Laborator 2 - RNSF

Extragerea trăsăturilor dintr-un semnal și clasificarea datelor liniar separabile

1. Introducere

Inteligența artificială reprezintă domeniul ce se ocupă cu dezvoltarea sistemelor computaționale capabile să îndeplinească sarcini ce necesită luarea unor decizii, în general în situații relativ noi, precum recunoașterea vizuală a unor obiecte/fețe/semne de circulație, recunoașterea vorbirii, determinarea funcționării normale/anormale a unui motor, detecția crizelor de epilepsie, detecția aritmiilor etc.

Inteligența artificială are multe subramuri, precum recunoașterea formelor, rețele neuronale, sisteme fuzzy, machine learning, însă aproape în toate cazurile scopul principal îl constituie dezvoltarea algoritmilor și metodelor ce stabilesc o anumită asociere între valorile unui spațiu de intrare și anumite valori ale spațiului de ieșire.

Există trei categorii de probleme:

a) Probleme de clasificare: acest tip de probleme presupun clasificarea valorilor de intrare în categorii sau clase de apartenență, scopul final fiind atribuirea unei etichete fiecărui vector de intrare.
b) Probleme de regresie (care includ și problemele de estimare/aproximare și predicție): problemele de regresie asociază o valoare reală pentru fiecare valoare sau set de valori de intrare
c) Probleme de analiză sintactică: abordarea sintactică se ocupă cu descrierea formelor de intrare prin descompunerea lor în forme primare/elementare și determinarea relațiilor (adiacență, subordonare) care definesc modul de alcătuire a sa. Pentru aceasta se folosesc structuiri matematice de tipul arbore sau graf.

2. Structura generală a unui sistemul de recunoaștere a formelor

Sistemele de recunoaștere a formelor au, în general, o structură ce urmează etapele prezentate în Fig. 1. Datele de intrare sunt preprocesate în vederea îmbunătățirii calității acestora prin operații de filtrare, normalizare sau eliminarea zgomotului. După ce datele de intrare au fost preprocesate, sistemul urmează o etapă de extragere de trăsături. În această etapă semnalul este procesat astfel încât caracteristicile extrase în final să fie cât mai reprezentaive pentru datele de intrare, pentru a se obține un rezultat cât mai bun în etapa de clasificare.

Fig. 1 – Structura generală a unui sistem de inteligență artificială

2.1. Etapa de preprocesare

În etapa de preprocesare se urmărește procesarea semnalului pentru îmbunătățirea calității acestuia. În general, această etapă conține filtre de eliminare a zgomotelor, normalizarea datelor, etc.

2.2. Extragere de trăsături

În etapa de extragere de trăsături se urmărește extragerea celor mai relevante informații din semnal în vederea caracterizării acestuia, facilitând învățarea datelor și generalizarea acestora. Astfel, se creează un vector de trăsături ce conține cele mai relevante informații din datele achiziționate, având și o dimensiune redusă.

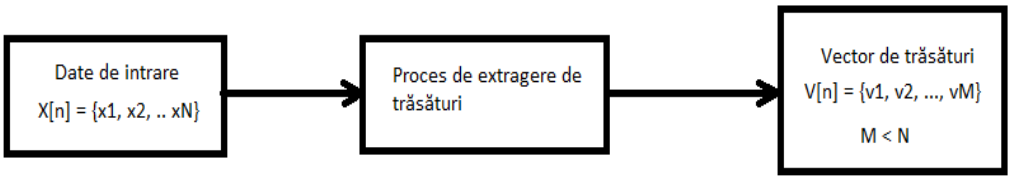

Fig. 2 – Extragere de trăsături

Exemplu:

Se dorește diferențierea vocalelor „a” și „e” analizând semnale achiziționate în timpul mai multor rostiri.

O metodă de extragere de trăsături din semnalul vocal este reprezentată de calcularea puterii pe benzi de frecvență.

Vocalele pot fi diferențiate analizând frecvențele acestora. În tipul rostirii, tractul vocal își modfică poziția pentru a articula sunetele care se doresc a fi rostite, rezultând sunete ce prezintă frecvențe diferite. În Fig. 3 sunt reprezentate spectrele obținute în urma aplicării transformatei Fourier pentru o rostire a fiecărei vocale.

