$$



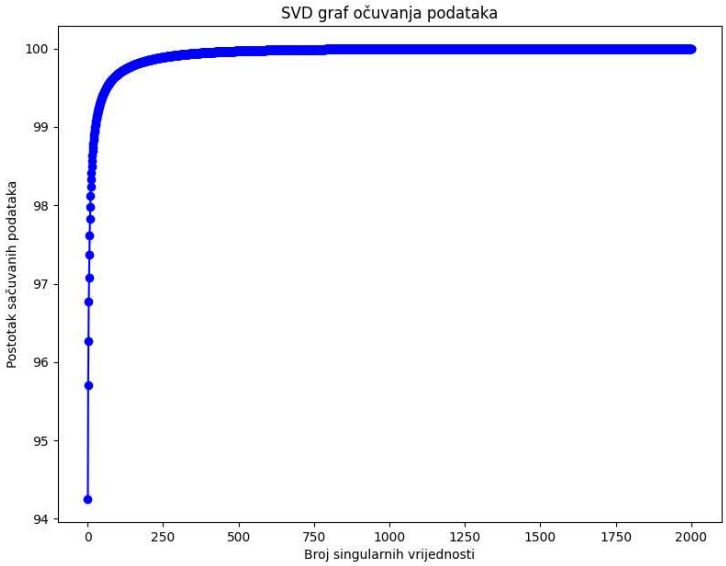
(a) Originalna slika (b) k = 1 (c) k = 5

(d) k = 10 (e) k = 20 (f) k = 200

Slika 1.2: Konačna kvaliteta slike ovisi o uzimanju različitog broja svojstvenih vektora u rekonstrukciji originalne slike

Tanki SVD

Neka je $A \in \mathbb{R}^{d_1 \times d_2}$ i $d_1 > d_2$ pa SVD možemo zapisati kao:



Slika 1.3: Očuvanje podataka u ovisnosti koliko sigularnih vrijednosti uzeli

$$A = U \begin{pmatrix} S \\ 0 \end{pmatrix} V^T. \tag{1.7}$$

Ako sada matricu U podijelimo u dva bloka $U = (U_1, U_2)$, gdje se zapravo U_2 u (1.7) množi s 0 pa onda možemo (1.7) zapisati kao:

$$A = U_1 S V^T,$$

to se onda naziva *tanki SVD* (eng. *thin SVD*).

Analiza glavnih komponenti

Kako smo rekli da SVD ima puno primjena u strojnom učenju tako je jedna od poznatijih Analiza glavnih komponenti ili *Principal Component Analysis (PCA)* (kako je poznata u stranoj literaturi). Neka je $C = A^T A$, gdje je A $\in \mathbb{R}^{m \times n}$, simetrična pozitivno definitna matrica. Tada uzimanjem k vodećih svojstvenih vrijednosti i vektora matrice C radimo PCA analizu matrice C, ali to je zapravo SVD matrice A. U statistici, $A^T A$ prestavlja upravo kovarijantnu matricu i sama PCA metoda je nastala proučavanjem svojstvenih vektora i vrijednosti te matrice.

Primjene PCA u medicini

PCA ima primjene u različitim područjima zahvaljujući tome što ne sadrži parametre, što omogućuje primjenu na raznolike skupove podataka bez potrebe za prilagođavanjem parametara ili poznavanja načina na koji su podaci zabilježeni.

Na primjeru Khanovog skupa podataka [24] o dječjem karcinomu, PCA je korištena za analizu podataka o ekspresiji gena različitih vrsta tumora. Skup podataka sastoji se od 63 djece s različitim tipovima tumora i 2308 gena koji odražavaju ekspresiju tih gena. PCA se koristi za smanjenje dimenzionalnosti podataka, pri čemu se pokušava objasniti što veći dio varijabilnosti podataka s manjim brojem glavnih komponenti. U ovom slučaju, dimenzionalnost podataka je određena brojem slučajeva minus jedan.

Primjenom PCA na Khanov skup podataka, mogu se identificirati geni koji značajno pridonose grupiranju tumora. Analiza pokazuje preklapanje individualnih tumora između različitih grupa, ali istovremeno razdvaja određene grupe tumora na temelju njihovih srednjih vrijednosti. Ova vrsta analize omogućuje istražiteljima da razumiju koja genetska obilježja doprinose grupiranju tumora i mogućim razlikama među različitim tipovima tumora.

Poglavlje 2

Tenzorske dekompozicije

U ovom ćemo poglavlju opisati četiri različite dekompozicije: CP dekompoziciju koja rastavlja tenzor na sumu tenzora ranga jedan, Tuckerovu dekompoziciju koja je generalizirana verzija HOSVD-a, HOSVD, koji je zapravo matrični SVD, ali za tenzore višeg reda i Tenzorski vlak dekompoziciju, koja rastavlja tenzor reda N u umnožak tenzora trećeg reda.

2.1 CP dekompozicija

The Canonical Polyadic (CP) je dekompozicija u kojoj se tenzori višeg reda rastavljaju na sumu tenzora ranga jedan.
Za primjer uzmimo tenzor trećeg reda $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ kojeg želimo zapisati kao:

$$\mathcal{X} \approx \sum_{r=1}^{\eta} a_r \circ b_r \circ c_r, \tag{2.1}$$

gdje nam je $\eta \in \mathbb{N}$, $a_r \in \mathbb{R}^{d_1}$, $b_r \in \mathbb{R}^{d_2}$, $c_r \in \mathbb{R}^{d_3}$ za $r = 1, \ldots, \eta$. Ako sada definiramo matrice:

$$\begin{aligned} A &= [a_1 \quad a_2 \quad \cdots \quad a_\eta], \\ B &= [b_1 \quad b_2 \quad \cdots \quad b_\eta], \\ C &= [c_1 \quad c_2 \quad \cdots \quad c_\eta], \end{aligned}$$

onda korištenjem tih novo definiranih matrica svaku matrizaciju našeg tenzora $\mathcal{X}$ možemo zapisati pomoću njih:

$$X_{(1)} \approx A(C \odot B)^T, \tag{2.2}$$
$$X_{(2)} \approx B(C \odot A)^T, \tag{2.3}$$
$$X_{(3)} \approx C(B \odot A)^T. \tag{2.4}$$

Na isti način možemo zapisati i horizontalne i bočne odsječke. Međutim, zapisivanje preko odsječaka nije lako proširiti u slučajeve kad imamo više od tri dimenzije. Sada CP model možemo zapisati kao i u [18]:

$$\mathcal{X} \approx [\![A, B, C]\!] \equiv \sum_{r=1}^{\eta} a_r \circ b_r \circ c_r. \tag{2.5}$$

Stupce matrica A, B, C želimo normalizirati na duljinu jedan, pa njihove težine možemo zapisati u vektor $\lambda \in \mathbb{R}^{\eta}$:

$$\mathcal{X} \approx [\![\lambda; A, B, C]\!] \equiv \sum_{r=1}^{\eta} \lambda_r a_r \circ b_r \circ c_r. \tag{2.6}$$

Sada ovo možemo poopćiti i na više dimenzija. Neka je dan tenzor N-tog reda, $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$. Tada je CP dekompozicija tenzora $\mathcal{X}$ dana kao:

$$\mathcal{X} \approx [\![\lambda; A^{(1)}, A^{(2)}, \ldots, A^{(N)}]\!] \equiv \sum_{r=1}^{\eta} \lambda_r a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)}, \tag{2.7}$$

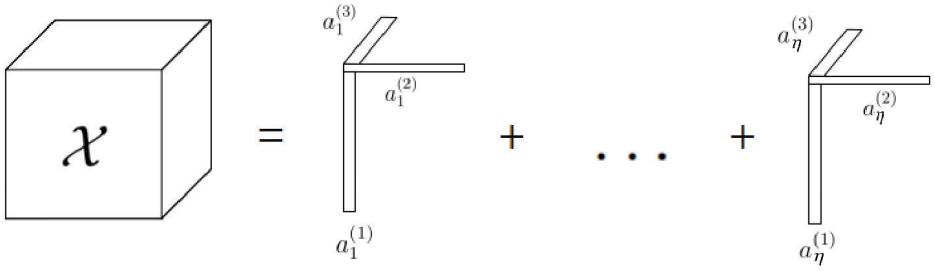
gdje je $a_r^{(i)} \in \mathbb{R}^{d_i}$, $\lambda \in \mathbb{R}^{\eta}$ i $A^{(i)} \in \mathbb{R}^{d_i \times \eta}$ $i = 1, ...N$, $r = 1, ..., \eta$.
Tenzor $\mathcal{X}$ u n-tom modu se sada može zapisati kao:

$$\mathcal{X}_{(n)} \approx A^{(n)} \Lambda (A^{(N)} \cdots \odot A^{(n+1)} \odot A^{(n-1)} \odot \cdots \odot A^{(1)}), \tag{2.8}$$

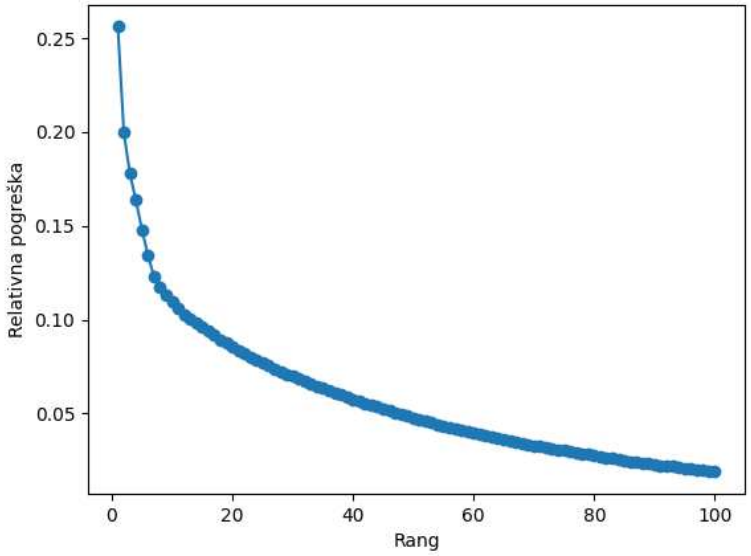
gdje je $\Lambda \in \mathbb{R}^{\eta \times \eta} = diag(\lambda_1, \ldots, \lambda_\eta)$.
Na slici 2.1 možemo i vizualno vidjeti kako izgleda CP dekompozicija. Kao što vidimo u definiciji (1.2.5) rang novonastalog tenzora je η. Postavlja se pitanje koji η da odaberemo da bi dekompozicija dala što bolju aproksimaciju, uzimajući u obzir da prevelik rang dovodi do korištenja previše memorije. Na primjer, povećavamo rang η dok ne dođemo do zadovoljavajuće aproksimacije početnog tenzora.

Prije nego što pogledamo kako se kvaliteta slike ponaša u odnosu na veličinu ranga, definirajmo relativnu pogrešku CP dekompozicije tenzora $\mathcal{X}_1$ na $\mathcal{X}_2$ kao:

$$\frac{\|\mathcal{X}_1 - \mathcal{X}_2\|_F}{\|\mathcal{X}_1\|_F}.$$

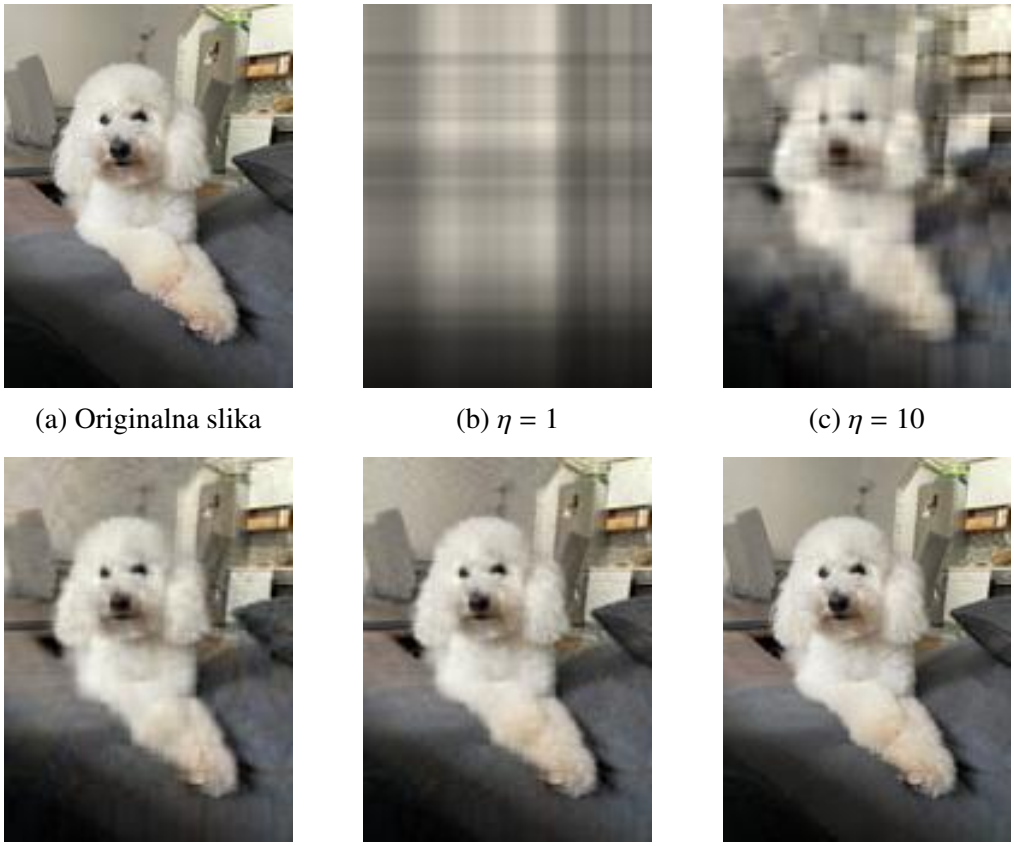


Slika 2.1: Vizualna reprezentacija CP dekompozicije



Slika 2.2: Smanjenje relativne pogreške povećanjem ranga

Na slikama 2.3 (a...f) možemo vidjeti kako nam se kvaliteta slike poboljšava što veći i veći rang uzimamo, što je i očekivano. Ujedno je i na grafu 2.2 moguće vidjeti kako se uzimanjem sve većeg i većeg ranga dobiva sve točnije i točnija aproksimacija početnog tenzora, tj. relativna greška je sve manja i manja (gdje se za početni tenzor uzela slika dimenzija $128 \times 128 \times 3$). Kao što vidimo pogreška i rang nisu u linearnom odnosu pa zato odabiranje ranga nije trivijalan zadatak već vrlo često želimo pogoditi točku nakon koje povećanjem ranga nećemo dobiti puno na točnosti aproksimacije, a da uštedimo memoriju.



