Tema 2 - Metode Numerice

Facultatea de Automatică şi Calculatoare
Universitatea Politehnică Bucureşti

March 31, 2019

Cuprins

1 Introducere

In 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 intr-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 intr-un spatiu de dimensiuni reduse.

Cele mai multe transformari utilizate pentru selectia caracteristicilor sunt cele liniare, in 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 (DVS)

Fiind data o matrice $A \in R^{m*n}$, descompunerea valorilor singulare (DVS, in eng. singular value decomposition - SDV) ale matricei A este data de factorizarea $A = USV^T$, unde:

1. $U \in R^{mxm}$ este o matrice ortonormata
2. $S \in R^{mxn}$ este o matrice diagonala
3. $V \in R^{nxn}$ este o matrice ortonormata

Elementele de pe diagonala principala a lui S sunt intotdeauna 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 in ordine descrescatoare, astfel incat $s_{11} \geq s_{22} \geq ... \geq s_{rr} > s_{r+1r+1} = ... = s_{pp} = 0$, unde $p = min(m,n)$.

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

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

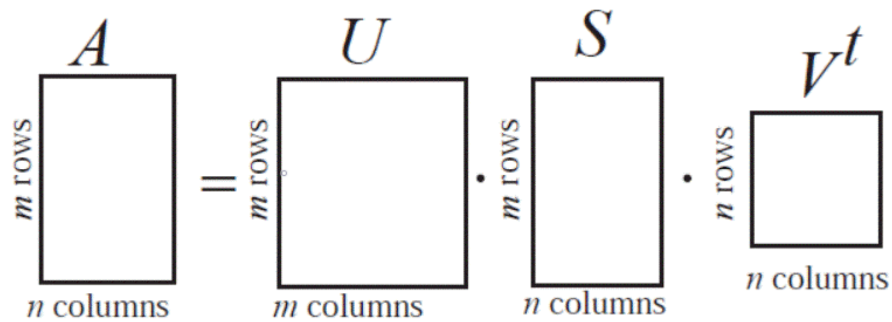

Figura 1: Descompunerea valorilor singulare pentru matricea A de dimensiune m*n, unde $m > n$.

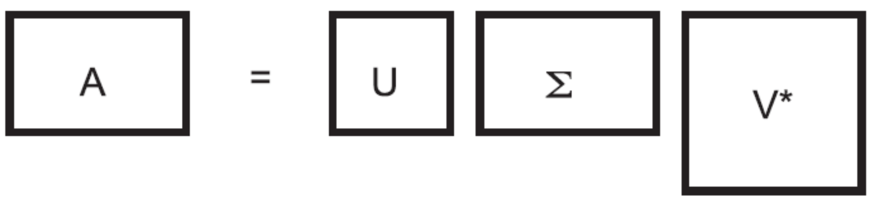

Figura 2: Descompunerea valorilor singulare pentru matricea A de dimensiune m*n, unde $n > m$, $S = \Sigma$, $V^t = V^*$.

2 Compresia imaginilor folosind DVS

Descompunerea redusa a valorilor singulare presupune descompunerea (factorizarea) matricei A astfel: $A \approx A_k = U_k S_k V_k^T$, unde $A_k \in R^{mxn}$, $U_k \in R^{mxk}$, $S_k \in R^{kxk}$, $V_k^T \in R^{kxn}$.

Intuitiv, descompunerea redusa a valorilor singulare semnifica eliminarea valorilor singulare nule sau a valorilor singulare de o valoare mica din matricea S (reprezentand informatia mai putin semnificativa). Acest lucru presupune si eliminarea coloanelor si a liniilor corespunzatoare acestor valori singulare din matricele U, respectiv din V (vezi Figura 3).

In cele ce urmeaza, presupunem ca matricea A reprezinta modelarea matematica pentru o imagine alb-negru clara si matricea A_k este modelarea matematica pentru o imagine alb-negru aproximativa a imaginii clare. Ambele imagini au dimensiune m*n pixeli. Fiecare element (i, j) din matricele A si A_k corespunde intensitatii de gri a pixelului (i, j) din imagine. Prin urmare, elementele matricelor A si A_k au valori cuprinse intre 0 (corespunzatoare culorii negre) si 255 (corespunzatoare culorii albe).