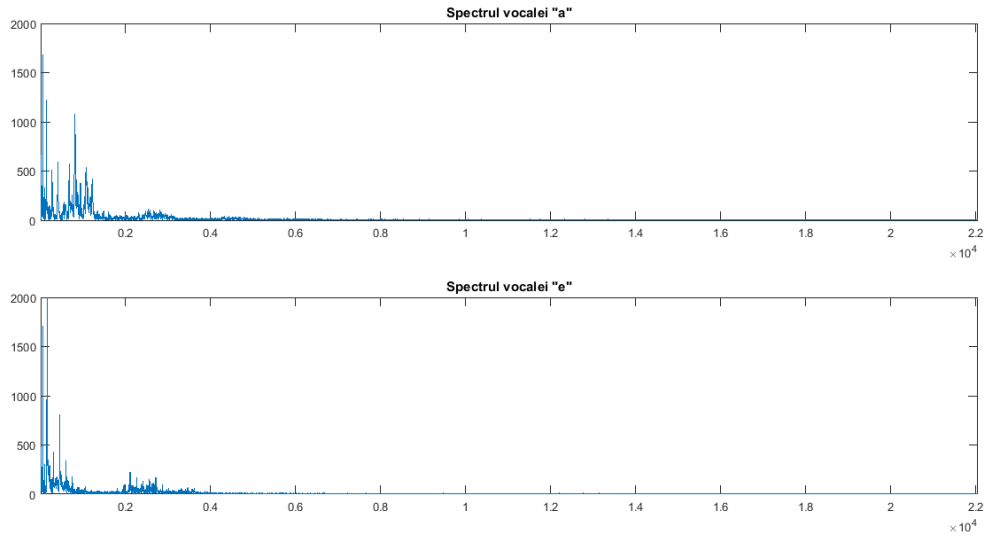

Fig. 3 – Spectrele celor două vocale (sus – vocala a, jos – vocala e)

În această lucrare au fost analizate benzile de frecvență [0 500] Hz și [500 1000] Hz. Se poate observa în Fig. 4 că puterea este distribuită diferit pe cele două benzi de frecvență pentru cele două vocale.

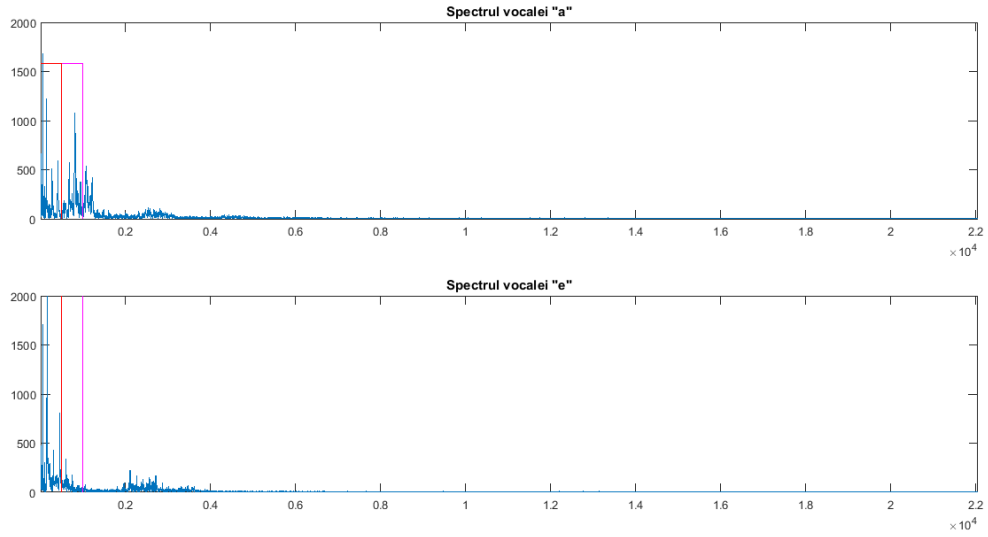

Fig. 4 – Spectrele unei rostiri pentru fiecare vocală împreună cu benzile de frecvență analizate

2.3. Clasificarea

Perceptronul simplu este cea mai simplă formă a rețelelor neuronale utilizat pentru clasificarea pattern-urilor linear separabile. Acesta conține un singur neuron cu sinapse ce pot fi ajustate prin modificarea ponderilor și a bias-ului. Prin acest algoritm, s-a demonstrat că dacă două clase sunt liniar separabile, atunci perceptronul converge către o suprafață de decizie de forma unui hiperplan între cele două clase. Deoarece perceptronul are un singur neuron, acesta este limitat la clasificarea vectorilor în doar două clase.

