Lucrarea nr.1
Perceptronul. Reguli de învățare

Definiție

Rețelele neuronale (RN) sunt sisteme neliniare formate dintr-un număr mare de procesoare elementare relativ simple, numite neuroni (sau unități, sau perpeptroane) care operează în paralel . Cunoştinţele sistemului sunt reprezentate prin ponderile asociate legăturilor dintre neuroni. Învățarea se realizează prin modificarea ponderilor interconexiunilor.

1. Scurt istoric

Conceptul de "neuron" a fost introdus în 1943 de McCulloch et Pitts, ca un model matematic pentru neuronul elementar biologic. Deşi caracterizat prin limitări majore acest model a permis explicarea funcționării unor ansambluri de neuroni elementari, demonstrând capacitatea lor de a efectua calcule matematice şi logice.

În 1957 Rosenblatt a descris perceptronul, primul model operațional cu capabilități de învățare, capabil să soluționeze probleme de clasificare. Rosenblat a demonstrat şi o teoremă de convergență a algoritmului de adaptare corespunzător. În 1958 a fost proiectat primul neurocomputer Mark I. Perceptronul care a funcționat cu succes în recunoaşterea de caractere. Perceptronul a stimulat cercetare până în 1969, când printr-un studiu riguros publicat sub numele de "Perceptrons" ei demonstrează limitările majore ale arhitecturilor neuronale cu un strat. Neexistând în acel moment nici o teorie referitoare la rețelele multistrat cei doi cercetători au lansat ideea ca limitările se extind şi asupra rețeleor multistrat. Prestigiul celor doi savanți au determinat refocalizarea atenției în alte domenii de cercetare, ducând practic la anularea investițiilor în următorii cincisprezece ani.

Puțini au fost cercetătorii care şi-au continuat studiile în domeniu. În 1960 Widrow şi colaboratorii săi propune ADALINE (ADaptive LInear NEuron) şi MADALINE (more ADALINE) pentru rezolvarea unor probleme filtrare, de recunoaştere a formelor. Primele succese au fost primite cu mare entuziasm de lumea ştiințifică. Specialiştii au emis ipoteza că şi alte procese asociate cu inteligența şi memoria umană pot fi modelate prin rețele neuronale.
La începutul anilor 80 descoperirile teoretice cât şi progresul tehnologic au dus la un reviriment al domeniului. Au fost descoperite şi implementate reguli de învățare noi (ca de exemplu bine cunoscuta regula a retropropagării erorii pentru rețelele multistrat) care au pus în valoare potențialul aplicativ al rețelelor neuronale. În 1982 J.J Hopfield introduce un punct de vedere elegant asupra rețelelor neuronale, care permite interpretarea lor ca sisteme energetice cărora li se poate asocia o funcție de energie. Evoluția sistemului este spre minimizarea funcției de energie similara unei funcții de cost specifică aplicației.

Prima conferință internațională consacrată rețelelor neuronale are loc în 1987 la San Diego (SUA). În următorii ani apar primele organizații de profil şi primele reviste dedicate domeniului.
În prezent majoritatea universităților au grupuri de cercetare heterogene, incluzând ingineri, matematicieni, informaticieni, psihologi, medici, biologi.

2. Scopul lucrării practice

Lucrarea este o introducere în domeniul rețelelor neuronale. Se prezintă câteva noțiuni fundamentale ca de exemplu modelul unui neuron (cunoscut şi sub denumirea de perceptron), funcția de activare , reguli de învățare şi câteva aplicații practice.

3. Noțiuni teoretice

Modelul unui neuron este prezentat în Fig. 1

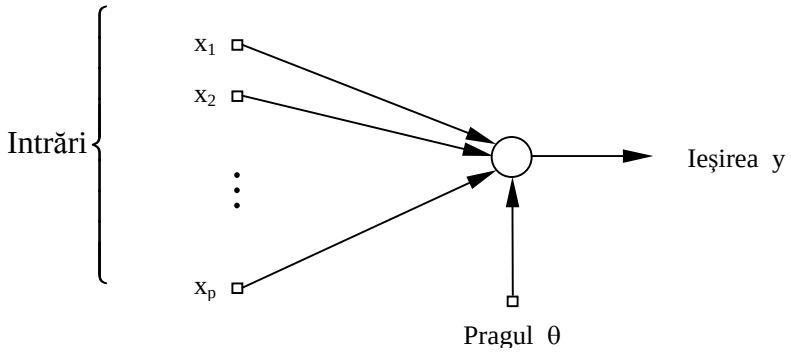


Fig. 1: Modelul unui neuron

Ieşirea neuronului este:

$$\mathbf{y = f\left(\sum_{i=1}^{p} w_i x_i - \theta \right)}$$

unde:

- $\mathbf{w_i}$ sunt numere reale numite ponderi ;
- $\mathbf{x_i}$ sunt intrări;
- θ este o constantă , numită prag;
- **f(.)** este funcția de activare;

O pondere pozitiva reprezintă o intrare excitatoare. O pondere negativă reprezintă o intrare inhibitorie . Ponderile $\mathbf{w_i}$ sunt modificate în timpul procesului de antrenament în conformitate cu regula de învățare.

3.1 Funcția de activare

Funcția de activare reprezintă funcția de transfer intrare-ieşire a neuronului. Ea combină intrarea curentă cu starea de activare existentă în neuron a(t) pentru a genera o nouă stare de activare a(t+1).

$$\mathbf{a_i(t+1) = f(a_i(t), net_i(t))}$$

Funcția de activare poate fi deterministă sau stohastică (probabilistică).Vom prezenta în cele ce urmează câteva exemple. Câteva exemple sunt:

- funcția Heaviside (cunoscută în rețelele neuronale şi sub numele de hard-limitatoare): unipolară $\sigma(t)$ (treaptă unitate) sau bipolară sgn(t);

$$\mathbf{a_i(t+1) = sgn(net_i(t))} \begin{cases} \mathbf{1 \ dacă \ net_i(t) \geq 0} \\ \mathbf{-1 \quad dacă \quad net_i(t) < 0} \end{cases}$$

Intrarea netă este dată de suma ponderată a intrărilor neuronului.

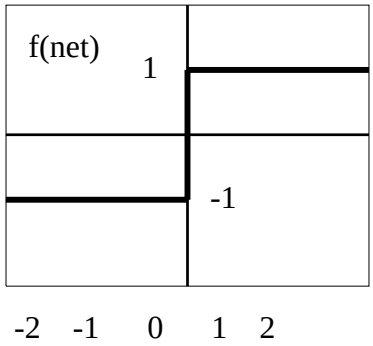
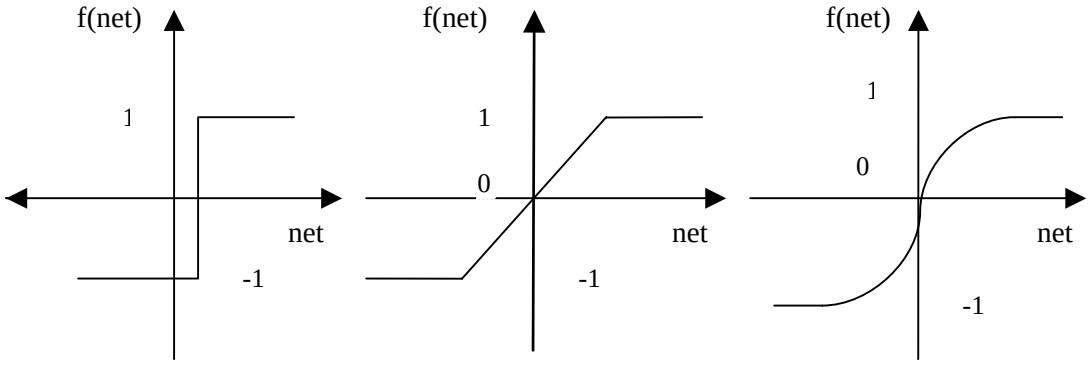


Fig.2 Funcția signum



Alte câteva exemple de funcții de activare

a) funcția bipolară cu prag ; b)semi liniară; c) sigmoidă bipolară ;

Adesea funcția de activare trebuie să fie o funcție neliniară, nedescrescătoare, astfel încât intrarea netă trebuie să depășească o valoare numită prag pentru determinarea unei noi activări :

$$a_i(t+1) = f_i\left(\sum_j w_{ij} a_j(t) - \theta_i(t)\right)$$

Uzual funcția de activare este o funcție neliniară, numită logistică sau sigmoidă, dată de relația :

$$a_i(t+1) = \begin{cases} 1 \text{ dacă } net_i(t) \geq \theta_i \\ net_i(t) \quad \text{dacă} \quad net_i(t) < \theta_i \\ 0 \quad \text{în} \quad \text{rest} \end{cases}$$

Un exemplu de funcție sigmoidă este :

$$a_i(t+1) = \frac{1}{1 + e^{-\beta . net_i(t)}}$$

unde β este un factor de proporționalitate. Avantajul acestei funcții este derivata sa, simplu de determinat :

$$f'(x) = f(x)[1 - f(x)]$$

Funcția tangentă hiperbolică este şi ea des utilizată deoarece intervalul său de răspuns este [-1, +1].:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

şi derivata sa se calculează relativ uşor :

$$\frac{d\tanh(x)}{dx} = [\sec h(x)]^2 = \frac{4}{(e^x + e^{-x})^2}$$

Există rețele care utilizează funcții de activare probabilistice. Probabilitatea ca neuronul să fie activ este :

$$p(a_i(t) \to 1) = \frac{1}{1 + e^{\frac{-net_i(t)}{T}}}$$

unde T este o constantă, numită temperatură. Aceasta dă panta curbei de probabilitate.

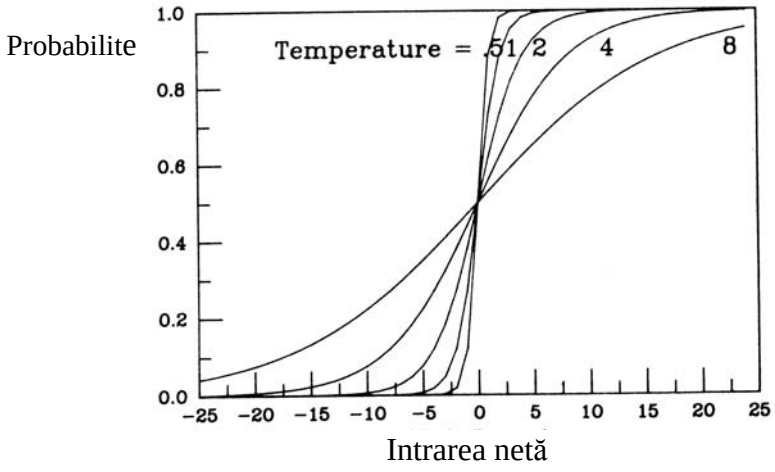


Fig.2.6 Funcție probabilistică de activare

Rețelele cu funcții neliniare sunt uzual utilizate, performanțele lor fiind superioare celor cu funcții liniare sau semiliniare. Se pot utiliza şi alte tipuri de funcții, unele cunoscute din teoria aproximării, care de la aplicație la aplicație conduc la performanțe superioare.

Deşi majoritate funcțiilor de activare sunt monotone, există şi funcții nemonotone care conduc la performanțe foarte bune în special în memoriile asociative.

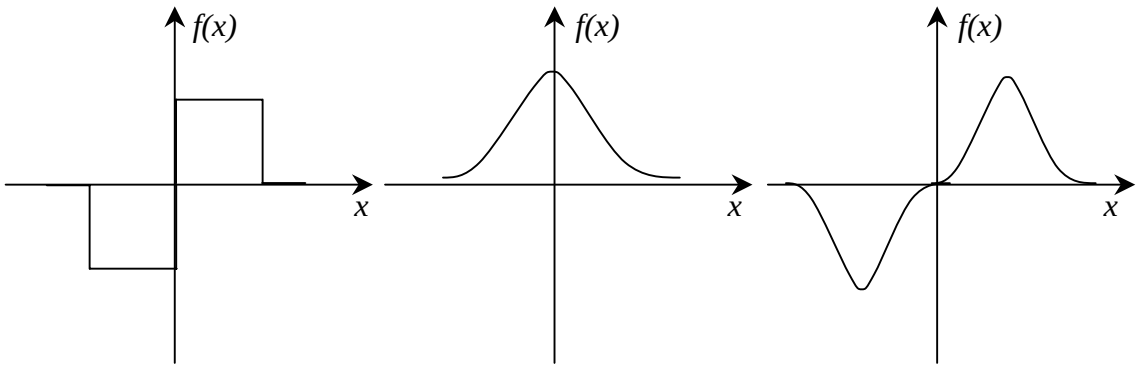


Fig.2.7 Funcții de activare nonmonotone

3.2 Reguli de învățare

Configurarea ponderilor unei rețele neuronale, respectiv a unui neuron trebuie făcută astfel încât aplicarea unui set de intrări să genereze un set de ieşiri dorite (ieşire dorită). Există diferite metode pentru determinarea ponderilor :

- fixarea explicită, utiliyând informație apriori referitoare la particularitățile şi eventual restricțiile la care este supusă aplicația considerată ;
- determinarea ponderilor prin antrenare, generând rețelei modele de învățat şi lăsând-o să-şi modifice ponderile conform unei reguli de învățare, în mod iterativ ;

O condiție esențială este ca algoritmul de antrenare să fie convergent, la un moment dat ponderile să rămână constante, indiferent de intrările aplicate.