(a) Originalna slika (b) $\eta = 1$ (c) $\eta = 10$

(d) $\eta = 20$ (e) $\eta = 30$ (f) $\eta = 60$

Slika 2.3: Uzimanjem različitih vrijednosti za η (tj. biranjem ranga) dobivamo različite kvalitete slika

Jedinstvenost CP dekompozicije

Dekompozicije tenzora višeg reda su često jedinstvene, za razliku od dekompozicija matrica. Radovi [15] i [27] nam daju informacije o jedinstvenosti CP dekompozicija kroz povijest.

Da bi pokazali nejedinstvenost matričnih dekompozicija, definirajmo $A \in \mathbb{R}^{d_1 \times d_2}$ kao matricu ranga η. Tada je dekompozicija ranga te matrice dana s:

$$A = BC^T = \sum_{r=1}^{\eta} a_r \circ b_r.$$

Ako sada napravimo SVD matrice A i dobijemo: $A = U\Sigma V^T$, možemo staviti $B = U\Sigma$ i $C = V$, ali također možemo staviti: $B = U\Sigma W$ i $C = VW$ gdje je W bilo koja $\eta \times \eta$

ortogonalna matrica. Upravo vidimo da na ovaj način možemo dobiti beskonačno rastava u kojem svi daju matricu A, ali različitih su vrijednosti.

Pokažimo sada jedinstvenost CP dekompozicije. Definirajmo prvo tenzor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ trećeg reda i ranga η:

$$\mathcal{X} = \sum_{r=1}^{\eta} a_r \circ b_r \circ c_r = [\![A, B, C]\!].$$

U ovom slučaju jedinstvenost znači da je ovo jedina kombinacija tenzora ranga jedan koji u sumi daju $\mathcal{X}$. Naravno, ne računajući skaliranje ili permutaciju. Skaliranje se odnosi na skaliranje pojedinačnih vektora, tj:

$$\mathcal{X} = \sum_{r=1}^{\eta} (\alpha_r a_r) \circ (\beta_r b_r) \circ (\gamma_r c_r),$$

gdje je $\alpha_r \beta_r \gamma_r = 1$ za $r = 1, \ldots, \eta$.

Permutacija se odnosi na to da komponente, koje čine tenzor ranga jedan, možemo presložiti na bilo koji način, tj:

$$\mathcal{X} = [\![A, B, C]\!] = [\![A\Pi, B\Pi, C\Pi]\!],$$

gdje je Π bilo koja $\eta \times \eta$ permutacijska matrica.

Računanje CP dekompozicije

S obzirom na to da je određivanje točne CP dekompozicije nekog tenzora kompleksan zadatak, sagledat ćemo ovaj problem iz drugog kuta. Neka je $\mathcal{T}$ tenzor koji želimo aproksimirati nekim tenzorom $\tilde{\mathcal{T}}$ koji ima CP strukturu nekog ranga η:

$$\min_{\tilde{\mathcal{T}}} \|\mathcal{T} - \tilde{\mathcal{T}}\|_F, \quad \text{gdje je } \tilde{\mathcal{T}} = \sum_{r=1}^{\eta} \lambda_r a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)}. \tag{2.9}$$

Metoda alternirajućih najmanjih kvadrata

Jedna metoda računanja CP dekompozicije tenzora $\mathcal{T}$, ujedno i najčešće korištena, je metoda alternirajućih kvadrata (ALS - *Alternating leas squares*), koju ćemo i poslije koristiti u konačnoj implementaciji, samo s drugom dekompozicijom. Cijela se metoda temelji na tome da svaku iteraciju fiksiramo sve članove dekompozicije osim jednoga kojeg onda i računamo. Da bi lakše objasnili ovaj algoritam, definirajmo $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ tenzor reda 3 te tada jednadžba (2.9) postaje:

$$\min_{\tilde{\mathcal{X}}} \|\mathcal{X} - \tilde{\mathcal{X}}\|_F, \qquad \text{gdje je } \tilde{\mathcal{X}} = \sum_{r=1}^{\eta} \lambda_r a_r \circ b_r \circ c_r = [\![\lambda; A, B, C]\!]. \tag{2.10}$$

Metoda alternirajućih kvadrata fiksira B i C te računa A, pa onda fiksira A i C pa računa B i konačno, fiksira A i B te računa C. Ovaj proces se ponavlja dok neki uvjet konvergencije nije zadovoljen.

Raspišimo sada jedan korak takvog postupka. Uzmimo da su B i C fiksirani i računamo A. Iz (2.4) možemo zapisati problem minimizacije kao:

$$\min_{\tilde{A}} \|X_{(1)} - \tilde{A}(C \odot B)^T\|_F,$$

gdje je $\tilde{A} = A \cdot \text{diag}(\lambda)$. Sada vidimo da se ovaj problem sveo na linearni problem najmanjih kvadrata, pa je rješenje dano s:

$$\tilde{A} = X_{(1)} \left[(C \odot B)^T\right]^{\dagger}.$$

Iz svojstva (1.5) možemo zapisati rješenje i kao:

$$\tilde{A} = X_{(1)}(C \odot B)(C^T C * B^T B)^{\dagger}.$$

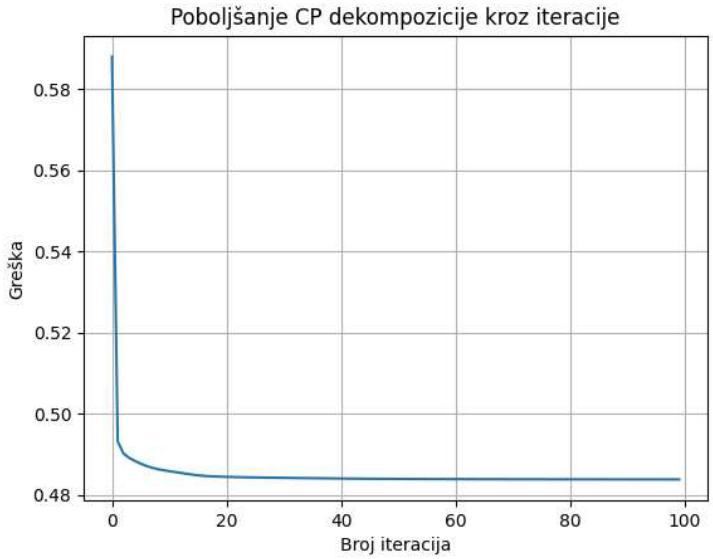
Kao što se može vidjeti mala prednost drugog zapisa u odnosu na prvi je da računamo pseudo inverz od matrice dimenzija $\eta \times \eta$, a ne $d_2 d_3 \times \eta$. Međutim ova verzija nije numerički stabilna.

Na grafu 2.4 vidimo glavni nedostatak metode alternirajućih kvadrata. Kroz iteracije pogreška CP dekompozicije naglo krene padati, ali onda se odjednom smanji brzina konvergencije i zapravo se čini da nema pomaka iz iteraciju u iteraciju jer je poboljšanje zanemarivo. U algoritmu 1 možemo vidjeti kako bi izgledao algoritam za računanje dekompozicije na ovako opisan način za tenzor reda N.

Algorithm 1 CP dekompozicija

Ulaz: Tenzor $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, rang tenzora, te neke početne vrijednosti za $A^{(1)}, \ldots, A^{(N)}$

for $it = 1$ to ... **do**
  for $n = 1$ to N **do**
    $V \leftarrow \left(A^{(1)T}A^{(1)}\right) * \cdots * \left(A^{(n-1)T}A^{(n-1)}\right) * \left(A^{(n+1)T}A^{(n+1)}\right) * \cdots * \left(A^{(N)T}A^{(N)}\right)$
    $A^{(n)} = X^{(n)}\left(A^{(N)} \odot \cdots \odot A^{(n+1)} \odot A^{(n-1)} \odot \cdots \odot A^{(1)}\right) V^{\dagger}$
    Izračunaj normu stupaca od $A^{(n)}$ i ažuriraj Λ
  end for
end for



Slika 2.4: Smanjenje relativne pogreške CP dekompozicije kroz iteracije metode alternirajućih najmanjih kvadrata

Kao što vidimo u samom algoritmu nismo naveli uvjete zaustavljanja jer ih ima nekoliko ovisno od toga što nam treba i koji od njih se postiže ako se ikad i postiže:

- faktorske matrice se jako malo mijenjaju ili se uopće ne mijenjaju
- vrijednost funkcije cilja ima jako malo poboljšanja ili uopće nema poboljšanja
- ciljana vrijednost je blizu nule
- postigli smo predodređeni broj iteracija

Kao što smo imali prilike i vidjeti metoda ALS je jednostavna za razumjeti i implementirati, ali zato ponekad treba jako puno iteracija do konvergencije. Ujedno i samo konačno rješenje može jako ovisiti o početnim izabranim vrijednostima. Jedno nepovoljno svojstvo je da ne možemo garantirati konvergenciju u globalni minimum ili čak stacionarne točke, već samo zapažamo da vrijednost funkcije cilja prestaje padati.

Primjene CP dekompozicije

Na samim početcima CP se koristio u psihometriji, jer su Carroll i Chang [13] u kontekstu analize matrica sličnosti ili različitosti uveli CANDECOMP. Metodu su primijenili na

skupu podataka o zvučnim tonovima iz Bell Labsa i na drugom skupu podataka o usporedbama država. Konačno, Harshman [6] uvodi PARAFAC kako bi eliminirao neodređenost povezanu s dvodimenzionalnom PCA metodom.

Appellof i Davidson [11] su pionirski primijenili CP u kemometriji dok je nekoliko autora koristilo CP dekompozicije u neuroznanosti.

Također je Acar u radovima [3] i [4] prvi put primijenio tenzore općenito u rudarenju podataka, pa se tako između svih ostalih i CP dekompozicija našla kao jedna od dekompozicija koja je korištena. Ujedno su Shashua i Levin [26] koristili CP dekompoziciju za klasifikaciju i kompresiju slika.

2.2 Tuckerova dekompozicija

Tucker prvi put uvodi pojam Tuckerove dekompozicije 1963. godine [29]. Ideja je rastaviti tenzor N-tog reda $\mathcal{W}$ u jezgreni tenzor $\mathcal{G}$ istog reda kao $\mathcal{W}$, pomnožen, u različitim modovima, s N matrica umnoška. Kao i kod CP dekompozicije, pogledajmo tenzor $\mathcal{W} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ trećeg reda radi lakšeg zapisivanja, pa imamo ([8]):

$$\mathcal{W} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_p \circ b_q \circ c_r = [\![\mathcal{G}; A, B, C]\!], \tag{2.11}$$

gdje su $A \in \mathbb{R}^{d_1 \times P}, B \in \mathbb{R}^{d_2 \times Q}, C \in \mathbb{R}^{d_3 \times R}$ matrice umnoška, a $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ jezgreni tenzor. Sada možemo istu stvar zapisati po elementima:

$$x_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr}, \tag{2.12}$$

za $i = 1, \ldots, d_1, \quad j = 1, \ldots, d_2 \quad i \quad k = 1, \ldots, d_3$.
Kao što se da uočiti P, Q, R mogu biti dosta manji od d_1, d_2, d_3 pa onda govorimo o kompresiranoj verziji Tuckerove dekompozicije. Razne verzije i primjene ove dekompozicije bit će objašnjene dalje u tekstu.

Navodimo jednadžbe, po modovima, matriciziranog oblika jednadžbe (2.11):

$$\begin{aligned} X_{(1)} &\approx A G_{(1)} (C \otimes B)^T, \\ X_{(2)} &\approx B G_{(2)} (C \otimes A)^T, \\ X_{(3)} &\approx C G_{(3)} (B \otimes A)^T. \end{aligned}$$

Poopćimo sada zapis na tenzore N-tog reda. Neka je $\mathcal{W} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, tada postoji jezgreni tenzor $\mathcal{G} \in \mathbb{R}^{J_1 \times \ldots \times J_N}$, takvi da je $J_k \leq I_k$ te matrice $U_i \in \mathbb{R}^{I_i \times J_i}$

$$\mathcal{W} = \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 \cdots \times_n U_n = [\![\mathcal{G}; U_1, U_2, \ldots, U_N]\!], \tag{2.13}$$

po elementima:

$$x_{i_1 i_2 \ldots i_N} \approx \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \cdots \sum_{j_N=1}^{J_N} g_{j_1 j_2 \ldots j_N} u_{i_1 j_1} u_{i_2 j_2} \cdots u_{i_N j_N}, \tag{2.14}$$

za $i_n = 1, \ldots, I_n, \quad n = 1, \ldots, N$. Matricizirana verzija (2.13):

$$X_{(n)} = U_n G_{(n)} (U_N \otimes \cdots \otimes U_{(n+1)} \otimes U_{(n-1)} \otimes \cdots \otimes U_1)^T.$$

Važno je spomenuti jednu varijaciju Tuckerove dekompozicije, a to je takozvana *Tucker2* dekompozicija [30]. Recimo da imamo tenzor reda tri, tada jednu matricu umnoška zamijenimo s matricom identiteta.

$$\mathcal{W} = \mathcal{G} \times_1 A \times_2 B = [\![\mathcal{G}; A, B, I]\!].$$

Vidimo da je jednadžba skoro identična (2.11), osim da je $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ s $R = K$ i C je $K \times K$ matrica identiteta. Također, u radu [30] se spominje *Tucker1* koji dvije matrice umnoška zamjeni s jediničnim matricama, tj. imamo:

$$\mathcal{W} = \mathcal{G} \times_1 A = [\![\mathcal{G}; A, I, I]\!].$$

Jedna od glavnih prednosti Tuckerove dekompozicije je da omogućuje fleksibilniji pristup samom modeliranju od nekih drugih tehnika (kao što su to PCA ili SVD). Na primjer, može se koristiti da se modeliraju podaci koji imaju nelinearne veze ili druge vrste složenijih struktura.