Arhitectura Perceptronului:

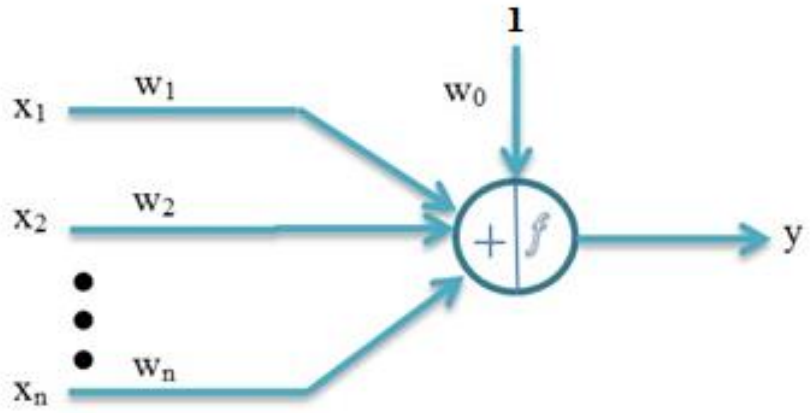

Fig. 6 – Arhitectura perceptronului simplu

În Fig. 6 este reprezentată arhitectura perceptronului simplu, unde $\boldsymbol{X} = (x_1, x_2, \dots, x_n)$, $x_i \in \mathbb{R}$, reprezintă intrările în neuron, iar $\boldsymbol{W} = (w_0, w_1, \dots, w_n)$, $w_i \in \mathbb{R}$, reprezintă ponderile neuronului.

Ieșirea neuronului reprezentată în Fig. 6 cu litera y, este dată de ecuația:

$$y = f\left(w_0 + \sum w_i x_i\right) \tag{3}$$

Unde f(x) reprezintă funcția de activare a neuronului și este de forma:

$$f(k) = \begin{cases} 0, dacă\ \left(w_0 + \sum w_i x_i\right) < 0 \\ 1, dacă\ \left(w_0 + \sum w_i x_i\right) > 0 \end{cases} \tag{4}$$

Algoritmul Perceptronului

1. Inițializarea aleatoare a ponderilor la momentul t=0 și alegerea ratei de învățare η.
2. Evaluarea ieșirii reale $y_j = f(W^T X_j)$, unde j=1:N, N – numărul de vectori de intrare
3. Compararea ieșirii reale y_j cu ieșirea dorită d_j

   $$e_j(t) = d_j - y_j(t) \tag{5}$$

   În acest caz, există 3 valori posibile pentru eroarea de la ieșire: $\{0,1,-1\}$
4. Ajustarea ponderilor cu o valoare care micșorează eroarea:

   $$\Delta w_i(t) = \eta e_j(t) x_{ij} \tag{6}$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \quad (7)$$

Unde η reprezintă rata de învățare.

5. Revenire la pasul 2 până când toate corespondentele $\{(X_j, d_j)\}$ sunt corecte

3. Desfășurarea lucrării

Se dorește diferențierea vocalelor „a” și „e” analizând semnale achiziționate în timpul mai multor rostiri și clasificarea acestora cu ajutorul Perceptronului.

Exercitiu 3.1: Să se calculeze puterea în benzile [0 500] Hz și [500 1000] Hz pentru toate semnalele vocale *„a”* prezente în folderul RNSF, respectiv în folderul *a*.

Rezolvare:
O metodă de extragere a trăsăturilor din semnalul vocal este reprezentată de calcularea puterii pe benzi de frecvență. Rostind cele două vocale, se poate observa că acestea prezintă componente spectrale semnificative în benzi de frecvență diferite. Astfel, calculând energia semnalului pe benzile de frecvență [0 500] Hz și [500 1000] Hz, este suficient pentru diferențierea vocalelor *a* și *e*.

Pasul 1: Se deschide un script nou utilizând comanda *File →New.* Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea Exercitiu03_1.