Câteva dintre regulile uzuale vor fi prezentate în continuare :

Câteva dintre regulile uzuale vor fi prezentate în continuare :

1) **Regula lui Hebb**

   Conform acestei reguli ponderea în pasul k+1 se modifică proporțional cu produsul dintre intrarea şi ieşirea neuronului :

$$\Delta w_{ij} = \eta \cdot o_i \cdot x_j$$

   unde:

   - Δw_{ij} este variația vectorului pondere w_{ij} de la neuronul j către neuronul i din pasul **k** în pasul **(k+1)**, dată de relația: $w_{ij}[k+1] = w_{ij}[k] + \Delta w_{ij}$;
   - o_i este ieşirea neuronului i;
   - x_j este intrarea în neuronul j ;
   - η este o constantă de care depinde viteza de învățare, $\eta \in (o,1)$;

   Această regulă de învățare este fără control (nesupravegheată) deoarece nu utilizează răspunsul dorit.

2) **Regula perceptronului :**

   Este o regulă supervizată pentru că în calculul variației ponderii se utilizează răspunsul dorit notat cu d_i :

$$\Delta w_{ij} = \eta.[d_i - sgn(w_i^T x)].x_j$$

   unde x este vectorul intrărilor în neuronul j $x=[x_1\, x_2 \ \ldots x_j \ldots \ x_n\ ]$

3) **Regula Delta (sau regula Widrow-Hoff, supervizată)**

   Denumirea de Delta este dată de diferența dintre ieşirea curentă pentru cazul părticular în care f(.)=1 şi răspunsul dorit :

$$\Delta w_{ij} = \eta.[d_i - o_i)].x_j$$

   Regula este cunoscută şi sub denumirea autorilor săi regula Widrow şi Hoff.

4) **Regula Delta generalizată (supervizată)**

   Modificările în ponderi se calculează cu :

$$\Delta w_{ij} = \eta.[d_i - o_i)].f^{'}(net_i)x_j$$

unde f′ este derivata funcției de activare, deci este valabilă doar pentru funcții de activare continue .

5) Regula de învățare a corelației

Este tot o variantă a regulii lui Hebb:

$$\Delta w_{ij} = \eta \cdot d_i \cdot x_j$$

6) Regulă de învățare de tip competitiv

$$\Delta w_{mj} = \eta \cdot (o_j \cdot w_{mj})$$

Se modifică doar ponderile neuronului m câştigător, în rest ponderile rămân nemodificate.

7) Regula outstar (a lui Grossberg)

$$\Delta w_{ij} = \eta \cdot (d_i \cdot w_{ij})$$

Există bineînțeles multe alte reguli de învățare dezvoltate pentru a ameliora performanțele RN. Pe parcursul cursului vor fi prezentate şi alte reguli de învățare.

4. Funcțiile Matlab utile pentru această lucrare practică sunt:

4.1)Funcții de inițializare:

- *randnc → pentru a da valori inițiale aleatoare ponderilor şi pragului, într-un vector coloană ;*
- *randnr → pentru a da valori inițiale aleatoare ponderilor şi pragului, într-un vector linie ;*
- *rands → pentru a da valori inițiale aleatoare simetrice;*

4.2) Funcții de activare:

- *hardlim → funcția Heaviside unipolară;*
- *hardlims → funcția Heaviside bipolară;*
- *logsig → funcția logistică logaritmică;*
- *purelin → funcția liniară;*
- *tansig → funcția sigmoidă de tip tangentă;*

4.3) Reguli de învățare:

- *learnh → regula de învățare Hebb (1);*
- *learnhd → regula de învățare Hebb (2);*
- *learnos → regula de învățare Hebb (3);*
- *learnp → regula de învățare de tip perceptron;*
- *learnwh → regula Widrow-Hoff;*

4.4) Funcții de afişare:

- *barrerr → histograma valorilor pentru fiecare vector de ieşire;*
- *errsurf → calculul suprafeței de eroare pentru perceptronul simplu cu o intrare şi un prag ;*
- *hintonw → reprezentarea simbolică a ponderilor;*
- *hintonwb → reprezentarea pragului;*
- *ploterr → reprezentarea simbolică a funcției eroare;*

- *plotfa → reprezentarea simbolică a valorilor dorite furnizate rețelei ;*
- *plotlr → reprezentarea simbolică a funcției de învățare ;*
- *plotpc → reprezintă linia de clasificare generată de perceptronul cu două intrări;*
- *plotpv → reprezintă vectorii de învățare pentru perceptronul cu două intrări;*

5. Desfăşurarea lucrării

5.1 Neuronul cu o intrare

Se lansează programul MATLAB. Se selectează din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks.

Se încarcă "Simple neuron and transfer functions".

5.1.1

Se alege funcția de activare cu F. Ieşirea neuronului este dată de relația:

a=f(w.p+b)

Reprezentați funcția de transfer selectată, în spațiul intrării selectând pragul b la zero.

Modificați pe rând intrarea p, pragul b şi ponderea conexiunii w. Observați schimbările asupra funcției de activare şi ieşirea curentă. Ce concluzie puteți trage?

5.1.2 Repetați punctul 5.1.1 pentru toate funcțiile de activare prevăzute în F.

5.2 Neuronul cu două intrări

Se încarcă "Neuron with vector input".

5.2.1 Selectați funcția de activare cu F. Modificați intrările p(1) şi p(2) mutând indicatoarele triunghiulare. In mod similar modificați w şi b. Intrarea netă şi ieşirea netă se vor modifica şi ele. Observați şi notați schimbările.

5.2.2 Repetați punctul 5.2.1 pentru toate funcțiile de activare prevăzute în F.

Lucrarea nr. 2
Limitele perceptronului
Separabilitatea liniară şi separabiltatea nonlineară

1. Noțiuni teoretice

Se poate demonstra că un neuron adaptabil (perceptronul) nu poate discrimina decât clase liniar separabile.

Definiție

Clasele distincte se pot separa prin aşa zise suprafețe de decizie. Pentru determinarea suprafețelor de decizie trebuie evaluat un set de funcții de discriminare. Două clase se numesc **liniar separabile** dacă funcțiile de discriminare au forma următoare:

$$g_i(X) = a_o + \sum_n^N a_n . x_n$$

unde a_0, a_1,…a_n sunt constante.

Deci două categorii sunt liniar separabile dacă pot fi separate printr-o dreaptă (Fig1). Funcțiile logice SI, SAU şi NU implică clasificări separabile printr-o dreaptă.

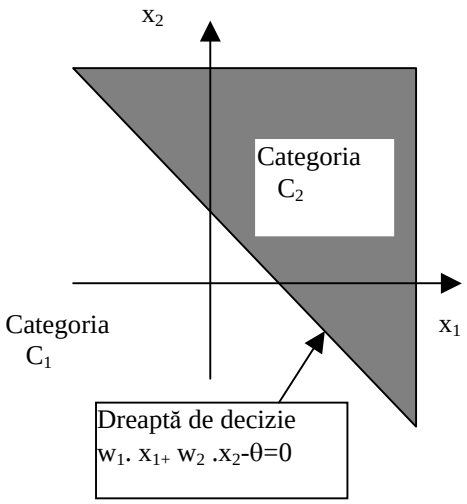


Fig.1 Două categorii liniar separabile

Cel mai cunoscut caz de categorii nonseparabile liniar este cel al funcției SAU EXCLUSIV (XOR). Tabelul de adevăr al funcției XOR este următorul :

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Regiunile de decizie ale funcțieie SAU Exclusiv nu pot fi separate printr-o dreaptă, ci prin două drepte , cum se poate vedea în Fig.2.

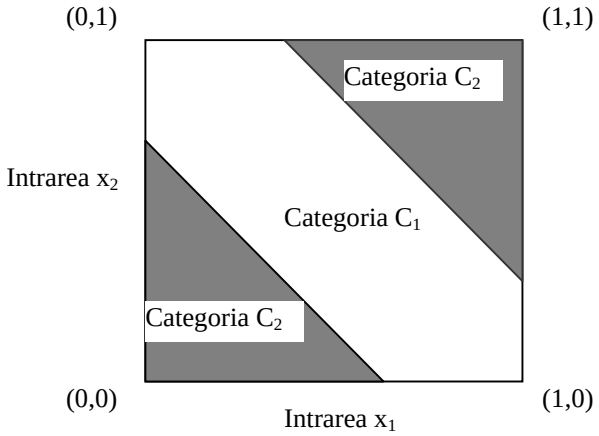


Fig.2 Spațiul modelelor de intrare pentru problema lui SAU EXCLUSIV

Limitele de învățare ale unei rețele neuronale cu un singur strat de neuroni adaptivi nu sunt date de algoritmul de învățare ci de topologia rețelei, care permite divizarea spațiului de intrare doar în două semiplane.

Dacă funcția SAU EXCLUSIV este adecvat codată se pot utiliza trei neuroni de intrare, celui de-al treilea aplicâdu-i-se la intrare produsul primelor două. Problema de nonseparabilitate liniară se transformă într-una de separabilitate liniară, concretizată prin următorul tabel de adevăr :

x_1	x_2	x_3	XOR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

După cum se poate observa cea de-a treia coloană se obține prin multiplicarea primelor două. Fie funcția de activare dată de relația:

$$f(net(t))\begin{cases} \mathbf{1\ dacă\ net}_i(t) > \theta \\ \mathbf{0\quad dacă\quad net}_i(t) \le \theta \end{cases}$$

Dacă se aleg ponderile $w_1 = 1$, $w_2 = 1$, $w_3 = -2$ și pragul neuronului $\theta = 0$, problema lui XOR devine o problemă de separabilitate liniară. Deci un aspect important în practica RN este o reprezentare adecvată a problemei de rezolvat.

O altă metodă pentru rezolvarea unor probleme non-separabile liniar este utilizarea de rețele cu mai multe straturi. Pentru limite de decizie mai complexe sunt necesare mai multe straturi. Pentru SAU EXCLUSIV sunt necesare două straturi de neuroni. Fig.3 reprezintă două variante pentru implementarea funcției .

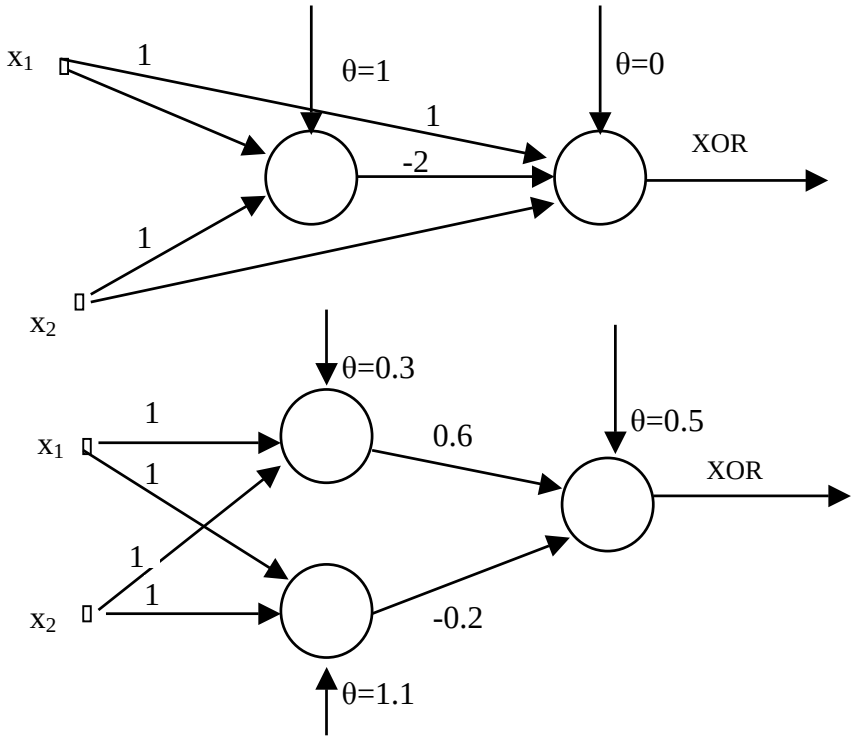


Fig.3 Două rețele pentru învățarea funcției logice XOR

Pentru limite de decizie mai complexe sunt necesare mai multe straturi de neuroni. Rețelele din Fig.3 se numesc cu două straturi, pentru că au două straturi de neuroni adaptabili.

