Laborator 5 - RNSF
Rețeaua neuronala cu Bază Radială (RBF)

1. Introducere

Reteaua neuronala cu functii de baza radiale este bazata pe aplicarea unei transformari neliniare asupra spatiului de intrare cu scopul ca în noul spatiu datele sa fie (aproape) liniar separabile. Neuronii din stratul ascuns efectueaza aceasta transformare asupra spatiului de intrare, iar neuronii de iesire lucreaza pe datele obtinute în urma transformarii neliniare. Fiecarui neuron din stratul ascuns îi corespunde un centroid, care actioneaza ca un prototip pentru vectorii din setul de date. Fiecare neuron din stratul ascuns semnalizeaza nivelul de similaritate al vectorului de intrare cu centroidul asociat acelui neuron.

Arhitectura rețelei RBF

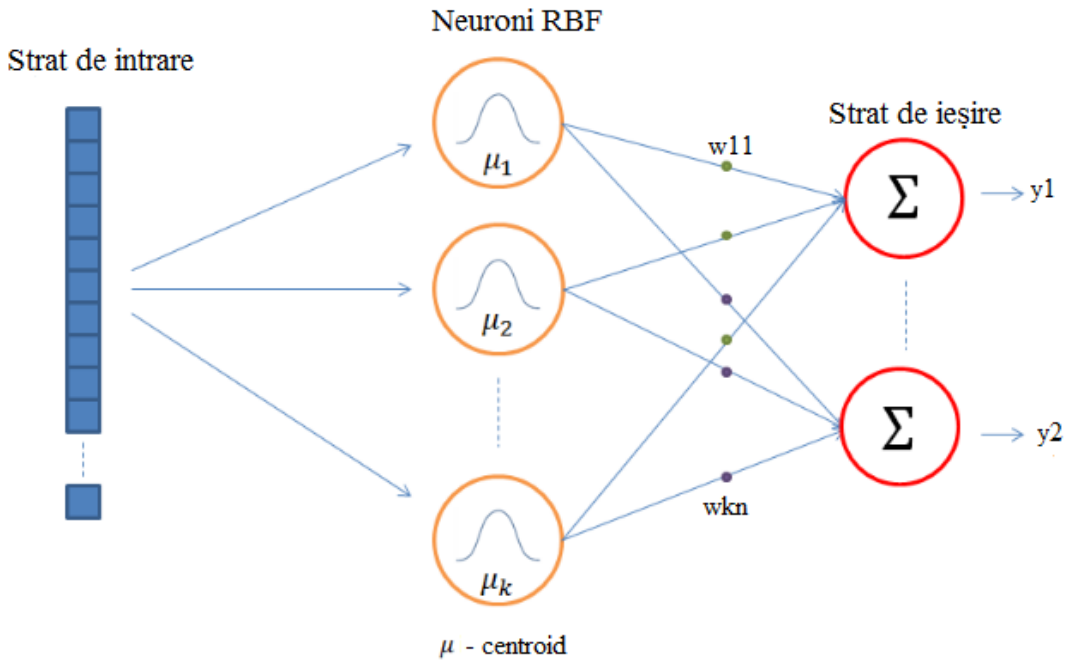

Fig. 1 – Arhitectura rețelei RBF

În Fig. 1 este reprezentată arhitectura tipică a unei rețele neuronale de tip RBF. Aceasta constă dintr-un strat de intrare, un strat de neuroni cu funcție de activare cu bază radială și un strat de ieșire cu un neuron pentru fiecare clasă a setului de date.

Stratul de intrare

Stratul de intrare este reprezentat de un vector n-dimensional care se dorește a fi clasificat.

Stratul ascuns

Fiecare neuron din stratul ascuns conține un centroid, μ_k. Vectorul de intrare este comparat cu fiecare centroid din stratul ascuns, rezultând valori de ieșire cuprinse între 0 și 1 reprezentând similaritatea dintre vectorul de intrare și neuronul respective. Dacă intrarea este egală cu centroidul, atunci ieșirea din neuronal RBF va fi 1. Cu cât crește distanța dintre intrare și centroid, răspunsul neuronului scade către 0.