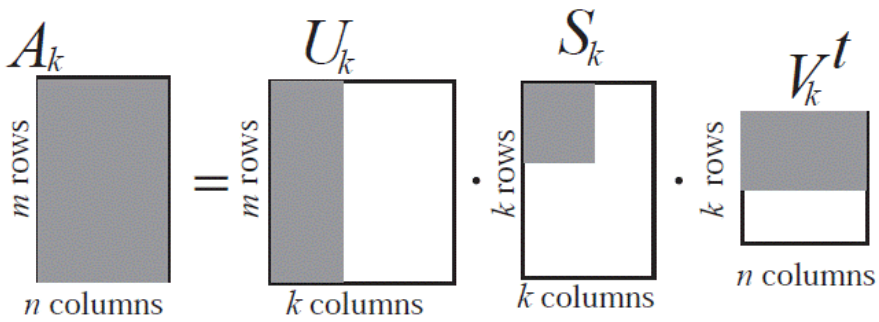

Figura 3: Exemplu de descompunere redusa a valorilor singulare pentru matricea A, m*n dimensionala, m>n. Aceasta descompunere presupune eliminarea portiunilor hasurate in alb din matricele U, S, respectiv V^t. Portiunile hasurate in gri (notate U_k, S_k, respectiv V_k^t) din matricele U, S, respectiv V se vor pastra. Astfel, matricea A_k va aproxima matricea initiala A.

2.1 Cerinta 1 [15p]

In cadrul acestei cerinte, va trebui sa scrieti o functie Octave pentru compresia unei imagini folosind descompunerea redusa a valorilor singulare. Semnatura functiei este: $function\ A\_k = task1\ (image, k)$, unde $image$ reprezinta calea catre imagine si k numarul de valori singulare. Functia trebuie sa intoarca matricea A_k avand semnificatia de mai sus.

2.2 Cerinta 2 [15p]

Scrieti o functie pentru a obtine urmatoarele 4 grafice pentru o imagine:

- folosind descompunerea valorilor singulare:

1. reprezentati grafic toate *valorile singulare* ale matricei A in ordine descrescatoare.

   - folosind descompunerea redusa a valorilor singulare (cerinta 1), pentru diferite valori ale lui k (de exemplu, k poate fi [1:19 20:20:99 100:30:min(m,n)]):

2. reprezentati grafic k (pe axa OX) si *informatia* data de primele k valori singulare rezultate din descompunerea valorilor singulare ale matricei A (pe axa OY) calculata dupa formula:

$$\frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{min(m,n)} s_{ii}}$$

3. reprezentati grafic k (pe axa OX) si *eroarea aproximarii* pentru matricea A (pe axa OY) calculata dupa formula:

$$\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(A(i,j)-A_k(i,j))^2}{m*n}$$

4. reprezentati grafic k (pe axa OX) si *rata de compresie a datelor* (pe axa OY) calculata dupa formula:

$$\frac{m*k+n*k+k}{m*n}$$

   Justificare formula: Pentru obtinerea imaginii aproximative avem nevoie sa memoram doar matricele U_k, V_k^T si primele k elemente de pe diagonala principala a matricei S_k. In total, m*k+n*k+k elemente trebuie memorate. Astfel, stocarea acestora ocupa memorie mai putina comparativ cu memoria necesara matricei A pentru m*n elemente. Tinand cont de faptul ca cea mai mare parte din informatia continuta in matrice este data de primele valori singulare, compresiea datelor folosind descompunerea redusa a valorilor singulare permite o reducere de memorie.

Semnatura functiei este $function\ task2()$.