O RN cu două straturi este capabilă să identifice orice fel de regiuni convexe, dacă numărul neuronilor din stratul ascuns este suficient şi ponderile sunt adecvat adaptate.

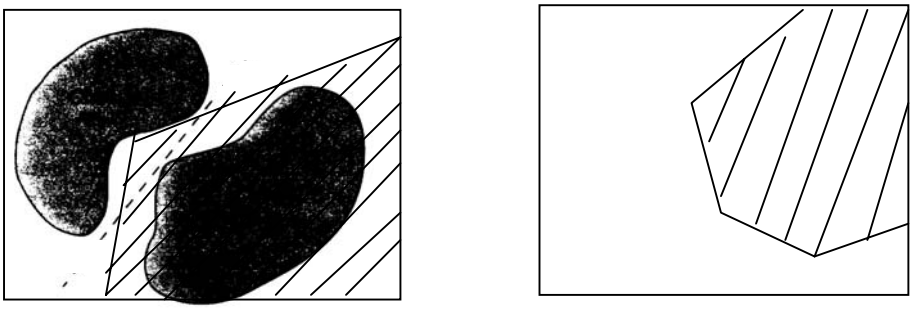


Fig.4 Regiuni de decizie pentru o RN cu două straturi de neuroni

Fig.4 prezintă două exemple

O RN cu două straturi poate forma regiuni de decizie convexe obținute prin intersecția semiplanelor de decizie ale neuronilor primului strat. Fiecare neuron al startului ascuns generează un hiperplan de separare. Neuronii stratului de ieşire generează regiuni de decizie mai complicate, formate prin intersecția semiplanelor primului strat.

O RN cu trei straturi poate implementa regiuni de decizie arbitrare, complexitatea fiind limitată de numărul de neuroni. S-a demonstrat că precizia unei clasificări neliniare de către o RN cu trei straturi (2 ascunse) poate fi făcută arbitrar de bună.

Deci cu alte cuvinte o RN cu trei straturi este capabilă să proceseze orice transformare neliniară continuă cu o precizie arbitrar de bună.

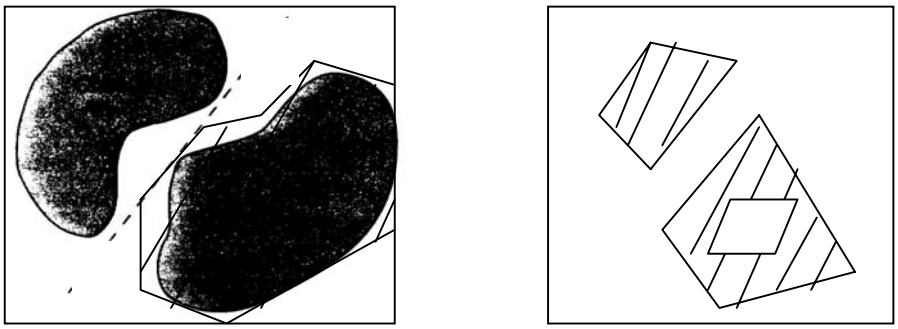


Fig.**5 Regiuni de decizie pentru o RN cu trei straturi de neuroni**

2. Scopul lucrării practice

Se prezintă problema separabilității şi a nonseparabilității liniare şi generalități despre performanțele RN cu un strat, două respectiv trei straturi de neuroni.
Se studiază performanțele unui neuron cu ponderi adaptabile cu funcție de activare hardlimitatoare, respectiv liniară.

3. Desfăşurarea lucrării

Se lansează programul MATLAB. Se selectează din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks.

3.1 Limitele neuronului adaptabil

Încărcați programul “Decision Bounderies ”.

Neuronul testat are două intrări cu valori în intervalul (-3,3).Selectați vectorii de intrare mutând cu mouse-ul punctele albe, respectiv negre, pentru a alege diferite probleme. Mutați dreapta de decizie a neuronului încercând să clasificați modelele de intrare astfel înc-t nici unul dintre modele să nu aibe contur roşu (caz în care nu poate fi clasificat nici într-una dintre cele două categorii posibile). Observați şi notați valorile ponderilor şi pragul. Ce concluzii puteți trage în legătură cu posibilitatea de clasificare a unui neuron?

3.2 Neuronul cu ponderi adaptabile

Încărcați programul “Perceptron learning rule ”.

3.2.1 Selectați cel mult cinci modele de intrare, trăgând cu mouse-ul punctul alb (negru) în planul modelelor de intrare. Click “Random ”pentru a inițializa aleator ponderile de intrare. Click “Learn” pentru a aplica regula de învățare unui singur vector de intrare. Click “Train” pentru a aplica regula de învățare pentru cinci vectori de intrare. Notați ponderile w şi pragul b obținute în urma antrenării. Ce fel de probleme poate soluționa un neuron adaptabil?

3.2.2 Repetați 3.1 definind mai multe probleme.

3.2.3 Alegeți mai mult de cinci modele de intrare. Ce puteți constata?

3.2.4 Vizualizați cu NOTEPAD programul sursa pentru implementarea neuronului cu ponderi adaptabile.

3.3 Selectați “Classification with a 2 inputs perceptron ”.
Dați “Start” pentru a inițializa ponderile. Pe ecran se pot vedea cei patru vectori de antrenament şi coordonatele lor. Ponderile se vor adapta dacă selectați Next pentru a clasifica în două categorii vectorii de intrare. Un al cincilea model cu care va fi confruntat neuronul va fi şi el clasificat într-una dintre cele două categorii.

3.4 Selectați “Outlier input vectors ”
Dați “Start” pentru a inițializa ponderile. Pe ecran se pot vedea cinci vectori de antrenament şi coordonatele lor. Unul dintre vectorii de intrare este departe de toți ceilalți vectori.El se numeşte "outlier". Ponderile se vor adapta dacă selectați Next pentru a clasifica în două categorii vectorii de intrare. Observați că antrenarea durează mult mai mult decât în cazul anterior. Odată antrenat neuronul , orice alt model planul modelelor de intrare va fi şi el clasificat într-una dintre cele două categorii.

3.5 Selectați “Normalised perceptron rule ”
Repetați paşii din 4.4. Observați că neuronul se va antrena mult mai rapid, în doar 3 epoci față de 32 în cazul anterior. Pentru antrenarea neuronului în acest caz se foloseşte o regulă de învățare ce implică normalizarea vectorilor de intrare, deci este mai puțin sensibilă la variațiile vectorilor de intrare.

3.6 Selectați “Linearly nonseparable input vectors ”
Rulare programului vă permite să constatați că un neuron adaptabil nu poate clasifica modele de intrare nonseparabile liniar.

3.7 Selectați “Pattern association showing error surface ”
Funcția de activare a neuronului utilizat este liniară. Ponderile se vor adapta astfel încât soluția va corespunde minimului suprafeței de eroare. Vizualizați suprafața de eroare în spațiul modelelor de intrare.

3.8 Rulați programul “Training a linear neuron ”
Prin acest program se antrenează un neuron pentru a răspunde unui model de intrare cu un anumit model de ieşire. Vizualizați suprafața de eroare. Câte răspunsuri posibile are neuronul?

3.9 Selectați programul “Linear fit to a nonlinear problem ”.
Programul vă permite să găsiți cea mai bună soluție pentru o problemă neliniară, de asociere a unui model de intrare cu un model de ieşire.

3.10 Rulați programul "Undetermined problem" pentru a antrena un neuron liniar să răspundă pentru un model de intrare dat cu un model de ieşire dorit. Întrucât aceasta este o problemă neliniară, soluția obținută nu va fi unică. Testați câteva variante.

3.11 Selectați "Linearly depenendent problem" Vectorii aplicați la intrarea neuronului sunt liniar dependenți. Încercați să dați o definiție pentru liniar dependență. Vectorii de ieşire care se doresc a se asocia nu sunt în aceeați relație de liniar dependență ca şi vectorii de intrare. Problema dată spre rezolvare este una de tip neliniar, astfel încât eroarea la ieşirea neuronului liniar nu va fi nulă.
Pentru toate subpunctele din "Desfăşurarea practică a lucrării" notați modele de intarre, modelele de ieşire dorite, ponderile şi pragul neuronului obținute ca soluție.

REȚELE MULTISTRAT

Incapacitatea rețelelor cu un strat adaptiv de a transforma modele arbitrare de intrare în modele arbitrare de ieşire, a fost depaşită utilizând un strat intermediar, numit ascuns între intrare si ieşire.

În forma lor generală, rețelele neuronale multistrat, (MLP multilayer perceptrons) Fig.1. au un strat de intrare, un număr de straturi ascunse şi un strat de ieşire. Când informația se propagă prin rețea de la intrare spre ieşire aceste rețele se numesc de tip "spre înainte" (feed-forward).

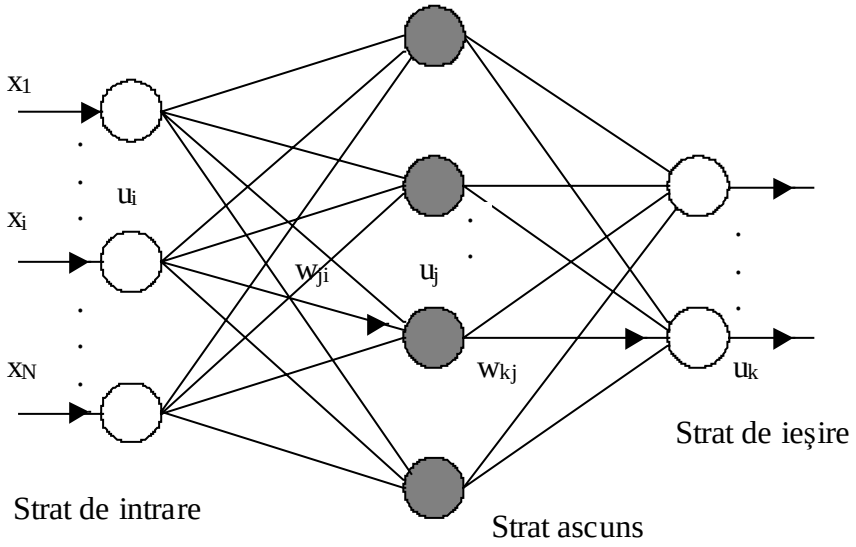


Fig.1 Arhitectura unei rețele neuronale multistrat

O funcție de activare liniară , cel mult semiliniară, pentru neuronii stratului de ieşire este suficientă pentru majoritatea aplicațiilor:

$$a_i(t+1) = f(net) = \begin{cases} 0, & \text{pentru } net(t) < -\theta \\ \dfrac{net(t)+\theta}{2\theta}, & \text{pentru } -\theta < net < \theta \\ 1, & \text{pentru } net(t) > \theta \end{cases} \quad (1)$$

Literatura demonstrează necesitatea ca funcția de activare a unităților ascunse să fie cel puțin semiliniară pentru a se depăşi performantele rețelelor cu un strat. Uzual ea este o funcție aşa zisă sigmoidă:

$$0_{pk} = \frac{1}{1+e^{-\beta(\sum_j w_{kj} \cdot o_{pj}(t)+\theta_k)}} \quad (2)$$

unde β este un factor de proporționalitate.

