Handwritten digit classification using higher order singular value decomposition

Eleonora Detić, Tea Maričić, Lara Milić i Josipa Radnić

Kolegij: Uvod u složeno pretraživanje podataka

Profesor: prof.dr.sc. Zlatko Drmač

Zagreb, 27. veljače 2022.

Sadržaj

1 Uvod

Klasifikacija rukom pisanih znamenki jedan je od najčešće spomenutih problema u algoritmu klasteriranja. Kroz povijest razvilo se nekoliko različitih pristupa tom problemu. Mi ćemo objasniti i implementirati jedan od njih.

U ovom seminarskom radu implementirat ćemo dva algoritma koja koriste HOSVD dekompoziciju tenzora. Oba algoritma razvijamo u 2 faze: train i test. Prvi algoritam konstruira 10 različitih tenzora za svaku znamenku te pomoću HOSVD dobivamo bazne matrice. Drugi algoritam konstruira jedan tenzor za cijeli skup podataka te konstruira reducirani tenzor koristeći HOSVD. U oba algoritma klasifikacija se zasniva na metodi najmanjih kvadrata.

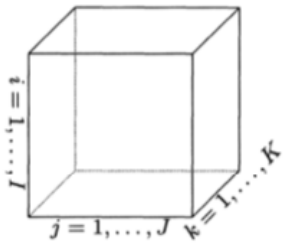
Za svaki algoritam mijenjati ćemo ulazne parametre na 2 različita skupa podataka rukom pisanih znamenki. Nakon što implementiramo gore navedene algoritme iznijet ćemo rezultate o brzini i točnosti algoritama.

2 Definicija tenzora

Za početak, definirajmo općeniti n-dimenzionalni tenzor.

Definicija 1. Ako je polje $\mathcal{X}$ određeno s N indeksa onda je $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ tenzor reda N nad $\mathbb{R}$, $\mathcal{X} = (x_{i_1 i_2 \ldots i_N}), i_1 = 1, \ldots, I_1; \ldots; i_N = 1, \ldots, I_N$

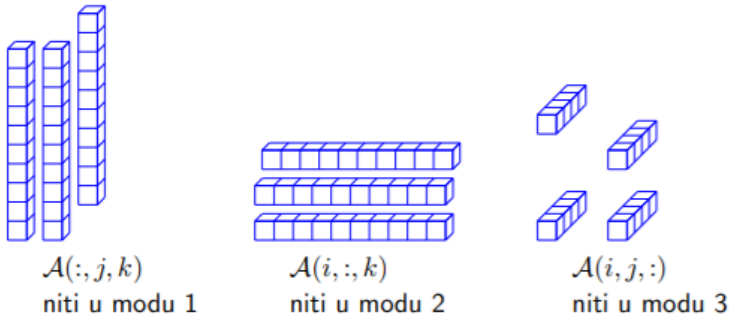
Tenzori s kojima smo se već susreli su: skalari (tenzori reda 0), vektori (tenzori reda 1) i matrice (tenzori reda 2). Pogledajmo sada tenzor reda 3 koji je određen s tri indeksa, $\mathcal{A} = (a_{ijk}) \in \mathbb{R}^{I \times J \times K} \equiv \mathbb{R}^{I_1 \times I_2 \times I_3}$



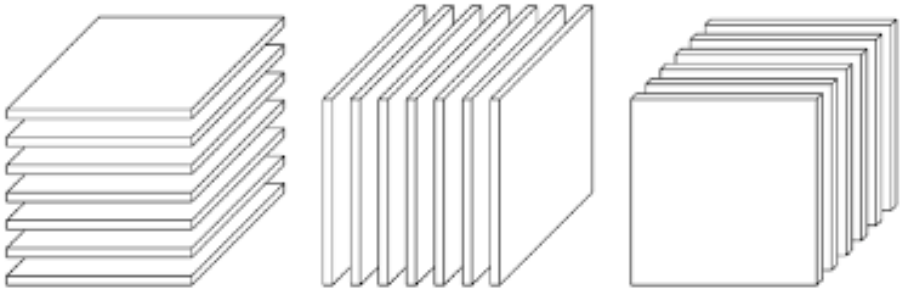
2.1 Anatomija tenzora reda 3

Kako bi s tenzorima računali potrebno je proučiti njihovu anatomiju.

1. Nit tenzora je 1D-sekcija tenzora definirana fiksiranjem svih osim jednog indeksa. Niti (koje su tenzori reda 1) po dogovoru reprezentiramo kao vektore stupce i sav račun se radi s tom pretpostavkom. Pogledajmo kako izgledaju niti u svim modovima tenzora reda tri.



2. Odsječak (slice) tenzora je 2D-sekcija tenzora definirana fiksiranjem svih osim dva indeksa.



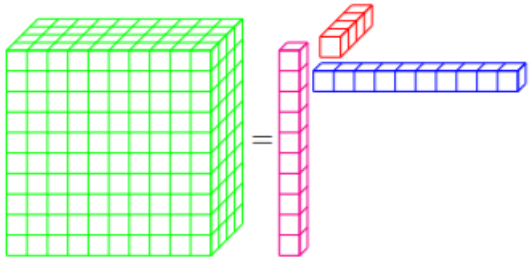
(a) Horizontal slices: $\mathbf{X}_{i::}$ (b) Lateral slices: $\mathbf{X}_{:j:}$ (c) Frontal slices: $\mathbf{X}_{::k}$ (or $\mathbf{X}_k$)

2.2 Aproksimacija pomoću tenzora ranga jedan

Kako bismo napravili aproksimaciju pomoću tenzora ranga jedan, trebat će nam definicija Hadamardovog matričnog produkta. Za $A, B \in \mathbb{R}^{m\times n}$, njihov Hadmardov produkt $C = A \circ B$ je definiran s $c_{ij} = a_{ij}b_{ij}$.