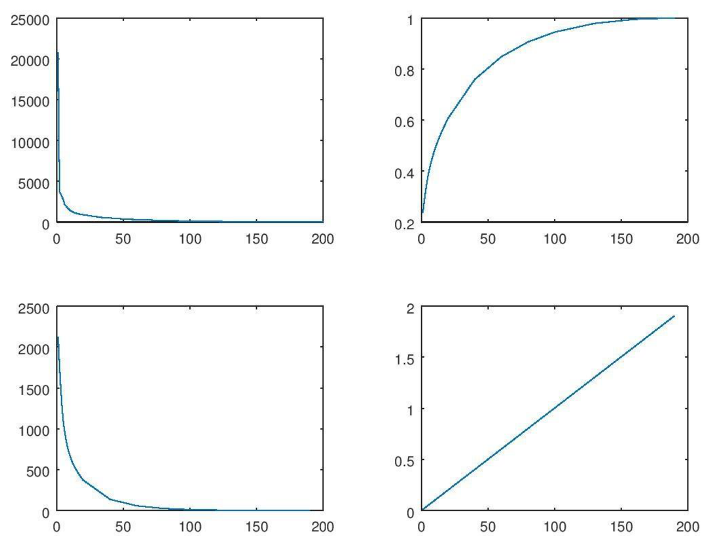

Figura 4: Graficele rezultate pentru cerinta 2 - image1.

Semnificatia graficelor din Figura 4 este urmatoarea:

1. Graficul din stanga-sus ilustreaza valorile singulare, in ordine descrescatoare. Primele valori singulare concentreaza o mare parte din informatia imaginii iar celelalte valori singulare din ce in ce mai putin.

2. Graficul din dreapta-sus reprezinta raportul dintre suma primelor k valori singulare si suma totala. Cu cat luam mai multe valori singulare, raportul tinde catre 1. Este o functie concava pentru ca primele valori singulare concentreaza o parte mai mare de informatie. De aceea, la inceput, cresterea este mai pronuntata. Cu cat luam mai multe valori, panta scade.

3. Graficul din stanga-jos ilustreaza eroarea aproximarii. Initial eroarea este mare, pentru ca imaginea rezultata din compresia cu un k mic difera foarte mult fata de imaginea initiala. Insa, cu cat luam mai multe valori singulare in considerare, imaginea devine mai fidela, iar eroarea din ce in ce mai mica.

4. Graficul din dreapta-jos reprezinta o dreapta de panta pozitiva si de ecuatie $\frac{m*k+n*k+k}{m*n}$, cu m si n constante, iar valoarea k necunoscuta. Concret, rata de compresie a datelor reprezinta raportul dintre memoria ocupata pentru retinerea imaginii la pasul k si memoria totala necesara pentru retinerea imaginii.

3 Compresia imaginilor folosind analiza componentelor principale

Scopul analizei componentelor principale (in eng. principal component analysis - PCA), este de a transforma date de tipul $A = [a_1, a_2, ..., a_n]$, dintr-un spatiu dimensional R^m intr-un spatiu dimensional R^k, unde $a_i \in R^m$ si k<m. Acest spatiu este dat de cele k componente principale (PC). Componentele principale sunt ortonormate, necorelate si reprezinta directia variatiei maxime. Prima componenta principala reprezinta directia variatiei maxime a datelor, urmand ca urmatoarele componente principale sa aduca variatii din ce in ce mai mici.

3.1 Cerinta 3 [15p]

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

1. Se calculeaza media pentru fiecare vector $a_i \in R^{1*n}, i = 1 : m$: $\mu_i = \frac{\sum_{j=1}^{n} a_i(j)}{n}$. Elementele μ_i formeaza componentele vectorului $\mu \in R^{m*1}$.

2. Se actualizeaza vectorii $a_i \in R^{1*n}, i = 1 : m$ astfel: $a_i = a_i - \mu_i$.

3. Se construieste matricea $Z = \frac{A^T}{\sqrt{n-1}}, Z \in R^{n*m}$.

4. Se calculeaza DVS pentru matricea Z: $Z = USV^T$.

5. Spatiul k-dimensional al componentelor principale (notat cu W) este dat de primele k coloane din matricea $V = [v_1, v_2, ..., v_m] \in R^{m*m}$ astfel: $W = [v_1, v_2, ..., v_k]$ (v_1 este prima PC, v_2 este a doua PC s.a.m.d).

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 a fost calculat la pasul 1.

Functia Octave care implementeaza acesta cerinta este: function $[A_k\ S]$ = task3(image, k), unde *image* reprezinta calea catre imagine si k numarul de componente principale. Functia intoarce matricele A_k si S cu semnificatia prezentata in acest algoritm.