Avantajul unei astfel de funcții este calculul simplu al derivatei:

$$f'(x) = f(x)[1-f(x)] \quad (3)$$

Adesea se utilizează funcția tangentă hiperbolică, deoarece are valori în intervalul [-1, +1]:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (4)$$

Derivata ei este:

$$\frac{d\tanh(x)}{dx} = [\sec h(x)]^2 = \frac{4}{(e^x + e^{-x})^2} \quad \textbf{(5)}$$

Pentru antrenarea rețelei MLP se utilizează uzual algoritmul cu "propagare inversă a erorii".

Algoritmul propagării inverse a erorii (back-propagation)

Algoritmul a fost fundamentat independent de mai mulți cercetători din domeniul analizei numerice (Bryson & Ho, 1969) şi al statisticii (Werbos în 1974) până în cel al RN (Parker 1982 , Le Cun 1986, Rumelhart, Hinton & Wiliam 1986).. Algoritmul este o învățare cu control în două faze şi este cunoscut şi sub denumirea de "regula delta generalizată", denumire introdusă în "Parallel Distributed Processing" de grupul creat de Rumelhart şi Mc Clelland .

Prima etapă

În prima faza, modelul p de intrare se propagă prin rețea din strat in strat, până la ieşire. Fie notațiile:

N_{input} : numărul intrărilor în RN (care dă dimensiunea vectorilor de intrare);

N_h : numărul unităților stratului ascuns;

N_{output} : numărul unităților stratului de ieşire;

Intrarea netă în fiecare neuron ascuns este pentru fiecare model de intrare p:

$$\mathbf{net_{pj}} = \sum_{i=1}^{Ninput} \mathbf{w_{ji}} . \mathbf{x_i} + \theta_j \quad \textbf{(6)}$$

Ieşirea fiecărui neuron ascuns este o funcție de suma ponderată a intrărilor şi pragul θ:

$$\mathbf{o_j} = \mathbf{f}\left(\sum_{i=1}^{N_{input}} \mathbf{w_{ji}} \mathbf{x_i} + \theta_j\right) \quad j=1,..,\ N_h \quad \textbf{(7)}$$

În cazul unei rețele neuronale cu un singur strat ascuns, ieşirea o_k a neuronului k de ieşire se exprimă în funcție de informația primită din stratul ascuns cu relația:

$$\mathbf{o_k} = \mathbf{f}\left(\sum_{j=1}^{N_h} \mathbf{w_{kj}} \mathbf{o_j} + \theta_k\right) \quad \textbf{ou} \quad \mathbf{o_k} = \sum_{j=1}^{N_h} \mathbf{w_{ij}} \mathbf{o_j} + \theta_k \qquad k=1,..,\ N_{output} \quad \textbf{(8)}$$

unde vectorul $w_k = (w_{k1}\ \ w_{k2} ... w_{kj})$ este vectorul ponderilor conexiunilor neuronului k de ieşire. Funcția globală realizată de RN la ieşirea neuronului k este deci:

$$\mathbf{o_k} = \mathbf{f}\left(\sum_{k=1}^{N_h} \mathbf{w_{kj}} \mathbf{f}\left(\sum_{i=1}^{N_{input}} \mathbf{w_{ji}} \mathbf{x_i} + \theta_j\right) + \theta_k\right) \quad k=1,..,\ N_{output} \quad \textbf{(9)}$$

Pentru neuronii unui strat funcția de transfer este în general aceeaşi.

Ieşirea curentă o_k se compară cu intrarea de control t_{pk}, generând, în unitățile de iesire, o eroare δ_{pk}:

$$\delta_{pk} = (t_{pk} - o_{pk}).f'(net_{pk}) \quad (10)$$

unde f' este derivata funcției de activare a neuronului.

Etapa a doua

În faza a doua, erorile se propagă de la ieşire spre intrare, din strat în strat determinând schimbarea ponderilor conexiunilor în sensul minimizării erorii la nivelul fiecărui neuron în parte. Regula de învățare pentru conexiunile dintre neuronii de ieşire şi cei ascunşi este:

$$\Delta_p w_{kj} = \eta.\delta_{pk}.o_{pj} \quad (11)$$

unde η este constanta de învățare care poate lua valori în intervalul (0,1).

Pentru unitățile ascunse, indexate după j, erorile δ_{pj} se calculează cu ajutorul erorilor de la neuronii de ieşire δ_{pk} cu relația:

$$\delta_{pj} = (\sum_k \delta_{pk}.w_{kj}).f`(net_{pj}) \quad (12)$$

Apoi se modifică ponderile conexiunilor dintre neuronii ascunşi şi intrări cu o regulă de învățare (care poate fi aceeaşi ca cea dată de relația 11):

$$\Delta_p w_{ji} = \eta.\delta_j.o_{pi} \quad (13)$$

Dacă în RN există mai multe straturi ascunse eroarea se evaluează pentru fiecare strat cu relația (12) în care erorile reprezintă erorile stratului ascuns superior şi apoi se determină noile ponderi dintre stratul anterior şi cel succesiv (într-o convenție de notare a straturilor de la intrare spre ieşire). În rețea pot exista şi ponderi fixe. Dacă exista unități de ieşire şi în straturile ascunse, acestea însumează două tipuri de erori: erori rezultate din compararea ieşirii cu răspunsul dorit şi erori obținute prin propagare de la unitățile de ieşire spre unitățile cu care sunt cuplate.

Observație

Această procedură minimizează eroarea pătratică medie în fiecare iterație (dacă viteză de învățare η este adecvat aleasă). Rezultatele practice arată că RN converge în general spre un minim local (care poate fi şi global), care reprezintă în unele cazuri o soluție acceptabilă. În literatura de specialitate sunt prezentate mai multe metode pentru evitarea minimelor locale, aspect care va fi abordat ulterior.

Se poate demonstra că algoritmul implementează un gradient descendent al erorii totale în spațiul ponderilor, adică:

$$w(t+1) = w(t) - \eta.\frac{dE}{dw} \quad (14)$$

unde dE/dw este gradientul aleatoriu necunoscut al erorii totale dintre modelele de intrare şi modele de ieşire dorite. White a demonstrat că relația (14) este o aproximare stohastică.

Demonstrație

Se demonstrează că o RN antrenată cu regulă de învățare dată de relația (11) minimizează eroarea pătratică medie pentru toate modele de intrare dacă funcția de activare a neuronilor este sigmoidă:

Fie funcția de activare a neuronilor o funcție sigmoidă dată de relația:

$$\mathbf{f(net_p) = o_p = \frac{1}{1 + e^{-net_p}}} \tag{15}$$

şi

$$\mathbf{net_{p_k} = \sum_j w_{kj} o_{pj} + \theta_k}$$

$$\mathbf{net_{p_j} = \sum_i w_{ji} o_{pi} + \theta_j}$$

Fie eroarea pătratică medie a RN pentru un model p :

$$\mathbf{E_p = \frac{1}{2} \sum_k (d_{pk} - o_{pk})^2} \tag{16}$$

Eroarea totală pentru toate modelele p este:

$$\mathbf{E_{totală} = \sum_p E_p} \tag{17}$$

Trebuie să se demonstreze de fapt că pentru fiecare model p:

$$\mathbf{\frac{-\partial E_p}{\partial w_{kj}} = \delta_{pk} . o_{pj}} \tag{18}$$

Se ştie că pentru algoritmul BKP variația ponderilor este dată de relația (11):

$$\mathbf{\Delta_p w_{kj} = \eta \delta_{p_k} o_{pj}}$$

Trebuie deci să se demonstreze că:

$$\mathbf{\Delta_p w_{kj} = -\eta \frac{\partial E_p}{\partial w_{kj}}} \tag{19}$$

Utilizând metoda derivării în lanț obținem:

$$\mathbf{\frac{\partial E_p}{\partial w_{kj}} = \frac{\partial E_p}{\partial net_{p_k}} \frac{\partial net_{p_k}}{\partial w_{kj}}} \qquad \text{şi} \qquad \mathbf{\frac{\partial E_p}{\partial net_{p_k}} = \frac{\partial E_p}{\partial o_{pk}} \frac{\partial o_{pk}}{\partial net_{p_k}}}$$

dar

$$\mathbf{\frac{\partial net_{p_k}}{\partial w_{kj}} = o_{pj}}$$

$$\mathbf{\frac{\partial o_{p_k}}{\partial netp_k} = o_{p_k}\left(1 - o_{p_k}\right) = f'\left(netp_k\right)}$$

$$\frac{\partial Ep}{\partial o_{pk}} = -(d_{pk} - o_{pk})$$

Dacă se înlocuieşte această ultimă relație în relația (19) se obține chiar relația (11):

$$\Delta_p w_{kj} = \eta f'(net_{p_k})(d_{pk} - o_{ok}) o_{pj} = \eta \delta_{p_k} o_{pj} \qquad \text{q.e.d.}$$

Pentru unitățile ascunse trebuie demonstrat că variația ponderilor :

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} = \eta \left(\sum_k \delta p_k w_{kj} \right) f'(net_{p_j}) o_{pi}$$

determină un gradient descendent al erorilor

$$\Delta_p w_{ji} = -\eta \frac{\partial E_p}{\partial w_{ji}} \qquad (20)$$

Pentru unitățile ascunse:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{p_j}}{\partial w_{ji}}$$

$$\frac{\partial E_p}{\partial net_{pj}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}}$$

$$\frac{\partial Ep}{\partial o_{pj}} = \sum_k \frac{\partial Ep}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}} = \sum_k \frac{\partial Ep}{\partial net_{p_k}} w_{kj}$$

$$\frac{\partial net_{p_j}}{\partial w_{ji}} = o_{pi}$$

$$\frac{\partial o_{pj}}{\partial net_{p_j}} = f'(net_{p_j})$$

Se ştie că:

$$\delta p_j = \left(\sum_k \delta_{pk} w_{kj} \right) f'(net_{p_j})$$

Dacă se înlocuieşte $\frac{\partial Ep}{\partial w_{ji}}$ în relația (20) se obține chiar relația (13), ceea ce trebuia demonstrat.

Observații

Algoritmul presupune o etapă de antrenament, una de testare şi apoi cea de utilizare. Crearea bazei de date, împărțirea acesteia în mod adecvat influențează major succesul sau eşecul funcționării unei RN. În timpul antrenamentului, fiecare pereche p de model de intrare-model de control, este prezentată repetat. În timpul testării se aplică doar modelele de intrare,

verificându-se statistic corectitudinea funcționarii. În cazul în care performanta este acceptabila rețeaua poate fi utilizată.

Dezavantajul metodei constă în necesitatea unei prezentări repetate a modelelor în timpul antrenamentului (deci la un timp îndelungat afectat antrenamentului).

Exemplu

Fie o RN cu trei neuroni de intrare I=3, doi neuroni ascunşi L=2 şi trei neuroni de ieşire J=3. Matricea interconexiunilor dintre neuronii de intrare şi cei ascunşi W de ordinul $L \times I=2 \times 3$ este:

$$\mathbf{W}=\begin{bmatrix} \mathbf{w}_{11} & \mathbf{w}_{21} & \mathbf{w}_{31} \\ \mathbf{w}_{12} & \mathbf{w}_{22} & \mathbf{w}_{32} \end{bmatrix}=\begin{bmatrix} \mathbf{0.5} & \mathbf{0.3} & \mathbf{0.1} \\ \mathbf{0.3} & \mathbf{0.2} & \mathbf{0.1} \end{bmatrix} \tag{21}$$

Matricea interconexiunilor dintre neuronii ascunşi şi cei de ieşire Z, de ordinul $J \times L=3 \times 2$ este:

$$\mathbf{Z}=\begin{bmatrix} \mathbf{z}_{11} & \mathbf{z}_{21} \\ \mathbf{z}_{12} & \mathbf{z}_{22} \\ \mathbf{z}_{13} & \mathbf{z}_{23} \end{bmatrix}=\begin{bmatrix} \mathbf{0.1} & \mathbf{0.2} \\ \mathbf{0.3} & \mathbf{0.4} \\ \mathbf{0.5} & \mathbf{0.6} \end{bmatrix} \tag{22}$$

Scopul este ca RN să învețe să asocieze un set de modele de ieşire cu un set de modele de intrare.

Fie de exemplu intrarea $\mathbf{x}=\begin{bmatrix} \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \end{bmatrix}$ care trebuie să genereze ieşirea dorită $\mathbf{t}=\begin{bmatrix} \mathbf{0.1} \\ \mathbf{0.3} \\ \mathbf{0.7} \end{bmatrix}$.

