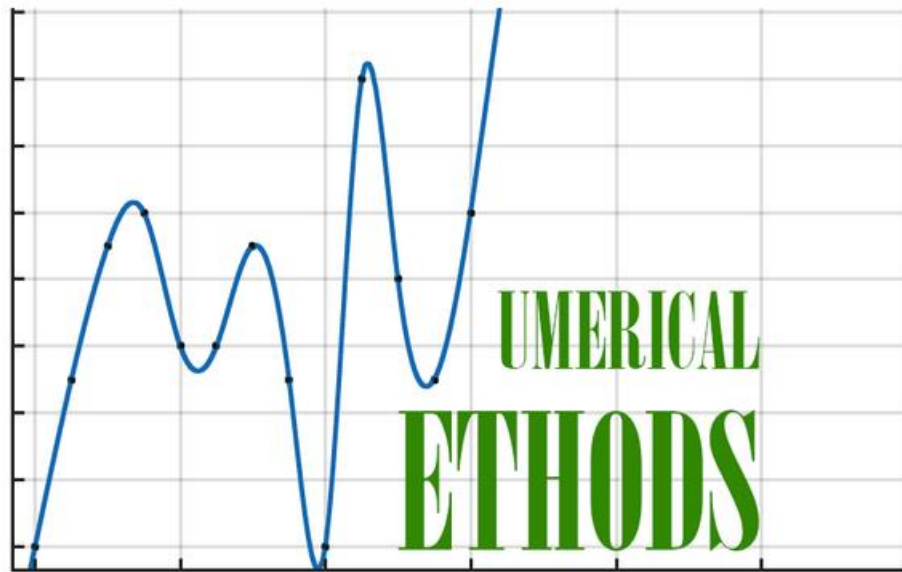


Tema 2 - Metode Numerice



Facultatea de Automatică și Calculatoare
Universitatea Politehnică București

6 Mai, 2023

Responsabil: Ionescu Andrei Ionuț

Cuprins

1	Introducere	3
1.1	Descompunerea valorilor singulare (SVD)	3
2	Compresia imaginilor folosind SVD	4
2.1	Task 1 [20p]	5
3	Compresia imaginilor folosind analiza componentelor principale	5
3.1	Task 2 [20p]	6
3.2	Task 3 [20p]	6
4	Task 4 – Recunoasterea cifrelor scrise de mana [35p]	7
5	Readme [5p]	11
6	Observatii finale	11
7	Bibliografie	11

1 Introducere

În recunoasterea formelor, selectia si extragerea caracteristicilor reprezinta o alegere decisiva pentru proiectarea oricarui clasificator. Selectia caracteristicilor poate fi vazuta si ca un proces de compresie de date, fiind similara cu o transformare liniara din spatiul initial al observatiilor într-un spatiu cu mai putine dimensiuni. O astfel de transformare este necesara deoarece poate pastra o mare parte din informatii (prin eliminarea informatiilor redundante sau a celor mai putin semnificative) si permite aplicarea unor algoritmi eficienti într-un spatiu de dimensiuni reduse.

Cele mai multe transformari utilizate pentru selectia caracteristicilor sunt cele liniare, în timp ce transformarile neliniare au o complexitate mai ridicata, sunt mai dificil de implementat, dar pot avea o eficienta mai mare asupra rezultatelor, exprimand mai bine dependenta dintre formele observate si caracteristicile selectate ale acestor forme.

1.1 Descompunerea valorilor singulare (SVD)

Fiind data o matrice $A \in R^{m \times n}$, descompunerea valorilor singulare (în engleza singular value decomposition - SVD) ale matricei A este data de factorizarea $A = USV^T$, unde:

1. $U \in R^{m \times m}$ este o matrice ortogonală;
2. $S \in R^{m \times n}$ este o matrice diagonală;
3. $V \in R^{n \times n}$ este o matrice ortogonală.

Elementele de pe diagonală principală a lui S sunt întotdeauna numere reale non-negative ($s_{ii} \geq 0$ pentru $i = 1 : \min(m, n)$) si se numesc *valorile singulare* ale matricei A. Acestea sunt asezate în ordine descrescătoare, astfel încat $s_{11} \geq s_{22} \geq \dots \geq s_{rr} > s_{r+1r+1} = \dots = s_{pp} = 0$, unde $p = \min(m, n)$.