Pasul 2: Se vor stabili parametrii utilizați. În realizarea scopului propus, vor fi inițial stabiliți parametrii cunoscuți, necesari calculării puterii pe benzi de frecvență: frecvența de eșantionare a semnalelor, *Fe*, care este 44100 Hz, prima banda de frecvență, *B1*, care este cuprinsă între 0 și 500 Hz, a doua bandă de frecvență, *B2*, care este cuprinsă între 500 și 1000 Hz și numărul total de înregistrări, *nr_inregistrari*, care pentru fiecare vocală este de 40.

```
Fe = 44100;
B1 = [0 500];
B2 = [500 1000];
nr_inregistrari = 40;
```

Pasul 3: Inițializarea cu zero a matricei *putere_a*, matrice ce va conține valorile finale după îndeplinirea scopului propus. Matricea *putere_a* este o matrice de dimensiune 2x40 (2 linii și 40 de coloane), linia 1 reprezentând puterea obținută în banda *B1*, iar linia 2 puterea obținută în banda *B2*, în timp ce pe fiecare coloana se va salva rezultatul puterilor (în banda *B1* și banda *B2*) pentru fiecare dintre cele 40 de înregistrări ale vocalei *„a”*. Inițializarea se va realiza utilizând funcția ***zeros***.

```
putere_a = zeros(2, nr_inregistrari);
```

Pasul 4: Se va calcula puterea pe cele două benzi de frecvență, *B1* și *B2*, pentru toate înregistrările vocalei „a”. Pentru realizarea acestui lucru, avem nevoie de o buclă ***for***, cu ajutorul căreia să încărcăm cele 40 de înregistrări prezente în folderul *„a”* . În continuare, în interiorul buclei se va realiza:

1. Încărcarea primei înregistrări din folderul *„a”*. Se poate observa că înregistrările sunt numerotate de la 1 la 40 (exemplu: „1.wav”, „2.wav”) cu scopul de a facilita încărcarea automată a acestora. Astfel, pentru încărcarea datelor având extensia *wav* se poate utiliza funcția ***audioread*** (sau ***wavread***). Parametrul de intrare al acestor funcții este reprezentat de numele înregistrării (un șir de caractere) sau calea la care se poate găsi fișierul respectiv, împreună cu numele înregistrării separate prin caracterul „/” (exemplu: ‘1.wav’ sau ‘C:/a/1.wav’). **În cazul în care înregistrarea NU se află în folderul curent, ESTE OBLIGATORIU să se specifice calea completă către fișierul respectiv!** În cazul în care înregistrarea se află într-unul din folderele prezente în folderul curent, atunci nu mai este nevoie să se specifice toată calea spre fișier, ci doar folderul în care se poate găsi înregistrarea dorită (exemplu: ‘a/1.wav’).
2. Transformarea în domeniul de frecvență a înregistrării încărcate în Octave. Acest lucru poate fi realizat utilizând funcția ***fft***. Rezultatul va fi salvat în variabila *FX*.

3. Transpunerea limitelor celor două benzi în domeniul discret. Asemănător semnalului, spațiul de frecvență (spectrul) este discretizat, fiecărei componente spectrale corespunzându-i un index . Relația dintre index și frecvența reală a componentei se stabilește cu ajutorul formulei:

$$bin = \frac{f\ [Hz] * lungime\_spectru}{Fe\ [Hz]}$$

Rezultatele transformării vor fi salvate în vectorii *bin1* și respectiv *bin2*. Primul element din vectorul *bin1* va corespunde transformării frecvenței de 0 Hz într-o valoare discreta a spectrului, iar al doilea element va corespunde transformării frecvenței de 500 Hz în valoarea discretă a spectrului corespunzătoare acestei frecvențe. Acelși lucru se întâmplă și în cazul vectorului *bin2*.
Deoarece indecșii unui vector sau matrice trebuie să fie valori întregi, se va folosi funcția ***floor*** pentru aproximarea valorii din domeniul discret cu o valoare mai mică sau egală cu elementul respectiv.

4. Stabilirea indecșilor care vor fi utilizați în calcularea celor două benzi de putere, *B1* și *B2*. La pasul anterior, au fost transformate în domeniul discret limitele celor două benzi de frecvență și anume 0 și 500 Hz pentru banda *B1* și 500 și 1000 Hz pentru banda *B2*. Însă, pentru calcularea puterii, avem nevoie de toate eșantioanele spectrului care se găsesc între cele două limite. Astfel, în vectorii *idx1* și *idx2* se vor păstra indecșii necesari calculării puterii pentru cele două benzi.

