Exemplul 1

Probleme de recunoastere a formelor

Sa se creeze in Matlab o retea neuronala de tip perceptron pentru recunoasterea si clasificarea unor segmente de dreapta orizontale, verticale si oblice.

Segmentele vor fi codificate cu ajutorul unor matrici 3x3, cu elemente binare (1 si 0):

```
0 1 0     0 0 0   0 0 1   1 0 0
0 1 0     1 1 1   0 1 0   0 1 0
0 1 0     0 0 0   1 0 0   0 0 1
```

Intrarile retelei vor fi reprezentate de 4 vectori cu 9 elemente fiecare, corespunzatori celor 4 matrici:

```
0  0  0  1
1  0  0  0
0  0  1  0
0  1  0  0
1  1  1  1
0  1  0  0
0  0  1  0
1  0  0  0
0  0  0  1
```

O metoda de codificare a iesirilor retelei de perceptroni, care reprezinta cele 4 tipuri de segmente, poate fi ilustrata in Figura 1.

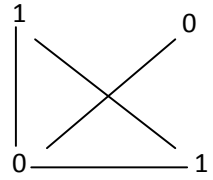

Figura 1. Codificarea iesirii retelei

%Program Matlab

```
vector={[0;1;0;0;1;0;0;1;0] [0;0;0;1;1;1;0;0;0] [0;0;1;0;1;0;1;0;0] [1;0;0;0;1;0;0;0;1]};

vector{1} %afisarea primei matrice coloana
```

```
d={[1;0] [0;1] [0;0] [1;1]};

net=perceptron; %creare retea de tip perceptron

net = configure(net,vector,d);

rez=net(vector)  %simularea retelei

disp('Afisare inainte de antrenare:');

disp(rez{1,1});

disp(rez{1,2});

disp(rez{1,3});

disp(rez{1,4});

%stabilirea numarului de iteratii
net.trainParam.epochs = 50;
%stabilirea erorii permise
net.trainParam.goal = 0.01;
%antrenarea retelei
net=train(net,vector,d);
rez=net(vector);

disp('Afisare dupa antrenare:');

disp(rez{1,1});

disp(rez{1,2});

disp(rez{1,3});

disp(rez{1,4});
```

La executia programului, dupa antrenarea retelei, se obtin codurile segmentelor recunoscute (de exemplu, prima matrice corespunde segmentului vertical, codificat 1 0 etc.):

Dupa antrenare:

```
1    0
0    1
0    0
1    1
```

Simularea retelei, pentru alte 4 segmente ‚imperfecte' codificate astfel:

```
0   1   1      0   0   0      0   0   1      1   0   1

0   1   0      1   1   1      1   1   0      0   1   0

0   1   0      1   0   0      1   0   0      0   0   1
```

```
vector1={[0;1;1;0;1;0;0;1;0] [0;0;0;1;1;1;1;0;0] [0;0;1;1;1;0;1;0;0] [1;0;1;0;1;0;0;0;1]};
```

```
rez=net(vector1);
disp('Rezultatul testelor:');
disp(rez{1,1});
disp(rez{1,2});
disp(rez{1,3});
disp(rez{1,4});
```

La simularea functionarii retelei pentru alte date, se obtin:

Rezultatul testelor:

```
    1    0
    0    1
    0    0
    1    1
```

Segmentele ,imperfecte' au fost clasificate corect, in cele 4 categorii:vertical, orizontal si oblic.

Exemplul 2

Probleme de algebra booleana. Functia XOR

Spre deosebire de perceptronul cu un singur strat care poate sa rezolve probleme de clasificare, numai daca sunt liniar separabile, pentru problemele non liniar separabile, este necesara utilizarea unor retele cu mai multe straturi (retele feed forward) sau perceptronul multistrat.

Un exemplu de functie non – liniar separabila este functia XOR (sau exclusiv), care nu poate fi reprezentata cu ajutorul unei retele neuronale de tip perceptron cu un singur strat. Functia XOR se poate reprezenta folosind o retea feedforward multistrat.

Tabelul functiei boolene XOR pentru doua propozitii logice p si q (p AND q) este prezentat in Figura 2:

p	q	p XOR q
1	1	0
1	0	1
0	1	1
0	0	0

Figura 2. Functia booleana XOR

Curbele de decizie delimiteaza regiunile spatiului de intrare. O separare non – liniara a spatiului de intrare, in cazul functiei XOR, foloseste o curba de decizie (Figura 3).

