Laborator 4

Triangularizare ortogonală cu reflectori. Problema CMMP. Clasificare.

4.1 Triangularizare ortogonală cu reflectori

4.1.1 Reflectori elementari

Definiţia 4.1 Un *reflector elementar* (sau Householder) de ordin m şi indice k este o matrice $U_k \in \mathbb{R}^{m\times m}$, de forma

$$U_k = I_m - \frac{1}{\beta_k} u_k u_k^T, \tag{4.1}$$

unde $u_k \in \mathbb{R}^m$ este vectorul

$$u_k = [\underbrace{0 \ \ldots \ 0}_{k-1} \ u_{kk} \ u_{k+1,k} \ \ldots \ u_{mk}]^T$$

şi

$$\beta_k = \|u_k\|^2/2.$$

Principalele proprietăţi ale transformărilor Householder sunt date în următoarea teoremă.

Teorema 4.2 *a) Un reflector elementar $U_k \in \mathbb{R}^{m\times m}$ este o matrice simetrică şi ortogonală.*

b) Dându-se un reflector elementar U_k şi un vector $x \in \mathbb{R}^m$, produsul lor $y = U_k x$ se poate calcula prin

$$y_i = \begin{cases} x_i, & i = 1 : k-1 \\ x_i - \tau u_{ik}, & i = k : m \end{cases} \tag{4.2}$$

unde

$$\tau = \frac{1}{\beta_k} \sum_{i=k}^{m} u_{ik} x_i.$$

c) Dându-se un indice $k \in 1 : m-1$ *şi un vector* $x \in \mathbb{R}^m$ *astfel încât*

$$\sigma^2 = \sum_{i=k}^{m} x_i^2 \neq 0, \tag{4.3}$$

reflectorul U_k *cu ajutorul căruia se pot introduce zerouri în poziţiile* $k+1 : m$ *ale vectorului* x*, mai precis care satisface relaţia*

$$(U_k x)_i = \begin{cases} x_i, & i = 1 : k-1 \\ -\sigma, & i = k \\ 0, & i = k+1 : m \end{cases}$$

este definit de vectorul $u_k \in \mathbb{R}^m$ *dat de*

$$u_{ik} = \begin{cases} 0, & i = 1 : k-1 \\ x_k + \sigma, & i = k \\ x_i, & i = k+1 : m \end{cases} \tag{4.4}$$

şi scalarul

$$\beta_k = \sigma u_{kk}.$$

4.1.2 Triangularizarea ortogonală

Rezolvarea problemei CMMP într-o manieră stabilă din punct de vedere numeric necesită o procedură pentru reducerea matricelor la formă triunghiulară prin transformări ortogonale.

Teorema 4.3 *Fie* $A \in \mathbb{R}^{m \times n}$*, cu* $m \geq n$*; atunci există reflectorii elementari* U_1*,* U_2*, ...,* U_p*, unde* $p = \min(m-1, n)$*, astfel încât*

$$U_p \ldots U_2 U_1 A = R, \tag{4.5}$$

unde R *este superior triunghiulară.*

Pentru triangularizarea ortogonală a matricei $A \in \mathbb{R}^{m \times n}$ putem folosi următoarea schemă (care calculează R în A):

1. **pentru** $k = 1 : \min(m-1, n)$
   1. Se determină U_k astfel încât $(U_k a_k)_i = 0$, pentru $i = k+1 : m$
   2. $A \leftarrow U_k A$

Schema de calcul de mai sus conduce la următorul algoritm.

ALGORITM 4.4 *(TORT – triangularizare ortogonală cu reflectori)* (Dată o matrice $A \in \mathbb{R}^{m\times n}$, algoritmul suprascrie matricea A cu matricea superior triunghiulară R şi calculează vectorii $u_k = U(:,k)$ şi scalarii β_k care definesc reflectorii elementari U_k, $k \in 1:p$, $p = \min(m-1,n)$, astfel încât are loc (4.5).)