5. Calcularea puterii semnalului pentru benzile *B1* și *B2*. Puterea semnalului poate fi calculată utilizând formula:

$$P = \frac{1}{N}\sum_{i=1}^{N} x_i^2$$

Unde:
- N reprezintă numărul de eșantioane
- x_i reprezintă aplitudinea eșantionului i

Puterea obținută în banda *B1* se va salva pe linia i și prima coloană a matricei *putere_a*, iar puterea obținută în banda B2 se va salva pe linia i cea de a doua coloană a matricei *putere_a*.

```
for i = 1:nr_inregistrari

%    Pasul 4.1
  x = audioread(sprintf('a/%d.wav', i));

%   Pasul 4.2
  FX = fft(x);

%   Pasul 4.3
  bin1 = [1 1] + floor(B1 * size(x,1) / Fe);
  bin2 = [1 1] + floor(B2 * size(x,1) / Fe);

%   Pasul 4.4
  idx1 = bin1(1):bin1(2);
  idx2 = bin2(1):bin2(2);

%   Pasul 4.5
  putere_a(1, i) = sum(abs(FX(idx1)).^2) / length(idx1);
  putere_a(2, i) = sum(abs(FX(idx2)).^2) / length(idx2);
end
```

Exercitiu 3.2: Să se calculeze puterea în benzile [0 500] Hz și [500 1000] Hz pentru toate semnalele vocale „e” prezente în folderul RNSF, respectiv în folderul e, asemănător exercițiului *Exercitiu 3.1* și să se salveze rezultatele în matricea putere_e.

Exercitiu 3.3: Să se reprezinte grafic trasaturile extrase la exercițiile *Exercitiu 3.1* și *Exercitiu 3.2,* ficare vocală reprezentându-se cu o culoare diferită.

Rezolvare:
Pasul 1: Acest exercițiu va fi realizat în continuarea scriptului Exercitiu03_1 salvat la Pasul 1 al exercițiului *Exercitiu 3.1*.

Pasul 2: Se va reprezenta grafic puterea calculată în banda *B2* în funcție de puterea calculată în banda *B1* la exercițiul *Exercitiu 3.1* cu marker de tip „*” și culoarea roșie pentru înregistrările în care a fost rostită vocala *„a”*.

```
plot(putere_a(1,:), putere_a(2,:), 'r*');
```

Pasul 3: Se va reține graficul cu ajutorul funcției ***hold*** pentru plotarea pe același grafic și a trăsăturilor obținute pentru vocala *e*.

```
hold on;
```

Pasul 3: Se va reprezenta pe același grafic obținut la Pasul 2 puterea calculată în banda *B2* în funcție de puterea calculată în banda *B1* la exercițiul *Exercitiu 3.2* cu marker de tip „*” și culoarea albastră pentru înregistrările în care a fost rostită vocala *„e”*.

```
plot(putere_e(1,:), putere_e(2,:), 'b*');
```

Pasul 4: Se va introduce și o legendă pentru recunoașterea celor două vocale.

```
legend('a', 'e');
```

Pasul 5: Se va seta parametrul funcției ***hold*** la *off*.

```
hold off;
```

Exercitiu 3.4: Să se creeze un set de date complet, cu denumirea *X*, care să conțină toate trăsăturile extrase pentru cele două vocale, *a* și *e* și să se creeze și un vector *D*, care să rețină etichetele trăsăturilor, astfel încât analizând vectorul *D* să se poată recunoaște din ce categorie face parte (vocala *a* sau vocala *e*) elementul respectiv din *X*.

Rezolvare:
Pasul 1: Acest exercițiu va fi realizat în continuarea scriptului Exercitiu03_1 salvat la Pasul 1 al exercițiului *Exercitiu 3.1*.

Pasul 2: Se vor concatena pe linie cele două matrice: *putere_a* și *putere_e* într-o singură matrice cu denumirea *X*, care va avea în final dimensiunea 80x2. Deoarece *putere_a* și *putere_e* sunt matrice de dimensiune 2x40, pentru a crea matricea *X* de dimensiune 80x2, se vor transpune *putere_a* și *putere_e* înainte sau în timpul concatenării.

```
X = [putere_a'; putere_e'];
```

Pasul 3: Se va crea vectorul *D* care conține etichetele necesare pentru diferențierea puterii a de puterea e din matricea *X*. Pentru realizarea acestui lucru, primele 40 de elemente din vectorul *D* vor conține doar valori de 0 (în consecință, puterile extrase din înregistrările care conțin vocala *„a”* vor avea eticheta 0), iar următoarele 40 de elemente (sau elementele de la 41 la 80) vor avea eticheta 1 (în consecință a fost dată eticheta 1 tuturor puterilor extrase pentru vocala *„e”*).

```
D = [zeros(nr_inregistrari,1); ones(nr_inregistrari,1)];
```