Stratul de ieșire

Ieșirea din rețeaua neuronală constă dintr-un set de neuroni, unul pentru fiecare clasă corespunzătoare setului de date. Fiecare neuroncalculează un scor pentru categoria asociată. În general, decizia de clasificare este luată conform scorului maxim.
Scorul este calculat prin însumarea ponderată a valorilor de activare obținute în urma trecerii vectorului de intrare prin fiecare neuron RBF.

Funcția de activare de tip RBF

Fiecare neuron cu bază radială calculează o măsură a similiarității dintre intrare și centroid. Vectorii de intrare care sunt similari cu prototipul returnează o valoare apropiată de 1.
Un exemplu de funcție cu bază radială este reprezentată în Fig. 2, având media=5 și varianța=1.

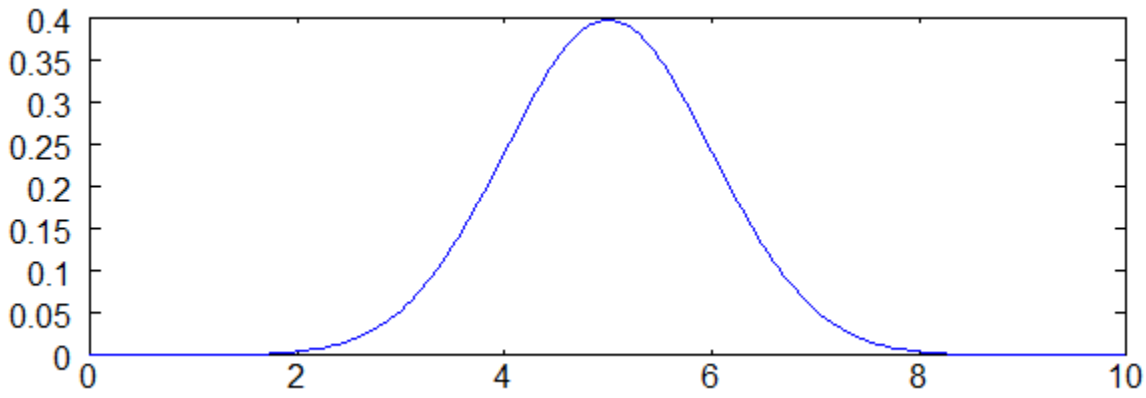

Fig. 2 – Funcție cu bază radială

Funcția de activare în cazul neuronilor cu bază radială este este, în general, scrisă astfel:

$$\varphi(x) = e^{-\beta\|x-\mu\|^2} \quad (1.2)$$

În cazul funcției de activare a neuronilor, μ se referă la centroid.
Parametrul, β controlează lărgimea funcției prezentată în Fig. 2.

2. Algoritmul RBF

Există mai multe variante de antrenare a rețelei neuronale de tip RBF:

1. Antrenarea simultană a tuturor parametrilor (similară algoritmului BackPropagation – doar regulile de ajustare ale centrilor se modifică)
2. Antrenare separată a parametrilor: centri, lărgimi, ponderi.

Antrenare separată a parametrilor

În cazul antrenării separate a parametrilor, se utilizează algoritmul KMeans (Basic Isodata) pentru determinarea centroizilor, după care se calculează varianța pentru fiecare centroid, urmând să se calculeze ponderile pentru stratul de ieșire.

I. Algoritmul KMeans

Algorimul K-Means este o metodă automată de clusterizare a datelor similare. Cu alte cuvinte, se dă un set de date:

$$\{x^{(1)}, .., x^{(m)}\}, \quad (2.1)$$

Unde $x^{(i)}$este n dimensional.
Și se dorește gruparea datelor într-un număr k de clustere.

Algoritmul K-Means este o procedură iterativă care începe prin alegerea aleatoare a unor centroizi, urmând ca aceștia să fie adaptați prin atribuirea vectorilor către cel mai apropiat centroid și recalcularea centroizilor în funcție de vectorii care le-au fost atribuiți.
Pașii parcurși în realizarea algoritmului sunt:

1. Inițializarea centroizilor
2. Repetă pentru fiecare vector $x^{(i)}$din set:
   a) Se calculează distanțele sale la mediile tuturor claselor

$$d_k\left(x^{(i)}, \mu_k\right) = \left\| x_j{}^{(i)} - \mu_k \right\|^2 = \sqrt{\sum_{j=1}^{m} \left(x_j - \mu_k\right)^2} \tag{2.2}$$

   b) Se stabilește clasa de apartenență (ce va fi utilizată la iterația viitoare) a vectorului $x^{(i)}$
   c) Reactualizarea mediilor
3. Dacă (față de iterația anterioară) nici un vector nu a fost mutat în altă clasă (sau media unei clase nu s-a modificat, algoritmul se încheie. Altfel, se reia algoritmul de la pasul 2.