Nejedinstvenost

Pokažimo na primjeru kako Tuckerova dekompozicija nije jedinstvena. Neka je $U \in \mathbb{R}^{P \times P}$, $V \in \mathbb{R}^{Q \times Q}$ i $W \in \mathbb{R}^{R \times R}$. I neka je:

$$\mathcal{X} = \mathcal{G} \times_1 A \times_2 B \times_3 C = [\![\mathcal{G}; A, B, C]\!],$$

gdje je $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. Sada je to ekvivalentno:

$$[\![\mathcal{G}; A, B, C]\!] = [\![\mathcal{G} \times_1 U \times_2 V \times_3 W; AU^{-1}, BV^{-1}, CW^{-1}]\!].$$

Vidimo da možemo promijeniti jezgru dok god radimo inverznu modifikaciju na matricama umnoška. Prednost je da, u nekim situacijama, možemo pojednostaviti jezgrenu strukturu tako da većina elemenata tenzora $\mathcal{G}$ bude jednaka nuli. Tucker je prvi primijetio ovo svojstvo, a mnogi drugi autori su ga proučavali kroz povijest. Razvili su se razni algoritmi koji poboljšavaju jezgreni tenzor na neki način. Kao što ćemo vidjeti u nastavku, HOSVD stvara jezgru koja je potpuno ortogonalna, što je opet neko svojstvo koje na svoj način može biti korisno.

Postupak računanja

Tucker je u svom radu iz 1966. godine [30] uveo tri metode za računanje Tuckerove dekompozicije. Jedna od njih je i HOSVD, ali nju ćemo obraditi kao posebnu dekompoziciju jer je popularna i ima mnoge primjene. Prva od metoda kojih se spominju je klasičan ALS (*Alternating Least Squares*). Ideja je minimizirati funkciju cilja s obzirom na jednu varijablu dok ostale fiksiramo. Na primjer, neka je $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ i cilj nam je doći do tri matrice umnoška: U_1, U_2 i U_3 te jezgrenog tenzora $\mathcal{G}$:

1. Izračunaj U_1 fiksiranjem U_2, U_3 i $\mathcal{G}$
2. Izračunaj U_2 fiksiranjem U_1, U_3 i $\mathcal{G}$
3. Izračunaj U_3 fiksiranjem U_1, U_2 i $\mathcal{G}$
4. Izračunaj $\mathcal{G}$ fiksiranjem U_1, U_2 i U_3

Ovdje ne navodimo detaljni opis kako izračunati slobodne varijable jer to ovisi o metodi koja se koristi. Prednosti ovakve metode je što se lako implementira i dosta je brza konvergencija za tenzore malih dimenzija.

HOOI

De Lathauwer, De Moor, i Vandewalle su 2000. godine predložili efikasniju ideju za računanje matrica umnoška te ju nazvali *Higher-order orthogonal iteration* (HOOI). Algoritam možemo vidjeti u 2.

Neka je dan tenzor$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, tada je optimizacijski problem dan s

$$\min_{\mathcal{G}, U_1, \ldots, U_M} \|\mathcal{X} - [\![\mathcal{G}; U_1, U_2, \ldots, U_M]\!]\|_F , \tag{2.15}$$

gdje je $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, $U_{(n)} \in \mathbb{R}^{I_n \times J_n}$ i ortogonalna je po stupcima za $n = 1, \ldots, N$. Ako zapišemo funkciju cilja iz jednadžbe (2.15) u vektorskoj formi dobivamo:

$$\left\| \operatorname{vec}(\boldsymbol{X}) - (\mathbf{U}_{(N)} \otimes \mathbf{U}_{(N-1)} \otimes \cdots \otimes \mathbf{U}_{(1)}) \operatorname{vec}(\mathcal{G}) \right\|_F ,$$

Algorithm 2 HOOI [8]

Ulaz: Tenzor $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ i $J_1, J_2, \ldots, J_N \in \mathbb{R}$ dimenzije jezgrenog tenzora.

1: inicijaliziraj $U_{(n)} \in \mathbb{R}^{I_n \times J_n}$, za $n = 1, ..., N$ koristeći HOSVD
2: **repeat**
3: **for** $n = 1$ to N **do**
4: $\mathcal{Y} \leftarrow \mathcal{X} \times_1 U_{(1)}^T \cdots \times_{n-1} U_{(n-1)}^T \times_{n+1} U_{(n+1)}^T \cdots \times_N U_{(N)}^T$
5: $U_{(n)} \leftarrow J_n$ vodećih singularnih vektora od $\mathcal{Y}_{(n)}$
6: **end for**
7: **until** Mala promjena u kvaliteti rješenja ili je postignut maksimalni broj iteracija
8: $\mathcal{G} \leftarrow \mathcal{X} \times_1 U_{(1)}^T \cdots \times_N U_{(N)}^T$
9: **return** $\mathcal{G}, U_{(1)}, U_{(2)}, \ldots, U_{(N)}$

pa vidimo da jezgreni tenzor mora zadovoljavati:

$$\mathcal{G} = X \times_1 \mathbf{U}_{(1)}^T \times_2 \mathbf{U}_{(2)}^T \cdots \times_N \mathbf{U}_{(N)}^T.$$

Kvadrirajmo sada funkciju cilja pa imamo:

$$\begin{aligned}
&\|\boldsymbol{X} - [\![\mathcal{G}; \mathbf{U}_{(1)}, \mathbf{U}_{(2)}, \ldots, \mathbf{U}_{(N)}]\!]\|_F^2 \\
&\quad = \|\boldsymbol{X}\|_F^2 - 2\left\langle \boldsymbol{X}, \left\|\mathcal{G}; \mathbf{U}_{(1)}, \mathbf{U}_{(2)}, \ldots, \mathbf{U}_{(N)}\right\|_F \right\rangle + \|\mathcal{G}; \mathbf{U}_{(1)}, \mathbf{U}_{(2)}, \ldots, \mathbf{U}_{(N)}\|_F^2 \\
&\quad = \|\boldsymbol{X}\|_F^2 - 2\left\langle \boldsymbol{X} \times {}_1\mathbf{U}_{(1)\mathrm{T}} \cdots \times {}_N\mathbf{U}_{(N)}^T, \mathcal{G} \right\rangle + \|\mathcal{G}\|_F^2 \\
&\quad = \|\boldsymbol{X}\|_F^2 - 2\langle \mathcal{G}, \mathcal{G} \rangle + \|\mathcal{G}\|_F^2 \\
&\quad = \|\boldsymbol{X}\|_F^2 - \|\mathcal{G}\|_F^2 \\
&\quad = \|\boldsymbol{X}\|_F^2 - \left\|\boldsymbol{X} \times {}_1\mathbf{U}_{(1)}^T \times_2 \cdots \times {}_N\mathbf{U}_{(N)}^T\right\|_F^2 .
\end{aligned}$$

Naravno da se (2.15) može riješiti s ALS metodom, ali kako je $\|\mathcal{X}\|^2$ konstanta, problem možemo pretvoriti u problem maksimizacije:

$$\max_{\mathbf{U}_{(n)}} \left\|\boldsymbol{X} \times {}_1\mathbf{U}_{(1)}^T \times_2 \mathbf{U}_{(2)}^T \cdots \times \times_N \mathbf{U}_{(N)}^T\right\|_F ,$$

gdje je $\mathbf{U}_{(n)} \in \mathbb{R}^{I_n \times J_n}$ i ortogonalna po stupcima. Ako sad tu novu funkciju cilja zapišemo u formi matrica imamo:

$$\left\|\mathbf{U}_{(n)}^T \mathbf{W}\right\|_F \text{ gdje je } \mathbf{W} = \mathbf{X}_{(n)} \left(\mathbf{U}_{(N)} \otimes \cdots \otimes \mathbf{U}_{(n+1)} \otimes \mathbf{U}_{(n-1)} \otimes \cdots \otimes \mathbf{U}_{(1)}\right).$$

Pomoću SVD-a možemo doći do rješenja: neka je $U_{(n)}$ R_n vodećih lijevih singularnih vektora od $\mathcal{W}$. Ova metoda će konvergirati do lokalnog optimuma gdje vrijednost funkcije cilja (2.15) pada, ali ne garantira konvergenciju do globalnog optimuma ili čak do stacionarne točke jednadžbe (2.15) ([8]).

Primjena

Tuckerova dekompozicija ima vrlo široku primjenu. U području obrade signala koriste je De Lathauwer i Vandewalle [14]. Također su ju upotrijebili Muti i Bourennane[16] za proširenje Wienerovih filtara u obradi signala. Između ostalog se koristi u psihometriji, na tu je temu objavljen rad od Kiers-a i Van Meschelen-a [12], te u kemijskoj analizi [7].

Svakako je bitno detaljnije spomenuti Vasilescu-a i Terzopoulos-a [9] koji su prvi koristili Tuckerovu dekompoziciju u sklopu računalnog vida (TensorFaces). U ovom istraživanju su uzeti u obzir podaci o slikama lica dobivenim od više subjekata, pri čemu je svaki subjekt imao više slika snimljenih u različitim uvjetima. Varijacija osvjetljenja je jedan od faktora koji su uzeti u obzir, te su podaci organizirani u tri kategorije: osoba, uvjeti osvjetljenja i pikseli. Također, mogu se dodati i dodatni faktori kao što su izražavanje lica i kut kamere, te potencijalno i drugi. Rezultati prepoznavanja lica korištenjem TensorFaces metode su značajno točniji u usporedbi sa standardnim PCA tehnikama [23]. TensorFaces također pruža korisnost u kompresiji podataka i može ukloniti nevažne efekte poput osvjetljenja, dok istovremeno zadržava ključne karakteristike lica [10]. Također, Vasilescu [31] je primijenio Tuckerovu dekompoziciju u analizi ljudskog kretanja.[8]

2.3 HOSVD

High order singular value decomposition (HOSVD) je ortogonalna Tuckerova dekompozicija. To je tehnika pomoću koje rastavljamo tenzor N-tog reda $\mathcal{W}$ u jezgreni tenzor $\mathcal{G}$, istog reda kao $\mathcal{W}$, pomnožen, u različitim modovima, s N matrica. Kao što je moguće primijetiti stvar je ista kao kod Tuckerove dekompozicije samo što su matrice u HOSVD-u ortogonalne i jezgreni tenzor ima svojstvo potpune ortogonalnosti. HOSVD se često koristi za obradu signala, slika, analizu podataka i u strojnom učenju.

Prije nego što opišemo postupak računanja navedimo teorem po kojem vidimo da za svaki N dimenzionalni tenzor postoji HOSVD i daje nam postupak računanja.

Teorem 2.3.1. *([19]) Neka je $\mathcal{A} \in \mathbb{C}^{d_1 \times \cdots \times d_N}$ proizvoljan tenzor reda N. Tada se on može zapisati kao:*

$$\mathcal{A} = \mathcal{S} \times_1 U_1 \times_2 U_2 \times_3 \cdots \times_N U_N, \tag{2.16}$$

gdje vrijedi:

- $U_n = \left(u_n^{(1)} u_n^{(2)} \dots u_n^{(I_n)}\right) \in \mathbb{C}^{d_n \times d_n}$ *su unitarne matrice i vrijedi da je vektor $u_n^{(i)}$ i-ti singularni vektor n-tog moda za $n = 1, ..., N$.*

- *Jezgreni tenzor* $\mathcal{S} \in \mathbb{C}^{d_1 \times \cdots \times d_N}$ *je kompleksni tenzor N-tog reda i za svaki njegov odsječak* $\mathcal{S}_{i_n=\alpha}$ *vrijedi:*

  – *Svojstvo potpune ortogonalnosti: svaka dva različita odsječka, istog tipa, su okomiti, tj. ako imamo dva odsječka istog tipa,* $\mathcal{S}_{i_n=\alpha}$ *i* $\mathcal{S}_{i_n=\beta}$*, tada vrijedi:*

$$\left\langle \mathcal{S}_{i_n=\alpha}, \mathcal{S}_{i_n=\beta} \right\rangle = 0 \quad \forall \alpha \neq \beta.$$

  – *Poredak singularnih vrijednosti: Singularna vrijednost u modu n je:*

$$\sigma_i^{(n)} = \left\| \mathcal{S}_{i_n=i} \right\|_F$$

  i uređene su tako da vrijedi:

$$\left\| \mathcal{S}_{i_n=1} \right\|_F \geq \left\| \mathcal{S}_{i_n=2} \right\|_F \geq \cdots \geq \left\| \mathcal{S}_{i_n=I_n} \right\|_F \geq 0 \quad \forall n = 1, \ldots, N. \tag{2.17}$$

Dokaz ovog teorema dostupan je [19]. Vidimo da kao direktnu posljedicu ovog teorema dobivamo da je:

$$\mathcal{S}_{i_n=\alpha} = \mathbf{0} \quad \forall \alpha > d_1 d_2 \cdots d_{i_n-1} d_{i_n+1} \cdots d_N.$$

Osnovna ideja

Kao što vidimo iz gore prikazanog teorema, HOSVD je rastavljanje tenzora na skup matrica umnoška i jezgreni tenzor. Matrica umnoška je matrica koja sadrži elemente tenzora u određenom modu, dok jezgreni tenzor sadrži ostale elemente:

$$\mathcal{A} = \mathcal{S} \times_1 U_1 \times_2 U_2 \times_3 \cdots \times_n U_n. \tag{2.18}$$

Računanje

Naivni pristup računanja HOSVD tenzora $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_n}$ je tako da matriciziramo tenzor u svih n modova i na svakoj od tih novo dobivenih matrica napravimo SVD pa dobivamo matrice $U_i, \Sigma_i, V_i \ \forall i \in \{1, ..., n\}$. Sada su U_i matrice umnoška. Jezgreni tenzor dobivamo preko formule (2.18):

$$\mathcal{S} = \mathcal{A} \times_1 U_1^T \times_2 U_2^T \times_3 \cdots \times_n U_n^T. \tag{2.19}$$

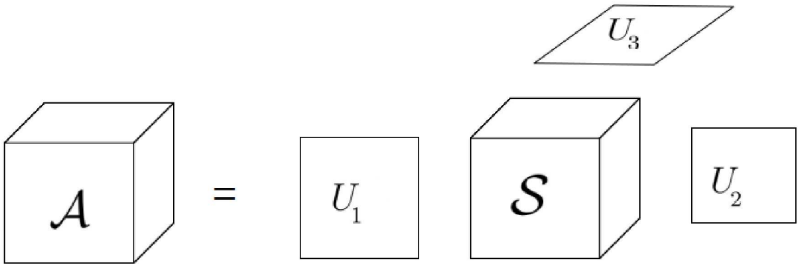
Detaljniji opis računanja nalazi se u algoritmu 3.

Na slici 2.5 vidimo vizualnu reprezentaciju HOSVD-a.