3.2 Cerinta 4 [15p]

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

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

- Pasii 1-2 sunt aceeasi ca la cerinta 3.
- Se construieste matricea de covarianta $Z = \frac{A*A^T}{n-1}, Z \in R^{m*m}$.
- Se aplica functia *eig* asupra matricei Z: $[V\ S] = eig(Z)$.
- Spatiul k-dimensional al componentelor principale (notat cu W) este dat de primele k coloane din matricea $V = [v_1, v_2, ..., v_m] \in R^{m*m}$: $W = [v_1, v_2, ..., v_k]$.
- Pasii 6-7 sunt aceeasi ca la cerinta 3.

Functia Octave care implementeaza acesta cerinta este: function $[A_k\ S]$= task4(image, k), unde *image* reprezinta calea catre imagine si k numarul de componente principale. Functia intoarce matricele A_k si S cu semnificatia prezentata in acest algoritm.

3.3 Cerinta 5 [10p]

Folosind *cerinta* 3, scrieti o functie pentru a obtine urmatoarele 4 grafice pentru o imagine:

1. reprezentati grafic vectorul $diag(S)$.

   pentru diferite valori ale lui k (de exemplu, k poate fi [1:19 20:20:99 100:30:min(m,n)]):

2. reprezentati grafic k (pe axa OX) si *informatia* data de primele k valori singulare rezultate din descompunerea valorilor singulare ale matricei Z (pe axa OY) calculata dupa formula:

$$\frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{min(m,n)} s_{ii}}$$

3. reprezentati grafic k (pe axa OX) si *eroarea aproximarii* pentru matricea A (pe axa OY) calculata dupa formula:

$$\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} (A(i,j) - A_k(i,j))^2}{m * n}$$

4. reprezentati grafic k (pe axa OX) si *rata de compresie a datelor* (pe axa OY) calculata dupa formula: $\frac{2*k+1}{n}$ Justificare formula: Matricea A are dimensiunea m*n. Pentru reconstructia matricei A este nevoie de W, Y si μ avand fiecare m liniii si k coloane (matricele W si Y), respectiv 1 coloana (vectorul μ). Prin urmare, numarul de coloane m-dimensionale de memorat a fost redus de la n la 2*k+1.

Semnatura functiei este $function\ task5()$.

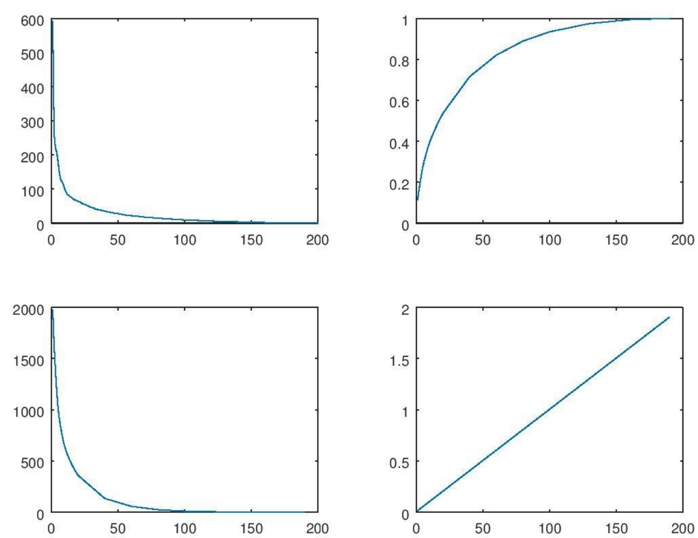

Figura 5: Graficele rezultate pentru cerinta 5 - image1. Interpretarea este aceeasi ca la cerinta 2. Rata de compresie se schimba, de data acesta fiind $\frac{2*k+1}{n}$. Cu toate acestea, ramane o dreapta cu panta pozitiva.