1. $p = \min(m-1,n)$
2. **pentru** $k = 1:p$
   1. $\sigma \leftarrow \text{sign}(a_{kk}) \cdot \sqrt{\sum_{i=k}^{m} a_{ik}^2}$
   2. **dacă** $\sigma = 0$ **atunci** $\beta_k \leftarrow 0$
      altfel
      1. $U(k,k) \leftarrow u_{kk} = a_{kk} + \sigma$
      2. **pentru** $i = k+1:m$
         1. $U(i,k) \leftarrow u_{ik} = a_{ik}$
      3. $\beta_k \leftarrow \sigma u_{kk}$
      4. $a_{kk} \leftarrow r_{kk} = -\sigma$
      5. (**pentru** $i = k+1:m$
         1. $a_{ik} \leftarrow 0$)
      6. **pentru** $j = k+1:n$
         1. $\tau \leftarrow \left(\sum_{i=k}^{m} u_{ik}a_{ij}\right)/\beta_k$
         2. **pentru** $i = k:m$
            1. $a_{ij} \leftarrow a_{ij} - \tau u_{ik}$

Algoritmul poate fi apelat cu sintaxa $[R,U,\beta] = TORT(A)$.

4.2 Problema celor mai mici pătrate

Fie $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ date. Considerăm sistemul liniar *supradeterminat* ($m > n$)

$$Ax = b.$$

În general, un astfel de sistem, având mai multe ecuaţii decât necunoscute, nu are soluţii. O reformulare naturală a rezolvării sistemului este problema găsirii unui vector $x \in \mathbb{R}^n$ astfel încât vectorul $y = Ax$ să fie cât mai aproape posibil de vectorul b. Când apropierea este definită de norma euclidiană, se obţine *problema celor mai mici pătrate* (CMMP):

$$\min_{x\in\mathbb{R}^n} \|r\| = \min_{x\in\mathbb{R}^n} \|b - Ax\|. \tag{4.6}$$

Vectorul $r = b - Ax$ este numit reziduu, iar norma lui este minimizată.

4.2.1 Rezolvarea problemei CMMP

Condiţiile în care problema CMMP are o soluţie şi când această soluţie este unică sunt date de următoarea teoremă.

Teorema 4.5 *Fie date $A \in \mathbb{R}^{m\times n}$ şi $b \in \mathbb{R}^m$, cu $m > n$. Problema CMMP (4.6) admite întotdeauna o soluţie. Soluţia este unică dacă şi numai dacă matricea A are coloanele liniar independente, adică are rang maxim. În acest caz soluţia CMMP a sistemului supradeterminat $Ax = b$ poate fi scrisă în forma*

$$x^* = (A^T A)^{-1} A^T b. \tag{4.7}$$

Relaţia (4.7) este utilizabilă direct doar pentru matrice de dimensiuni mici. Un mijloc de calcul numeric stabil a soluţiei CMMP constă în triangularizarea ortogonală a matricei sistemului (algoritmul $TORT$ din secţiunea precedentă). Notăm $U = U_p \dots U_2 U_1$ în (4.5) şi observăm că putem scrie

$$R = \begin{bmatrix} R' \\ 0 \end{bmatrix},$$

cu R' o matrice pătrată $n \times n$ superior triunghiulară. Matricea R are coloanele liniar independente întrucât A are această proprietate şi, deci, R' este o matrice nesingulară. Transformările ortogonale conservând norma euclidiană, avem

$$\|b - Ax\| = \|U(b - Ax)\| = \|d - Rx\|, \tag{4.8}$$

unde $d = Ub$. Folosind partiţia

$$d = \begin{bmatrix} d' \\ d'' \end{bmatrix}, \tag{4.9}$$

unde $d' \in \mathbb{R}^n$ şi $d'' \in \mathbb{R}^{m-n}$, relaţia (4.8) poate fi scrisă

$$\|b - Ax\| = \left\| \begin{bmatrix} d' - R'x \\ d'' \end{bmatrix} \right\| = \sqrt{\|d' - R'x\|^2 + \|d''\|^2},$$

care este minimă când $d' - R'x = 0$. Deci, soluţia CMMP a sistemului supradeterminat $Ax = b$ poate fi calculată rezolvând sistemul superior triunghiular nesingular

$$R'x = d'.$$

4.2.2 Algoritm

Operaţiile de mai sus sunt implementate de următorul algoritm.