Se consideră pentru simplitate viteza de învățare unitară şi pentru toți neuronii o aceeaşi funcție de activare, sigmoidă.

Într-o primă etapă informația se transmite de la intrare spre ieşire. Se calculează intrarea netă în neuronii ascunşi b cu relația:

$$\mathbf{b}=\mathbf{x}.\mathbf{W}=\begin{bmatrix} \mathbf{0.5+0.6+0.3} \\ \mathbf{0.3+0.4+0.3} \end{bmatrix}=\begin{bmatrix} \mathbf{1.4} \\ \mathbf{1.0} \end{bmatrix} \tag{24}$$

Aplicând funcția de activare sigmoidă, ieşirea neuronilor ascunşi va fi:

$$\mathrm{h}=f(\mathbf{b})=\frac{\mathbf{1}}{\mathbf{1+e^{-b}}}=\begin{bmatrix} \mathbf{0.8022} \\ \mathbf{0.7311} \end{bmatrix} \tag{25}$$

Această informație se transmite neuronilor stratului de ieşire, a căror activare devine:

$$\mathbf{a}=\mathbf{Z}\cdot\mathbf{h}=\begin{bmatrix} \mathbf{0.1\cdot 0.8022+0.2\cdot 0.7311} \\ \mathbf{0.3\cdot 0.8022+0.4\cdot 0.7311} \\ \mathbf{0.5\cdot 0.8022+0.6\cdot 0.7311} \end{bmatrix}=\begin{bmatrix} \mathbf{0.2264} \\ \mathbf{0.5331} \\ \mathbf{0.8397} \end{bmatrix} \tag{26}$$

Aplicându-se şi în aceşti neuroni funcția de activare sigmoidă se obține vectorul ieşirilor:

$$\mathbf{o} = \mathbf{f}(\mathbf{a}) = \begin{bmatrix} \mathbf{0.5564} \\ \mathbf{0.6302} \\ \mathbf{0.6984} \end{bmatrix} \tag{27}$$

Astfel a fost generat un prim răspuns al rețelei. Se calculează apoi diferența dintre răspunsul dorit t şi ieşirea curentă a neuronilor de ieşire:

$$\mathbf{e} = \mathbf{t} - \mathbf{o} = \begin{bmatrix} \mathbf{-0.4564} \\ \mathbf{-0.3302} \\ \mathbf{0.0016} \end{bmatrix} \tag{28}$$

Pentru calculul semnalelor de eroare cu relația (10) se determină derivata funcției de activare a neuronilor de ieşire:

$$\mathbf{f}\copyright(\mathbf{a}) = \mathbf{o} \otimes (\mathbf{1} - \mathbf{o}) = \begin{bmatrix} \mathbf{0.5564} \\ \mathbf{0.6302} \\ \mathbf{0.6984} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{0.4436} \\ \mathbf{0.3698} \\ \mathbf{0.3016} \end{bmatrix} = \begin{bmatrix} \mathbf{0.2468} \\ \mathbf{0.2330} \\ \mathbf{0.2106} \end{bmatrix} \tag{29}$$

Se evaluează erorile stratului de ieşire cu relația (10):

$$\delta_{\mathbf{iesire}} = \mathbf{f}'(\mathbf{a}) \otimes \mathbf{e} = \mathbf{o} \otimes (\mathbf{1} - \mathbf{o}) \otimes (\mathbf{t} - \mathbf{0}) = \begin{bmatrix} \mathbf{-0.1126} \\ \mathbf{-0.0770} \\ \mathbf{0.0003} \end{bmatrix} \tag{30}$$

Acum poate începe etapa a doua, de învățare.Erorile se transmit înapoi spre neuronii straturilor anterioare permițând modificarea ponderilor în conformitate cu regula de învățare. Se calculează ca o etapă intermediară o matrice R ai cărei termeni $r_{j,l}$ se obțin prin multiplicarea semnalului de eroare a neuronului de ieşire j cu ponderea conexiunii de la neuronul ascuns l la neuronul de ieşire j, cu relația: $r_{j,l} = \delta_j^{iesire} \times z_{j,l}$ (s-a notat cu δ_j^{iesire} componenta j a vectorului $\delta_{ieşire}$)

$$\mathbf{R} = \mathbf{Z} \otimes \left(\mathbf{1}^{\mathbf{T}} \otimes \delta_{\mathbf{iesire}}\right) = \mathbf{Z} \otimes \left[\delta_{\mathbf{iesire}} \delta_{\mathbf{iesire}}\right]$$

$$= \begin{bmatrix} \mathbf{.1} & \mathbf{.2} \\ \mathbf{.3} & \mathbf{.4} \\ \mathbf{.5} & \mathbf{.6} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{-0.1126} & \mathbf{-0.1126} \\ \mathbf{-0.0770} & \mathbf{-0.0770} \\ \mathbf{0.0003} & \mathbf{0.0003} \end{bmatrix} = \begin{bmatrix} \mathbf{-0.0113} & \mathbf{-0.0225} \\ \mathbf{-0.0231} & \mathbf{-0.0308} \\ \mathbf{0.0002} & \mathbf{0.0002} \end{bmatrix} \tag{31}$$

(1^T este un vector linie cu elemente de valoare 1)

Erorile fiind propagate se modifică ponderile conexiunilor dintre neuronii ascunşi şi cei de ieşire $\Delta\mathbf{Z}$ cu relația (11):

$$\mathbf{Z}[\mathbf{t}+\mathbf{1}] = \mathbf{Z} + \Delta\mathbf{Z} = \mathbf{Z} + \eta\delta_{\mathbf{iesire}}\mathbf{h}^{\mathbf{T}}$$

unde:

$$\eta\delta_{\mathbf{iesire}}\mathbf{h}^{\mathbf{T}} = \begin{bmatrix} \mathbf{-0.1126} \\ \mathbf{-0.0770} \\ \mathbf{0.0003} \end{bmatrix} \begin{bmatrix} \mathbf{0.8022} & \mathbf{0.7311} \end{bmatrix} = \begin{bmatrix} \mathbf{-0.0904} & \mathbf{-0.0823} \\ \mathbf{-0.0617} & \mathbf{-0.0563} \\ \mathbf{0.0003} & \mathbf{0.0002} \end{bmatrix}$$

Rezultă:

$$\mathbf{Z}[\mathbf{t}+\mathbf{1}] = \begin{bmatrix} \mathbf{.1} & \mathbf{.2} \\ \mathbf{.3} & \mathbf{.4} \\ \mathbf{.5} & \mathbf{.6} \end{bmatrix} + \begin{bmatrix} \mathbf{-0.0904} & \mathbf{-0.0823} \\ \mathbf{-0.0617} & \mathbf{-0.0563} \\ \mathbf{0.0003} & \mathbf{0.0002} \end{bmatrix} = \begin{bmatrix} \mathbf{0.0096} & \mathbf{0.1177} \\ \mathbf{0.2383} & \mathbf{0.3437} \\ \mathbf{0.5003} & \mathbf{0.6002} \end{bmatrix} \tag{32}$$

Se determină erorile neuronilor ascunşi. Notăm cu f suma ponderată a erorilor neuronilor de ieşire cu care este cuplat fiecare dintre cei doi neuroni ascunşi. Se obține:

$$\mathbf{f} = \mathbf{R}^T \mathbf{1} = \mathbf{Z}^T \delta_{\mathbf{iesire}} = \begin{bmatrix} -0.0342 \\ -0.0531 \end{bmatrix}$$

Cu relația (12) se calculează:

$$\delta_{\mathbf{ascuns}} = \mathbf{f}'(\mathbf{b}) \otimes \mathbf{f} = \mathbf{h} \otimes (\mathbf{1} \otimes \mathbf{h}) \otimes (\mathbf{Z}^T \delta_{\mathbf{iesire}}) = \begin{bmatrix} -0.0054 \\ -0.0104 \end{bmatrix}$$

Se modifică ponderile dintre intrări şi neuronii ascunşi cu relația (13):

$$\mathbf{W[t+1]} = \mathbf{W} + \Delta\mathbf{W} = \mathbf{W} + \eta \delta_{\text{ascuns}} \mathbf{x}^T = \begin{bmatrix} 0.5 & 0.3 & 0.1 \\ 0.3 & 0.2 & 0.1 \end{bmatrix} + \left(\begin{bmatrix} -0.0054 \\ -0.0104 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \right)$$

$$= \begin{bmatrix} .5 & .3 & .1 \\ .3 & .2 & .1 \end{bmatrix} + \begin{bmatrix} -0.0054 & -0.0108 & -0.0163 \\ -0.0104 & -0.0209 & -0.0313 \end{bmatrix} = \begin{bmatrix} .4946 & .2892 & .0837 \\ .2896 & .1791 & .0687 \end{bmatrix} \quad (33)$$

Viteza de învățare are o importanta deosebită în evoluția procesării. O viteza de învățare mare, asigura o convergență rapidă, dar poate determina oscilații ale rețelei. O viteză mică are ca efect mărirea timpului de procesare şi poate duce la împotmolirea în minime locale cu o probabilitate mai mare.

Prin introducerea unei relații între schimbarea curentă a ponderii şi modificarea ei anterioarî , se pot realiza paşi mai mari (convergență mai rapidă), fără a utiliza viteze mari de învățare:

$$\Delta_p w_{ij}(n+1) = \eta . \delta_{pj} . o_{pi} + \alpha . \Delta_p w_{ij}(n)$$

unde: α este o constantă, numită momentum, ce determină efectul ponderilor anterioare asupra ponderii curente.

Aceasta relație practic realizează o filtrare a variațiilor mari ale erorilor în spațiul pondere.

Lucrarea nr.3

1. Scopul lucrării

Scopul lucrării este :

- testarea unor aplicații implementabile cu neuroni adaptivi liniari ce utilizează regula Widrow-Hoff: filtrarea adaptivă, aproximarea de funcție, identificarea de sistem;
- studiul diferitelor variante ale algoritmului retropropagării erorii şi a influenței parametrilor asupra convergenței algoritmului ;
- testarea unor aplicații ce utilizează RN multistrat antrenate cu algoritmul retropropagării erorii: filtrarea adaptivă, controlul unui sistem, identificarea de sistem neliniar;

2. Desfăşurarea lucrării

Se lansează programul MATLAB. Se selectează din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks.

2.1 Selectați "Adaptive linear prediction". Aplicația realizează predicția unui semnal pe baza ultimelor cinci eşantioane, utilizând regula Widrow Hoff şi o viteză de învățare η=0.1.Se vizualizează semnalul de intrare, semnalul de ieşire curent marcat cu albastru şi diferența dintre ele, eroarea. Se observă că deşi semnalul de intrare are două frecvența diferite, filtrul se adaptează la schimbările din semnal pentru a prezice cea mai bună estimare (în medie pătratică a semnalului de intrare).

2.2 Selectați "Linear prediction". Aplicația realizează o aproximare de funcție cu ajutorul unui neuron linear, din ultimele cinci eşantioane ale semnalului de intrare. Funcția aproximată este sin(4.π.t).Vizualizați semnalul de intrare, de ieşire şi eroarea.

2.3 Selectați "Linear system identification". Aplicația implementată este de identificare de sistem, utilizând un neuroni adaptivi liniari. Sistemul necunoscut generează sin((sint).10t). Intrările în RN. Sunt ultimele trei eşantioane ale semnalului de intrare. Răspunsul dorit este ieşirea sistemului necunoscut. Vizualizați intrarea, ieşirea şi eroarea RN. Ce concluzie puteți trage?

2.4 Î ncărcați programul "Too large a learning rate ".
Cu ajutorul acestui program se poate constata că în cazul în care viteza de învățare este prea mare rețeaua oscilează şi nu poate găsi soluția optimă.

2.5 Rulați programul "Generalization ". Modificați numărul neuronilor din stratul ascuns şi gradul de dificultate al funcției aproximate de RN. Observați şi notați diferitele situații studiate şi performanțele obținute de RN.

2.6 Selectați "Steepest descent backpropagation ". Rețeaua din figură implementează o aproximare de funcție. Există trei posibilități de selectare a parametrilor rețelei ce se pot antrena cu algoritmul BKP. Selectați cu mouse-ul punctul de plecare al antrenării pe graful curbelor de nivel ale erorii.(Curbele de nivel sunt desenate pe scara nuanțelor de gri, negrul corespunzând minimului. Punctul marcat cu roşu reprezintă soluția optimă. Alegeți diferite puncte de plecare şi observați comportarea rețelei în

preajma minimelor locale. Deschideți fișierul nnd12sd1 pentru a observa modul de implementare, intrarea şi răspunsul dorit.