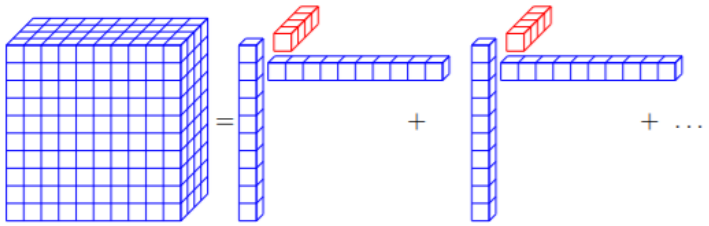
Kažemo da je tenzor $\mathcal{A}$ ranga jedan ako se može prikazati u obliku:

$$\mathcal{A} = a^{(1)} \circ a^{(2)} \circ \cdots \circ a^{(N)}, \quad a_{i_1 i_2 \ldots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N}$$



Ako tenzor nije ranga jedan možemo ga aproksimirati tenzorom ranga jedan na način:

$$\mathcal{A} \approx \sum_{\ell=1}^{L} \sigma_\ell u_\ell \circ v_\ell \circ w_\ell$$



2.3 Matricizacija tenzora

Matricizacija tenzora $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ u n-tom modu, $\mathcal{X}_{(n)}$ se dobije tako da se niti u n-tom modu poslože kao stupci matrice u nekom unaprijed zadanom fiksnom poretku.

$$\mathcal{A}_{(1)} = \underbrace{\text{unfold}_{\,1}(\mathcal{A})}_{\ell \times (m \cdot n)} = \underbrace{(\mathcal{A}(:,1,:), \mathcal{A}(:,2,:), \ldots, \mathcal{A}(:,m,:))}_{\text{niti u modu 1 uzete redom po lateralnim sliceovima}}$$

$$\mathcal{A}_{(2)} = \underbrace{\text{unfold}_2(\mathcal{A})}_{m \times (\ell \cdot n)} = \underbrace{(\mathcal{A}(:,:,1)^T, \mathcal{A}(:,:,2)^T, \ldots, \mathcal{A}(:,:,n)^T)}_{\text{transponirani frontalni sliceovi, redom}}$$

$$\mathcal{A}_{(3)} = \underbrace{\text{unfold}_3(\mathcal{A})}_{n \times (m \cdot \ell)} = \underbrace{(\mathcal{A}(1,:,:)^T, \mathcal{A}(2,:,:)^T, \ldots, \mathcal{A}(l,:,:)^T)}_{\text{transponirani horizontalni sliceovi}}$$

fold je inverzna operacija od unfold i ona matricu pretvara u tenzor zadanih dimenzija.

2.4 Množenje tenzora matricom u modu

Neka je dan tenzor $\mathcal{T} \in \mathbb{R}^{J_1 \times \cdots \times J_{n-1} \times J_n \times J_{n+1} \times \cdots \times J_N}$, $\mathcal{T} = (t_{j_1 \ldots j_N})$ i neka je zadana matrica $A \in \mathbb{R}^{I \times J_n}$. Produkt $\mathcal{T} \times_n A \in \mathbb{R}^{J_1 \times \cdots \times J_{n-1} \times I \times J_{n+1} \times \cdots \times J_N}$ je zadan formulama

$$\underbrace{(\mathcal{T} \times_n A)_{j_1 \ldots j_{n-1} i j_{n+1} \ldots j_N}}_{\text{element niti u modu } n} = \sum_{k=1}^{J_n} a_{ik} \underbrace{t_{j_1 \ldots j_{n-1} k j_{n+1} \ldots j_N}}_{\star}, i = 1, \ldots, I$$

$\star = v_k =$ fiksirani svi ind. osim n-tog $=$ nit u modu n; $\sum_{k=1}^{J_n} a_{ik} v_k$

Također, vrijedi formula $(\mathcal{T} \times_n A)_{(n)} = A\mathcal{T}_{(n)}$.

Teorem 2. *(Osnovna svojstva množenja u modu)*
Neka je $\mathcal{T} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$. *Tada vrijedi: - Za* $A \in \mathbb{R}^{I_m \times J_m}, B \in \mathbb{R}^{I_n \times J_n}$ *i* $m \neq n$ *je*

$$\mathcal{T} \times_m A \times_n B = (\mathcal{T} \times_m A) \times_n B = (\mathcal{T} \times_n B) \times_m A$$

Za $A \in \mathbb{R}^{I \times J_n}, B \in \mathbb{R}^{K \times I}$ *je*

$$(\mathcal{T} \times_n A) \times_n B = \mathcal{T} \times_n (BA)$$

Neka je $A \in \mathbb{R}^{I \times J_n}$ *punog ranga. Tada*

$$\mathcal{X} = \mathcal{T} \times_n A \Longrightarrow \mathcal{T} = \mathcal{X} \times_n A^{\dagger}$$

Ako je A ortonormalna $\left(A^T A = I\right)$ *onda gornje vrijedi s* $A^{\dagger} = A^T$.