> ALGORITM 4.6 *(CMMP – problema celor mai mici pătrate)* (Date o matrice $A \in \mathbb{R}^{m\times n}$, $m > n$, cu coloanele liniar independente, şi un vector $b \in \mathbb{R}^n$, acest algoritm calculează soluţia CMMP a sistemului liniar supradeterminat $Ax = b$.)

{ Triangularizarea ortogonală a lui A }
1. $[R, U, \beta] = TORT(A)$
{ Calculul vectorului d care suprascrie b }
2. **pentru** $k = 1 : n$
   1. $\tau \leftarrow \left(\sum_{i=k}^{m} u_{ik} b_i\right) / \beta_k$
   2. **pentru** $i = k : m$
      1. $b_i \leftarrow b_i - \tau u_{ik}$
{ Calculul soluţiei CMMP }
3. $x = UTRIS(R(1 : n, :), b(1 : n))$

4.3 Clasificare binară cu CMMP

4.3.1 Problema, soluţia CMMP şi interpretare

Se presupune că avem exemple de semnale (vectori) din două clase şi dorim ca, pe baza lor, să creăm o regulă prin care să clasificăm semnale viitoare. Notăm N_1, N_2 numărul de vectori (de antrenare) aparţinând claselor $\mathcal{C}_1$, respectiv $\mathcal{C}_2$; vectorii au dimensiune ℓ. Un clasificator foarte simplu este cel bazat pe CMMP. (Atragem atenţia că există metode de clasificare mai eficiente, dar şi mai sofisticate.) Acesta construieşte funcţia liniară

$$f(v) = c^T v + d \tag{4.10}$$

unde $v \in \mathbb{R}^\ell$ este un semnal de antrenare, iar $c \in \mathbb{R}^\ell$ şi $d \in \mathbb{R}$ sunt parametrii încă necunoscuţi ai funcţiei.

Pentru a afla aceşti parametri, fixăm ca obiectiv satisfacerea relaţiei

$$f(v) = \begin{cases} 1, & v \in \mathcal{C}_1 \\ -1, & v \in \mathcal{C}_2 \end{cases} \tag{4.11}$$

Desigur, această relaţie nu poate fi satisfăcută exact, de aceea căutăm o soluţie în sens CMMP. Notând $\mathbf{1}_n$ un vector având toate elementele egale cu 1 şi dimensiune n, relaţia (4.11) conduce la sistemul CMMP

$$\begin{bmatrix} V_1^T & \mathbf{1}_{N_1} \\ V_2^T & \mathbf{1}_{N_2} \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \mathbf{1}_{N_1} \\ -\mathbf{1}_{N_2} \end{bmatrix}, \tag{4.12}$$

în care $V_1 \in \mathbb{R}^{\ell \times N_1}$ şi $V_2 \in \mathbb{R}^{\ell \times N_2}$ sunt matrice ale căror coloane sunt vectorii de antrenare din cele două clase. Rezolvarea sistemului (4.12), a cărui matrice are dimensiunea $(N_1 + N_2) \times (\ell + 1)$, produce valorile optime ale parametrilor c şi d pentru această metodă.

Clasificarea efectivă a unui semnal nou se face ţinând cont de care dintre valorile 1 sau -1 este mai aproape valoarea funcţiei (4.10). Aceasta decizie se poate lua pe