2.7 Încărcați programul "Momentum backpropagation ". Deschideți fișierul nnd12mo pentru a observa modul de implementare, intrarea şi răspunsul dorit. Testați influența vitezei de învățare şi a momentumului asupra modului de funcționare al RN.Pentru aceasta alegeți viteza de învățare 0,1; 0,5; 0,9; 1; 4; 10. Pentru fiecare dintre aceste valori alegeți constanta momentum de diferite valori 0,1, 0,5; 0,9;1.

2.8 Selectați "Conjugate gradient backpropagation ".

Algoritmul conjugate gradient backpropagation este o variantă a algoritmului BKP care utilizează o viteză de învățare variabilă pentru a mări convergența.

$$w(t+1) = w(t) + \eta(t).p(t)$$

- unde p(t) este direcția de variație ponderii

$$p(0) = -g(0)$$

- unde g(0) este gradientul erorii în momentul t=0

$$p(t+1) = -g(t+1) + \beta(t).p(t)$$

Există mai multe metode de obținere a valorii lui β

1. Formula Fletcher-Reeves

$$\beta(t) = \frac{g^T(t+1).g(t+1)}{g^T(t).g(t)}$$

2. Formula Patak – Ribiere

$$\beta(t) = \frac{g^T(t+1).[g(t+1) - g(t)}{g^T(t).g(t)}$$

Selectați cu mouse-ul punctul de plecare al antrenării pe graful curbelor de nivel ale erorii. Punctul marcat cu roşu reprezintă soluția optimă. Alegeți diferite puncte de plecare şi observați comportarea rețelei. Care este performanța rețelei comparativ cu algoritmul standard BKP?

2.9 Selectați "Adaptive linear system identification". Aplicația realizează predicția unui semnal pe baza ultimelor două eşantioane, utilizând regula retropropagării erorii şi o viteză de învățare η=0.5.

2.10 Încărcați programul "Model reference control ". O rețea neuronală antrenată BKP realizează controlul unui pendul inversat. Pendulul inversat este un pendul vertical prins într-o articulație pe un cărucior mobil. Pendulul tinde să cadă la orizontală datorită forței gravitaționale. Un motor controlat de RN deplasează căruciorul pentru a menține în poziție verticală pendulul.

2.11 Încărcați programul "Nonlinear control system identification ". O rețea neuronală este antrenată BKP pentru a realiza identificarea unui sistem neliniar necunoscut, în acest caz un pendul inversat.

Lucrarea numărul 4

1. Rețeaua Hopfield . Noțiuni teoretice

Rețeaua Hopfield este o rețea neuronală cu reacție, în care starea fiecărui neuron la un moment dat depinde şi de ieşirile tuturor celorlalți neuroni la momentul anterior.

Structura ei este prezentată în Fig.1. Are un singur strat de neuroni, complet interconectați, adică fiecare neuron este conectat cu toți ceilalți. Ieşirile neuronilor sunt binare 0 şi 1, sau bipolare -1 şi +1 . Matricea ponderilor este simetrică $w_{ij}=w_{ji}$.Autoreacția est nulă $w_{ii}=0$ (fenomen observat de alfel şi în neuronii biologici). Aceasta îmbunătățeşte performanțele obținute cu modele bipolare.

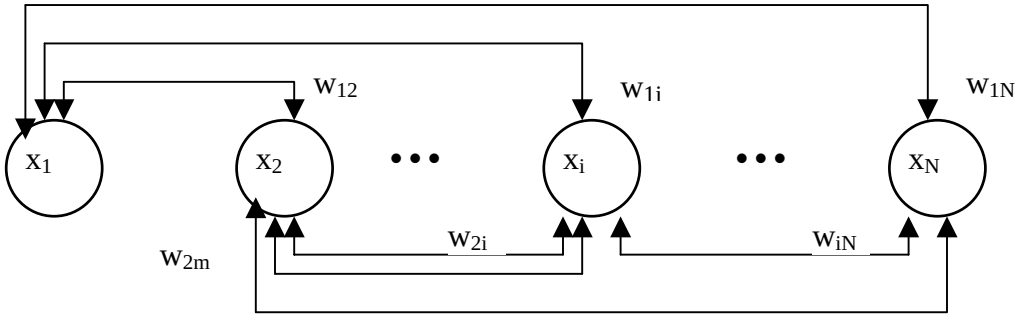


Fig.1 Rețeaua neuronală Hopfield

Există două modalități de implementare a unei rețele neuronale Hopfield determinate de aplicația în care este utilizată: ca sistem discret pentru o memorie asociativă şi ca sistem continuu pentru o problemă de optimizare.

1.1 Rețeaua neuronală Hopfield ca memorie asociativă

În funcționarea unei rețele neuronale ca memorie asociativă există două faze:

- de înmagazinare a informațiilor;
- de regăsire a informației dorite din memorie (recall sau retrieval);

1. **Înmagazinarea informațiilor** Fie un set de p modele bipolare X_1, X_2, X_3, … X_P, de dimensiune N, pe care dorim să le memorăm.Aceste modele se numesc şi modele prototip sau modele fundamentale. Ponderile interconexiunilor se determină cu o generalizare a regulii lui Hebb, regula bipolară hebbiană (outer product rule):

$$w_{ij} = \sum_{k=1}^{p} X_{ki} X_{kj} \tag{1}$$

Ținând cont şi de faptul că autoreacția trebuie să fie nulă $w_{ii}=0$, relația (5.1) se poate scrie sub formă matriceală:

$$W = \sum_{k=1}^{P} X_k^T . X_k - p.I \tag{2}$$

unde :

- I este matricea unitate de dimensiune N x N ;
- p este numărul modelelor memorate;
- X_k este model prototip $X_k = [X_{k1}\ X_{k2}\ X_{k3} \ldots X_{kN}]$;
- termenul p.I a fost introdus pentru anularea ponderilor de autoreacție;

De exemplu pentru a coda modelul bipolar [1,-1] într-o rețea cu doi neuroni se obține matricea W_1 :

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix}.\begin{bmatrix} 1 & -1 \end{bmatrix} - 1.\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \quad (3)$$

Pentru a simplifica formalismul matematic al regăsirii informației se poate utiliza şi un termen de proporționalitate $\frac{1}{N}$ în relația (5.2):

$$W = \frac{1}{N}\sum_{k=1}^{P} X_k^T . X_k - \frac{p.I}{N} \quad (4)$$

Regăsirea informației dorite din memorie

Un model de intrare bipolar x =[x_1 x_2 x_3 … x_N] , N dimensional, este impus ca stare a RN Hopfield . Tipic el este o versiune incompletă sau afectată de zgomot al unui model memorat. Actualizarea stării neuronilor este asincronă, un singur neuron îşi schimbă starea la un moment dat în conformitate cu funcția de activare. Intrarea netă a acestui neuron depinde de ieşirile tuturor celorlalți:

$$net_i[k+1] = \sum_{j=1}^{N} w_{ji} . x_j[k] + I_i \quad (5)$$

unde:

- x_j este starea de activare a neuronului j;
- I_i este o intrare constantă, numită curent de polarizare ;
- N este numărul neuronilor rețelei ;

Se aplică apoi funcția de activare care poate fi o funcție bipolară cu prag , dată de relația :

$$f(net_i[k+1]) = \begin{cases} 1 & \text{dacă} \quad net_i[k] > \theta_i \\ f(net_i[k]) & \text{dacă} \quad net_i[k] = \theta_i \\ 0 & \text{dacă} \quad net_i[k] < \theta_i \end{cases} \quad (6)$$

sau funcția signum:

$$f(net_i[k+1] = sign(net_i[k])) \quad (7)$$

La o nouă iterație un alt neuron îşi schimbă starea în conformitate cu regula de actualizare.Se determină pentru acesta intrarea netă cu relația (5) şi apoi noua stare cu relația (6). În final RN ajunge într-o stare invariantă în timp care satisface condiția de stabilitate, adică într-unul dintre atractori.

Observații

1.Rețeaua Hopfield este stabilă.

Fie X= [1,1] aplicat la intrarea rețelei anterior introduse (care a memorat modelul [1 –1]).

Printr-o actualizare asincronă un singur neuron îşi schimbă starea. Fie acesta neuronul unu. Doar prima coloană a matricii pondere este implicată (ponderile neuronului unu către neuronul doi). Starea neuronului 2 rămâne 1.

$$X.W_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}.\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

Aplicând din nou rețelei vectorul [-1,1] prin W şi actualizând neuronul 2 se obține acelaşi vector [-1,1] . Deci rețeaua neuronală Hopfield este stabilă.

2. Actualizarea asincronă permite interpretarea informației procesate de rețeaua Hopfield ca un proces aleator. Pentru actualizarea neuronilor uneori se stabileşte o schemă de actualizare astfel încât în medie fiecare neuron să fie actualizat de acelaşi număr de ori. Actualizarea asincronă după o lege probabilistică permite caracterizarea statistică a rețelei Hopfield (ancorarea ei în fizica statistică). Evoluția stărilor rețelei nu este în mod unic definită de o anume stare inițială, în spațiul $\{0,1\}^m$, ea depinde de schema de actualizare.

3. Contribuția cea mai importantă a lui Hopfield este introduction unei funcții de energie în analiza comportamentului RN. Aceasta permite abordarea RN într-o manieră similară sistemelor fizice, marcându-le evoluția. Fie funcția ce caracterizează rețeaua Hopfield :

$$E(x) = -\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}.x_i.x_j + 2\sum_{i=1}^{n}\theta_i.x_i \tag{8}$$

Se poate demonstra că de fiecare dată când un neuron îşi schimbă starea, E(x) descreşte :

$$\Delta E = E(x^n) - E(x^v) = -\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}.x_i^n.x_j^n + 2\sum_{i=1}^{n}\theta_i.x_i^n + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}.x_i^v.x_j^v - 2\sum_{i=1}^{n}\theta_i.x_i^v =$$

$$-2.x_k^n.\sum_{j=1,j\neq k}^{n} w_{ij}.x_j^n + 2.\theta_k.x_k^n + 2.x_k^v.\sum_{j=1,j\neq k}^{n} w_{ij}.x_j^v - 2.\theta_k.x_k^v \tag{9}$$

unde $\mathbf{x_k^n}$ ($\mathbf{x_k^v}$) este starea nouă (veche) a neuronului k, singurul neuron care îşi schimbă starea în iterația curentă. Toți ceilați neuroni i≠k rămân în aceeaşi stare $x_k^n = x_k^v$ astfel încât vor exista termeni care se anulează.

Există două situații posibile:

1.când starea neuronului k a fost $x_k^v = -1$ şi devine $x_k^n = 1$

Pentru că $\sum_{j=1,j\neq k}^{n} w_{ij}.x_j^n = \sum_{j=1,j\neq k}^{n} w_{ij}.x_j^v$ se poate rescrie relația (5.6) :

$$\Delta E = 2(x_k^v - x_k^n).(\sum_{j=1,j\neq k}^{n} w_{ij}.x_j^v - \theta_k) \tag{10}$$

Semnul celui de-al doilea factor al produsului din relația (5.7) este plus pentru că starea neuronului k devine +1 dacă este îndeplinită condiția $(\sum_{j=1,j\neq k}^{n} w_{ij}.x_j^v - \theta_k) > 0$. Diferența $(x_k^v - x_k^n) < 0$ este negativă, deci ΔE este şi ea negativă.

2. când starea neuronului k a fost quand $\mathbf{x_k^v = 1}$ şi devine $\mathbf{x_k^n = -1}$.

Semnul celui de-al doilea factor al produsului din relația (6) este minus pentru că starea neuronului k devine –1 dacă este îndeplinită condiția $(\sum_{j=1,j\neq k}^{n} w_{ij}.x_j^v - \theta_k) < 0$. Diferența $(x_k^v - x_k^n) > 0$ este pozitivă, deci ΔE este negativă.

Observații

1. Aplicațiile posibile ale rețelei Hopfield sunt de memorie asociativă şi de optimizare.
2. **Memoriile asociative** implementează o transformare între o mulțime de modele aparținând spațiului de intrare şi o mulțime de modele aparținând spațiului de ieşire. Ori de câte ori la intrare se aplică un model particular , la ieşire se obține modelul asociat acestuia.