Množenje u modu uz pomoć fold i unfold funkcija.

$$\mathcal{T} \times_i M = \text{fold}_i \left(M \cdot unfold_i(\mathcal{T}),\ \text{dimenzije tenzora}\ \right), \text{tj.}$$
$$\mathcal{T} \times_i M = \text{fold}_i \left(M \cdot \mathcal{T}_{(i)}, \text{dimenzije tenzora}\right)$$

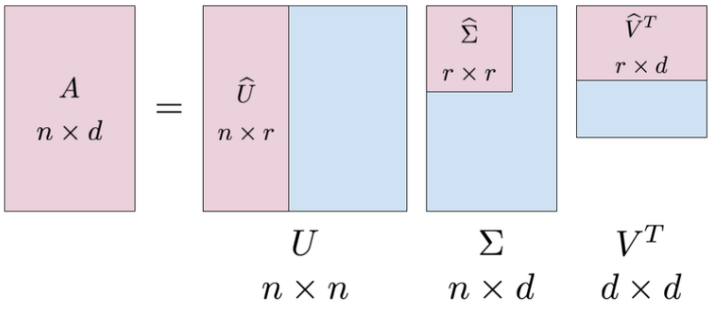
3 HOSVD

Ponovimo SVD dekompoziciju matrice. Neka je $A \in \mathbb{R}^{m\times n}$ matrica ranga r. Tada postoje unitarne matrice $U = (u_1, \ldots, u_m), V = (v_1, \ldots, v_n)$ $(U^*U = I_m, V^*V = I_n)$ i dijagonalna Σ tako da je (npr. $m \geq n$)

$$A = U\Sigma V^* = \sum_{k=1}^{r} \sigma_i u_i v_i^*$$

$$\Sigma = \operatorname{diag}(\sigma_i), \quad \sigma_1 \geq \cdots \geq \sigma_r > 0 = \sigma_{r+1} = \cdots = \sigma_{\min(m,n)}, \text{ gdje su}$$

σ_i = singularna vrijednost matrice A
u_i = lijevi singularni vektor
v_i = desni singularni vektor



Sljedeći teorem nam omogućava da ako imamo SVD dekompoziciju matrice A možemo pronaći njoj najbližu matricu manjeg ranga te znati kolika je pogreška u aproksimaciji.

Teorem 3. *(Eckart - Young - Mirsky - Schmidt)*
Neka je $\ell < r$ *i* $A_\ell = \sum_{i=1}^{\ell} \sigma_i u_i v_i^*$. *Tada je*

$$\min_{\operatorname{rang}(X)\leq \ell} \|A - X\|_F = \|A - A_\ell\|_F = \sqrt{\sum_{i=\ell+1}^{r} \sigma_i^2}$$
$$\min_{\operatorname{rang}(X)\leq \ell} \|A - X\|_2 = \|A - A_\ell\|_2 = \sigma_{\ell+1}$$

Gornje rezultate želimo generalizirati na tenzorima reda 3 kako bi tenzore u oba algoritma koja želimo obraditi mogli aproksimirati slično kao što smo matrice.

Teorem 4. *(Higher order singular value decomposition)*
Tenzor $\mathcal{A} \in \mathbb{R}^{\ell \times m \times n}$ *možemo zapisati kao*

$$\mathcal{A} = \mathcal{S} \times {}_1U^{(1)} \times {}_2U^{(2)} \times {}_3U^{(3)}$$

gdje su $U^{(1)} \in \mathbb{R}^{\ell \times \ell}, U^{(2)} \in \mathbb{R}^{m \times m}, U^{(3)} \in \mathbb{R}^{n \times n}$ *ortogonalne matrice,* $\mathcal{S}$ *je tenzor istih dimenzija kao i* $\mathcal{A}$ *i ima svojstvo potpune ortogonalnosti: svaka dva različita odsječka istog tipa su okomiti: za sve* $i \neq j$ *vrijedi*

$$\langle \mathcal{S}(i,:,:), \mathcal{S}(j,:,:) \rangle_F = \langle \mathcal{S}(:,i,:), \mathcal{S}(:,j,:) \rangle_F = \langle \mathcal{S}(:,:,i), \mathcal{S}(:,:,j) \rangle_F = 0$$

Singularne vrijednosti u modu 1 su

$$\sigma_j^{(1)} = \|\mathcal{S}(j,:,:)\|_F, \quad j = 1, \ldots, \ell$$

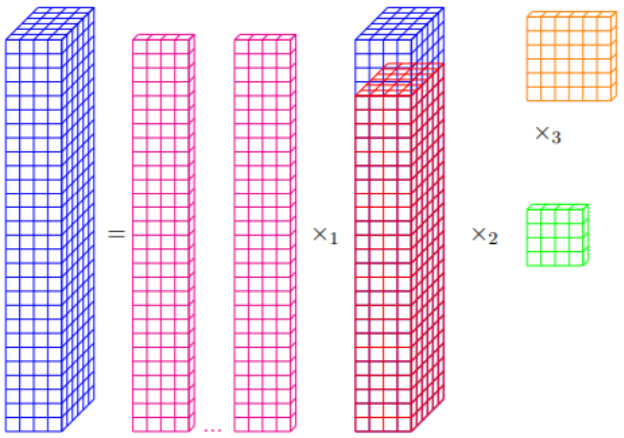
i uređene su tako da je $\sigma_1^{(1)} \geq \sigma_2^{(1)} \geq \cdots \geq \sigma_\ell^{(1)} \geq 0$. *Ako je* $\ell > mn$, *onda je* $\mathcal{S}(i,:,:) = \mathbf{0}$ *za* $i > mn$. *Analogno se definiraju singularne vrijednosti u ostalim modovima. Vrijedi i da je*

$$\mathcal{S} = \mathcal{A} \times_1 \left(U^{(1)}\right)^T \times_2 \left(U^{(2)}\right)^T \times_3 \left(U^{(3)}\right)^T$$