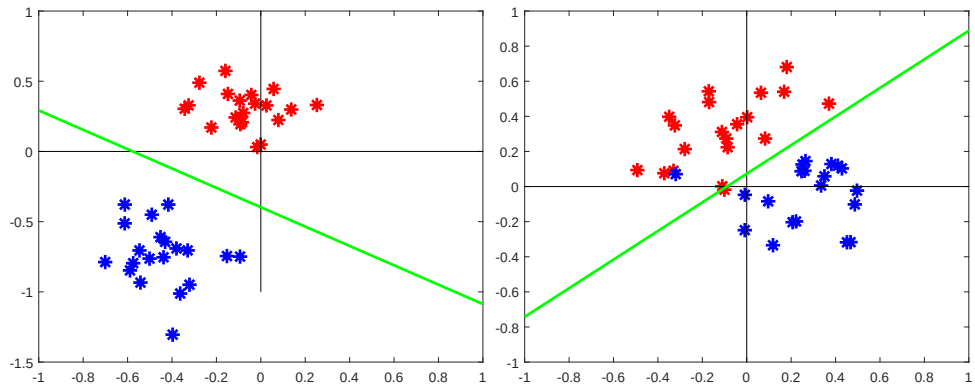

Figure 4.1: Exemple de clasificator CMMP.

baza semnului funcţiei (4.10). Mai precis, dacă v este acum un semnal oarecare, decizia de clasificare este

$$v \in \begin{cases} \mathcal{C}_1, & \text{sign}(c^T v + d) = 1, \\ \mathcal{C}_2, & \text{altfel} \end{cases} \tag{4.13}$$

Semnificaţia geometrică este simplă. Hiperplanul $c^T v + d = 0$ separă spaţiul $\mathbb{R}^\ell$ în două, asociind fiecare jumătate unei clase. Exemple pentru $\ell = 2$, când hiperplanul separator devine o dreaptă, sunt prezentate în figura 4.1. Vectorii de antrenare din cele două clase sunt reprezentaţi cu roşu şi albastru, iar dreapta separatoare cu verde (cu negru sunt axele de coordonate). In exemplul din stânga, cele două clase sunt bine separate de dreaptă; aceasta este situaţia ideală, mai rar întâlnită în practică. In exemplul din dreapta nu există o dreaptă separatoare; cea furnizată de metoda CMMP este însă un compromis acceptabil pentru a generaliza din exemple clasele "roşie" şi "albastră".

Rezultatele clasificării semnalelor de antrenare este cuantificată prin matricea de confuzie, al cărui element (i, j) reprezintă numărul de semnale din clasa i care sunt clasate în clasa j; deci pe diagonală se află numărul de semnale clasificate corect, iar pe celelalte poziţii numărul de semnale clasificate incorect. Pentru figura 4.1 din dreapta (clasa roşie fiind $\mathcal{C}_1$ şi fiecare clasă având 20 de semnale), matricea de confuzie este

$$\begin{bmatrix} 20 & 0 \\ 1 & 19 \end{bmatrix}.$$

Toate semnalele roşii sunt clasificate corect, dar unul albastru este clasificat incorect.