Coloanele $u_j \in R^m, j = 1 : m$ ale lui U se numesc *vectori singolari stanga* ai matricei A. Coloanele $v_j \in R^n, j = 1 : n$ ale lui V se numesc *vectori singolari dreapta* ai matricei A.

De exemplu, pentru matricea:

$$A = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}$$

se obtine urmatoarea descompunere a valorilor singulare:

$$A = USV^T = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{bmatrix}$$

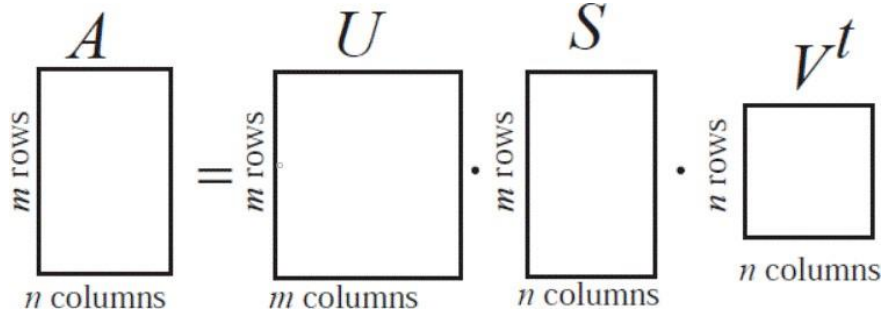


Figura 1: Descompunerea valorilor singulare pentru matricea A de dimensiune $m \times n$, unde $m > n$.

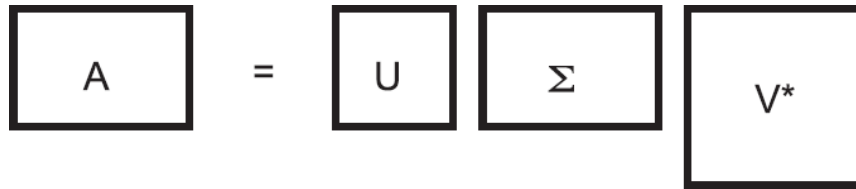


Figura 2: Descompunerea valorilor singulare pentru matricea A de dimensiune $m \times n$, unde $n > m$, $S = \Sigma$, $V^t = V^*$.

2 Compresia imaginilor folosind SVD

Descompunerea redusă a valorilor singulare presupune descompunerea (factorizarea) matricei A astfel: $A \approx A_k = U_k S_k V_k^T$, unde $A_k \in \mathbb{R}^{m \times n}$, $U_k \in \mathbb{R}^{m \times k}$, $S_k \in \mathbb{R}^{k \times k}$, $V_k^T \in \mathbb{R}^{k \times n}$.

Intuitiv, descompunerea redusă a valorilor singulare semnifică eliminarea valorilor singulare nule sau a valorilor singulare de o valoare foarte mică din matricea S (reprezentând informația puțin semnificativă). Acest lucru presupune și eliminarea coloanelor și a liniilor corespunzătoare acestor valori singulare din matricele U , respectiv din V (vezi Figura 3). Astfel, obținem o imagine aproximativă care are aproape toată informația stocată de imaginea originală, doar că o putem păstra în memorie sub o formă mai restrânsă.

În cele ce urmează, presupunem că matricea A reprezintă modelarea matematică pentru o imagine alb-negru clară și matricea A_k este modelarea matematică pentru o imagine alb-negru care aproximează imaginea clară. Ambele imagini au dimensiune $m \times n$ pixeli. Fiecare element (i, j) din matricele A și A_k corespunde intensității de gri a pixelului (i, j) din imagine. Prin urmare, elementele matricelor A și A_k au valori cuprinse între 0 (corespunzătoare culorii negre) și 255 (corespunzătoare culorii albe).