Kako bi došli do HOSVD dekompozicije tenzora $\mathcal{A}$ dobijemo računanjem ortogonalnih matrica $U^{(n)}$ SVD dekompozicijom matrice dobivene matricizacijom tenzora $\mathcal{A}$ u modu n što označavamo $\mathcal{A}_{(n)}$.
Postupak je:

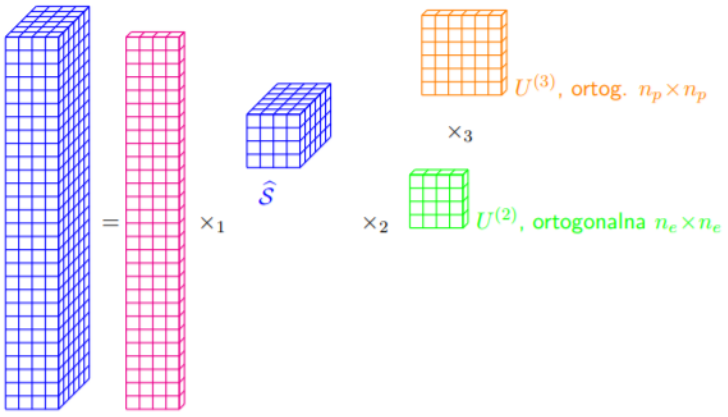
1. napraviti SVD dekompoziciju: $\mathcal{A}_{(n)} = U^{(n)} \sum^{(n)} V^{(n)}$ za n=1,2,3
2. izračunati jezgreni tenzor: $\mathcal{S} = \mathcal{A} \times_1 \left(U^{(1)}\right)^T \times_2 \left(U^{(2)}\right)^T \times_3 \left(U^{(3)}\right)^T$

Slikovni prikaz HOSVD-a:



$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)}$ $(\ell > mn,\ \mathcal{S}(i,:,:) = \mathbf{0}$ za $i > mn)$

Kao što nam SVD dekompozicija pomaže u kompresiji, tako i kod prikaza tenzora kako bi uštedjeli memoriju i ubrzali rad s njima možemo izbaciti odsječke čije su singularne vrijednosti 0 ili jako blizu nule. Izbacivanjem dijela podataka nastaje aproksimacija tenzora dana na slici ispod.



$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)} = \widehat{\mathcal{S}} \times_1 \widehat{U^{(1)}} \times_2 U^{(2)} \times_3 U^{(3)}$

Kada raspišemo zapis tenzora $\mathcal{A} = \mathcal{S} \times {}_1U^{(1)} \times {}_2U^{(2)} \times {}_3U^{(3)}$ dobijemo da je

$$\mathcal{A} = \sum_{i=1}^{n} \left(\mathcal{S}(:,:,i) \times_1 U^{(1)} \times_2 U^{(2)}\right) \times_3 u_i^{(3)} = \sum_{i=1}^{n} \mathbb{A}_i \times_3 u_i^{(3)}.$$

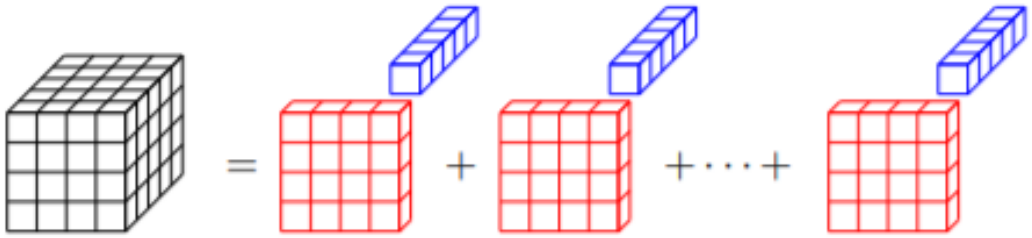
gdje su nam:

$$\mathbb{A}_i = \left(\mathcal{S}(:,:,i) \times_1 U^{(1)} \times_2 U^{(2)}\right) = U^{(1)}\mathcal{S}(:,:,i)\left(U^{(2)}\right)^T.$$

One su ortogonalne i nazivamo ih bazne matrice.

$$\begin{aligned}\langle \mathbb{A}_i, \mathbb{A}_j \rangle_F &= \text{Trag}\left(\mathbb{A}_j^T \mathbb{A}_i\right) = \text{Trag}\left(U^{(2)}\mathcal{S}(:,:,j)^T \left(U^{(1)}\right)^T U^{(1)}\mathcal{S}(:,:,i)\left(U^{(2)}\right)^T\right) \\ &= \text{Trag}\left(\mathcal{S}(:,:,j)^T \mathcal{S}(:,:,i)\right) = 0 \quad \text{za } i \neq j\end{aligned}$$

Uz pomoć njih reprezentacija tenzora $\mathcal{A}$ izgleda:



4 Algoritmi

Implementirali smo dva algoritma koja koriste HOSVD dekompoziciju. Prvi algoritam kreira u training fazi tenzore za svaku znamenku te pohranjuje na frontalnom odsječku sliku znamenke u obliku matrice $m \times n$. Pomoću HOSVD tih tenzora dođemo do baznih matrica svake znamenke koje koristimo u test fazi. Drugi algoritam kreira tenzor kojem frontalni odsječak prezentira jednu znamenku, te stupac u odsječku prezentira sliku znamenke u obliku vektora stupca, te HOSVD koristimo nad tim tenzorom training podataka. Sada ćemo detaljnije objasniti kako rade algoritmi u obe faze: train i test.