4 Recunoasterea faciala [25p]

O alta aplicatie importanta care se bazeaza pe calcularea valorilor si vectorilor proprii este cea de Recunoastere Faciala. La aceasta cerinta, algoritmul va indica pe baza unei multimi de imagini M daca alte imagini sunt fete cunoscute, necunoscute sau nu reprezinta fete umane. Pasii algoritmului sunt urmatorii:

1. Se citeste fiecare imagine aflata in directorul *dataset* si se transforma intr-un vector coloana. Toti vectorii obtinuti se pun intr-o matrice T.
2. Se calculeaza media pe fiecare linie din matricea T. Rezultatele obtinute se vor salva in vectorul m.
3. Se calculeaza matricea $A = T$ - m.
4. Se calculeaza matricea cu fetele proprii $EigFaces = A * V$, unde matricea V contine vectorii proprii corespunzatori valorilor proprii mai mari decat 1 ale matricei $A^T * A$. $EigFaces$ reprezinta spatiul fetelor.
5. Se calculeaza proiectia fiecare imagini din multimea de imagini M in spatiul fetelor astfel: $PrImg = EigFaces^T * A$.
6. Data fiind o imagine de test, aceasta se transforma intr-un vector coloana si se extrage din ea media similar pasilor 1 si 3.
7. Se calculeaza proiectia imaginii de test in spatiul fetelor astfel: $PrTestImg = EigFaces^T * vectorul\ coloana\ rezultat\ la\ pasul$ 6.
8. Se determina cea mai mica distanta intre proiectia imaginii de test obtinuta la pasul 7 si proiectiile obtinute la pasul 5. Intuitiv, la acest pas, imaginea din multimea M cea mai asemanatoare cu imaginea de test se determina.

Pasii 1-5 se vor implementa in urmatoarea functie:
$function\ [m\ A\ eigenfaces\ pr\_img] = eigenface\_core(database\_path)$

Pasii 6-8 se vor implementa in urmatoarea functie:
$function\ [min\_dist\ output\_img\_index] = face\_recognition(image\_path,$ m, A, $eigenfaces$, $pr\_img)$
Ultima functie returneaza cea mai mica distanta si indicele proiectiei imaginii de la pasul 5 pentru care se obtinea cea mai mica distanta.

Observatii pentru cerinta curenta:

1. Imaginile trebuie sa fie convertite in formatul grayscale, deoarece imaginele in format RGB reprezinta matrice 3D, nu 2D. In acest scop, se va folosi comanda $double(rgb2gray(imread(image\_path)))$. De asemenea, transformarea unei matrice intr-un vector coloana se va face parcurgand elementele primei linii, apoi elementele de pe a doua linie s.a.m.d. pana la ultima linie.

2. Puteti decomenta codul din functia $task6.m$ pentru a vedea imaginile obtinute, in rest nu motificati aceasta functie.

5 Observatii finale

1. Imaginile de test pentru cerintele 1-5 sunt doar in format alb-negru. Pentru a citi o imagine in program folositi functiile *imread* si *double* din Octave astfel: $image\_matrix = double(imread(image\_path))$.

2. Pentru vizualizarea imaginilor pe care le obtineti la cerintele 1-5 folositi functiile *imshow* si $uint8$ din Octave astfel: $imshow(uint8(image\_matrix))$. Aceasta cerinta nu este obligatorie (checker-ul verifica doar datele returnate de functiile obligatorii), dar va ajuta sa observati diferentele pe imaginile modificate.

3. Puteti sa definiti functii auxiliare in cazul in care aveti nevoie de acestea.

4. In rezolvarea temei, aveti voie sa folositi functiile din Octave (inclusiv functiie *svd* si *eig*) cu o singura restrictie: NU folositi functia *princomp* din Octave.

5. Fisierul *Readme* (cu extensia *.pdf*) va contine detalii despre cum ati rezolvat cerintele si graficele obtinute la cerintele 2 si 5.

   Nu este nevoie sa interpretati graficele obtinute la cerintele 2 si 5 este suficient doar sa le afisati in Readme. Cerintele 2 si 5 sa le testati pentru oricare 2 imagini diferite de prima imagine din directorul *in*. In total, Readme va contine 16 grafice. Tot in Readme puteti pune si poze cu imaginile obtinute la cerintele 2 si 5.

6. Checker-ul face testarea automata doar a cerintelor 1, 3, 4 si 6, cerintele 2 si 5 le vom corecta manual.

7. Fisierul *Readme* se puncteaza cu 5 puncte.

8. Tema se va incarca atat pe Moodle cat si pe vmchecker.

6 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