Pasul 4: Se vor salva cele două variabile, *X* și *D*, în folderul curent, folderul cu denumirea Grupa [Nr. Grupei], utilizând funcția ***save***. Primul parametru de intrare trebuie să fie un șir de caractere și corespunde numelui cu care se dorește să se salvaeze variabila respectivă, iar cel de-al doilea parametru este tot un șir de caractere și corespunde variabilei prezente în Workspace care se dorește să fie salvată.

```
save('X.mat', 'X')
save('D.mat', 'D')
```

(5p) Exercitiu punctat! Exercitiu 3.5: Să se încarce pe Moodle la exercițiul cu numele **GR[Nr.Grupa]_RNSF_LAB02_ex1** (unde Nr.Grupa reprezinta grupa de apartenență a fiecărui student) cele două variabile rezultate în urma rulării exercițiului *Exercitiu 3.4* (matricea X și vectorul D) și să se afișeze suma trăsăturilor primului vector din setul de date ***X*** aproximat la 4 zecimale* și eticheta acestuia.

Date de intrare: Setul de date (X, D) calculat la exercițiul 3.4

Date de ieșire: Suma primului vector de trăsături al setului de date X (aproximat la 4 zecimale), separat prin spațiu de eticheta vectorului respectiv.**

Exemplu de afișare:
25483.7453 0

*pentru afișarea rezultatului aproximat la 4 zecimale se poate folosi următoarea linie de cod:
printf(“%.4f”, valoare)
unde valoare corespunde valorii care se dorește a fi aproximată

**NU SE VA AFIȘA ALTCEVA ÎN AFARĂ DE CEEA CE SE SPECIFICĂ LA DATE DE IEȘIRE. În caz contrar, rezultatul nu se punctează.

(5p) Exercitiu punctat! Exercitiu 3.6: Să se încarce pe Moodle la exercițiul cu numele **GR[Nr.Grupa]_RNSF_LAB02_ex2** (unde Nr.Grupa reprezinta grupa de apartenență a fiecărui student) cele două variabile rezultate în urma rulării exercițiului *Exercitiu 3.4* (matricea X și vectorul D), să se calculeze ieșirea dintr-un Perceptron simplu pentru primul vector de intrare cunoscând vectorul de ponderi:

W = [-0.03 118.56 -203.08]

Pe următoarea linie se va afișa și mesajul "Clasificat corect!" sau "Clasificat gresit!" dacă vectorul a fost clasificat corect, respectiv greșit utilizând ponderile date*.

Date de intrare: Setul de date (X, D) calculat la exercițiul 3.4

Date de ieșire: Pe prima linie ieșirea dintr-un Perceptron simplu utilizând ponderile W. Pe a doua linie mesajul: "Clasificat corect!" sau "Clasificat greșit!"

Exemplu de afișare:
1
Clasificat gresit!

Pentru obținerea ieșirii dintr-o rețea de tipul Perceptron se folosește următoarea ecuație (conform teoriei din Secțiunea 2.3):

$$y_i = f\left(w_0 + \sum w_j x_{ij}\right)$$

Unde:

- w_0 reprezintă biasul, în cazul de față este echivalent cu W(1) – primul element din vectorul de ponderi
- w_i reprezintă ponderea i, în cazul de față ponderile W(2:end) – restul elementelor din vectorul de ponderi
- x_{ji} reprezintă vectorul de pe linia j a matricei X, având trăsătura de pe coloana i (exemplu: x_{11}= X(1,1) adică puterea calculată în banda 1 pentru prima înregistrare, de pe linia 1 a matricei X)
- y_j reprezintă ieșirea reală din rețea, adică vectorul j (linia j din matricea X care corespunde unei inregistrări din cele două clase: a sau e) este trecut prin rețea și i se acordă o etichetă (0 sau 1) care poate fi egală sau poate fi diferită de eticheta dorită, sau stabilită din vectorul D.
- f reprezintă funcția de activare și reprezintă funcția treaptă $f(k) = \begin{cases} 0, dacă\ k < 0 \\ 1, dacă\ k > 0 \end{cases}$

Pentru a determina dacă ieșirea din rețea este corectă se va verifica dacă ieșirea obținută (y) este egală cu cea dorită, din vectorul de etichete D.

*Mesajul afișat trebuie să fie exact ca cel specificat. În caz contrar, rezultatul nu se punctează.

!Codul va fi verificat. În cazul în care se observă probleme în cod sau hardcodare punctajul va fi anulat.