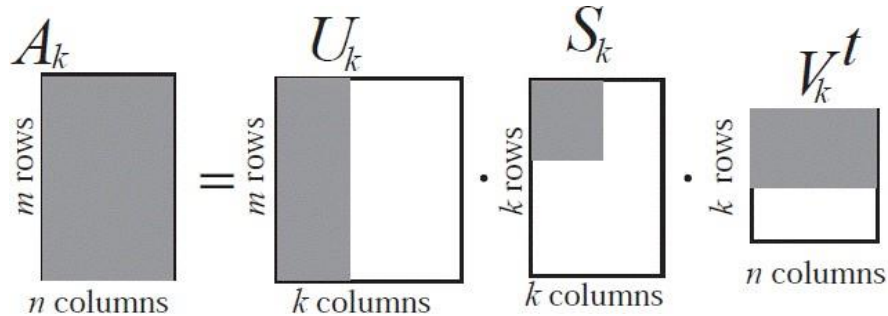


Figura 3: Exemplu de descompunere redua a valorilor singulare pentru matricea A , $m \times n$ dimensională, $m > n$. Această descompunere presupune eliminarea porțiunilor hasurate în alb din matricele U , S , respectiv V^t . Porțiunile hasurate în gri (notate U_k , S_k , respectiv V_k^t) din matricele U , S , respectiv V se vor păstra. Astfel, matricea A_k va aproxima matricea inițială A .

2.1 Task 1 [20p]

În cadrul acestei cerințe, va trebui să scrieți o funcție Octave pentru compresia unei imagini folosind descompunerea redusă a valorilor singulare. Semnatura funcției este: `function new_X = task1 (photo, k)`, unde *photo* reprezintă imaginea sub forma unei matrici și k numărul de valori singulare. Funcția trebuie să întoarcă matricea A_k având semnificația de mai sus.

3 Compresia imaginilor folosind analiza componentelor principale

Scopul analizei componentelor principale (în engleză principal component analysis - PCA), este de a transforma date de tipul $A = [a_1, a_2, \dots, a_n]$, dintr-un spațiu dimensional R^m într-un spațiu dimensional R^k , unde $a_i \in R^m$ și $k < m$. Acest spațiu este dat de cele k componente principale (PC). Componentele principale sunt ortonormate, necorelate și reprezintă direcția variației maxime. Prima componentă principală reprezintă direcția variației maxime a datelor, urmând ca următoarele componente principale să aducă variații din ce în ce mai mici.

O explicație foarte bună găsiți în următorul videoclip pe care vi-l recomand să îl urmăriți: <https://www.youtube.com/watch?v=TJdH6rPA-TI> (poate să para lung la început, dar merita).

3.1 Task 2 [20p]

Urmatorul algoritm calculeaza componentele principale folosind metoda SVD:
Fie o matrice $A \in \mathbb{R}^{m \times n}$. Notam coloanele lui A cu $b_j \in \mathbb{R}^{m \times 1}$ unde $j = 1 : n$ iar liniile lui A cu $a_i \in \mathbb{R}^{1 \times n}$ unde $i = 1 : m$.

1. Se calculeaza media pentru fiecare vector $a_i \in \mathbb{R}^{1 \times n}$, $i = 1:m$

$$\mu_i = \frac{\sum_{j=1}^n a_i(j)}{n}$$

Elementele μ_i formeaza componentele vectorului $\mu \in \mathbb{R}^{m \times 1}$.

2. Se actualizeaza vectorii $a_i \in \mathbb{R}^{1 \times n}$, $i = 1:m$ astfel: $a_i = a_i - \mu_i$
3. Se construiesc matricea $Z \in \mathbb{R}^{n \times m}$.

$$Z = \frac{A^T}{\sqrt{n-1}}$$

4. Se calculeaza SVD pentru matricea Z : $Z = USV^T$.
5. Spatiul k -dimensional al componentelor principale (notat cu W) este dat de primele k coloane din matricea $V = [v_1, v_2, \dots, v_m]$ astfel:
 $W = [v_1, v_2, \dots, v_k]$ (v_1 este prima componenta principala, v_2 este a doua si asa mai departe).
6. Se calculeaza proiectia lui A in spatiul componentelor principale, adica matricea $Y = W^T A$.
7. Se aproximeaza matricea initiala astfel: $A_k = WY + \mu$, unde μ este cel calculat la pasul 1.

Functionia Octave care implementeaza aceasta cerinta este:
`function new_X = task2 (photo, pcs)`, unde *photo* reprezinta imaginea sub forma unei matrici si *pcs* numarul de componente principale. Functionia intoarce matricea `new_X` care este matricea A_k din algoritmul explicat mai sus.

3.2 Task 3 [20p]

Componentele principale se pot calcula folosind si un algoritim bazat pe matricea de covarianta. Pasii pentru acest algoritim sunt:

Fie o matrice $A \in \mathbb{R}^{m \times n}$. Notam coloanele lui A cu $b_j \in \mathbb{R}^{m \times 1}$ unde $j = 1 : n$ iar liniile lui A cu $a_i \in \mathbb{R}^{1 \times n}$ unde $i = 1 : m$.