4.1 Prvi algoritam

U prvom algoritmu svaku znamenku promatramo kao točku iz $\mathbb{R}^{16\times 16}$ i razumno je pretpostaviti da training podaci će napraviti 10 dobro odvojena klastera, u suprotnom bi algoritam loše klasificirao.

TRAIN FAZA:

Za svaku znamenku konstruiramo tenzor koji će sadržavati sve slike zadane znamenke gdje će frontalni odsječak sadržavati sliku znamenke u obliku matrice $\mathbb{R}^{m\times n}$ (u našim podacima će to biti 16×16 i 28×28). Za svaki tenzor izračunamo HOSVD i konstruiramo ortogonalne bazne matrice na način opisan u prethodnom poglavlju. U prethodnom poglavlju konačnu sumu nakon dobivenog HOSVD smanjimo na manji i dominantni k-dimenzionalni potprostor za svaki klaster. Nakon toga normiramo bazne matrice kako bi vrijedilo:

$$\langle \mathbb{A}_v^\mu, \mathbb{A}_\lambda^\mu \rangle = \delta_{v\lambda}, \quad \mu = 0, 1, \ldots, 9$$

gdje je $\delta_{v\lambda}$ Kronecker delta.

TEST FAZA:

Prvo normiramo svaku znamenku D iz testnog uzorka, te ćemo svaku znamenku gledati u matričnom zapisu D koja je već normirana. Cilj nam je naći linearnu kombinaciju baznih matrica pojedine znamenke koja će najbolje opisati neklasificiranu znamenku D. Određujemo koeficijente α_v^μ koristeći metodu najmanjih kvadrata:

$$\min_{\alpha_v^\mu} \left\| D - \sum_{v=1}^{k} \alpha_v^\mu A_v^\mu \right\|, \quad \mu \quad \text{fiksirani indeks klase}$$

S obzirom da su A_v^μ ortonormalne za fiksirani μ imamo rješenje:

$$\hat{\alpha}_y^\mu = \langle D, \mathbb{A}_v^\mu \rangle, \quad v = 1, 2, \ldots, k$$

Ubacivajući rješenje u prethodni izraz i koristeći ortonormalnost baznih matrica dobijemo sljedeći izraz za kvadriranu normu:

$$\begin{aligned} R(\mu) &= \left\| D - \sum_{v=1}^{k} \hat{\alpha}_v^\mu \mathbb{A}_v^\mu \right\|^2 \\ &= \left\langle D - \sum_{v=1}^{k} \hat{\alpha}_v^\mu \mathbb{A}^\mu, D - \sum_{v=1}^{k} \hat{\alpha}_v^\mu \mathring{\mathbb{A}}_v^\mu \right\rangle \\ &= \langle D, D \rangle - \sum_{v=1}^{k} \langle D, \mathbb{A}_v^\mu \rangle^2 = 1 - \sum_{v=1}^{k} \langle D, \mathbb{A}_v^\mu \rangle^2 \end{aligned}$$

Dakle, znamenku ćemo svrstati u klaster za koji je $R(\mu)$ najmanji.

Pseudokod prvog algoritma:

Algorithm 1. Classification by HOSVD

- Training phase:
(1) Sort the training digits into tensors with digits of the same type.
(2) Compute the HOSVD of the tensors.
(3) Compute and store the normalized basis matrices $(A_v^\mu)_{v=1}^k$, $\mu = 0, 1, \ldots, 9$.

- Test phase:
(1) Normalize the unknown digit.
(2) Compute $R(\mu) = 1 - \sum_{v=1}^{k} \langle D, A_v^\mu \rangle^2$, $\mu = 0, 1, \ldots, 9$.
(3) Determine $\mu_{\min} := \operatorname{argmin}_\mu R(\mu)$ and classify D as $\mu_{\min}$.

4.2 Drugi algoritam

U drugom algoritmu koristimo HOSVD kako bi kompresirali training podatke. Prednost u odnosu na prvi algoritam je u tome što ćemo znamenke iz različitih klastera projicirati na jedan zajednički potprostor, time će neklasificirana znamenka biti projicirana samo jednom. Time bi testna faza algoritma zahtjevala puno veći broj računanja i memorije. Također ćemo pretpostaviti da imamo 10 klastera, u suprotnom bi algoritam loše klasificirao.

TRAIN FAZA:

Kreirati ćemo tenzor koji će sadržavati 10 frontalnih odsječaka koji se sastoji od znamenaka pojedinog klastera tako da stupac reprezentira sliku znamenke rastegnute u stupac. Tako dobiveni tenzor označit ćemo s $\mathcal{D}$. Računajući njegovu HOSVD dekompoziciju dolazimo do tenzora $\mathcal{F}$ koji smo dobili uzimanjem prvih p stupaca u matrice U te prvih q stupaca matrice V. Točnije,

$$\mathscr{D} = \mathscr{S} \times_1 U \times_2 V \times_3 W \approx \mathscr{F} \times_1 U_p \times_2 V_q$$

gdje je $U_p = U(:, 1:p), V_q = V(:, 1:q)$, te $\mathcal{F} = (1:p, 1:q, :) \times_3 W$.