Un caz particular este **autoasocierea**, când un model este asociat cu el însuşi. Scopul procesării este completarea de model sau eliminarea zgomotului, asociindu-se modelul de intrare , incomplet, sau afectat de zgomot cu el însuşi la ieşire. Rețeaua Hopfield est o RN auto-asociativă .

Capacitatea de memorare cea mai mare dintre toate memoriile asociative cunoscute**. Capacitatea de memorare** este numărul modelelor distincte pe care sistemul le poate învăța cu precizie şi rememora, deci coda şi decoda.

$$C = \frac{N}{\log_2 N} \tag{11}$$

Pentru determinarea capacității de memorare se poate folosi şi o relație empirică aproximativă C=0,15.m.

3. **Optimizarea** este o tehnică pentru rezolvarea unor probleme ce implică minimizarea unei unei funcții de cost asociate în raport cu nişte constrângeri impuse . Funcția de cost este funcția de energie asociată RN. Astfel încât pentru rețeua Hopfield optimizarea este o aplicație directă. Prin minimizare RN converge către o stare stabilă producând o soluție optimă (sau lângă optim).

O problemă celebră de optimizare este problema comis voiajorului. Acesta trebuie să viziteze N oraşe , trecând o singură dată prin fiecare oraş. Comis voiajorul cunoaşte distanța dintre oraşele pe care trebuie să le viziteze, problema este de a determina traseul optim (în ce ordine să treacă prin fiecare oraş) , astfel încât distanța pe care o parcurge să fie minimă.

Aceasta este o problemă ce necesită explorarea a unui număr de N! =1x2x3x….xN variante distincte. Numărul este relativ comod pentru valori mici ale lui N , dar creşte exponențial cu valoarea lui N. se spune că este o problemă de tip NP- complet .

Exemplul 1

Construiți o rețea Hopfield care să memoreze modelele X_1=[1 -1] şi X_2=[-1 1], printr-o codare bipolară hebbiană.

a) Determinați matricea ponderilor. De ce este suficientă memorarea unui singur model?

b) Determinați stările succesive ale rețelei până în starea finală pentru toate intrările posibile. Ce observați?

c) Se ştie că pentru cazul particular al pragurilor şi ponderilor nule funcția de energie a rețelei este : $\mathbf{E = -\frac{1}{2} X.W.X^T}$. Determinați evoluția rețelei pentru intrările de la punctul b. Ce puteți spune despre stabilitatea rețelei ?

Soluție

a) Rețeaua are 2 neuroni. Este suficientă memorarea unui singur model pentru că un artefact al codării bipolare hebbiene este memorarea modelelor complementare. Matricea ponderilor este :

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix}.\begin{bmatrix} 1 & -1 \end{bmatrix} - 1.\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \tag{12}$$

b)Intrările posibile sunt : X=[1 1], [-1 -1], [1 -1] et [-1 1].

Fie starea inițială X_1= [1,1]. Fie neuronul 1 cel care işi schimbă starea. Starea neuronului 2 rămâne neschimbată.Noua stare va fi :

$$X_1.W_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}.\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 1 \end{bmatrix} \tag{14}$$

Se aplică funcția de activare, funcția signum. Stare rețelei rămâne aceeaşi [-1,1]. După actualizarea neuronului 2 starea rețelei va fi tot [-1,1], care este un punct de echilibru.

Dacă se aplică la intrare X_2 = [-1 -1] se obține prin actualizarea primului neuron:

$$X_2.W_1 = \begin{bmatrix} -1 & -1 \end{bmatrix}.\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 \end{bmatrix} \tag{15}$$

$$[\text{sign}(1) \quad 1] = [1 \quad -1] \tag{16}$$

Prin actualizarea neuronului 2:

$$\begin{bmatrix} 1 & -1 \end{bmatrix}.\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 \end{bmatrix} \tag{17}$$

$$[1 \quad \text{sign}(-1)] = [1 \quad -1] \tag{18}$$

Rețeaua va ajunge într-un alt punct de echilibru [1 -1]. Pentru celelalte stări posibile, care reprezintă modelele memorate [1 -1] şi [-1 1], rețeaua nu-şi va schimba starea . În concluzie, indiferent de starea inițială RN, va evolua înspre unul dintre punctele sale de echilibru

c) Pentru starea inițială X_1= [1 1] energia RN este:

$$E_{1i} = -\frac{1}{2}[1 \quad 1].\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.\begin{bmatrix} 1 \\ 1 \end{bmatrix} = -\frac{1}{2}[-1 \quad -1]\begin{bmatrix} 1 \\ 1 \end{bmatrix} = -\frac{1}{2}(-2) = 1 \tag{19}$$

A doua stare , care este şi cea finală are energia următoare:

$$E_{1f} = -\frac{1}{2}[-1 \quad 1].\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.\begin{bmatrix} -1 \\ 1 \end{bmatrix} = -\frac{1}{2}[-1 \quad 1]\begin{bmatrix} -1 \\ 1 \end{bmatrix} = -\frac{1}{2}(2) = -1 \tag{20}$$

Pentru starea inițială X_2= [-1 -1] energia rețelei este:

$$E_{2i} = -\frac{1}{2}[-1 \quad 1].\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.\begin{bmatrix} -1 \\ -1 \end{bmatrix} = -\frac{1}{2}[-1 \quad 1]\begin{bmatrix} -1 \\ -1 \end{bmatrix} = -\frac{1}{2}(-2) = 1 \tag{21}$$

Starea finală are energia :

$$E_{2f} = -\frac{1}{2}[1 \quad -1].\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.\begin{bmatrix} 1 \\ -1 \end{bmatrix} = -\frac{1}{2}[-1 \quad 1]\begin{bmatrix} 1 \\ -1 \end{bmatrix} = -\frac{1}{2}(2) = -1 \tag{22}$$

În mod similar se poate calcula energia şi pentru celelalte intrări care sunt puncte de echilibru ale rețelei. Pentru aceste cazuri energia rămâne la o aceeaşi valoare –1. Deci în toate cazurile energia are o evoluție descrescătoare astfel încât satisface condiția de stabilitate

Exemplul 2

Construiți o rețea Hopfield cu patru neuroni care să memoreze modelul X_1=[1 1 1 -1], printr-o codare bipolară hebbiană. Fie funcția de activare funcția signum, pragurile şi curenții de polarizare nuli.

a) Determinați matricea ponderilor.
b) Fie starea inițială una dintre următoarele:

X^0=[1 1 1 1] ;
X^0=[1 1 -1 -1] ;
X^0=[1 -1 1 -1] ;
X^0=[-1 1 1 -1] ;

Actualizați neuronii în ordinea 1, 2, 3, et 4. Determinați stările succesive ale rețelei până în starea finală pentru toate intrările posibile .
c) Se ştie că pentru cazul particular al pragurilor şi ponderilor nule funcția de energie a rețelei este : $\mathbf{E = -\frac{1}{2} X.W.X^T}$. Determinați evoluția rețelei pentru intrările de la punctul b. Ce puteți spune despre stabilitatea rețelei ?

Soluție
Structura rețelei este cea din Fig.2. Matricea ponderilor este dată de relația:

$$W = X_1^T.X_1 - U \tag{23}$$

$$W = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}.\begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix} - 1.\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

$$W = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

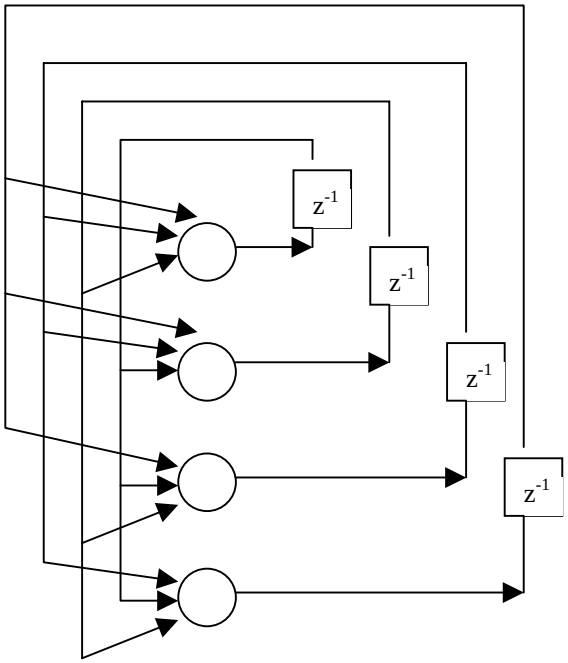


Fig.2 Structura rețelei din exemplul 2

b) Fie starea inițială: X=[1 1 1 1]. În mod secvențial, conform schemei de actualizare câte un neuron îşi actualizează starea:

$$X^0.W = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & -1\\ -1 & -1 & -1 & 0\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix} \quad (25)$$

Noua stare a neuronului 1 este $\begin{bmatrix}f(1) & 1 & 1 & 1\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}.$ (26)

Într-un mod similar se actualizează neuronul 2, $\begin{bmatrix}1 & f(1) & 1 & 1\end{bmatrix}$, apoi neuronul 3 , a cărui stare devine $\begin{bmatrix}1 & 1 & f(1) & 1\end{bmatrix}$. Neuronii 2 şi 3 rămân în aceeaşi stare 1. Doar neuronul 4 îşi schimbă starea :

$$\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & -1\\ -1 & -1 & -1 & 0\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & -1\end{bmatrix} \quad (27)$$

Stare finală va fi $\begin{bmatrix}1 & 1 & 1 & f(-1)\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & -1\end{bmatrix}$, care este un atractor al rețelei .
Dacă starea inițială este X=[1 -1 -1 1], după actualizarea primului neuron ea devine:

$$X.W = \begin{bmatrix}1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & -1\\ -1 & -1 & -1 & 0\end{bmatrix} = \begin{bmatrix}-3 & -1 & -1 & 1\end{bmatrix} \quad (28)$$

$$\begin{bmatrix}sgn(-3) & -1 & -1 & 1\end{bmatrix} = \begin{bmatrix}-1 & -1 & -1 & 1\end{bmatrix} \quad (29)$$

Se actualizează al doilea neuron:

$$X.W = \begin{bmatrix}-1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & -1\\ -1 & -1 & -1 & 0\end{bmatrix} = \begin{bmatrix}-1 & -3 & -1 & 1\end{bmatrix} \quad (30)$$

$$\begin{bmatrix}-1 & sgn(-3) & -1 & 1\end{bmatrix} = \begin{bmatrix}-1 & -1 & -1 & 1\end{bmatrix} \quad (31)$$

După actualizarea celui de-al patrulea neuron starea rețelei devine:

$$\begin{bmatrix}-1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & -1\\ -1 & -1 & -1 & 0\end{bmatrix} = \begin{bmatrix}-1 & -1 & -3 & 1\end{bmatrix} \quad (32)$$

$$\begin{bmatrix}-1 & -1 & sgn(-3) & 1\end{bmatrix} = \begin{bmatrix}-1 & -1 & -1 & 1\end{bmatrix} \quad (33)$$

Starea finală va fi:

$$\begin{bmatrix}-1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & -1\\ -1 & -1 & -1 & 0\end{bmatrix}=\begin{bmatrix}-1 & -1 & -1 & 3\end{bmatrix} \quad (34)$$

$$\begin{bmatrix}-1 & -1 & -1\,\mathrm{sgn}(3)\end{bmatrix}=\begin{bmatrix}-1 & -1 & -1 & 1\end{bmatrix} \quad (35)$$

Modelul X=[-1 –1 –1 1] este un model nedorit, dar este un atractor al rețelei memorat printr-o codare bipolară hebbiană .

d) Pentru starea inițială X= [1 –1 –1 1] energia rețelei este:

$$E_i=-\frac{1}{2}\begin{bmatrix}1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & -1\\ -1 & -1 & -1 & 0\end{bmatrix}\begin{bmatrix}1\\ -1\\ -1\\ 1\end{bmatrix}=-\frac{1}{2}\begin{bmatrix}-3 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\\ -1\\ 1\end{bmatrix}=0 \quad (36)$$

Pentru starea succesivă, care este şi cea finală energia este:

$$E_f=-\frac{1}{2}\begin{bmatrix}-1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & -1\\ -1 & -1 & -1 & 0\end{bmatrix}\begin{bmatrix}-1\\ -1\\ -1\\ 1\end{bmatrix}=-\frac{1}{2}\begin{bmatrix}-3 & -3 & -3 & 1\end{bmatrix}\begin{bmatrix}-1\\ -1\\ -1\\ 1\end{bmatrix}=-6 \quad (37)$$

Asfel încât evoluția energiei rețelei este descrescătoare, deci satisface condiția de stabilitate..