- Pasii 1-2 sunt aceeasi ca la Task-ul 2.
- Se construiesc matricea de covarianta $Z \in \mathbb{R}^{m \times m}$.

$$Z = \frac{A * A^T}{n-1}$$

- Se calculeaza valorile si vectorii proprii folosind functionia *eig* asupra matricei Z : $[V \ S] = \text{eig}(Z)$.

- Spatiul k-dimensional al componentelor principale (notat cu W) este dat de primele k coloane din matricea $V = [v_1, v_2, \dots, v_m] \in \mathbb{R}^{m \times m}$:

$$W = [v_1, v_2, \dots, v_k].$$

- Pasii 6-7 sunt aceeasi ca la cerinta 3.

Functia Octave care implementeaza acesta cerinta este: `function new_X = task3(photo, pcs)`, unde *photo* reprezinta imaginea sub forma unei matrici si *pcs* numarul de componente principale. Functia intoarce matricea *new_X* care este echivalentul matricei A_k din algoritm.

4 Task 4 - Recunoasterea cifrelor scrise de mana [35p]

Cu ajutorul valorilor si vectorilor proprii se poate rezolva si problema recunoasterii cifrelor. Aceasta problema este un punct de plecare si pentru alti algoritmi asemanatori, cum ar fi recunoasterea scrisului de mana in general. Recunoasterea cifrelor (sau Digit Recognition cum este des intalnita) este folosita de banci in procesarea cecurilor, citirea automata a adreselor persoanelor si citirea automata a formularelor completate. In marea majoritate a cazurilor, problema de fata este una de baza pentru persoanele care lucreaza in domeniul de *machine learning*.

Algoritmul PCA este folosit si el in acest domeniu, de multe ori fiind util pentru a micsora dimensiunea datelor, pastrand ce este mai important din inputul primit. Astfel, acuratetea nu va avea mult de suferit, dar timpul de procesare al informatiilor va fi semnificativ mai mic, ceea ce este foarte util atunci cand trebuie sa lucrezi cu seturi de date de dimensiuni foarte mari.

Inainte de a prezenta modul in care se va face recunoasterea, mai intai sa vorbim putin de setul de date. „MNIST” este un set de date de 70000 de imagini etichetate, cu cifre scrise de mana colectate de la o multitudine de persoane. Setul are o parte de 60000 de imagini de antrenament si 10000 de imagini care vor fi folosite ca test. Imaginile sunt alb-negru si au o dimensiune de 28x28 pixeli.

Pentru rezolvarea problemei de fata, exista multe variante, cele mai utilizate fiind cele care se folosesc de retelele neuronale. Pentru a va arata cum poate sa arate o astfel de rezolvare, v-am pregatit un program in Python folosind GoogleColab care recunoaste cifrele cu o acuratete de 98.4% ce poate fi gasit la link-ul:

https://colab.research.google.com/drive/1o_O4E-75-G1CFjY599E1Cd5yPcw-XO5s?usp=sharing

Pentru aceasta tema, insa, am propus un algoritm care are la baza PCA-ul si care are o acuratete de aproximativ 93.3%, dar care este mai simplu de inteles si de aplicat decat cele care se bazeaza pe metode de *machine learning*.



Figura 4 – imagini cu cifre din setul MNIST

Pentru aceasta problema, avand nevoie de multe imagini de antrenament, nu ar fi fost practic sa lucrati cu un folder cu zeci de mii de imagini, asa ca am ales optiunea de a stoca informatiile necesare in fisierul „mnist.mat”. Din acest fisier veti avea nevoie doar de imaginile de antrenament. Pentru a lucra cu acest tip de fisier, mai intai trebuie incarcat prin comanda:

`d = load ('nume_fisier')`. Pentru a obtine datele de antrenament trebuie data comanda `X = d.trainX`, iar pentru a obtine etichetele care ne indica ce numar este in fiecare imagine folositi comanda `y = d.trainY`. Astfel, `X` este o matrice `60000x784`, unde pe fiecare linie se afla valorile pixelilor unei imagini, matricea `28x28` fiind transformata intr-un vector de lungime 784.