Ovim postupkom reducirali smo prikaz pojedine znamenke s $\mathbb{R}^{256}$ na prvih $\mathbb{R}^p$ i broj znamenaka u svakom klasteru na q. Reducirani tenzor $\mathcal{F}$ može se izračunati kao:

$$\mathbb{R}^{p \times q \times 10} \ni \mathcal{F} = D \times_1 U_p^T \times_2 V_q^T$$

Pod pretpostavkom da su i p i q puno manji od pripadajućih dimenzija tenzora $\mathcal{D}$ time smo dosta reducirali svoj početni skup podataka. Kako bi ta aproksimacija bila dobra nužno je da su singularne vrijednosti odrezanih dijelova su vrlo male.

Kako bismo dobili bazne vektore koristimo $F^\mu := \mathscr{F}(:, :, \mu)$ čiji stupci predstavljaju bazne vektore znamenki klase $\mu = 0, 1, ..., 9$. Kako bismo dobili ortonormalne bazne matrica uzimamo prvih k lijevih singularnih vektora iz SVD dekompozicije matrice F^μ. Dakle, vrijedi

$$F^\mu = \left[B^\mu \left(B^\mu \right)^\perp \right] \Sigma^\mu \left(Q^\mu \right)^T, \quad \mu = 0, 1, \ldots, \quad 9, B^\mu \in \mathbb{R}^{p \times k}$$

gdje $B^\mu \in R^{p \times k}$ razapinju k dimenzionalni potprostor od F^μ

TEST FAZA:

Pretpostavimo kako je $d \in R^{256}$ znamenka koju treba klasificirati. U test fazi ovog algoritma računamo $d_p = U_p^T d$ te ponovno koristimo metodu najmanjih kvadrata kako bi odredili pripadnost klasi:

$$\min_{x^\mu} \|d_p - B^\mu x^\mu\| \quad \mu = 0, 1, \ldots, \quad 9, \quad \hat{x}^\mu = (B^\mu)^T d_p$$

što nam se na kraju svodi na minimizaciju

$$R(\mu) = \left\| d_p - B^\mu \left(B^\mu\right)^T d_p \right\| \quad \mu = 0, 1, \ldots, \quad 9$$

Ponovno, znamenku ćemo svrstati u klaster za koji je $R(\mu)$ najmanji.

Pseudokod drugog algoritma:

Algorithm 2

- Training phase:

(1) Sort and vectorize the digits from the training set into a tensors $\mathscr{D}$.
(2) Compute the HOSVD of $\mathscr{D}$, Eq. (21).
(3) Compute the reduced tensor $\mathscr{F}$ representing the training set, Eq. (22).
(4) Compute and store the basis matrices B^μ for each class.

- Test phase:

(1) Compute the low dimensional representation $d_p = U_p^T d$ of an unknown digit.
(2) Compute the residuals $R(\mu) = \|d_p - B^\mu (B^\mu)^T d_p\|$, $\mu = 0, 1, \ldots, 9$.
(3) Determine $\mu_{\min} = \text{argmin}_\mu R(\mu)$ and classify d as $\mu_{\min}$.

5 Implementacija u Pythonu

Opisane algoritma ćemo implmentirati u Pythonu. Implementirali smo i neke pomoćne funkcije kako bi olakšali rad u algoritmima.

5.1 Implementacija prvog algoritma

Za početak ćemo definirati pomoćne funkcije, te ćemo onda definirati train i test fazu prvog algoritma.

1. Funkcija HOSVD za prvi algoritam:

```
def HOSVD(T):

    U1, _, _ = np.linalg.svd(tl.unfold(T, 0), full_matrices=False)
    U2, _, _ = np.linalg.svd(tl.unfold(T, 1), full_matrices=False)
    U3, _, _ = np.linalg.svd(tl.unfold(T, 2), full_matrices=False)

    S = tl.tenalg.multi_mode_dot(T,
        [np.transpose(U1),np.transpose(U2),np.transpose(U3)],
        modes = [0,1,2], transpose=False)

    return(S, U1, U2)
```

2. Funkcija množenja tenzora matricom u modu:

```
def multiplication(T, M, mode):

    descriptor = list(T.shape)
    descriptor[mode] = M.shape[0]
    pom = tl.unfold(T, mode)

    return (tl.fold( M @ pom, mode, descriptor))
```

3. TRAIN FAZA prvog algoritma:

```
def alg1_train(A_train, D_train,  k):

    tensor = [None] * k

    for i in range(k):
```

```
            T = tl.vec_to_tensor(A_train[:, np.where(D_train == i)[1]],
                (int(math.sqrt(A_train.shape[0])),
                int(math.sqrt(A_train.shape[0])),
                A_train[:, np.where(D_train == i)[1]].shape[1]))
                S, U1, U2 = HOSVD(T)

            tensor[i] = np.zeros((int(math.sqrt(A_train.shape[0])),
                        int(math.sqrt(A_train.shape[0])), S.shape[2]))

            for j in range(S.shape[2]):

                pom_1 = multiplication(tl.vec_to_tensor(S[:,:,j],
                        (int(math.sqrt(A_train.shape[0])),
                        int(math.sqrt(A_train.shape[0])),1)), U1, 0)
                pom_2 = multiplication(pom_1, U2, 1)

                tensor[i][:, :, j] = np.reshape(pom_2/np.linalg.norm(pom_2
                                    (int(math.sqrt(A_train.shape[0])),
                                     int(math.sqrt(A_train.shape[0]))))

        return(tensor)
```

4. TEST FAZA prvog algoritma:

```
    def alg1_test(A_test, tensor, k, l):

        digits = []

        for i in range(A_test.shape[1]):

            D = np.reshape((A_test[:, i] / np.linalg.norm(A_test[:, i])),
                (int(math.sqrt(A_train.shape[0])),
                int(math.sqrt(A_train.shape[0]))))
            max_R = np.NINF
            digit = 0

            for j in range(k):
```

```
            R = 0

            for z in range(l):
                R+=np.trace(np.matmul(np.transpose(D),tensor[j][:,:,z]

            if( R > max_R):
                max_R = R
                digit = j

        digits.append(digit)

    return digits
```

5.2 Implementacija drugog algoritma

Za početak ćemo definirati pomoćne funkcije uz korištenje već gore definiranje funkcije za množenje tenzora u modu s matricom, te ćemo onda definirati train i test fazu drugog algoritma.