II. Calcularea parametrului β

Odată cunoscuți centroizii și clasele de apartenență a vectorilor de intrare pentru fiecare centroid, se poate determina parametrul β astfel:

$$\beta = \frac{1}{2\sigma^2} \tag{2.3}$$

Unde σ^2 reprezintă varianța și poate fi calculat utilizând următoarea ecuație:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{2.4}$$

Estimarea câmpurilor receptive mai poate fi calculată și în alte moduri:

$$\sigma = \frac{d_{max}}{\sqrt{K}}, unde\ d_{max} = distanța\ maximă\ dintre\ centrii$$

III. Calcularea ponderilor pentru stratul de ieșire

În urma aplicării funcției de activare cu bază radială pentru vectorii de intrare, utilizând ecuația 1.2, se obține ieșirea din stratul ascuns. Valorile obținute astfel reprezintă intrări pentru stratul de ieșire.
Funcția de activare pentru neuronii din stratul de ieșire poate fi o funcție liniară de forma:

$$f(x) = x \tag{2.5}$$

3. Desfășurarea lucrării

Exercitiu 3.1: Să se creeze functia ***plotDataset*** cu ajutorul căreia să poată fi reprezentate grafic trăsăturile din setul de date prezent în fișierul *lab05_data01.mat* din folderul *RNSF_Lab5*. Funcția va primi ca parametrii de intrare setul de date *X* și clasele de apartenență prezente în vectorul *D*, pentru reprezentarea fiecărei clase cu o culoare.

Exercitiu 3.2: Să se antreneze o rețea neuronală de tipul RBF (Radial Basis Function) cu ajutorul funcțiilor ***rbf*** și ***rbftraindemo*** prezente în folder-ul RNSF_Lab5 →RBFToolbox, care să clasifice setul de date *lab05_data01.mat*. Setul de date conține setul de antrenare *X*, care prezintă pe fiecare linie un vector de intrare, iar pe fiecare coloană este stocată o trăsătură și vectorul *D*, care conține clasele de apartenență pentru fiecare vector de intrare din *X*. Se vor folosi **6 neuroni** în stratul ascuns.

Se va utiliza funcția ***rbftraindemo*** și nu ***rbftrain***, deoarece funcția ***rbftraindemo*** reprezintă grafic antrenarea rețelei utilizând algoritmul Kmeans. Mai exact, această funcție permite vizualizarea modificării centroizilor la fiecare pas al algoritmului și de asemenea, mai pot fi observați si vectorii care aparțin fiecărui centroid. Funcția ***rbftraindemo*** poate fi utilizată doar dacă setul de date prezintă două trăsături, iar numărul de centroizi din stratul ascuns nu depășește valoarea 6!!

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_2.
Pasul 2: Se va încărca setul de date, *lab05_data01.mat* utilizând funcția ***load*** si se va introduce toolboxul in path-ul de cautare.

```
clear all, close all, clc

load('lab05_data01.mat');
addpath('path\netToolbox');
```

Pasul 3: Se vor stabili parametrii cunoscuți, necesari rezolvării problemei: numărul vectorilor de intrare, *nvect*, și numărul neuronilor de intrare, *nin*, din rețea pot fi determinate din setul de date *X* utilizând funcția ***size***, numărul de neuroni din stratul ascuns, *nhidden*, va fi setat la 6, numărul neuronilor din stratul de ieșire, *nout*, poate fi extras din vectorul *D*, utilizând funcțiile ***length*** și ***unique*** (funcția ***unique*** returnează toate valorile distincte care se găsesc într-un vector sau matrice), iar funcția de activare a neuronilor din stratul de ieșire, *outfunc*, va fi setată ca funcție liniară și metoda de antrenare, *method*, va fi reprezentată de algoritmul de clustering kmeans.

```
nvect = size(X,1);
nin = size(X,2);
nhidden = 6;
nout = length(unique(D));
outfn = 'linear';
actfn = 'gaussian';
method = 'kmeans';
```