Algorithm 3 HOSVD

Ulaz: Tenzor $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$

1: **for** $n = 1$ to N **do**
2: Matriciziraj $\mathcal{X}$ u modu n, dobivamo $X_{(n)}$
3: Izračunamo SVD od $X_{(n)}$, $X_{(n)} = U\Sigma V^T$
4: Postavi $U^{(n)} = U$
5: **end for**
6: Izračunaj $\mathcal{S} = \mathcal{X} \times_1 (U^{(1)})^T \times_2 (U^{(2)})^T \times_3 \cdots \times_n (U^{(N)})^T$



Slika 2.5: Vizualna reprezentacija HOSVD-a za tenzor trećeg reda

Smanjenje dimenzije

Neka je dan tenzor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$. Ponekad se može desiti da je dimenzija u jednom modu puno veća od umnoška dimenzija u ostalim modovima, kao što je promatrano u [20]. Takva situacija je moguća kod, na primjer, slika, gdje je dimenzija prostora piksela veći od umnoška dimenzija značajki, tj. imamo $d_1 > d_2 d_3 \dots d_N$. Kako smo prije pokazali tada za jezgreni tenzor vrijedi

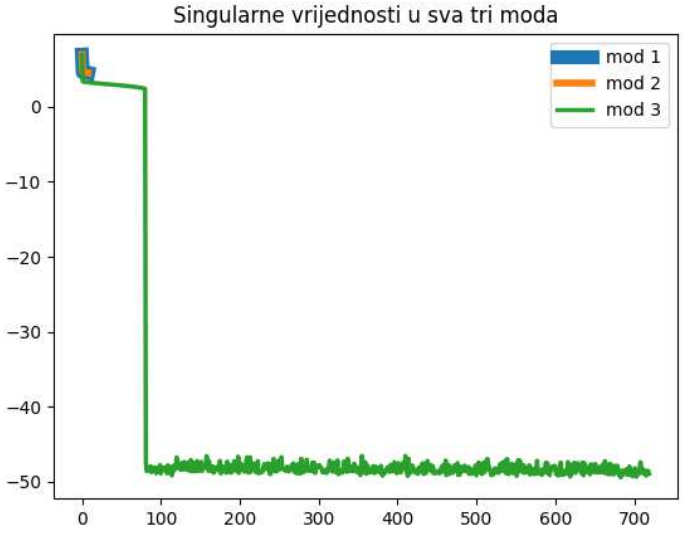
$$\mathcal{S}_{i_1=\alpha} = 0 \quad \alpha > d_2 d_3 \dots d_N. \tag{2.20}$$

Pa se sada (2.16) može zapisati tako da se odbaci dio jezgrenog tenzora koji je 0:

$$\mathcal{A} = \tilde{\mathcal{S}} \times_1 \tilde{U}_1 \times_2 U_2 \times_3 \cdots \times_N U_N, \tag{2.21}$$

gdje je $\tilde{\mathcal{S}} \in \mathbb{R}^{d_2 d_3 \dots d_N \times d_2 \times \cdots \times d_N}$ i $\tilde{U}_1 \in \mathbb{R}^{d_1 \times d_2 d_3 \dots d_N}$. Ovako zapisan HOSVD se također naziva i *tanki HOSVD* (eng. *thin HOSVD*).
Možemo primijetiti da ako je $A \in \mathbb{R}^{d_1 \times d_2}$ tanki HOSVD, onda je zapravo tanki SVD.

Primjer 2.3.2. *Imamo skup podataka videa u kojem svaki video ima isti broj frame-ova. Jedan video prikazuje jednu riječ iz znakovnog jezika. Ideja je da spojimo sve riječi istog znaka u jedan tenzor dimenzija* $frame_{height} \times frame_{width} \times (numberOfFrames * numberOfVideos)$*. Na njemu radimo HOSVD i dobivamo jezgreni tenzor* $\mathcal{S}$ *te matrice umnoška* U_1, U_2, U_3*. Kako je:*



Slika 2.6: Pad normi odsječka u svakom modu

- $frame_{height} = 9$
- $frame_{width} = 9$
- $numberOfFrames = 120$
- $numberOfVideos = 6$

pa onda iz (2.20) *graf, koji vidimo na 2.6, koji prikazuje kako vrijednosti normi odsječaka jezgrenog tenzora* $\mathcal{S}$ *u svakom modu padaju, kao što je navedeno u teoremu 2.3.1. Sada možemo lako odrediti koliko horizontalnih, lateralnih i frontalnih odsječaka želimo zadržati u jezgrenom tenzoru i tako dosta smanjiti dimenziju.*

Dodatni rezultati

Napomena 2.3.3. *([19]) Neka je dan tenzor* $\mathcal{A}$ *i njegov HOSVD kao u* (2.16)*. Tada su singularne vrijednosti u modu m poredane u padajućem poretku, tj:*

$$\sigma_1^{(n)} \geq \sigma_2^{(n)} \geq \cdots \geq \sigma_{I_n}^{(n)} \geq 0, \quad n = 1, \ldots, N.$$

Napomena 2.3.4. *([19]) Neka je dan tenzor* $\mathcal{A}$ *i njegova HOSVD kao u* (2.16)*. Tada vrijedi* $\|\mathcal{A}\|_F^2 = \|\mathcal{S}\|_F^2$

Napomena 2.3.5. *([19]) Singularne vrijednosti n-tog moda su jedinstveno definirane. Nadalje, ze tenzore s realnim vrijednostima, singularni vektori n-tog moda su jedinstveni do predznaka ili umnoška s ortogonalnom matricom.*

Skraćeni HOSVD

U primjenama nam je često potrebno napraviti kompresiju podataka, ali bez smanjenja kvalitete. Da bi postigli to, skraćena verzija HOSVD-a može bit primijenjena (eng. *truncated HOSVD*). Međutim, treba primjetiti da, za razliku od SVD-a, HOSVD nema svojstvo najbolje aproksimacije.

Ipak, postoji teorem koji daje gornju granicu za aproksimaciju pogreške.

Teorem 2.3.6. *([19]) Neka je $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$ te neka je dan njegov HOSVD ((2.16)). Neka je sada $r_n (1 \leq n \leq N)$ rang n-tog moda tenzora $\mathcal{A}$ i definirajmo tenzor $\hat{\mathcal{A}}$ tako da odbacimo n najmanjih singularnih vrijednosti $\sigma^{(n)}_{d'_n+1}, \sigma^{(n)}_{d'_n+2}, \ldots, \sigma^{(n)}_{r_n}$ n-tog moda za zadane vrijednosti od d'_n. Tada:*

$$\|\mathcal{A} - \hat{\mathcal{A}}\|_F^2 \leq \sum_{i_1=d'_1+1}^{r_1} \left(\sigma^{(1)}_{i_1}\right)^2 + \sum_{i_2=d'_2+1}^{r_2} \left(o^{(2)}_{i_2}\right)^2 + \cdots + \sum_{i_N=d_N+1}^{d_N} \left(o^{(N)}_{i_N}\right)^2.$$

Dokaz. Neka je $\hat{\mathcal{S}}$ tenzor koji je dobiven iz tenzora $\mathcal{S}$ tako da n najmanjih singularnih vrijednosti u n-tom modu stavimo da budu jednaki nula. Pa sad imamo:

$$\begin{aligned}
&\|\mathcal{A} - \hat{\mathcal{A}}\|_F^2 = \|\mathcal{S} - \hat{\mathcal{S}}\|_F^2 \\
&= \sum_{i_1=1}^{r_1} \sum_{i_2=1}^{r_2} \cdots \sum_{i_N=1}^{r_N} s^2_{i_1,i_2\cdots,i_N} - \sum_{i_1=1}^{d'_1} \sum_{i_2=1}^{d'_2} \cdots \sum_{i_N=1}^{d_N} s^2_{i_1,i_2\cdots,i_N} \\
&= \sum_{i_1=d'_1+1}^{r_1} \sum_{i_2=d'_2+1}^{r_2} \cdots \sum_{i_N=d'_N+1}^{r_N} s^2_{i_1,i_3,\cdots,i_N} \\
&\leq \sum_{i_1=d'_1+1}^{r_1} \sum_{i_2=1}^{r_2} \cdots \sum_{i_N=1}^{r_N} s^2_{i_1,i_2,\ldots,i_N} + \sum_{i_1=1}^{r_1} \sum_{i_2=d'_2+1}^{r_2} \cdots \sum_{i_N=1}^{r_N} s^2_{i_1,i_2,\ldots,i_N} + \ldots \\
&+ \sum_{i_1=1}^{r_1} \sum_{i_2=1}^{r_2} \cdots \sum_{i_N=d'_N+1}^{r_N} s^2_{i_1,i_2,\ldots,i_N} \\
&= \sum_{i_1=d'_1+1}^{r_1} \left(\sigma^{(1)}_{i_1}\right)^2 + \sum_{i_2=d'_2+1}^{r_2} \left(\sigma^{(2)}_{i_2}\right)^2 + \cdots + \sum_{i_N=d'_N+1}^{r_N} \left(\sigma^{(N)}_{i_N}\right)^2.
\end{aligned}$$

□

Primjene

Glavna primjena HOSVD-a je izvlačenje bitnih informacija (značajki) iz višedimenzionalnih tenzora. Osim toga, primjena HOSVD-a obuhvaća različita područja kao što su prepoznavanje lica, obrada signala, veliki skupovi podataka i otkrivanje događaja u stvarnom vremenu. Također se koristi u dizajnu upravljača i analizi podataka s više pogleda. HOSVD ima sposobnost razdvajanja uzročnih faktora podataka, što ga čini snažnim alatom za istraživanje i analizu kompleksnih skupova podataka. Primjena HOSVD-a ima potencijal u različitim područjima kao što su medicina, genomska analiza i in silico otkrivanje lijekova. Ova tehnika pruža nove načine za prikaz podataka i donošenje informiranih odluka na temelju višedimenzionalnih podataka.

2.4 Tenzorski vlak (Tenzor Train)

I. V. Oseledets je 2010. godine uveo pojam Tenzorske vlak (TT) dekompozicije. Ideja joj je da riješi jedan od glavnih problema dekompozicija Tuckerovih formata, npr. ako pogledamo HOSVD nekog proizvoljnog tenzora $\mathcal{A}$, moramo u memoriju pohraniti jezgreni tenzor $\mathcal{G}$ i N matrica modova, a to postaje jako nezgodno za velike N. Nadalje, kako bi riješio prokletstvo dimenzije Tenzorski vlak dekomponira tenzor N-tog reda u produkt tenzora trećeg reda. Također, kako se algoritam bazira na SVD da se dobiju tenzori, svojstva koja je HOSVD očuvao i ovdje su očuvana.
Neka je $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, tada je Tenzorska vlak dekompozicija dana s ([25]):

$$\mathcal{X}(i_1, ..., i_N) = A_1(i_1, :)\mathcal{A}_2(:, i_2, :)...\mathcal{A}_{N-1}(:, i_{N-1}, :)A_N(:, i_N), \tag{2.22}$$

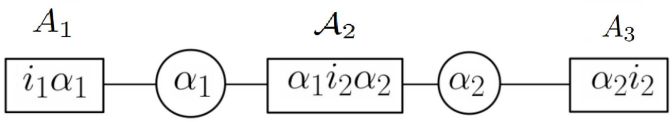
gdje je:

- $A_1 \in \mathbb{R}^{r_1 \times d_1}, A_N \in \mathbb{R}^{r_{N-1} \times d_N}$ i $\mathcal{A}_k \in \mathbb{R}^{r_{k-1} \times d_k \times r_k}$, za $k = 2, ..., N-1$, nazivamo ih TT-jezgre
- r_k, $k = 1, ..., N$ se naziva rang Tenzorskog vlaka vlaka i vrijedi da $r_0 = r_N = 1$
- $r = \max\limits_{1 \le k \le N} r_k$ se naziva maksimalni rang Tenzorskog vlaka

Ako formulu (2.22) zapišemo pomoću indeksa imamo:

$$\mathcal{X}(i_1, ..., i_N) = \sum_{\alpha_1, ..., \alpha_{N-1}} A_1(i_1, \alpha_1)\mathcal{A}_2(\alpha_1, i_2, \alpha_2)...A_N(\alpha_{N-1}, i_N). \tag{2.23}$$

Definicija 2.4.1. *([25]) Neka je $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ tenzor. Kažemo da je $\mathcal{X}$ u TT formatu ako su mu elementi dani s* (2.22).



Slika 2.7: Vizulana reprezentacija dekompozicije Tenzorskog vlaka zapisanog pomoću indeksa

Na slici 2.7 možemo vidjeti prikaz dekompozicije za tenzor trećeg reda. Kao što se može vidjeti pravokutni čvorovi sadrže indekse i_k originalnog tenzora i barem jedan pomoćni indeks dok kružni čvorovi sadrže samo jedan pomoćni indeks. Vrijedi da su dva pravokutna čvora povezana ako i samo ako imaju zajednički pomoćni indeks α_k. Da bi odredili vrijednost tenzora, pomnožimo elemente tenzora odgovarajućih pravokutnih čvorova i onda ih sumiramo po svim pomoćnim indeksima (formula (2.23)). Kao što vidimo slika podsjeća na vlak s vagonima, pa je po tome dekompozicija i dobila ime Tenzorski vlak ([25]).

Teorem 2.4.2. *([25]) Ako za svaku matrizaciju, matrica* $\mathcal{A}_{[k]}$ *forme kao u* (1.6)*, tenzora* $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ *sa*

$$\operatorname{rang}(\mathcal{A}_{[k]}) = r_k, \tag{2.24}$$

tada postoji dekompozicija (2.22) *s TT-rangovima ne većim od* r_k*.*

Dokaz. Za početak uzmimo matrizaciju, u prvom modu, tenzora $\mathcal{A}$ i njezinu dijadnu (eng. *dyadic*) dekompoziciju $\mathcal{A}_{[1]} = UV^T$, koja zapišemo s indeksima:

$$\mathcal{A}_{[1]}(i_1, \overline{i_2 i_3 \dots i_N}) = \sum_{\alpha_1=1}^{r_1} U(i_1, \alpha_1) V(\alpha_1, \overline{i_2 \dots i_N}),$$

kako je $\operatorname{rang}(\mathcal{A}_{[1]}) = r_1$.