Dupa ce avem setul de date, se poate incepe propriu-zis lucrul. Pasii algoritmului sunt urmatoarii:

1. Pregatirea setului de date prin incarcarea acestora in Octave cu comanda `load`.
2. Se calculeaza media fiecarei coloane, iar apoi este scazuta din fiecare element al coloanei. Astfel, pixelii fiecarei imagini au fost normalizati.
3. Se calculeaza matricea de covarianta folosind formula $cov_matrix = X' * X / (m - 1)$
4. Se calculeaza vectorii si valorile proprii ale matricii de covarianta.
5. Se sorteaza valorile proprii in ordine descrescatoare si pastrand aceeasi ordine se sorteaza si vectorii proprii.
6. Vectorii proprii ordonati la pasul 5 sunt folositi pentru crearea unei noi matrici `V`.
7. Din `V` sunt selectate primele `k` coloane. Pentru aceasta versiune a algoritmului `k` este ales sa fie egal cu 23.
8. Se calculeaza proiectia lui `X` in spatiul componentelor principale: $Y = X * V_k$
9. Se calculeaza aproximatia lui `X` notata \underline{X} folosind doar 23 de

componente principale: $\underline{X} = Y * V_k'$

10. Se transforma imaginea de test intr-un vector de lungime 784 si se proiecteaza in spatiul componentelor principale prin inmultirea cu V -ul obtinut la pasii anteriori.
11. Se aplica algoritmul k-nearest neighbours pentru care se alege $k = 5$, iar rezultatul obtinut este predictia cautata.

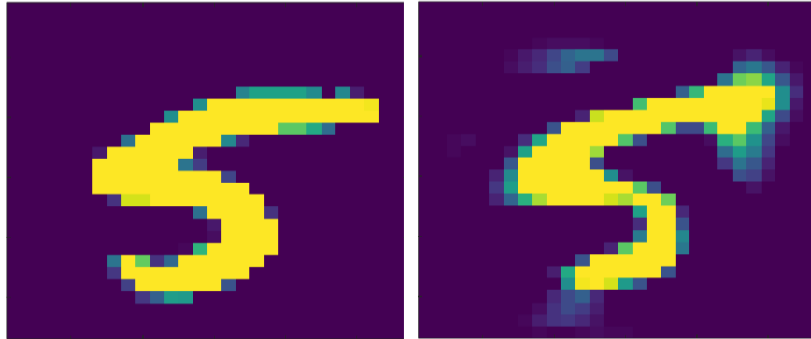


Figura 2 – Cifra initiala si cifra aproximata cu 23 de componente principale

Pentru a calcula predictia finala ne vom folosi de algoritmul k-nearest neighbours care calculeaza cele mai „apropiate” k imagini din setul de antrenament. Alegerea k-ului este o problema in sine deoarece un k prea mic poate duce la gasirea gresita a predictiei, iar un numar prea mare poate duce la suprasaturare de informatii, ceea ce duce, din nou la raspunsuri gresite. Pentru algoritmul de recunoastere a cifrelor prezentat mai sus este suficient un $k = 5$. Algoritmul k-nearest neighbours are urmatoorii pasi:

1. Se foloseste matricea Y care reprezinta proiectia matricii cu datele de intrare in spatiul componentelor principale. Se ia fiecare rand al matricii care reprezinta forma vectorizata a unei imagini si se calculeaza distanta euclidiana dintre acest vector si vectorul de test:
$$\text{distance} = \sqrt{(y_1 - x_1)^2 + \dots + (y_{784} - x_{784})^2}$$
2. Se sorteaza crescator distantele obtinute si se pastreaza doar primele 5 si se pastreaza intr-un vector valorile care reprezinta ce numar era in imaginea respectiva (i.e. prima poza din set este un 5, a doua este un 0 etc.).
3. Se calculeaza mediana vectorului obtinut mai sus si se obtine astfel predictia (HINT: functia median din Octave).

Pentru acest Task veti avea de implementat in total 6 functii: *prepare_data*, *visualise_image*, *magic_with_pca*, *prepare_photo*, *KNN* si *classify_image*.

Funcția `prepare_data` are semnatura: `[train_mat, train_val] = prepare_data (name, no_train_images)` și cu ajutorul ei veți importa datele necesare din fisierul „mnist.mat”.