Pasul 4: Normalizarea datelor. Setul de date X va fi normalizat astfel încât să conțină valori între 0 și 1, utilizând următoarea ecuație:

$$X_{norm} = \frac{X - x_{min}}{x_{max} - x_{min}}$$

```
minim = min(X);
maxim = max(X);

X = (X-minim)./(maxim-minim);
```

Pasul 5: Vizualizarea datelor de intrare.

```
plotdata_multiple_class(X,D,nout)
```

Pasul 6: Se va transforma vectorul *D* într-o matrice, *Out*, de dimensiune *[nvect x nout]*, astfel încât să se cunoască care dintre neuronii de ieșire va trebui să se activeze pentru a oferi răspunsul corect. Acest lucru poate fi realizat astfel: se presupune că primul neuron din stratul de ieșire răspunde pentru clasa cu eticheta 1, al doilea neuron din stratul de ieșire răspunde pentru clasa cu eticheta 2, etc. Pentru ca un neuron să se activeze, acesta trebuie să fie setat pe 1, iar restul neuronilor să fie setați pe 0.

```
Out = zeros(nvect,nout);

for i=1:nvect
       Out(i,D(i)) = 1;
end
```

Pasul 7: Stabilirea arhitecturii și parametrii necesari rețelei. Se va apela funcția ***rbf***, cu parametrii de intrare: *nin*, *nhidden*, *nout*, *actfn*, *outfn*, *method*. Unde *nin* reprezintă numărul de neuroni din stratul de intrare, *nhidden* reprezintă numărul de neuroni din stratul ascuns, *nout* reprezintă numărul de neuroni din stratul de ieșire, *actfn* reprezintă funcția de activare a neuronilor din stratul ascuns, *outfunc* reprezintă funcția de activare a neuronilor din stratul de ieșire, iar *method* reprezintă metoda de antrenare, care poate fi 'kmeans' sau 'bkp' (în această lucrare antrenarea se va realiza cu ajutorul algoritmului kmeans). Parametrul de ieșire a funcției, denumit *net*, va fi reprezentat de o structură ce conține câmpurile:

- type = `rbf`
- nin = numărul neuronilor de intrare
- nhidden = numărul neuronilor din stratul ascuns
- nout = numărul neuronilor din stratul de ieșire
- actfn = șir de caractere ce descrie funcția de activare a neuronilor din stratul ascuns
- outfn = șir de caractere ce descrie funcția de activare a neuronilor din stratul de ieșire
- method = șir de caractere ce descrie metoda de antrenare
- c = ponderile din stratul ascuns, reprezentate de centroizi, dimensiune *[nin x nhidden]*
- sigma = lărgimea câmpurilor receptive, dimensiune *[1 x 1] sau [1 x nhidden]*, în problema curentă fiind calculată după ecuația de mai jos, având dimensiune *[1 x 1]*:

$$\sigma = \frac{d_{max}}{\sqrt{K}}, unde\ d_{max} = distanța\ maximă\ dintre\ centrii$$

- wo = ponderile neuronilor din stratul de ieșire, împreună cu biasul, dimensiune *[nout x nhidden]*
- bo = biasul neuronilor din stratul de ieșire, dimensiune *[1 x nout]*

Câmpurile se pot apela utilizând operatorul ***punct „.”***, exemplu: *net.type*, *net.nout*. De asemenea, funcția ***rbf*** inițializează aleator ponderile.

```
net = rbf(nin, nhidden, nout, actfn, outfn, method);
```

Pasul 8: Antrenarea rețelei. Se va antrena rețeaua neuronala RBF utilizând funcția ***rbftraindemo***. Parametrii de intrare ai funcției sunt: *net*, structura creată anterior la Pasul 5, *X*, setul de date de intrare, *Out*, matricea valorilor corespunzătoare ieșirii dorite din rețea și numărul maxim de interații ale rețelei, care va fi setat în acest caz 100. Parametrul de ieșire din funcția ***rbftraindemo*** este *net*, structura creată la Pasul 5 având ponderile obținute în urma antrenării și nu ponderile inițializate aleator.