Kako je matrica U pravokutna, a nama treba kvadratna da bi došli do inverza pomnožimo početnu jednadžbu u U^T slijeva:

$$U^T \mathcal{A}_{[1]} = U^T U V^T.$$

Sada znamo da postoji inverz od $U^T U$ pa pomnožimo jednadžbu s lijeva s inverzom:

$$(U^T U)^{-1} U^T \mathcal{A}_{[1]} = (U^T U)^{-1} (U^T U) V^T = V^T. \tag{2.25}$$

Sada, prva TT-jezgra je dana s matricom G_1 tako da $G_1(i_1, \alpha_1) = U(i_1, \alpha_1)$. Iz (2.25) vidimo da ako transponiramo lijevu i desnu stranu, za matricu V dobivamo:

$$V = \mathcal{A}_{[1]}^T U \left(U^T U\right)^{-1} = \mathcal{A}_{[1]}^T W$$

ili ako zapišemo s indeksima

$$V\left(\alpha_1, \overline{i_2 \dots i_N}\right) = \sum_{i_1=1}^{d_1} \mathcal{A}(i_1, \dots, i_N)\, W(i_1, \alpha_1)\,.$$

Ako sada gledamo na matricu V kao na tenzor $(N-1)$ reda onda:

$$\mathcal{V}\left(\overline{\alpha_1 i_2}, i_3, \dots, i_N\right) = V\left(\alpha_1, \overline{i_2 \dots i_N}\right).$$

Sada pogledajmo matrizaciju $\mathcal{V}_{[k]}\left(\overline{\alpha_1 i_2 \dots i_k}, \overline{i_{k+1} \dots i_N}\right)$ za $k = 2, \dots, N$. Može se pokazati da vrijedi $\operatorname{rang}(\mathcal{V}_{[k]}) \leq r_k$. Kako je sada zadovoljeno (2.24),

$$\mathcal{A}_{[k]}\left(\overline{i_1 \dots i_k}, \overline{i_{k+1} \dots i_N}\right) = \sum_{\beta=1}^{r_k} F\left(\overline{i_1 \dots i_k}, \beta\right) G\left(\beta, \overline{i_{k+1} \dots i_N}\right).$$

Koristeći ovo upravo dobiveno imamo:

$$\begin{aligned}
\mathcal{V}_{[k]}\left(\overline{\alpha_1 i_2 \dots i_{k+1}}, \overline{i_{k+2} \dots i_N}\right) &= \sum_{i_1=1}^{d_1} \mathcal{A}_{[k]}\left(\overline{i_1 \dots i_k}, \overline{i_{k+1} \dots i_N}\right) W(i_1, \alpha_1) \\
&= \sum_{i_1=1}^{d_1} \sum_{\beta=1}^{r_k} F\left(\overline{(i_1 \dots i_k}, \beta\right) G\left(\beta \overline{i_{k+1} \dots i_N}\right) W(i_1, \alpha_1) \qquad (2.26) \\
&= \sum_{\beta=1}^{r_k} H\left(\overline{\alpha_1 i_2 \dots i_k}, \beta\right) G\left(\beta \overline{i_{k+1} \dots i_N}\right),
\end{aligned}$$

gdje

$$H\left(\overline{\alpha_1 i_2 \dots i_k}, \beta\right) = \sum_{i_1=1}^{d_1} F\left(\overline{i_1 \dots i_k}, \beta\right) W(i_1, \alpha_1)\,.$$

Iz (2.26) imamo da je $\operatorname{rang}(\mathcal{V}_{[k]}) \leq r_k$, za $k = 1, \dots, N$. Pogledajmo sada matrizaciju $\mathcal{V}_{[1]}$, imamo:

$$\mathcal{V}_{[1]}\left(\overline{\alpha_1 i_2}, \overline{i_3 \dots i_N}\right) = \sum_{\alpha_2=1}^{r_2} G_2\left(\overline{\alpha_1 i_2}, \alpha_2\right) V'(\alpha_2, i_3 \dots i_N)\,.$$

Sada samo postavimo $(\mathcal{G}_2)_{[2]} = G_2$ i ovaj postupak ponovimo za ostale modove. Tim postupkom dobivamo ostale jezgrene tenzore $\mathcal{G}_k$, za $k = 3, \dots, N$. □

Algorithm 4 TT-SVD[28]

Ulaz: Tenzor $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$

1: Izračunaj dimenziju matrizacije u prvom modu od $\mathcal{X}$, X_1:

$$N_l = d_1, N_r = \prod_{k=2}^{N} d_k;$$

2: spremi originalni tenzor u pomoćnu varijablu $\mathcal{M}_1 = \mathcal{X}$;
3: matriziraj $\mathcal{M}_1$ u dobivene dimenzije: $M_1 = \text{reshape}(\mathcal{M}_1, [N_l, N_r])$;
4: izračunaj mršavi SVD matrice M_1, $M_1 = U\Sigma V^T$;
5: Postavi prvi jezgreni tenzor $G_1 := U$;
6: izračunaj $M_2 = \Sigma V^T = U^T M$ i neka je $[\sim, r_1] = \text{size}(U)$;
7: **for** $k = 2$ to $N - 1$ **do**
8: Izračunaj:

$$N_l = d_k, N_r = \frac{N_r}{d_k};$$

9: postavi $M_k = \text{reshape}(M_k, [r_{k-1} \cdot N_l, N_r])$;
10: izračunaj mršavi SVD od M_k, $M_k = U\Sigma V^T$;
11: postavi $G_k = \text{reshape}(U, [r_{k-1}, d_k, r_k])$;
12: Izračunaj:

$$M_{k+1} = U^T M_k, [\sim, r_k] = \text{size}(U);$$

13: **end for**
14: $G_N := M_N$;

Skraćeni TT-SVD

Kada je riječ o HOSVD u Tenzorskom vlaku (TT), često se koriste strategije smanjenja ranga. Umjesto da računamo točan SVD, možemo odrediti najbolju SVD aproksimaciju s rangom r_k. Kroz ovaj pristup se uvodi određena pogreška, ali ju možemo procijeniti, dok se uspjeh aproksimacije može analizirati prema rezultatima koje navodi sljedeći teorem.

Teorem 2.4.3. *([25] Neka je dan tenzor $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ takav da njegova problikovanja tenzora $\mathcal{A}_{[k]}$ mogu biti aproksimirani matricom nižeg ranga $\hat{A}_k$*

$$\mathcal{A}_{[k]} = \hat{A}_k + E_k, \quad \text{rang}(\hat{A}_k) = r_k, \quad \|E_k\| = \epsilon_k, \quad k = 1, \ldots, N-1.$$

Tada TT-SVD računa tenzor $\mathcal{B}$ u TT-formatu s TT-rangovima r_k i vrijedi

$$\|\mathcal{A} - \mathcal{B}\|_F \leq \sqrt{\sum_{k=1}^{N-1} \epsilon_k^2}.$$

Ako sada primijenimo teorem 2.4.3 možemo napraviti varijantu algoritma 4 u kojem koristimo skraćeni SVD umjesto originalnog. Možemo vidjeti kako izgleda takav algoritam u 5.

Algorithm 5 Skraćeni TT-SVD[25]

Ulaz: Tenzor $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, željena točnost ϵ

1: Izračunaj parametar skraćivanja $\delta = \frac{\epsilon}{\sqrt{d-1}} \|\mathcal{A}\|_F$;
2: Izračunaj dimenziju matrizacije u prvom modu od $\mathcal{X}$, $X_{[1]}$:

$$N_l = d_1, N_r = \prod_{k=2}^{N} d_k;$$

3: Spremi orginalni tenzor u pomoćnu varijablu $\mathcal{M}_1 = \mathcal{X}$;
4: Matriciziraj $\mathcal{M}_1$ u dobivene dimenzije: $M_1 = \text{reshape}(\mathcal{M}_1, [N_l, N_r])$;
5: Izračunaj skraćeni SVD matrice M_1, $M_1 = U_1 \Sigma_1 V_1^T + E_1$, gdje je $\|E_1\|_F \leq \delta$ i vrijedi $r_1 = \text{rang}(U_1)$;
6: Postavi prvi jezgreni tenzor $G_1 := U$;
7: Izračunaj $M_2 = \Sigma_1 V_1^T = U_1^T M$;
8: **for** $k = 2$ to $N - 1$ **do**
9: Izračunaj:

$$N_l = d_k, N_r = \frac{N_r}{d_k};$$

10: Postavi $M_k = \text{reshape}(M_k, [r_{k-1} \cdot N_l, N_r])$;
11: Izračunaj skraćeni SVD od M_k, $M_k = U_k \Sigma_k V_k^T + E_k$, gdje je $\|E_k\|_F \leq \delta$ i vrijedi $r_k = \text{rang}(U_k)$;
12: Postavi $\mathcal{G}_k = \text{reshape}(U_k, [r_{k-1}, d_k, r_k])$;
13: Izračunaj: $M_{k+1} = U_k^T M_k$;
14: **end for**
15: $G_N := M_N$;
16: **return** TT-jezgre $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times d_k \times r_k}$ za $k = 1, \ldots, N$

Korolar 2.4.4. *([25] Neka je dan tenzor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_N}$ i granice ranga r_k, tada najbolja aproksimacija od $\mathcal{A}$ u Frobenijusovoj normi s TT-rangovima ograničenima s r_k uvijek postoji, označimo je s $\mathfrak{A}$, i TT-aproksimacija koja je izračunata s TT-SVD algoritmom je kvazi optimalna, tj. vrijedi:*

$$\|\mathcal{A} - \mathcal{B}\|_F \leq \sqrt{(N-1)} \|\mathcal{A} - \mathfrak{A}\|_F.$$

Primjene

Tenzorski vlak koristi se u mnogim znanstvenim granama [5], ponajviše u fizici i kemiji. U kemiji se koristi za efikasno rješavanje visoko-dimenzionalnih kemijskih master jednadžbi u kemijskim reakcijskim mrežama. U kontekstu kemijskih reakcija, TT dekompozicija omogućuje reprezentaciju i manipulaciju složenih prostora stanja sustava smanjujući dimenzionalnost problema. To omogućuje brže rješavanje kemijskih master jednadžbi i analizu ponašanja kemijskih sustava. Kroz primjere, TT dekompozicija pokazuje svoju učinkovitost u analizi vremenski ovisne progresije i simulaciji kemijskih procesa poput izražavanja gena i prijenosa signala. S obzirom na niske brojeve interakcija i reakcija u kemijskim sustavima, TT dekompozicija omogućuje precizno i brzo modeliranje složenih kemija, otvarajući put za daljnje istraživanje i razumijevanje kemijskih procesa u prirodi.

S druge strane, u fizici, TT dekompozicija koristi se prilikom analize dinamike fluida te se može primijeniti kako bi se učinkovito rješavali problemi strujanja fluida. U ovom slučaju, TT dekompozicija se može koristiti za aproksimaciju i diskretizaciju varijabli koje se pojavljuju u Navier-Stokesovim jednadžbama, poput brzine fluida i tlaka, na računalnoj mreži. Iskorištavanjem niske rang-reprezentacije TT dekompozicije omogućuje se efikasno rješavanje problema s visokom dimenzionalnošću, otvarajući mogućnosti za naprednu analizu i simulaciju fluidnih sustava.

Poglavlje 3

Implementacija i testiranje

U ovom poglavlju koristit ćemo Tuckerovu dekompoziciju kao algoritam za prepoznavanje akcije koji je opisan u [17]. Na kraju ćemo pokazati rezultate algoritma na nekim podacima.

3.1 Uvod

Prepoznavanje ljudskih radnji je izazovan zadatak u računalnom vidu. Značajne prostorne informacije o ljudskim radnjama odnose se na oblik i izgled ljudskog tijela, na primjer, položaj ruke u serviranju tenisa često je viši nego u udarcu teniskim reketom. Slično, položaj noge kod udarca često je viši nego kod trčanja. Ove prostorne informacije su uključene u prirodne reprezentacije vizualnih podataka. Tenzorska reprezentacija može sačuvati prostorne informacije jer tenzor može biti smatran prirodnim reprezentacijama vizualnih podataka.

U prošlim desetljećima su razvijeni brojni algoritmi čiji su ulazni podatci vektori, poput K najbližih susjeda (KNN), potpornog vektorskog stroja (SVM) i *ridge* regresije (RR). Međutim, kada se primjenjuju tenzorski podatci, oni se moraju pretvoriti u vektorske podatke. Jedan uobičajen način je slaganje elemenata tenzorskih podataka u više ili manje proizvoljnom redoslijedu. Doduše, ovi vektorizirani pristupi mogu stvoriti nekoliko problema u obradi tenzorskih podataka. Prvo, kada se tenzorski podaci preoblikuju u vektor, često dobivamo vektor potencijalno vrlo visoke dimenzionalnosti. To može uzrokovati probleme poput prenaučenosti ili velikih zahtjeva za memorijom. Drugo, kada se tenzor proširi kao vektor, prostorna struktura i korelacija u obliku izgleda se zanemaruju.

Da bi se riješili ovi problemi, nedavno su razvijeni algoritmi koji koriste tenzorske reprezentacije umjesto vektorskih. U ovom radu koristi se HOG opisnik ulazne slike prikazane kao tenzor reda 3 i predlaže se algoritam za učenje tenzora kako bi se izravno obradio taj tenzor.

Za bolje iskorištavanje prostornih informacija HOG tenzora koristimo Tuckerovu dekompoziciju za dekomponiranje parametarskog tenzora. Prednosti Tuckerove dekompozicije u odnosu na CP dekompoziciju je da ne moramo odrediti rang dekomponiranog tenzora na početku samog algoritma i možemo odabrati koje prostorne informacije su nam bitnije od ostalih duž svakog reda, tako što podešavamo dimenzije jezgrenog tenzora.

Predloženi algoritam ide sljedećim redosljedom. Prvo parametarski tenzor reda tri dekomponiramo u jezgreni tenzor pomnožen s matricama umnoška u tri moda. Onda u svakoj iteraciji jezgreni tenzor i matrice umnoška dobivamo rješavanjem *ridge* regresije, ali tako da prvo fiksiramo jezgreni tenzor i dvije matrice umnoška pa izračunamo za tu jednu slobodnu matricu, onda fiksiramo tu slobodnu, a iduća matrica umnoška postaje slobodna. Nakon što smo dobili sve matrice umnoška računamo i jezgreni tenzor. Ovaj postupak ponavljamo do konvergencije.

Algoritam učinkovito istražuje prostorne informacije HOG tenzora tijekom postupka učenja i time poboljšava performanse prepoznavanja. Ova metoda se naziva *Tucker Ridge Regression (TuRR).*

3.2 Opis algoritma

Objasnimo sad detaljnije algoritam Tuckerove *ridge* regresije (TuRR). Za funkciju gubitka, u problemu regresije, odaberimo funkciju najmanjih kvadrata. Tada, koristeći $\ell_2 - normu$ dobivamo:

$$\min_{w,b} \sum_{i=1}^{n} (\langle x_i, w \rangle - y_i + b)^2 + \lambda \|w\|_2^2 , \tag{3.1}$$

gdje je x_i opisnik slike u vektorskom obliku, w je vektor parametar, y_i je izlaz za podatak x_i, te je b pomak, a λ regularizacijski parametar.