1. Funkcija koja konstruira tenzor originalnih podataka:

```
def construct_tensor(A_train, D_train, k):

    makszn = (D_train[0]== np.bincount(D_train[0]).argmax()).sum()

    T = np.zeros((A_train.shape[0],makszn,k))

    for i in range(k):
        for j in range(makszn):
            if(j >= len(np.where(D_train==i)[1])):
                mjesto = np.random.randint(0,
                        len(np.where(D_train==i)[1])-1)
                T[:,j,i]=A_train[:,np.where(D_train == i)[1]][:,mjesto

            else:
                T[:,j,i]=A_train[:, np.where(D_train == i)[1]][:,j]

    return(T)
```

2. Funkcija HOSVD za drugi algoritam:

```
def HOSVD(T, p, q):

    U1, _, _ = np.linalg.svd(tl.unfold(T, 0), full_matrices=False)
    U2, _, _ = np.linalg.svd(tl.unfold(T, 1), full_matrices=False)

    F_ = multiplication(T, np.transpose(U1[:, 0:p]), 0)
    F = multiplication(F_, np.transpose(U2[:, 0:q]), 1)

    return F, U1[:, 0:p]
```

3. TRAIN FAZA drugog algortima:

```
def alg2_train(F, k, l):

    B = np.zeros((F.shape[0], l, k))

    for mi in range(F.shape[2]):
        U, _, _ = np.linalg.svd(F[:, :, mi], full_matrices=False)
        B[:, :, mi] = U[:, 0:l]

    return(B)
```

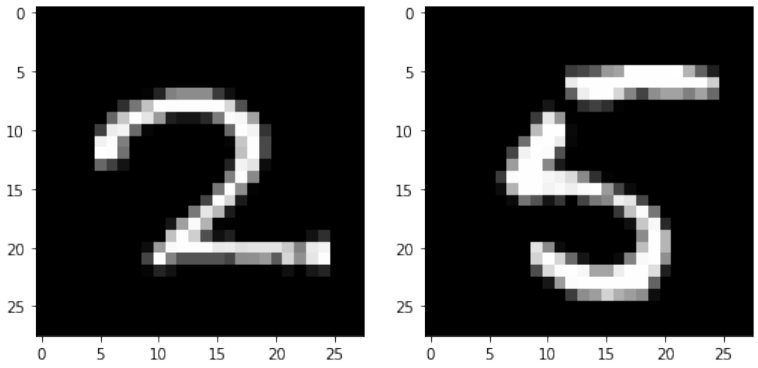
4. TEST FAZA drugog algortima:

```
def alg2_test(A_test, B, U_p):

    digits = []

    for j in range(A_test.shape[1]):

        d = A_test[:, j]
        d = np.transpose(U_p) @ d

        r_min =  np.Inf

        for k in range(B.shape[2]):

            r=np.linalg.norm(d - (B[:,:,k] @ (np.transpose(B[:,:, k])

            if(r < r_min):
                r_min = r
                digit = k

        digits.append(digit)

    return digits
```

6 Opis podataka i rezultati

6.1 Opis podataka

Obe skupine podataka su nam rukom pisane znamenke. Prvi skup podataka ćemo zvati USPS, a drugi MNIST. Prvu skupinu podataka čine slike rezolucije 16×16 piksela. U train setu imamo 1707 znamenki, a u test skupu 2007. Također, imamo i vektore D_train i D_test koji nam označavaju na kojem indeksu se nalazi pojedina znamenka. Drugi skup podataka je tzv. MNIST. On je veći u odnosu na prvi i sastoji se od 60000 train znamenki i 10000 slika test skupa rezolucije 28×28 piksela. Ponovno, imamo vektore D_train i D_test koji nam predstavljaju na kojem indeksu se nalazi koja znamenka.