Se va utiliza funcția ***rbftraindemo*** și nu ***rbftrain***, deoarece funcția ***rbftraindemo*** reprezintă grafic antrenarea rețelei utilizând algoritmul kmeans. Mai exact, această funcție permite vizualizarea modificării centroizilor la fiecare pas al algoritmului și de asemenea, mai pot fi observați si vectorii care aparțin fiecărui centroid.

Pentru a obține valori asemănătoare celor din platformă, antrenarea se va face utilizând ponderile prezente în structura „*net_init.mat*”.

```
load('net_init.mat');
net = rbftraindemo(net, X, Out, 50);
```

Pasul 9: Afișarea suprafețelor de separație

```
plotdata_multiple_class(X,D,nout)
plot_boundary_rbf(X, net)
```

Rezultatul obținut (aproximativ): Cost: 2.639991e-02

Pasul 10: Se va salva structura *net* pentru utilizarea acesteia in exercițiul următor, precum și parametrii utilizați pentru normalizare.

```
save('net.mat','net')
save('parametrii_norm.mat','minim', 'maxim')
```

(3p) Exercițiu Punctat! *Exercitiu 3.3:* Să se definească funcția cu denumirea ***rbfFeedForward*** care primește ca parametrii de intrare un set de date **X** și o structura **net** inițializată și oferă ca parametru de ieșire ieșirea din rețea pentru toți vectorii de intrare din setul de date **X**. Funcția de activare a neuronilor va fi *f1** pentru toți neuronii din stratul ascuns și *f2** din stratul de ieșire.

Să se încarce funcția **(doar funcția definită în cerință)** pe Moodle la exercițiul cu numele **GR[Nr.Grupa]_RNSF_LAB05_ex1.** Exercițiul restricționează încărcarea altui fișier decât cel din cerință: un fișier cu extansia **.m** și denumirea ***rbfFeedForward*** care conține declarația funcției:

```
function y = rbfFeedForward(X, net)
% Corp Functie de completat
end
```

*Funcția de activare pentru stratul ascuns (*f1*) este:

$f_1(u) = e^{-\frac{u^2}{2\sigma^2}}$

*Funcția de activare pentru stratul de ieșire (f2) este:

$f_2(u) = u$

OBS: i) Acest fișier (definiția funcției) se poate găsi și în secțiunea Edit pe Moodle.
ii) Setul de date cu care va fi testat este setul "lab05_data01.mat" și structura net salvată anterior la *Exercițiul 3.2*.

Instrucțiuni opționale de 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_3.

Pasul 2: Se încarcă setul de date *lab05_data01.mat* și structura *net.mat* salvată anterior.

Pasul 3: Se calculează ieșirea din rețea. Pentru a obține ieșirea din rețea:

1. Inițial se calculează ieșirea din stratul ascuns pentru fiecare neuron (**ecuația este caracteristică unui singur vector de intrare**):

$$y_i^h = f_1\left(\sqrt{\sum_{j=1}^{nin}\left(x_j - c_{ij}\right)^2}\right)$$

   Unde, i reprezintă neuronul din stratul ascuns: i = $\overline{1, nhidden}$ (nhidden corespunde numărului de neuroni din straul ascuns), j reprezintă trăsătura: j = $\overline{1, nin}$ (nin reprezintă numărul de intrări în rețea, echivalent cu numărul de trăsături), x_j corespunde valorii j din vectorul de trăsături introdus în rețea, c_{ij} corespunde ponderii j pentru neuronul i din stratul ascuns, iar f_1 reprezintă funcția de activare a neuronilor din stratul ascuns. Ecuația funcției de activare specifică problemei pentru stratul ascuns este funcția:

$$f_1(u) = e^{-\frac{u^2}{2\sigma^2}}$$

2. Calcularea ieșirii pentru neuronii din stratul de ieșire (ecuația este caracteristică unui singur vector de intrare):

$$y_i^o = f_2\left(\sum_{j=1}^{nhidden} y_j^h w_{ij}^o + w_{i0}^o\right)$$

   Unde i reprezintă neuronul din stratul de ieșire: i = $\overline{1, nout}$ (nout corespunde numărului de neuroni din straul de ieșire), j reprezintă intrarea în neuronul de ieșire: j = $\overline{1, nhidden}$ (nhidden reprezintă numărul de neuroni din stratul ascuns), y_j^h corespunde valorii j din vectorul obținut în urma caluclării ieșirilor din stratul ascuns, w_{ij}^o corespunde ponderii j pentru neuronul i din stratul de ieșire, w_{i0}^o corespunde bias-ului neuronului i din stratul de ieșire, iar f_2 reprezintă funcția de activare a neuronilor din stratul de ieșire. Ecuația funcției de activare specifică problemei pentru stratul de ieșire este funcția liniară:

$$f_2(u) = u$$

Primii 5 vectori de ieșire vor fi de forma (sunt valori aproximative):

1.2743e-01	9.6369e-01	-3.8662e-02	-3.7470e-03
-9.5587e-02	1.0568e+00	-3.3638e-04	-2.3741e-03
-2.0142e-01	9.9708e-01	1.2880e-02	8.8495e-02
7.4966e-03	-1.7118e-02	9.7506e-02	9.8420e-01
-1.4166e-01	1.9783e-01	-6.4110e-02	8.6041e-01

(2p) Exercițiu Punctat! Exercitiu 3.4: Să se definească funcția cu denumirea ***calculCentroizi*** care primește ca parametrii de intrare un set de date **X** și etichetele, **D**, obținute după etapa de stabilire a etichetelor a algoritmului KMeans (vezi Secțiunea 2.I. Algoritmul KMeans). Să se calculeze centroizii cunoscând etichetele pentru fiecare vector în parte.

Să se încarce funcția **(doar funcția definită în cerință)** pe Moodle la exercițiul cu numele **GR[Nr.Grupa]_RNSF_LAB05_ex2.** Exercițiul restricționează încărcarea altui fișier decât cel din cerință: un fișier cu extansia **.m** și denumirea ***calculCentroizi*** care conține declarația funcției:

```
function centroizi = calculCentroizi(X, D)
% Corp Functie de completat
end
```

Instrucțiuni opționale de rezolvare:

Se vor stabili toate clasele, etichetele prezente in D parcurgând vectorul și stabilind toate valorile unice din el. De exemplu, după parcurgerea vectorului s-au găsit etichetele 1, 2 și 3.
Se calcuează media aritmetică pentru toți vectorii care au eticheta 1, apoi pentru toți vectorii care au eticheta 2 și în final pentru toți vectorii care au eticheta 3 și se savlează într-o matrice cu denumirea **centroizi** de dimensiune *[nr. centroizi x nr. trasaturi]*.

Ecuația matematică:

$$c_i = \frac{1}{N_i} \sum_{X_j \in i} X_j$$

Unde i reprezintă numărul clasei, i = $\overline{1, nr.etichete}$, X_j reprezintă vectorul de trăsături j din setul de date **X**, j=$\overline{1, nr.vectori}$, iar N_i reprezintă numărul de vectori din setul de date care fac parte din clasa i.

(5p) Exercițiu Punctat! Quiz Moodle cu denumirea **GR[Nr.Grupa]_RNSF_LAB05_ex3.** Quiz-ul va dura 10 minute și va fi vizibil la finalul labortorului (interval de timp (ora_incepere+1):30 - (ora_incepere+1):40))

Exercitiu 3.5: Să se calculeze eroarea medie pătratică pentru setul de date *lab05_data01.mat*, utilizând ponderile obținute la *Exercitiu 3.2*.

Cunoscând ieșirile din rețea, se va calcula eroarea medie pătratică conform ecuației:

$$E = \frac{1}{2L} \sum_{j=1}^{L} \sum_{i=1}^{N} (d_i^j - y_i^j)^2$$

Verificare (aprox.): E = 0.026383

Exercitiu 3.6: Să se calculeze **eroarea de clasificare**, *Ecl*, pentru setul de date de test *lab05_data01_test.mat.*

Ne reamintim din laboratorul anterior, *RNSF_Lab4*, că eroarea de clasificare este reprezentată de:

$$E_{cl} = \frac{Nr.vectori\ clasificați\ greșit}{Nr.total\ vectori}$$

Pentru verificare, se va afișa suma ieșirilor din rețea pentru *primul* vector de intrare.
Verificare (aprox.): 1.0303

Verificare: Ecl = 0