Sada želimo transformirati jednadžbu (3.1) da umjesto vektora koristimo tenzore. Definirajmo sada $\mathcal{X} = [\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n] \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_M \times n}$, gdje nam tenzor M-tog reda $\mathcal{X}_i$, predstavlja i-ti opisnik slike, a n nam je broj slika koje su nam dostupne za trening. Sada još definiramo $y = [y_1, y_2, \ldots, y_n] \in \{0, 1\}^{n \times 1}$, gdje je $y_i = 1$ ako je i-ti podatak pozitivan primjerak i $y_i = 0$ inače. Dobili smo *ridge* regresiju s tenzorima:

$$\min_{\mathcal{W},b} \sum_{i=1}^{n} (\langle \mathcal{X}, \mathcal{W} \rangle - y_i + b)^2 + \lambda \|\mathcal{W}\|_F^2 , \tag{3.2}$$

gdje je $\mathcal{W} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_M}$ parametar tenzor, λ regularizacijski parametar i b pomak. Sada je cilj naučiti parametre $\mathcal{W}$ i b. Napravimo Tuckerovu dekompoziciju tenzora $\mathcal{W}$ (kao u (2.13)) i zapišimo taj rastav skraćeno:

$$\mathcal{W} = \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 \cdots \times_n U_n = [\![\mathcal{G}; U_1, U_2, \ldots, U_M]\!], \tag{3.3}$$

pa sada jednadžbu (3.2) možemo zapisati kao:

$$\min_{\mathcal{G}, U_1, \ldots, U_M, b} \sum_{i=1}^{n} (\langle \mathcal{X}, [\![\mathcal{G}; U_1, U_2, \ldots, U_M]\!] \rangle - y_i + b)^2 + \lambda \left\| [\![\mathcal{G}; U_1, U_2, \ldots, U_M]\!] \right\|_F^2 . \tag{3.4}$$

Funkcija cilja (3.4) nažalost nije zajednički konveksna za sve elemente od $\mathcal{W}$ nakon Tuckerove dekompozicije. Da bi se riješio ovaj problem predlaže se alternirajući optimizacijski algoritam, gdje u svakom koraku, rješavamo konveksni optimizacijski problem za jedan element dok su ostali fiksirani.

Znači ideja je da prvo rješavamo jednadžbu za fiksni U_k redom za $k = 1, ..., M$, te onda za $\mathcal{G}$ i tako u krug do konvergencije. Za lakše zapisivanje označimo s W_k, G_k, X_i^k, matricizirane tenzore $\mathcal{W}, \mathcal{G}, \mathcal{X}_i$ u k-tom modu, redom.

Jednadžba (3.2), za matricizirane tenzore u k-tom modu, može biti zapisana kao:

$$\min_{W_k, b} \sum_{i=1}^{n} \left(\mathrm{Tr}\left(W_k X_i^k \right) - y_i + b \right)^2 + \lambda \left\| W_k \right\|_F^2 . \tag{3.5}$$

Da bi dalje raspisivali ovu formulu uvedimo još par oznaka. Kronekerov produkt od M matrica označimo s:

$$U_\otimes = U_M \otimes \cdots \otimes U_1.$$

Također, definirajmo i $\widetilde{U}_k$ kao:

$$\widetilde{U}_k = U_M \otimes \cdots \otimes U_{k+1} \otimes U_{k-1} \otimes \cdots \otimes U_1.$$

Sada slijedi:

$$\begin{aligned} W_k &= U_k G_k (U_M \otimes \cdots \otimes U_{k+1} \otimes U_{k-1} \otimes \cdots \otimes U_1)^T \\ &= U_k G_k \tilde{U_k}^T. \end{aligned} \tag{3.6}$$

Ako sada zamijenimo W_k u (3.5) sa 3.6, i fiksiramo jedan $U_l|_{l=1, l\neq k}^{M}, G_k$, onda dobivamo:

$$\min_{U_k, b} \sum_{i=1}^{n} \left(\mathrm{Tr}\left(U_k G_k \widetilde{U}_k^T X_i^k \right) - y_i + b \right)^2 + \lambda \, \mathrm{Tr}\left(U_k G_k \widetilde{U}_k^T \widetilde{U}_k G_k^T U_k^T \right). \tag{3.7}$$

Definirajmo sada $\widetilde{X}_i^{kT} = G_k \widetilde{U}_k^T X_i^{kT}$ i $D_k = G_k \widetilde{U}_k^T \widetilde{U}_k G_k^T$, pa gornja jednadžba postaje:

$$\min_{U_k, b} \sum_{i=1}^{n} \left(\mathrm{Tr}\left(U_k \widetilde{X}_i^{kT} \right) - y_i + b \right)^2 + \lambda \, \mathrm{Tr}\left(U_k D_k U_k^T \right). \tag{3.8}$$

Sada definirajmo:

$$\mathrm{Tr}\left(U_k \widetilde{X}_i^{kT}\right) = \left[\mathrm{vec}\,(U_k)^T\ b\right]\left[\mathrm{vec}\left(\widetilde{X}_i^k\right)^T\ 1\right]^T = \mathbf{v}_k^T \widehat{\mathbf{x}}_i,$$

$$\widetilde{D}_k = \begin{bmatrix} D_k \otimes I_{d_k \times d_k} & 0 \\ 0 & 1 \end{bmatrix},$$

pa (3.8) postaje:

$$\min_{\mathbf{v}_k} \sum_{i=1}^{n} \left(\mathbf{v}_k^T \widehat{\mathbf{x}}_i - y_i\right)^2 + \lambda\, \mathrm{Tr}\left(\mathbf{v}_k^T \widetilde{D}_k \mathbf{v}_k\right).$$

Označimo sada $\widehat{X} = [\widehat{\mathbf{x}}_1, \widehat{\mathbf{x}}_2, \ldots, \widehat{\mathbf{x}}_n]$ i $\mathbf{y} = [y_1, y_2, \ldots, y_n]$, te dobivamo:

$$\min_{\mathbf{v}_k} \mathrm{Tr}\left(\mathbf{v}_k^T \left(\widehat{X}\widehat{X}^T + \lambda \widetilde{D}_k\right)\mathbf{v}_k\right) - 2\,\mathrm{Tr}\left(\mathbf{v}_k^T \widehat{X}\mathbf{y}^T\right) + \mathrm{Tr}\left(\mathbf{y}^T\mathbf{y}\right). \tag{3.9}$$

Sada smo dobili da optimizacijski problem za U_k, b u jednadžbi (3.7) formuliran kao problem *ridge* regresije s obzirom na $\mathbf{v}_k$. Deriviranjem (3.9) s obzirom na $\mathbf{v}_k$ i izjednačavanjem s nulom dobivamo:

$$\begin{aligned} &2\left(\widehat{X}\widehat{X}^T + \lambda \widetilde{D}_k\right)\mathbf{v}_k - 2\widehat{X}^T\mathbf{y}^T = 0 \\ &\Rightarrow \mathbf{v}_k = \left(\widehat{X}\widehat{X}^T + \lambda \widetilde{D}_k\right)^{-1} \widehat{X}_{\mathbf{y}}{}^T. \end{aligned} \tag{3.10}$$

Nakon što smo dobili rješenje za $\{U_1, U_2, \ldots, U_M\}$, zato jer se jezgreni tenzor $\mathcal{G}$ može matricizirati u bilo kojem modu (naravno $k \leq M$), računat ćemo za $\mathcal{G}$ matriciziran u prvom modu, tj. G_1, jer se takav pristup koristio i u [17]. Sada prvo vidimo vektorizirani oblik jednadžbe 3.6:

$$\mathrm{vec}\,(W_k) = U_\otimes\, \mathrm{vec}\,(G_k),$$

pa specijalno imamo i $\mathrm{vec}\,(W_1) = U_\otimes\, \mathrm{vec}\,(G_1)$, te uvrštavanjem u (3.5)

$$\min_{G_1, b} \sum_{i=1}^{n} \left(\mathrm{vec}\,(G_1)^T\, U_\otimes^T\, \mathrm{vec}\left(X_i^k\right) - y_i + b\right)^2 + \lambda\, \mathrm{vec}\,(G_1)^T\, U_\otimes^T U_\otimes\, \mathrm{vec}\,(G_1). \tag{3.11}$$

Sada opet slijedi slični postupak, označimo

$$\bar{\mathbf{x}}_i = \left[\left(U_\otimes\, \mathrm{vec}\left(X_i^k\right)\right)^T\ 1\right], \quad \mathbf{g}_1 = \left[(\mathrm{vec}\,(G_1))^T\ b\right],$$

$$D = \begin{bmatrix} U_\otimes^T U_\otimes & 0 \\ 0 & 1 \end{bmatrix} \quad \text{i} \quad \bar{X} = [\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_n].$$

Pa sada (3.11) može biti prikazana kao problem *ridge* regresije o g_1:

$$\min_{\mathbf{gin}_1, b} \operatorname{Tr}\left(\mathbf{g}_1^T \left(\bar{X}\bar{X}^T + \lambda D\right)\mathbf{g}_1\right) - 2\operatorname{Tr}\left(\left(\mathbf{g}_1^T \bar{X}^T\right) + \operatorname{Tr}\left(\mathbf{y}^T\mathbf{y}\right)\right). \tag{3.12}$$

Deriviranjem gornje jednadžbe, s obzirom na $\mathbf{g}_1$, i izjednačavanjem s nulom, imamo:

$$\begin{aligned} &2\left(\bar{X}\bar{X}^T + \lambda D\right)\mathbf{g}_1 - 2\bar{X}^T\mathbf{y}^T = 0 \\ &\Rightarrow \mathbf{g}_1 = \left(\bar{X}\bar{X}^T + \lambda D\right)^{-1}\bar{X}\mathbf{y}^T. \end{aligned} \tag{3.13}$$

Optimiziramo $\{U_1, U_2, \ldots, U_M; \mathcal{G}\}$ tako što ponavljamo ove korak redom do konvergencije. Detaljno raspisan proces iteriranja je dan u 6.

Algorithm 6 Tuckerova *ridge* regresija

Input:

Ulazni tenzori $\mathcal{X}_i$, $i = 1, \ldots, n$. $\mathcal{X}_i \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_M}$ i labele $Y \in \mathbb{R}^{c \times n}$, gdje je c broj klasa akcija, a n broj primjera. Dimenzije $R_1, R_2, \ldots, R_M$ za jezgreni tenzor.

Output: Matrice za svaku klasu $\left\{U_1^r, U_2^r, \ldots, U_M^r, G_1^r, b^r\right\}\Big|_{r=1}^{c}$

for $r = 1$ to c **do**
  for $i = 1$ to n **do**
    if $Y(r, i) = c, y(r, t) = 1$; else, $y(r, 1) = -1$.
  end for
  Postavi $t = 0$ i inicijaliziraj matrice $U_1^y, U_2^r, \ldots, U_M^r, G_1^y$ s nasumičnim vrijednostima.
  repeat
    $r = t + 1$
    for $k = 1$ to M **do**
      Izračunaj $\mathbf{v}_k$ po (3.10).
    end for
    Izračunaj $\mathbf{g}_1$ po (3.13).
  until Convergence
end for
return $\left[U_1^r, U_2^r, \ldots, U_{M^+}^r, G_1^r, b^r\right]_{\mathrm{r}=1}^{c}$.

Jednom kada imamo $U_1, U_2, \ldots, U_M, G_1, b$ za c klasa, uz pomoć 7, dolazimo do odgovarajućih oznaka za određeni testni podatak.

Algorithm 7 Proces prepoznavanja

Input:

Ulazni tenzori (testni podaci) $\mathcal{X}_i$, $i = 1, \ldots, n$. $\mathcal{X}_i \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_M}$, set parametara $\{U_1^r, U_2^r, \ldots, U_M^r, G_1^r, b^r\}\Big|_{r=1}^{c}$ za svaku od c klasa.

Output: Predviđene oznake **y** za testne podatke

1: **for** $r = 1$ to c **do**
2: Izračunaj tenzor parametar za r-tu klasu, po formuli: $\mathcal{W}^r = \mathcal{G}^r \times_1 U_1^r \times_2 U_2^r \times_3 \cdots \times_M U_M^r$
3: **end for**
4: **for** $i = 1$ to n **do**
5: Izračunaj predviđenu oznaku za $\mathcal{X}_i$ za

$$y_i = \arg\max_r \left(\langle \mathcal{X}_i, \mathcal{W}^r \rangle + b^r\right)\Big|_{r=1}^{c}$$

6: **end for**
7: **return** Prepoznate oznake **y**

3.3 Podaci za treniranje i testiranje

Korišteni podaci za treniranje i testiranje opisanog algoritma su skup podataka pod nazivom *People Playing Musical Instrument (PPMI)* ([32]). U skupu podataka se nalaze slike ljudi koji sviraju sedam različitih instrumenata. Instrumenti su: fagot, erhu, flauta, francuski rog, gitara, saksofon i violina. Za svaki instrument dostupno je po sto slika za treniranje i isto toliko za testiranje. Možemo vidjeti kako izgledaju slike na 3.1. Svaka slika je preoblikovana na dimenzije 128×128 piksela.

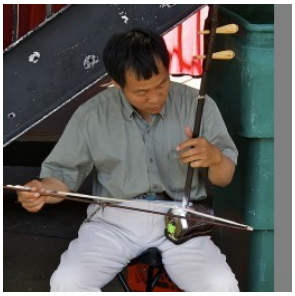
Prije ulaza slika u glavni algoritam postoji predobrada podataka koja se odvija u dvije faze. U prvoj fazi svaka se slika provuče kroz algoritam za detekciju rubova. Za testiranje odabrani algoritmi za detekciju ruba su:

1. *Canny* algoritam za detekciju ruba, jedan od najpoznatijih algoritama za detekciju ruba. Poznat po vrlo točnoj detekciji ruba i minimizaciji šuma.

2. *Differenece of Gaussians (DOG)* algoritam uključuje konvoluciju slike s dvije Gaussove funkcije različitih veličina kako bi se dobila aproksimacija gradijenta slike.

3. *EdgeBoxes* je algoritam koji uzima u obzir i snagu ruba i povezanost ruba kako bi identificirao regije koje vjerojatno sadrže objekte.

(a) Fagot



(b) Erhu

(c) Flauta

(d) Gitara

(e) Saksofon

(f) Violina

Slika 3.1: Slike iz skupa podataka PPMI bez ikakve obrade.

4. *Laplacian of Gaussian (LOG)* slično kao DOG, samo koristi Laplaceovu Gaussovu funkciju.