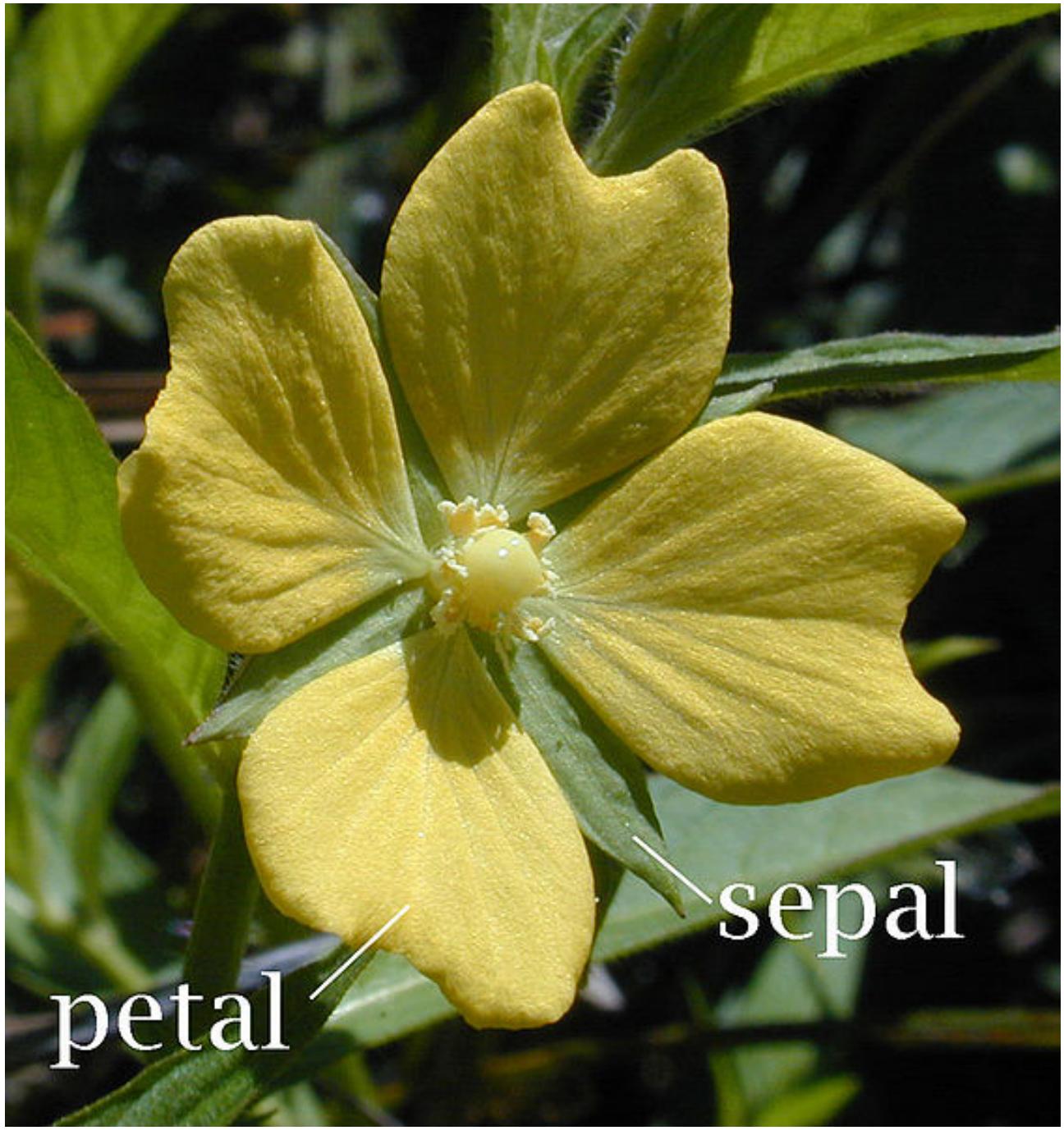

Figure 4.2: Petale şi sepale. (In engleză, în figură.)

1 Clasificarea ochilor

Scopul acestei aplicatii este de a gasi un clasificator care sa poata decide daca ochiul unei persoane este închis sau deschis. Amintim ca pentru orice clasificator avem nevoie de o perioada de antrenare, în care se gaseste parametri acestuia, si una de testare, în care de obicei se stabileste precizia acestuia.

1.1 Faza de antrenare

Pentru aceasta aplicatie, se va folosi în faza de antrenare 100 de poze alb negru (50 de imagini sunt cu ochi închisi si restul cu ochi deschisi), de dimensiune 24x24.

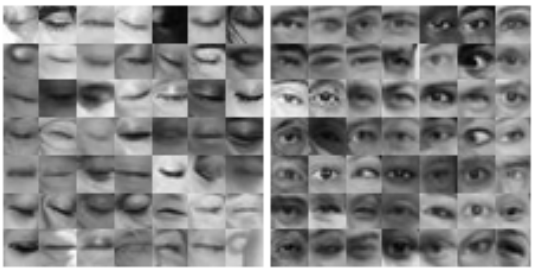

Figure 1: Poze din baza de date

Primul pas este de a încarca baza de date:

```
load('eye_data_lab4');
```

Aceasta comanda va furniza în spatiul de lucru doua variabile:

- Matricea *features* de dimensiune 100x144, unde numarul de linii indica numarul de poze utilizat pentru antrenare, iar numarul de coloane specifica dimensiunea vectorului de caracteristici.
- Vectorul *labels* de dimensiune 100x1, unde numarul de linii indica numarul de poze. Elementul labels(i) este eticheta pentru vectorul caracteristic features(i,:). Valorile din acest vector sunt:

-1 pentru ochi închis

1 pentru ochi deschis

Urmatoarea comanda adauga la finalul matricei features o linie plina cu unu pentru a putea forma sistemul CMMP:

```
features =[features, ones(100,1)];
```

Ultima etapa din faza de antrenare este rezolvarea efectiva a sistemului CMMP (folosim comanda matlab *backslash*):

```
classifier = features \ labels;
```

1.2 Faza de testare

Se va citi o imagine ce se gaseste în folderul de lucru:

```
testImg = imread('closed_eye_test1_lab4.jpg');
```

Se va obtine o matrice testImg de 24x24 ce contine valori intre 0 si 256, reprezentând intensitatea pixelilor. Mai departe, aceasta matrice va fi binarizata (adica elementele matricea vor fi transformate în 1 sau 0) pentru a putea extrage trasaturile din aceasta imagine:

```
testImgMat = imbinarize(testImg);
%Extract the feature vector from the image
testFeatures = extractHOGFeatures(testImgMat);
```

S-a ales pentru analiza trasaturile tipul *Histogram Oriented Gradient*(HOG),acestea putând fi extrase cu ajutorul comenzii Matlab *extractHOGFeatures*.
Dimensiunea vectorul de trasaturi *testFeatures* depinde de dimensiunea imaginii analizate, cât si de 4 parametri ai functiei: *BlockSize, CellSize, BlockOverlap, si NumBins*. Acestia au ramas la valorile lor default(BlockSize =[2 2], CellSize = [8 8], BlockOverlap = 1, si NumBins =9). Pentru mai multe delatii legat de acest aspect puteti apela în linia de comanda:

```
>> doc extractHOGFeatures
```

Remarca: Datele din faza de antrenare au fost create în aceeasi maniera ca si datele de antrenare.

Dupa ce am extras vectorul de trasaturi din imagine, ramâne doar sa evaluam functia f utilizând clasificatorul pe care l-am gasit în faza de antrenare:

```
test_label = testFeatures * classifier(1:end-1) +
   classifier(end)
```

Afisati imaginea pentru verificare:

```
imshow('closed_eye_test1_lab4.jpg')
```

2 Clasificarea florilor Iris

În aceasta aplicatie, clasificatorul gasit va trebui sa decida daca o floare Iris este de tipul Setosa sau Versicolor.
Fisierul iris_data_lab4.mat contine 4 variabile:

- trainingIris - este o matrice de dimensiune 50x4, în care numarul de linii reprezinta numarul de masuratori, iar numarul de coloane reprezinta câte componente au fost evaluate (lungimea si latimea sepalei, respectiv a petalei)
- trainigLabel - este un vector de dimensiune 50x1, ce contine etichetele florilor:
  1 - pentru Iris Setosa
  2 - pentru Iris Versicolor
- testIris - are aceeasi dimensiune si semnificatie ca trainingIris, doar ca aceste date au fost salvate pentru faza de testare
- testLabel - etichetele pentru datele din faza de testare

Spre deosebire de aplicatia de clasificare a ochilor, de aceasta data se vor testa 50 de masuratori si se va calcula eroarea de predictie (adica câte predictii au fost gresite)

3 Sarcini de lucru

3.1 In laborator

- Elaboratti si editati programul MATLAB pentru implementarea algoritmului TORT de triangularizare ortogonala. Testatti functionarea programului pentru matrice cu elemente aleatoare de dimensiuni cel putin 7×5.
- Fie $H \in R^{m \times n}, (m > n)$ o matrice superior Hessenberg. Scrieti algoritmi eficienti si programe pentru triangularizarea ortogonala a matricei H.
- Elaboratti si editati programul MATLAB pentru implementarea algoritmului CMMP de rezolvare a sistemelor supradeterminate si verificati corectitudinea pentru sisteme cu 20 de ecuatii si 6 necunoscute.