Funcția `visualise_image` are semnatura: `im = visualise_image (train_mat, number)` și cu ajutorul ei puteți vizualiza imaginea cu numărul `number` din setul de date. Dacă comentați ultima comandă, atunci puteți vedea imaginea cu care lucrați.

Funcția `magic_with_pca` are semnatura: `[train, miu, Y, Vk] = magic_with_pca (train_mat, pcs)` și aplică PCA asupra matricii de antrenament, implementând pașii 2-9 din algoritmul de predicție.

Funcția `prepare_photo` are semnatura: `sir = prepare_photo (im)` și primește imaginea de test pe care o modifică și o transformă într-un `sir` pentru a se putea face mai ușor predicția. Imaginile de antrenament au fundalul negru și cifra albă, pe când cele de test au fundalul alb și cifra neagră, așa ca trebuie să se inverseze culorile.

Funcția KNN are semnatura: `prediction = KNN (labels, Y, test, k)` și pune aplică algoritmul de k-nearest neighbours.

Funcția `classify_image` are semnatura: `prediction = classifyImage (im, train_mat, train_val, pcs)` și pune cap la cap funcțiile anterioare pentru a returna predicția pe care o face.

Testele sunt împartite pe 4 categorii: „data image”, „data processing image”, „invert image” și „prediction”.

- Pentru a trece oricare dintre teste, mai întâi trebuie să implementați funcția „visualise_image”.
- Pentru a trece testele „data image” trebuie implementată și funcția „prepare_data”
- Pentru a trece testele „data processing image” mai trebuie implementată și funcția „magic_with_pca”
- Pentru a trece testele „invert image” trebuie implementată funcția „prepare_photo”
- Pentru a trece testele „prediction” trebuie implementate toate funcțiile



Figura 3 – Modul în care sunt pastrate în memorie datele de antrenament (stanga) și modul în care sunt pastrate imaginile de test (dreapta)

5 Readme [5p]

Pentru aceasta tema va trebui sa scrieti si un Readme care va fi in format PDF si inclus in arhiva finala. In acest fisier veti explica voi cum ati rezolvat cerintele propuse pentru tema si ce observatii ai avut in legatura cu calitatea imaginii la diferite numere de componente principale alese.

!!! Fisierul se va numi `Readme.pdf`, nu in alt mod pentru ca altfel nu va fi punctat de catre checker. Acesta se va pune in radacina arhivei!!!

6 Observatii finale

1. Arhiva va contine cele 5 foldere cu care a venit, checkerul si tabela MNIST din radacina. Pe langa acestea, tot in radacina va fi pus si Readme-ul. Voi veti scrie cod doar in functiile din folderul functions.
2. Imaginile de test pentru task-urile 1-4 sunt doar in format alb-negru. Pentru a nu avea probleme de calcul, pentru fiecare imagine va trebui sa fie facut mai intai un cast la double, iar apoi la final cand vrem sa vedem imaginea, un cast la uint8 pentru a obtine la final o imagine valida. Urmariti comentariile din cod pentru mai multe detalii.
3. Pentru vizualizarea imaginilor pe care le obtineti la task-urile 1-4 folositi functia `imshow` din Octave astfel: `imshow(image_matrix)`. Aceasta cerinta nu este obligatorie (checker-ul verifica doar datele returnate de functiile obligatorii), dar va ajuta sa observati diferentele pe imaginile modificate.
4. Puteti sa definiti functii auxiliare in cazul in care aveti nevoie de acestea.
5. In rezolvarea temei, aveti voie sa folositi functii Octave (inclusiv functiile `svd` si `eig`) cu urmatoarele restrictii: NU folositi functiile `princomp`, `svds`, `eigs` din Octave.
6. Checker-ul face testarea automata a cerintelor. Pentru a rula checker-ul trebuie sa fiti in folder-ul principal si sa rulati comanda `checker` in Octave in Command Window. Checker-ul va da punctajul pe Readme daca exista, dar acesta va fi primit doar dupa verificarea sa manuala.
7. Readme-ul se va intitula `Readme.pdf`
8. Tema se va incarca pe Moodle.

7 Bibliografie

1. Richard L. Burden, J. Douglas Faires, *Numerical Analysis*, Editia 9, Subcapitolul 9.6
2. http://www.cs.utexas.edu/users/inderjit/public_papers/HLA_SVD.pdf

3. <https://www.researchgate.net/publication/309165405>
4. Principal Component Analysis – Mark Richardson