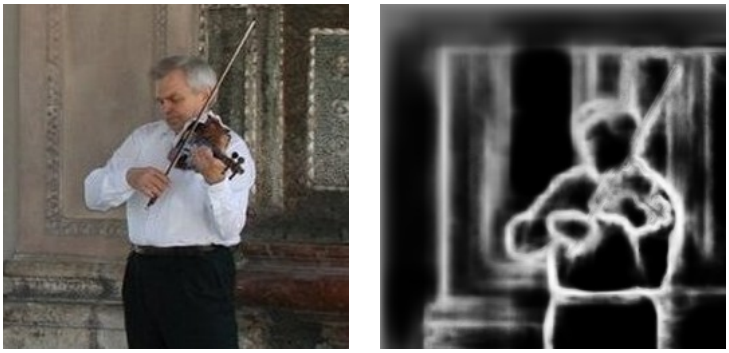
5. *Pb* operatori je algoritam koji procjenjuje vjerojatnost prisutnosti granice na svakom pikselu slike. Ovaj algoritam je korišten i u radu.

6. *Sobel* operator je jednostavan i često korišten algoritam koji izračuna magnitudu gradijenta slike konvolucijom s dvije male matrice filtera u okomitom i vodoravnom smjeru.

7. *Holistically-nested edge (HED)* je model dubokog učenja koji koristi potpuno spojene konvolucijske neuronske mreže za predikciju ruba.

Od svih algoritama na kraju je HED dao najbolje rezultate, pa se on i koristio u konačnom testiranju i na slici 3.2 je moguće vidjeti kako izgleda slika prije ulaza u algoritam i kako nakon izlaza iz algoritma.

Nakon što smo sve slike provukli kroz HED algoritam slijedi druga faza pripreme podataka, a to je da se za svaku sliku dobije njezin HOG opisnik.



(a) Orginalna slika (b) Slika nakon

Slika 3.2: Slika prije i poslje prolaska kroz algoritam za detekciju ruba, točnije HED algoritam

HOG

Histogram of Oriented Gradiens (HOG) je popularna tehnika za dobiti opisnik značajki korišten u računalnom vidu i obradi slike. Ukratko, metoda računa veličinu gradijenta i orijentacije za svaki piksel na slici i zatim dijeli cijelu sliku u manje ćelije u kojima onda gradi histogram orijentacija gradijenata kako bi se predstavila raspodjela gradijenata u cijeloj jednoj ćeliji.

Na slici 3.3 možemo vidjeti kako izgleda slika prije i nakon primjene HOG metode na sliku. Kao što vidimo na slici preostaju samo bitnije informacije o rubovima i samim oblicima, dok su nepotrebne informacije (u ovom kontekstu gdje nam je cilj prepoznati objekte), poput boje, osvjetljenja itd. zanemarene.

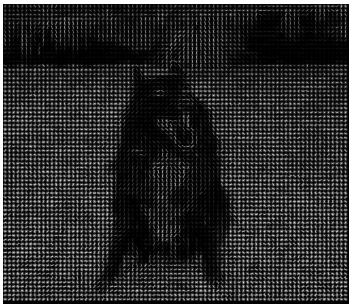
HOG metoda ima nekoliko prednosti:

- invarijantna je na osvjetljenje, lokalne teksture i geometrijske deformacije
- efikasna je i može se primijeniti na različite vrste objekata

Sama HOG metoda ima nekoliko parametara koje je moguće prilagoditi našem skupu podataka i našem cilju, a neki od njih su:

- *broj orijentacija*: Broj smjerova iz kojih računamo gradijente
- *veličina ćelije*: koliko piksela se nalazi u svakoj ćeliji
- *veličina bloka*: koliko ćelija čine jedan blok


(a) Originalna slika


(b) HOG metoda

Slika 3.3: Slika prije i nakon prolaska kroz HOG algoritam

- *veličina preklapajućih blokova*
- *pomak*: koliko su pomaknute ćelije od ćelije

U radu se koriste nepreklapajući blokovi dimenzije 16×16, s 4 orijentacije, pa je svaka slika prikazana kao tenzor dimenzija $16 \times 16 \times 16$, dok smo u ovom radu koristili veličinu ćelija 8×8 piksela, s preklapajućim blokovima dimenzija 2×2 i također 4 orijentacije, pa se dobivamo tenzor dimenzija $15 \times 15 \times 16$.

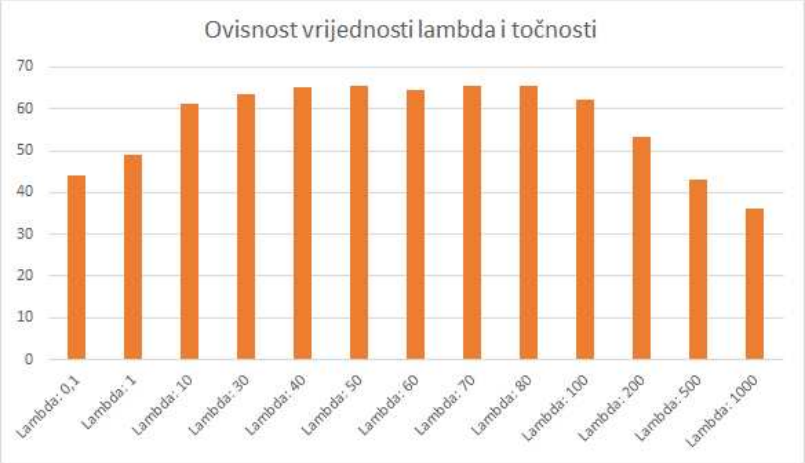
Konačno, nakon što sve slike prođu kroz obje faze dolazimo do konačnog skupa podataka koji se onda koristi u algoritmu.

3.4 Rezultati

Za testiranje točnosti algoritama dostupno nam je po sto slika za svaki instrument, tj. ukupno 700 slika. Postoje dva hiperparametra koje su u radu [17] namještali pa ćemo ih i mi istestirati.

Prvi od njih je λ. Za vrijednosti koje testiramo ćemo uzeti: 0.1, 1, 10, 30, 40, 50, 60, 70, 80, 100, 200, 500, 1000. Na grafu 3.4 vidimo da najbolje rezultate dobijemo za $\lambda \in \{50, 70, 80\}$ dok su u [17] najbolje rezultate dobili za $\lambda = 100$. Zasad nastavljamo testiranje s $\lambda \in \{50, 70, 80\}$.

Sljedeći parametar koji testiramo je R. R određuje dimenziju jezgrenog tenzora, a onda i jednu dimenziju matrica umnoška. Na grafu 3.5 je prikazana točnost u ovisnosti o vrijednosti R-a, dok je $\lambda = 50$. U slučaju da smo dobili više R-ova koji daju istu vrijednost birali bi onaj manji. To nam je indirektni cilj, da nam jezgreni tenzor bude što manje dimenzije, jer tako kompresiramo podatke.

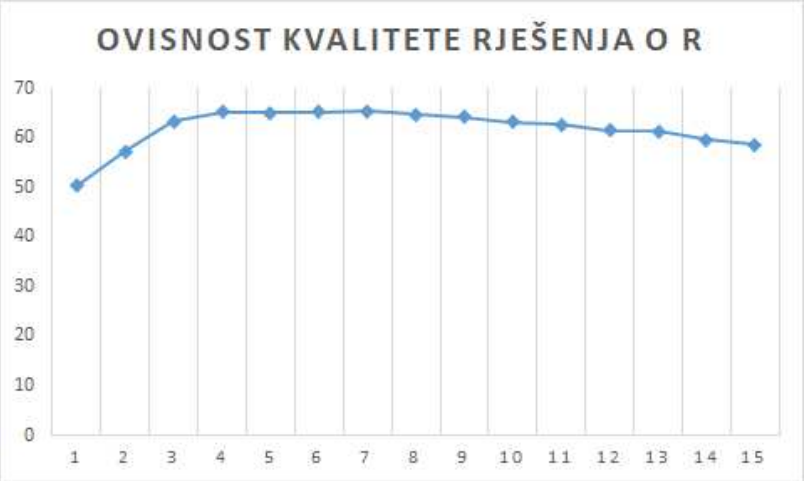


Slika 3.4: Ovisnost kvalitete rješenja i hiperparametra λ

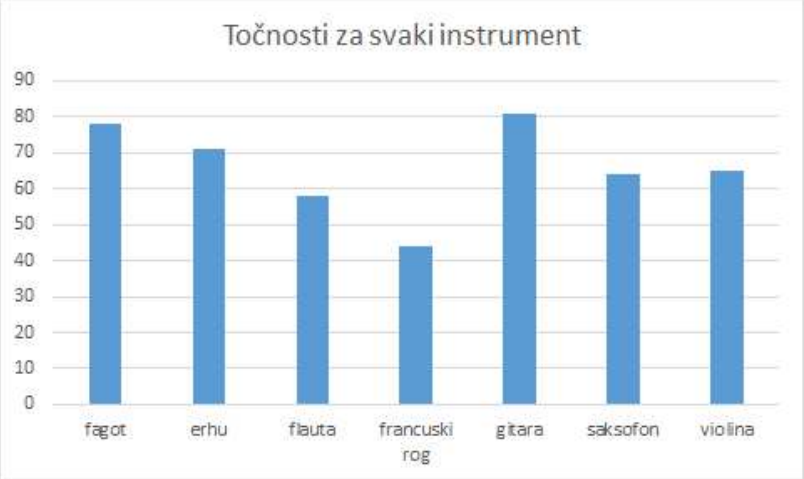
Na kraju odabiremo $\lambda = 50$ i R = 7 te je broj slika instrumenata koje smo uspiješno klasificirali, s tako odabranim hiperparametrima, 485. Možemo reći da nam algoritam radi s točnosti **65.43%** s obzirom na ovaj skup testnih podataka.

U radu [17] su dobili točnost od 71.57%, ali kako nisu točno objasnili parametre HOG opisnika, tome pripisujemo razlike u točnosti. U radu spominju još i algoritme koji se baziraju na vektorskom prikazu podataka i navode podatak da je njihova točnost 64.57%.

Za kraj je moguće još vidjeti na grafu 3.6 kolika je točnost po svakom instrumentu i tu se vide očekivani rezultati. Algoritam lakše razaznaje one instrumente koji imaju specifičan oblik i držanje npr. gitara i fagot.



Slika 3.5: Ovisnost kvalitete rješenja od hiperparametru R



Slika 3.6: Točnost algoritama za svaki instrument posebno

Poglavlje 4

Dodatak - implementacija u Python-u

```
1  from skimage.feature import hog
2  from skimage import io
3  from skimage.transform import resize
4  import numpy as np
5  import tensorly as tl
6  from tensorly.tenalg import mode_dot, inner
7  import os
8
9  # Kreiranje skupa podataka za treniranje
10 filenames = ["flute", "erhu", "bassoon", "frenchhorn", "guitar",
       "saxophone", "violin"]
11 cnt_all_img = 0
12 dataset = []
13 for i in range(len(filenames)):
14     print("Dohvat slika:", filenames[i])
15     for file in os.listdir("./dataset_hed/" + filenames[i] + "/
           train"):
16         image = io.imread("dataset_hed/" + filenames[i] + "/train
               /" + file)
17         image = resize(image, (128,128))
18         image = image / np.max(image)
19         hog_image = hog(image, orientations=4, pixels_per_cell
               =(8,8), cells_per_block=(2,2), visualize=False,
               multichannel=False, feature_vector=False)
20
21         hog_vector = np.array(hog_image)
22         hog_tensor = np.reshape(hog_vector, (15,15,16))
23
```

```
24          dataset.append(hog_tensor)
25          cnt_all_img += 1
26  print("Ukupan broj slika za treniranje:", cnt_all_img)
27
28
29  # Kreiranje skupa podatak za testiranje
30  cnt_test_img = 0
31  dataset_test = []
32  for i in range(len(filenames)):
33      print("Dohvat slika:", filenames[i])
34      for file in os.listdir("./dataset_hed/" + filenames[i] + "/
            test"):
35          image = io.imread("dataset_hed/" + filenames[i] + "/test/
                " + file)
36          image = resize(image, (128,128))
37          image = image / np.max(image)
38          hog_image = hog(image, orientations=4, pixels_per_cell
                =(8,8), cells_per_block=(2,2), visualize=False,
                multichannel=False, feature_vector=False)
39
40          hog_vector = np.array(hog_image)
41          hog_tensor = np.reshape(hog_vector, (15,15,16))
42
43          dataset_test.append(hog_tensor)
44          cnt_test_img += 1
45  print("Ukupno slika za testiranje:", cnt_test_img)
46
47
48  NUM_INS = 7
49  NUM_OF_TRAIN_FILES = cnt_all_img
50  M = 3
51  nm_iter = 25
52  R = 7
53  lamda = 50
54  dim = [15, 15, 16]
55  G_final = []
56  U_0_final = []
57  U_1_final = []
58  U_2_final = []
59  b_final = []
60  for i in range(NUM_INS):
61
```

```
62      y = np.ones(NUM_OF_TRAIN_FILES)
63      for j in range(NUM_OF_TRAIN_FILES):
64          if (i+1) * 100 <= j or j < i*100:
65              y[j] = -1
66
67      # Incijalizacija G, U_1, U_2, U_3
68      G = tl.tensor(np.random.rand(R, R, R))
69      U = [tl.tensor(np.random.randn(15, R)),tl.tensor(np.random.
            randn(15, R)), tl.tensor(np.random.randn(16, R))]
70      b = 0
71
72      # Racunanje matricizacija od G za brze racunanje
73      G_unfold = [tl.unfold(G, 0), tl.unfold(G, 1), tl.unfold(G, 2)
            ]
74      cnt = 0
75      while True:
76          cnt += 1
77          X_cap = np.zeros(M, dtype=object)
78          for k in range(M):
79              X_cap[k] = np.zeros((dim[k]*R+1, NUM_OF_TRAIN_FILES))
80          for j in range(M):
81              U_tilde = np.kron(U[1],U[2])
82              if j == 1:
83                  U_tilde = np.kron(U[0], U[2])
84              elif j == 2:
85                  U_tilde = np.kron(U[0], U[1])
86              for k in range(NUM_OF_TRAIN_FILES):
87                  X_tilde =  tl.unfold(dataset[k],j) @ U_tilde @ np
                        .transpose(G_unfold[j])
88                  X_cap[j][:,k] = np.append(X_tilde.flatten(), 1)
89              D_j = G_unfold[j] @ np.transpose(U_tilde) @ U_tilde @
                    np.transpose(G_unfold[j])
90              D_tilde_j_help = np.kron(D_j, np.identity(dim[j]))
91              D_tilde_j = np.zeros((dim[j]*R+1, dim[j]*R+1))
92              D_tilde_j[0:dim[j]*R, 0:dim[j]*R] = D_tilde_j_help
93              D_tilde_j[dim[j]*R, dim[j]*R] = 1
94              v =np.linalg.inv(X_cap[j] @ np.transpose(X_cap[j]) +
                    lamda * D_tilde_j)  @ X_cap[j] @ y
95              # Spremanje novog U_j
96              U[j] = np.reshape(v[:dim[j]*R], (dim[j], R))
97              b = v[dim[j]*R]
98
```

```
99          # Racunanje novog G
100         U_kron = np.kron(np.kron(U[0], U[1]), U[2])
101         X_overline = np.zeros((R*R*R + 1, 700))
102
103         for k in range(NUM_OF_TRAIN_FILES):
104             X_tmp_1 = np.transpose(U_kron) @ tl.unfold(dataset[k
                    ],0).flatten()
105             X_overline[:,k] = np.append(X_tmp_1, 1)
106         D = np.zeros((R*R*R+1, R*R*R+1))
107         D[0:R*R*R, 0:R*R*R] = np.transpose(U_kron) @ U_kron
108         D[R*R*R, R*R*R] = 1
109         G_tmp_1 = np.linalg.inv(X_overline @ np.transpose(
                X_overline) + lamda * D) @ X_overline @ y
110         # Spremanje G-a i racunanje matricizacija za sljedecu
                iteraciju
111         G = tl.fold(G_tmp_1[:R*R*R], 0, (R, R, R))
112         G_unfold = [tl.unfold(G, 0), tl.unfold(G, 1), tl.unfold(G
                , 2)]
113
114         if cnt == nm_iter:
115             break
116
117     G_final.append(G)
118     U_0_final.append(U[0])
119     U_1_final.append(U[1])
120     U_2_final.append(U[2])
121     b_final.append(b)
122     print(f"Zavrsio za {filenames[i]}.")
123
124 # Za svaki instrument racunanje W-a
125 c = 7
126 W = np.zeros(c, dtype=object)
127 for i in range(c):
128     W[i] = mode_dot(mode_dot(mode_dot(G_final[i], U_0_final[i],
            0), U_1_final[i], 1), U_2_final[i], 2)
129
130 # Skup podataka za testiranje je slozen tako da
131 # prvo idu sve slike od flaute, pa onda erhua, itd.
132 def classification(a):
133     if a < 100: return 0
134     if a < 200: return 1
135     if a < 300: return 2
```

```
136     if a < 400: return 3
137     if a < 500: return 4
138     if a < 600: return 5
139     return 6
140 
141 # Racunanje konacne tocnosti
142 right_guessed = 0
143 for i in range(cnt_test_img):
144     maximum = -float("inf")
145     label = -1
146     for r in range(c):
147         tmp = inner(dataset_test[i], W[r]) + b_final[r]
148         if tmp > maximum:
149             maximum = tmp
150             label = r
151     if label == classification(i):
152         right_guessed += 1
153 
154 print(f"Konacna tocnost: {round(right_guessed/cnt_test_img*100,2)
    }")
```