- Studiati si executati programul script clasificare-ochi-lab4, care implementeaza metoda CMMP de clasificare binara pentru ochi. Testati cele patru imagini test si interpretati rezultatele. Inlocuiti backslash cu algoritmul CMMP scris de voi.

3.2 Acasa

- Scrieti si testati programe MATLAB pentru: a) calculul elementelor definitorii ale unui reflector U_k care anuleaza componentele $k+1:n$ ale unui vector $x \in R^n$ dat; b) inmultirea unei matrice A pe stanga cu un reflector U_k: $A = U_k A$; c) inmultirea unei matrice A pe dreapta cu un reflector U_k: $A = AU_k$.

- Rescrieti algoritmul de triangularizare ortogonala a unei matrice cu folosirea procedurilor de mai sus.

- Consideram dat vectorul $x \in R^n$, de norma unitara. Scrieti un program de calcul al unei matrice cu coloanele ortogonale $Y \in R^{n\times(n-1)}$ astfel incat $x^T Y = 0$ (i.e. matricea $[x\ Y]$ sa fie ortogonala).

- Fie $A \in R^{m\times n}, m > n$, o matrice cu rang maxim. Elaborati un algoritm si un program de calcul al pseudo-inversei $A^+ = (A^T A)^{-1} A^T$ a lui A.

- Studiati si executati programul script clasificare-iris-lab4-BD, care implementeaza metoda CMMP de clasificare binara. Modificati diversi parametri pentru a vedea rezultate diferite. (Figura 4.1 este generata cu acest program.)

- Studiati si executati programul script clasificare-iris-lab4, care implementeaza metoda CMMP de clasificare binara pentru flori. Calculati matricea de confuzie.

- Gasiti alte date pentru clasifcare si incercati metoda CMMP (exemplu pentru clasificare email-uri).

4 Cod Matlab pentru clasificare ochi/flori

CMMP OCHI

```
clear; clc;
%% ============ Description ====================
% features = n x m feature matrix where n is the number of
   img and m is the
%            dimension of the feature vector of one img
% labels = is the label vector; Semnification: 1- open eye;
    -1 - eye closed
% * The data base used gray images of 24x24 pixel.
%% ============ Training Step ====================
%Load data to workspace.
%This data base contains 2 variables: features and labels
load('eye_data_lab4');

features =[features, ones(100,1)];
%Finding the classifier by solving the CMMP problem
classifier = features\labels; % replace it with the
   function writen in lab

%% ============ Testing Step ====================
%Read the image
testImg = imread('closed_eye_test1_lab4.jpg');% replace the
    name of the img
% with another image
%Pre-process step
testImgMat = imbinarize(testImg);
%Extract the feature vector from the image
%Default values for the paramters of the function:
% BlockSize =[2 2], CellSize = [8 8], BlockOverlap = 1  and
    NumBins =9
testFeatures = extractHOGFeatures(testImgMat);
%Test the image
test_label = testFeatures * classifier(1:end-1) +
   classifier(end)
% Display image to check result
imshow('closed_eye_test1_lab4.jpg')% replace the name of
   the img
% with another image
```

CMMP IRIS

```
clear; clc;
%% ============ Description ========================
% trainingIris - matrix of dim nxm where n is the number of
    measurement of
%                the petals and sepals and m is the
   dimension of this
%                measurements
% trainingLabel - vector of dim nx1 that contains the
   labels;
%                 Semnification: 1- Iris Setosa; 2- Iris
   Versicolor
% testIris, testLabel are the data for testing the
   classifier.
%
%% ============ Training Step ======================
%Load data to workspace.
load('iris_data_lab4');

features =[trainingIris, ones(50,1)];
%Finding the classifier by rezolving the CMMP problem
classifier = features\trainingLabel; % replace it with the
   function write in lab

%% ============ Testing Step =======================
test_label = testIris * classifier(1:end-1) + classifier(
   end)
%Determine the class label by looking at the most closed
   label
for i = 1:50
if test_label(i) > 1.5
predictedL(i,1) = 2;
else
predictedL(i,1) =1;
end
end
% Testing how manny prediction were wrong
error = sum(predictedL ~= testLabel)
```