Exercitiu 3.7: Să se antreneze o rețea de tipul Perceptron care să clasifice cele două clase: vocala a și vocala e, utilizând trăsăturile extrase la exercițiile Exercitiu 3.1 și Exercitiu 3.2 și salvate în matricea X. Se va lucra doar cu matricea X și vectorul D.

Ponderile inițiale vor fi:
W = [0.5 -1 0.75]

Rata de învățare: 0.0001

Rezolvare:
Pasul 1: Se deschide un script nou utilizând comanda *File →New.* Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea Exercitiu03_5.

Pasul 2: Se va elibera toată memoria din Octave utilizând funcția ***clear*** și se va elibera fereastra de comandă utilizând instrucțiunea ***clc***.

```
clear all, clc
```

Pasul 3: Se vor încărca variabilele salvate la *Exercitiu 3.4* la Pasul 4 utilizând funcția ***load***.

```
load('X.mat')
load('D.mat')
```

Pasul 4: Se vor stabili parametrii necesari îndeplinirii obiectivului. În vederea rezolvării acestui exercițiu va trebui să stabilim numărul de exemple (observații) din setul de date *X* utilizat pentru antrenare, cu denumirea *nr_vectori*, și numărul trăsăturilor extrase la exercițiile anterioare, cu denumirea *nr_trasaturi*. Numărul de vectori din setul de date corespunde numărul tuturor înregistrărilor (atât înregistrările vocalei a cât și înregistrările vocalei e), adică 80, iar numărul de trăsături extrase este 2, corespunzător celor două benzi: *B1* și *B2*. Acești parametrii pot fi stabiliți utilizând funcția size pentru setul de date *X*. În continuare, vom mai utiliza variabila *eta*, care corespunde ratei de învățare a algoritmului și va avea valoarea de 0.01.

```
[nr_vectori, nr_trasaturi] = size(X);
eta = 0.0001;
```

Pasul 5: În continuare, se va începe dezvoltarea algoritmului de învățare a rețelei Perceptron. Conform algoritmului prezentat în Secțiunea 2.3, primul pas al algoritmului îl presupune inițializarea aleatoare a ponderilor. Pentru a realiza un număr mai mic de pași în vederea găsirii suprafeței de separație, în acest caz ponderile, stocate în vectorul *W*, vor fi inițializate cu valorile: W = [0.5 -1 0.75]. Primul element din vectorul *W* corespunde biasul-ui, *w0,* al doilea element corespunde ponderii w1, iar cel de-al treilea element corespunde ponderii *w2.* Și se inițiază *E*, eroarea, cu 1.

```
W = [0.5 -1 0.75];
E = 1;
```

Pasul 6: Se va introduce o buclă ***while*** care are condiția de oprire ca *E* (eroarea) să fie diferită de 0. În momentul în care eroarea devine 0, adică toți vectorii au fost clasificați corect, se va ieși din bucla ***while***.

Pentru a înțelege mai bine cum se obține suprafața de separație în raport cu ponderile, se recomandă afișarea la fiecare iterație a algoritmului vectorii în spațiul trăsăturilor (trasatura 1 în funcție de

trăsătura 2 sau invers) împreună cu suprafața de separație. Pentru a afișa suprafața de separație se poate utiliza funcția ***plotpc(),*** funcție care se găsește în folderul RNSF_LAB2.
Un exemplu de afișare a datelor de intrare împreună cu suprafața de separație este:

```
figure, plot(X(find(D==0),1),X(find(D==0),2),'r*')
hold on, plot(X(find(D==1),1),X(find(D==1),2),'b*')
plotpc(W(2:3),W(1))
```

Explicație linii de cod:

Funcția ***find*** are rolul de a găsi într-un vector sau o matrice valoarea dorită și de a returna indexul la care a fost găsită valoarea respectivă. În acest caz, cunoaștem faptul că fiecare index din vectorul *D* (sau poziția din vectorul *D*) care conține valoarea 0 corespunde unei linii din *X* (care în esență este tot un index) în care au fost salvate puterile în benzile *B1* și *B2* pentru vocala a. Mai exact, pe fiecare poziție din vectorul *D* unde se află un element cu valoarea 0, găsim pe linia matricei *X* puterile extrase pentru o *vocală a* (exemplu: Dacă ne uităm în vectorul *D* pe poziția 25, adică D(25), putem observa că valoarea de la acea poziție este 0, adica D(25) = 0. Deci știm că în matricea *X*, pe poziția 25, vom găsi puteri extrase dintr-o înregistrare în care a fost rostită *vocala a*. Adica, X(25,1) reprezintă puterea calculată în banda 1 pentru o înregistrare din clasa *a*, iar X(25,2) reprezintă puterea calculată în banda 2 pentru o înregistrare din clasa *a*). Astfel că, instrucțiunea find(D==0) va returna indecșii (sau valorile pozițiilor) unde *D* prezintă valoarea 0. De asemenea, știm că pozițile din *D* unde se găsește valoarea 0 corespunde linilor din *X* unde se găsește *clasa a*. În concluzie, instrucțiunea X(find(D==0),:) reprezintă extragerea din *X* a tuturor vectorilor care fac parte din *clasa a*. Acest lucru se poate extinde și pentru *clasa b*, unde eticheta este 1.
Ne dorim afișarea celor două clase pentru a putea vizualiza suprafața de separație în timp ce aceasta se modifică odată cu updatarea ponderilor.

Funcția ***plotpc()***, primește ca parametrii de intrare: ponderile pentru cele două trasături (primul parametru de intrare) și biasul (al doilea parametru de intrare).

În interorul buclei while se va calcula:

5.1. Ieșirea din rețea pentru toți vectorii de intrare utilizând ecuația (conform teoriei din Secțiunea 2.3):

$$y_i = f\left(w_0 + \sum w_j x_{ij}\right)$$

Unde:

- w_0 reprezintă biasul, în cazul de față este echivalent cu W(1) – primul element din vectorul de ponderi
- w_i reprezintă ponderea i, în cazul de față ponderile W(2:end) – restul elementelor din vectorul de ponderi
- x_{ji} reprezintă vectorul de pe linia j a matricei X, având trăsătura de pe coloana i (exemplu: x_{11}= X(1,1) adică puterea calculată în banda 1 pentru prima înregistrare, de pe linia 1 a matricei X)
- y_j reprezintă ieșirea reală din rețea, adică vectorul j (linia j din matricea X care corespunde unei inregistrări din cele două clase: a sau e) este trecut prin rețea și i se acordă o etichetă (0 sau 1) care poate fi egală sau poate fi diferită de eticheta dorită, sau stabilită din vectorul D.
- f reprezintă funcția de activare și reprezintă funcția treaptă $f(k) = \begin{cases} 0, dacă\ k < 0 \\ 1, dacă\ k > 0 \end{cases}$

5.2. Determinarea erorii pentru vectorul i. În acest caz, se va calcula diferența dintre ieșirea dorită (*D(i)*) și ieșirea reală din rețea (*y(i)*)

5.3. Modificarea ponderilor în funcție de eroarea obținută. Modificarea ponderilor se face conform ecuației:

$$w_j(t+1) = w_j(t) + \eta e_i(t) x_{ij}$$

Unde:

- w_j reprezintă ponderea j din vectorul de ponderi W
- η reprezintă rata de învățare, eta, stabilită ca fiind 0.01
- e_i reprezintă eroarea obținută pentru vectorul de trăsături *i* (obținută la pasul 5.2)
- x_{ij} reprezintă valoarea de pe linia i și coloana j a matricei X

5.4. Calcularea erorii globale, *E*. Se va testa dacă în urma modificării ponderilor există vectori clasificați greșit. Acest lucru se poate realiza prin însumarea valorilor absolute a diferenței dintre *D* și *y*. În cazul în care *E* este 0, adică vectorul *D* este egal cu vectorul *y*, înseamnă că toți vectorii au fost clasificați correct, iar programul va ieși din bucla ***while***. Altfel, orice valoare diferită de 0 îndeplinește condiția buclei while și în concluzie, se va repeta procesul începând de la *Pasul 5.1*.

Un exemplu de pași de urmat pentru implementare:

```
      while E~=0

    %  Calcularea ieșirii din rețea pentru fiecare vector din setul de date


    %  Calcularea erorii pentru fiecare vector din setul de date


    %  Modificarea ponderilor pe baza erorii calculate


    %  Afișarea vectorilor de intrare împreună cu suprafața de separație
dată de ponderile W

      figure, plot(X(find(D==0),1),X(find(D==0),2),'*')
      hold on, plot(X(find(D==1),1),X(find(D==1),2),'r*')
      plotpc(W(2:3),W(1))

      pause(1) % oprirea apariției graficului pt 1s (pt vizualizare)
      hold off % închiderea graficului
      close all

    %    Calcularea erorii globale
      E = sum(abs(D-y));

      end
```