Bibliografija

[1] R. Bro i P. Geladi A. Smilde, *Multi-Way Analysis: Applications in the Chemical Science*, Wiley, 2004.

[2] Susan Blackford, *Singular value decomposition*, 1999, https://netlib.org/lapack/lug/node32.html, posjećena 2023-06-30.

[3] M. S. Krishnamoorthy i B. Yener E. Acar, S. A. Camtepe, *Modeling and multiway analysis of chatroom tensors*, ISI 2005:Proceedings of the IEEE International Conference on Intelligence and Security Informatics, Lecture Notes in Comput. Sci. 3495, Springer (2005), 256–268, https://doi.org/10.1007/11427995_21.

[4] S. A. Camtepe i B. Yener E. Acar, *Collective sampling and analysis of high order tensors for chatroom communications*, ISI 2006:Proceedings of the IEEE International Conference on Intelligence and Security Informatics, Lecture Notes in Comput. Sci. 3975, Springer (2006), 213–224, https://doi.org/10.1007/11760146_19.

[5] Patrick Gelß, *The Tensor-Train Format and Its Applications*, (2017).

[6] R. A. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), 1–84.

[7] R. Henrion, *N -way principal component analysis theory, algorithms and applications*, Chemometrics and Intelligent Laboratory Systems, 25 (1994), 1–23, https://doi.org/10.1016/0169-7439(93)E0086-J.

[8] T. G. Kolda i B. W. Bader, *Tensor decompositions and applications*, SIAM review (2009), 455–500, https://doi.org/10.1137/07070111X.

[9] M. A. O. Vasilescu i D. Terzopoulos, *Multilinear analysis of image ensembles: TensorFaces*, ECCV 2002:Proceedings of the 7th European Conference on Computer Vision, Lecture Notes in Comput. Sci. 2350, Springer, (2002), https://link.springer.com/chapter/10.1007/3-540-47969-4_30.

[10] M.A.O. Vasilescu i D. Terzopoulos, *Multilinear subspace analysis of image ensembles*, CVPR 2003:Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society Press (2003), 93–99, https://doi.org/10.1109/CVPR.2003.1211457.

[11] C. J. Appellof i E. R. Davidson, *Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents*, Anal. Chem., 53 (1981), 2053–2056, https://doi.org/10.1021/ac00236a025.

[12] H. A. L. Kiers i I. Van Mechelen, *Three-way component analysis: Principles and illustrative application*, Psych. Methods, 6 (2001), 84–110, https://doi.org/10.1037/1082-989x.6.1.84.

[13] J. D. Carroll i J. J. Chang, *Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition*, Psychometrika, 35 (1970), 283–319, https://doi.org/10.1007/BF02310791.

[14] L. De Lathauwer i J. Vandewalle, *Dimensionality reduction in higher-order signal processing and rank-(R1 , R2, ..., RN) reduction in multilinear algebra*, Linear Algebra Appl., 391 (2004), 31–55, https://www.sciencedirect.com/science/article/pii/S0024379504000886?via%3Dihub.

[15] N. D. Sidiropoulos i R. Bro, *On the uniqueness of multilinear decomposition of N-way arrays*, J. Chemometrics, 14 (2000), 229–239, https://doi.org/10.1002/1099-128X(200005/06)14:3<229::AID-CEM587>3.0.CO;2-N.

[16] D. Muti i S. Bourennane, *Multidimensional filtering based on a tensor approach*, Signal Process., 85 (2005), 2338–2353, https://www.sciencedirect.com/science/article/abs/pii/S0165168405001234?via%3Dihub.

[17] Jianmin Jiang Jianguang Zhang, Yahong Han, *Tucker decomposition-based tensor learining for human action recognition*, Multimedia Systems 22 (2016), 343–353, https://doi.org/10.1007/s00530-015-0464-7.

[18] T. G. Kolda, *Multilinear Operators for Higher-Order Decompositions*, Tech. Report SAND (2006), 2006–2081, https://doi.org/10.2172/923081.

[19] B. De Moor i J. Vandewalle L. De Lathauwer, *A multilinear singular value decomposition*, SIAM journal on Matrix Analysis and Applications (2000), 1253–1278, https://doi.org/10.1137/S0895479896305696.

[20] L.Elden, *Matrix Methods in Data Mining and Pattern Recognition(Fundamentals of Algorithmas)*, USA: Society for Industrial and Applied Mathematics, 2007, https://doi.org/10.1137/1.9780898718867.

[21] Intel® Math Kernel Library, *Singular Value Decomposition - LAPACK Computational Routines*, 2014, http://cali2.unilim.fr/intel-xe/mkl/mklman/GUID-31E03413-9B5E-4F72-A4B0-743AFD7E387F.htm, posjećena 2023-06-30.

[22] C. F. Van Loan, *The ubiquitous Kronecker product*, J. Comput. Appl. Math. (2000), 85–100, https://doi.org/10.1016/S0377-0427(00)00393-9.

[23] D. Terzopoulos M. A. O. Vasilescu, *Multilinear image analysis for facial recognition*, ICPR 2002:Proceedings of the 16th International Conference on Pattern Recognition (2002), 511–514, https://doi.org/10.1109/ICPR.2002.1048350.

[24] T. Hastie i A. Iodice d'Enza M. Greenacre, P. J. F. Groenen, *Principal Component Analysis*, Nature Reviews Methods Primers (2023), https://econ-papers.upf.edu/papers/1856.pdf.

[25] I.V. Oseledets., *Tensor-train decomposition*, SIAM Journal on Scientific Computing (2011), 2295–2317, https://doi.org/10.1137/090752286.

[26] A. Shashua i A. Levin, *Linear image coding for regression and classification using the tensor-rank principle*, CVPR 2001:Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2001), 42–49, https://doi.org/10.1109/CVPR.2001.990454.

[27] J.M.F. ten Berge, *The typical rank of tall three-way arrays*, Psychometrika 65 (2000), 525–532, https://doi.org/10.1007/BF02296342.

[28] D. Tock, *Tensor Decomposition and its Applications*, SIAM journal on Matrix Analysis and Applications (2010), 1253–1278.

[29] L. R. Tucker, *Implications of factor analysis of three-way matrices for measurement of change*, Problems in Measuring Change, C. W. Harris, ed., University of Wisconsin Press (1963), 122–137.

[30] L.R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), 279–311, https://doi.org/10.1007/BF02289464.

[31] M. A. O. Vasilescu, *Human motion signatures: Analysis, synthesis, recognition*, ICPR 2002:Proceedings of the 16th International Conference on Pattern Recognition, IEEE Computer Society Press (2002), https://ieeexplore.ieee.org/document/1047975.

[32] Bangpeng Yao i Li Fei-Fei, *Grouplet: A Structured Image Representation for Recognizing Human and Object Interactions*, IEEE Conference on Computer Vision and

Pattern Recognition (CVPR) (2010), 9–16, https://doi.org/10.1109/CVPR.2010.5540234.

Sažetak

Ovaj rad započeli smo detaljnim istraživanjem nekih ključnih područja u vezi s tenzorima i njihovim dekompozicijama u kontekstu računalnog vida. Prvo smo se posvetili definiranju nekih matričnih množenja koja su neophodna pri radu s tenzorima i njihovim dekompozicijama. Nakon toga, usredotočili smo se na definiciju tenzora i njihovih osnovnih komponenti, kao što su niti i odsječci tenzora. Uz to, definirali smo osnovne operacije s tenzorima koje su neophodne za njihovu obradu. Konačno, u kraćem opisu, predstavili smo SVD (Singular Value Decomposition) koji je tehnika za dekompoziciju matrica. Sva ova teorijska podloga predstavlja osnovu za daljnje istraživanje i primjenu tenzorskih dekompozicija u računalnom vidu.

Nadalje, opisali smo četiri dekompozicije tenzora koje imaju ključnu ulogu u obradi visokodimenzionalnih podataka. Prva dekompozicija koju smo predstavili je CP dekompozicija. Ova dekompozicija razdvaja tenzor na zbroj tenzora ranga jedan. Njena primjena omogućuje izdvajanje bitnih informacija iz tenzora te smanjenje dimenzionalnosti uz minimalan gubitak informacija. Zatim smo opisali Tuckerovu dekompoziciju koja predstavlja snažan alat za analizu tenzora. Tuckerova dekompozicija rastavlja tenzor na njegove osnovne komponente - jezgru i faktorske matrice za svaku dimenziju. Ova dekompozicija omogućuje nam razumijevanje strukture tenzora i izdvajanje bitnih karakteristika. Posebno smo istaknuli HOSVD, specijalnu verziju Tuckerove dekompozicije koja se može smatrati generaliziranom verzijom SVD-a za višedimenzionalne tenzore. Naposljetku, predstavili smo Tenzorski vlak dekompoziciju, najmlađu dekompoziciju koja se razvila kako bi riješila određene probleme ostalih dekompozicija. Ova dekompozicija pruža fleksibilnost i efikasnost u radu s tenzorima velikih dimenzija.

Nakon što smo detaljno opisali ove četiri dekompozicije, prešli smo na primjenu Tuckerove dekompozicije s *ridge* regresijom. Implementirali smo ovaj algoritam s ciljem analize i razumijevanja radnji ljudi iz slika. Testirali smo algoritam na skupu podataka People playing musical instruments (PPMI) te smo dobili rezultate koji potvrđuju učinkovitost Tuckerove dekompozicije u prepoznavanju ljudskih radnji.

Summary

We initiated this work with in-depth research on key areas related to tensors and their decompositions in the context of computer vision. First, we focused on defining certain matrix multiplications that are necessary when working with tensors and their decompositions. After that, we delved into the definition of tensors and their basic components, such as fibers and tensor slices. Additionally, we defined fundamental tensor operations required for their processing. Finally, in a brief description, we presented Singular Value Decomposition (SVD) as a technique for matrix decomposition. All this theoretical foundation serves as a basis for further research and application of tensor decompositions in computer vision.

Furthermore, we described four tensor decompositions that play a crucial role in processing high-dimensional data. The first decomposition we introduced is CP decomposition. This decomposition separates a tensor into a sum of rank-one tensors. Its application allows us to extract essential information from tensors and reduce dimensionality with minimal information loss. Next, we described Tucker decomposition, which is a powerful tool for tensor analysis. Tucker decomposition breaks down a tensor into its core and factor matrices for each dimension. This decomposition enables us to understand the structure of tensors and extract significant characteristics. We particularly highlighted HOSVD, a special version of Tucker decomposition that can be considered a generalized version of SVD for multidimensional tensors. Lastly, we presented the Tensor Train decomposition, the most recent decomposition developed to address specific issues of other decompositions. This decomposition provides flexibility and efficiency in handling large-dimensional tensors.

After thoroughly describing these four decompositions, we proceeded to the application of Tucker decomposition with ridge regression. We implemented this algorithm with the aim of analyzing and understanding human actions from images. We tested the algorithm on the People playing musical instruments (PPMI) dataset and obtained results that confirm the effectiveness of Tucker decomposition in recognizing human actions.

Životopis

Rođen sam 3. rujna 1999. godine u Glini. Završio sam opću gimnaziju u Srednjoj školi Glina 2018. godine. Nakon završetka srednje škole upisujem Preddiplomski sveučilišni studij Matematike na Prirodoslovno-matematičkom fakultetu u Zagrebu. U ljeto 2021. godine odlazim na ljetnu praksu u tvrtku Ericsson Nikola Tesla te upisujem Diplomski studij Računarstva i matematike na Prirodoslovno-matematičkom fakultetu. Na posljednjoj godini studija prijavljujem se na ljetnu praksu u tvrtku Visage Technologies gdje nakon prakse ostajem kao student developer i na kraju kao stalni zaposlenik.