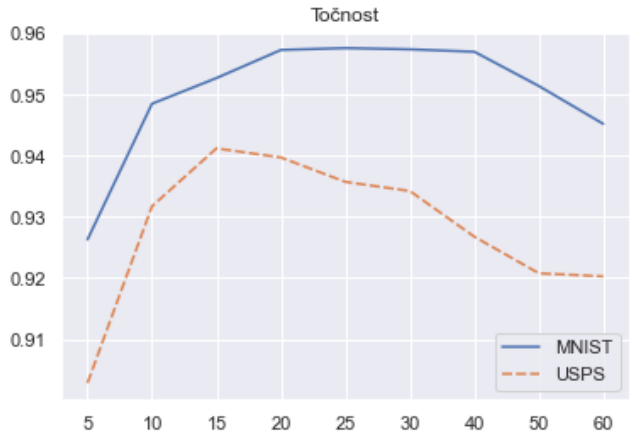


Slika 1: Rukom pisane znamenke

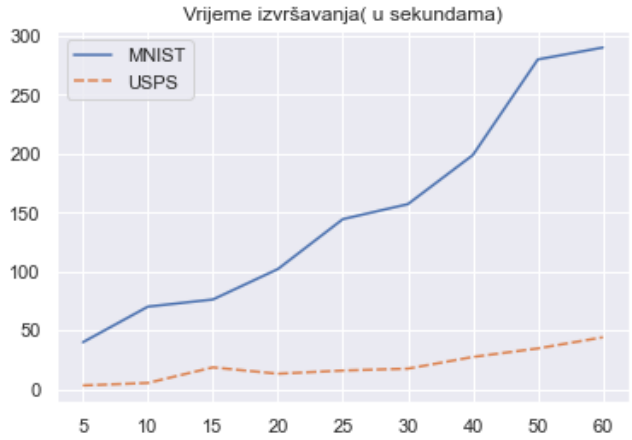
6.2 Rezultati

6.2.1 Rezultati - Algoritam 1

Pogledat ćemo točnost prvog algoritma za obe skupine podataka ovisno o broju baznih matrica.



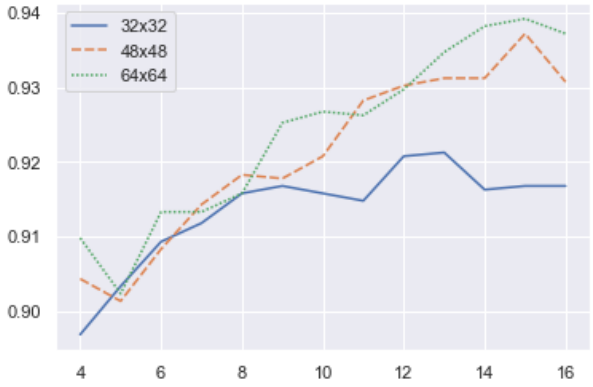
Slika 2: Točnost algortima 1 ovisno o broju baznih matrica



Slika 3: Vrijeme izvršavanja ovisno o broju baznih matrica

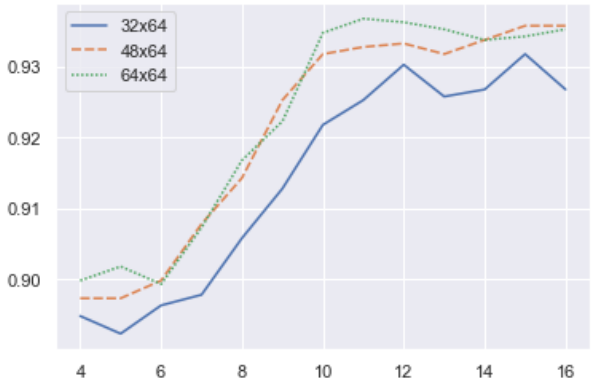
6.2.2 Rezultati - Algoritam 2

Pogledat ćemo točnost drugog algoritma ovisno o fiksiranim parametrima p i q i broju uzetih baznih vektora na USPS podacima.

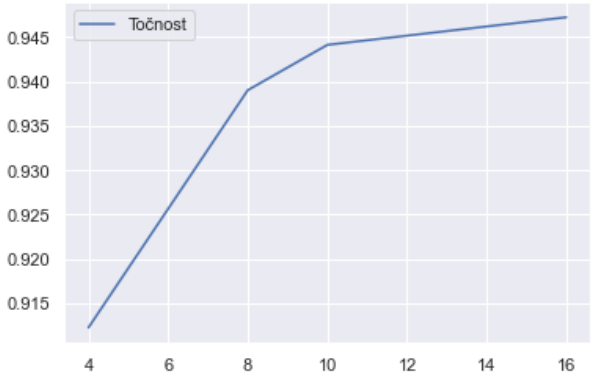


Slika 4: Točnost predikcije za jednake p i q ovisno o broju baznih vektora

Slika 5: Točnost predikcije za fiksan parametar p ovisno o broju baznih vektora

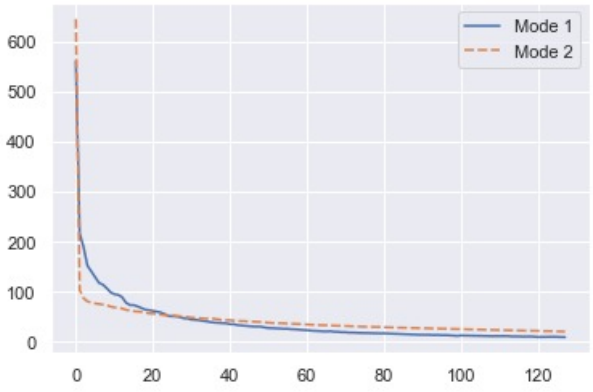


Slika 6: Točnost predikcije za fiksan parametar q ovisno o broju baznih vektora



Slika 7: Točnost predikcije ovisno o broju baznih vektora - MNIST

Na Slici 7 gledali smo točnost predikcije na MNIST podacima za fiksan p i q t.d. $p = q = 64$ ovisno o broju baznih vektora (gledali smo za $l = 4, 8, 10, 16$).



Slika 8: Singularne vrijednosti za pixel i digit mode - USPS

Literatura

[1] Zlatko Drmač: *USPP-2021-22-P6-P7-P8, predavanja*, Zagreb, (2021)

[2] Berkant Savas, Lars Eldén: *Handwritten digit classification using higher order singular value decomposition*

[3] T. Hastie, R. Tibshirani, J. Friedman: *The elements of statistical learning, Springer Series in Statistics*, Springer, New York, (2001)

[4] Andrea Giuliodori, Rosa Lillo and Daniel Peña: *Handwritten digit clasification*, Getafe, Spain (2011)