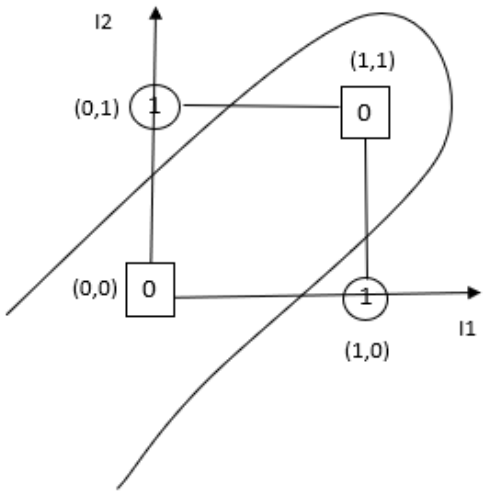

Figura 3. Separare non liniara a spatiului de intrare.

O retea neuronala trebuie sa invete sa identifice aceste regiuni de clasificare (parametrii curbelor de decizie liniara) si sa le asocizeze cu raspunsul corect (iesirea corecta a retelei).

Pentru exemplificare, vom alege o retea neuronala cu 2 intrari, 2 straturi de neuroni, 2 neuroni pe stratul 1 si 1 neuron pe stratul 2 (Figura 4).

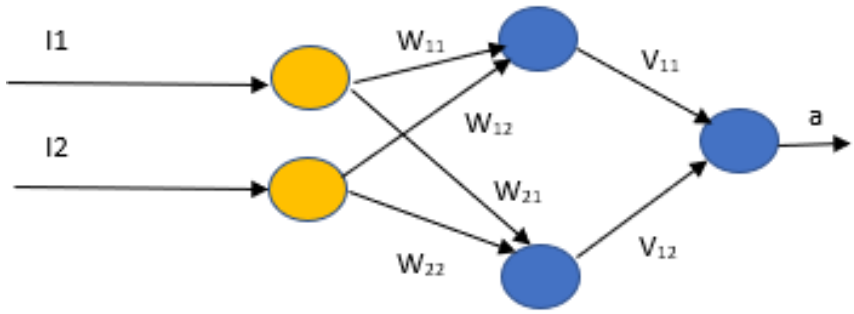

Figura 4. Retea feed forward cu 2 straturi functionale.

In retele feed forward, se foloseste invatarea supervizata, metoda ce presupune existenta unui set de antrenare format din perechi de tipul (intrare, iesire dorita). Procesul de antrenare in astfel de retele multistrat cu propagare directa se bazeaza pe algoritmul Backpropagation (numit si algoritmul gradientului descendent sau regula Delta generalizata).

Denumirea algoritmului Backpropagation provine de la faptul ca eroarea se propaga inapoi, de la ultimul strat inspre primul strat. Eroarea reprezinta diferenta dintre iesirile dorite (tinta), din setul de antrenare, si valorile de iesire determinate de retea, in timpul antrenarii.

Algoritmul Backpropagation cauta minimul functiei Eroare, in spatiul ponderilor, utilizand metoda gradientului descendent (derivata erorii in raport cu ponderea). Deci, solutia problemei invatarii, in retele feed forward, este considerata combinatia de ponderi care minimizeaza functia Eroare.

Algoritmul Backpropagation implica 2 faze:

- Faza Forward, in care parametrii retelei sunt fixati si semnalul de intrare este propagat prin retea , strat cu strat. La sfarsitul acestei faze se calculeaza eroarea astfel:

  $$e_{j=}t_{j}.a_{j},$$

  unde j=1....m, unde m reprezinta numarul de perechi (intrari si iesiri dorite) ale setului de antrenare, a_j noteaza iesirea produsa de retea ca raspuns la intrarea i_j, iar t_j este iesirea dorita (tinta).
- Faza Backward, in care eroarea e_j se propaga prin retea inapoi. In aceasta faza, se modifica parametrii pondere si bias, in scopul minimizarii erorii e_j.

Metoda de antrenare a unei retele feed forward, cu algoritmul Backpropagation, folosind functia predefinita Matlab train este prezentata in exemplul care urmeaza.

```
%program Matlab
p = [0 1 0 1 ; 0 0 1 1];
t = [0 1 1 0];
%creare retea feed forward cu 1 strat ascuns cu 2 neuroni si un strat de
%iesire cu 1 neuron; functia de antrenare setata este ‘trainlm’
net = feedforwardnet(2,'trainlm');
%stabilirea erorii permise
net.trainParam.goal = 0.01;
%seteaza ca reteaua sa foloseasca toate datele pentru antrenare, si nu pentru
%validare sau testare
net.divideFcn = 'dividetrain';
%antrenarea retelei
net = train(net,p,t);
%La simularea retelei, dupa antrenare, se obtine:
a = net(p)
%a =  0.0300    1.0609    1.0305    0.0103
%La simularea functionarii retelei, pentru alt set de date de intrare,
%reteaua raspunde corect (valori apropiate de cele corecte 1, 1, 0, 0):
p1=[0 1 1 0; 1 0 1 0];
a=net(p1)
%a =  1.0305    1.0609    0.0103    0.0300
```