2.Scopul lucrării

Scopul lucrării este studiul rețelei Hopfield:

- Implementarea unei rețele care să memoreze anumite modele ;
- Verificarea posibilității de apel din memorie a unui anumit model memorat atunci când la intrare se aplică modelul eronat ;
- Studiul memorării de modele nedorite ;

3. Desfăşurarea lucrării

Se lansează programul MATLAB. Se selectează din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks.

3.1 Încărcați programul “Hopfield two neuron design ”. Acesta permite construirea unei rețele Hopfield cu doi neuroni cu două puncte de echilibru, [+1 -1] şi [-1 +1].

Verificați cu “help NEWHOP ” modul de implementare a RN.

Funcția de activare utilizată este SATLINS. Extrageți expresia analitică a funcției.

Cei doi atractori ai rețelei sunt reprezentați în planul stărilor de intrare posibile prin puncte roşii. Rulați programul pentru diferite intrări posibile. Ce constatați?

3.2 Selectați programul “Hopfield unstable equilibria ”.

Rețeaua Hopfield anterior construită are şi un punct de echilibru nedorit. Verificați că toate stările inițiale aflate la egală distanță (notați aceste stări) de cei doi atractori doriți vor conduce în punctul de echilibru nedorit [0 0].

3.3 Încărcați programul “Hopfield three neuron design ”. Acest program implementează o rețea Hopfield cu trei neuroni cu două puncte de echilibru dorite, [+1 -1-1] şi [+1 +1 -1], marcate în spațiul stărilor posibile cu puncte roții. Testați rețeaua pentru diferite modele de intrare. Ce puteți constata?

Aplicați la intrare unul dintre modelele [+1 0 -1] şi [+1 0 -1]. Rețeaua va evolua înspre centrul spațiului stărilor. Ce reprezintă acest model?

3.4 Implementați o rețea Hopfield folosind o codare bipolară hebbiană şi funcția de activare SATLINS.

Considerați modelul de intrare [-1 -1 +1] şi determinați evoluția rețelei. Ce reprezintă acest model pentru RN implementată ? Verificați evoluția rețelei şi pentru [0.8 0.5 -1].

Comparați rezultatele obținute cu cele ale rețelei implementate soft. Ce concluzii puteți trage?

3.5 Încărcați programul “ Hopfield spurious stable points”. Programul implementează o rețea Hopfield cu cinci neuroni care memorează patru modele dorite. Notați aceste modele. Verificați evoluția rețelei pentru diferite modele de intrare. Ce atractori nedoriți ați mai găsit?

Lucrarea nr.5

1. Noțiuni teoretice

1.1 Rețele neuronale cu învățare competitivă

Arhitectura unei rețele neuronale cu învățare competitiva este prezentata in fig.1. Învățarea competitiva este o procedura cu sau fără control, aplicabilă în timp real.

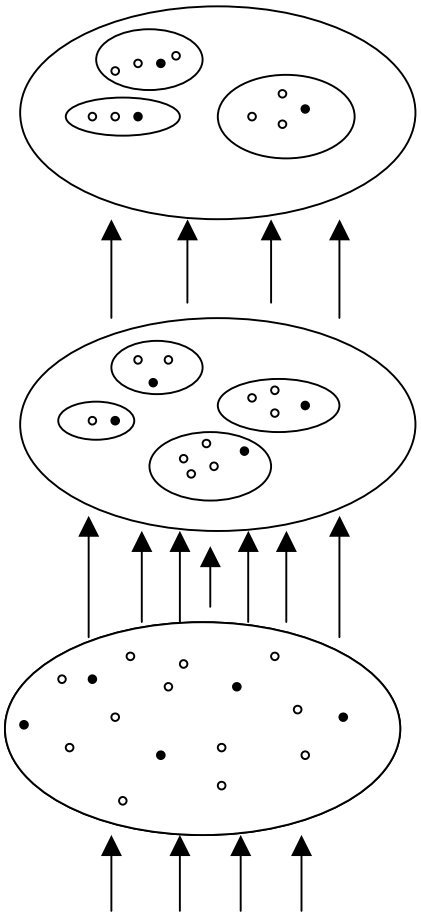


Fig.1 Structura generală a unei rețele competitive

Unitățile unui strat se împart în grupări (bazine), care nu se suprapun. Numărul de unități poate varia de la o grupare la alta. Fiecare unitate primește conexiuni excitatorii de la toate unitățile din stratul anterior şi transmite conexiuni inhibitorii către toate unitățile grupării din care face parte. Unitățile unei grupări intra în competiție pentru a răspunde unui model de intrare dat. Unitatea care are intrarea cea mai mare tinde către valoarea de activare maxima, în timp ce toate celelalte tind către valoarea minima. Învățarea are loc prin modificarea ponderilor, conform unei strategii cunoscute în literatura sub numele
de "câştigătorul ia totul ".In final o singura unitate va fi activa într-o grupare. Astfel are loc specializarea unităților asupra seturilor de modele de intrare similare. Fiecare unitate devine un detector de caracteristica sau clasificator de model.

In decursul anilor, mai mulți cercetători au elaborat modele competitive, având la baza diferite reguli de învatare: von der Malsburg (1973), Grossberg (1972, 1976), Fukushima (1975), Bienenstock , Cooper şi Munro (1980), Rumelhart şi Ziepser (1985).

1.2 Caracteristici

1. Unitățile unui strat sunt asemănătoare, cu excepția unui set de caracteristici care le face sa răspundă în mod diferit la aplicarea unui model de intrare. Dacă exista doua unități intr-o grupare, fiecare este activă pentru valorile opuse ale unei caracteristici binare. Dacă exista trei unități, fiecare este activa la una dintre valorile unei caracteristici ternare. Astfel o grupare poate fi considerata ca formatoare a unei caracteristici M -are, în care fiecare model este clasificat ca având una dintre cele M valori posibile ale acestei caracteristici.

2.Dacă exista o structura în modelele de stimuli unitatea va fragmenta modelele în caracteristici structural relevante. Dacă stimulii sunt puternic structurați, clasificatoarele vor fi stabile. Dacă stimulii nu au o structura clară, când o unitate , când alta, va câstiga competiția, generând instabilitate.

3. Clasificarea depinde atât de valorile inițiale ale ponderilor , cât şi de secvența curenta de stimuli. Un stimul S_j consta dintr-un model binar de 1 şi 0.

4. Pentru o unitate u_j suma ponderilor liniilor de intrare este constantă

$$\sum_i w_{ij} = 1$$

O unitate învață prin trecerea ponderilor de la liniile active la cele inactive. Doar unitățile care vor câștigă competiția învață.

1.3 Procedura de învățare competitivă conține,în esență, următorii paşi:

1. Se inițializează vectorii pondere**:**

$$w_i(0) = x(i); \quad i = 1,2,\ldots n$$

Pentru eşantionul de intrare aleator x(t) se găseşte cel mai apropiat vector pondere, $w_j(t)$, care va fi declarat câştigător:

$$\left\| w_j(t) - x(t) \right\| = \min_i \left\| w_i(t) - x(t) \right\|$$

unde $|x| = (x_1)^2 + (x_2)^2 + \ldots (x_n)^2$ este norma euclidiană a lui x

2.Se actualizează ponderea nodului câştigător conform relației:

- pentru **învățarea competitivă nesupravegheată:**

$$w_j(t+1) = w_j(t) + c_t[x(t) - w_j(t)]$$

unde c_t defineşte o secvență de coeficienți de învățare descrescătoare:

$$c_t = 0.1[1 - \frac{t}{10000}]$$

pentru 10000 de eşantioane de învățare ale lui x(t)

- pentru **învățarea competitivă supravegheată:**

$$w_j(t+1) = w_j(t) + c_t . r_j(x(t)).[x(t) - w_j(t)]$$

unde $-r_j(x(t))$= 1 dacă clasificarea făcută a fost corectă

$-r_j(x(t))$= -1 dacă clasificarea făcută nu a fost corectă

- pentru **învățarea competitivă diferențială:**

$$w_j(t+1) = w_j(t) + c_t . \Delta f_j(y_j)).[x(t) - w_j(t)]$$

unde $\Delta f_j(y_j(t))$ reprezintă variația în timp a ieşirii neuronului j câştigător din stratul 2

$$\Delta f_j(y_j(t)) = f_j(y_j(t+1)) - f_j(y_j(t))$$

Practic se utilizează doar semnul diferenței.

4. Pentru neuronii necâştigători ponderile rămân neschimbate**.**

1.4 O interpretare geometrică

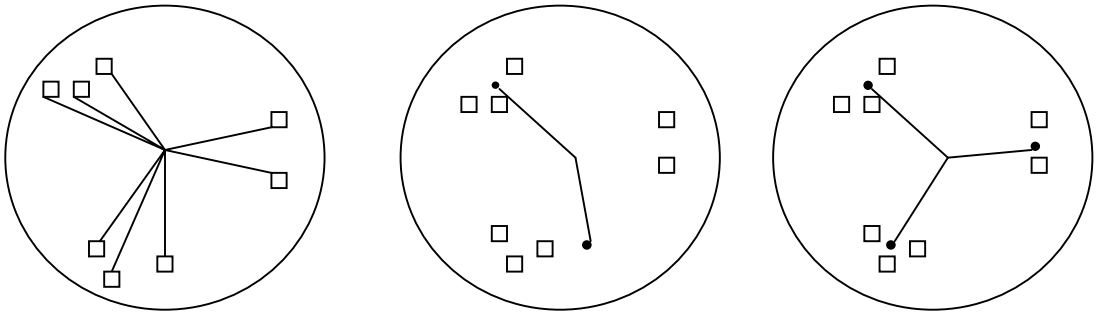
Dacă toate modelele S_k au acelaşi număr de linii active de intrare (unități active în stratul inferior) modele stimul au o aceeaşi lungime. Fiecare model de intrare poate fi văzut ca un punct pe sfera N dimensională. Modelele de intrare similare vor fi reprezentate prinn puncte apropiate pe sferă.

Şi vectorii pondere au o lungime fixă (deoarece suma lor este 1) deci şi ei se vor reprezenta printr-un punct pe sfera N dimensională. La prezentarea unui model la intrarea rețelei, neuronul care are cea mai mare stare de activare este cel al cărui vector pondere este mai aproape de stimul. El va fi declarat câştigător. Ori de câte ori un neuron câştigă competiția vectorul său pondere se va roti din locația curentă înspre locația modelului de intrare.

În Fig.2a sunt reprezentați 8 stimuli, grupați în trei regiuni pe sferă.

Fig.2b. Când se aplică un model de intrare va câştiga competiția neuronul al cărui vector pondere este mai apropiat. Vectorul pondere se va deplasa spre gruparea cea mai apropiată de modele.

Fig3c Vectorii pondere ai celor trei neuroni din figură au migrat înspre centrul grupării mai apropiate.



2a 2b 2c

Fig.2 O interpretare geometrică a unei rețele competitive

1.5 Rețele cu autoorganizare "Kohonen"

Pornind de la constatarea ca pentru modelarea interna a structurilor de date creierul utilizează transformări (mapping) spațiale, Kohonen a realizat o rețea detectoare de caracteristici. Rețeaua Kohonen este capabila sa identifice caracteristicile comune ale modelelor de intrare (dacă acestea există), şi sa realizeze o clasificare a acestora. Date multidimensionale pot fi reprezentate intr-un spațiu cu dimensiune mult redusa, uzual cu doua dimensiuni. Funcționarea cortexului uman este asemănătoare.

O rețea tipica Kohonen are un singur strat de neuroni adaptivi. Fiecare intrare se conectează la toți neuronii din rețea. Fiecare neuron interacționează cu vecinii săi.

Printr-un proces ciclic se compară vectorii de intrare cu vectorii "înmagazinați" (prin ponderile conexiunilor) în fiecare dintre nodurile de ieşire.

Se evalueaza distanța:

$$d_j(t) = \sum_{i=1}^{n} [x_i(t) - w_{ji}(t)]$$

Dacă intrarea x(t) se potriveşte cel mai bine cu vectorul $w_j(t)$ (deci dacă d_j este minimă) neuronul j este declarat câştigător.

Ponderile conexiunilor se modifică pentru toți neuronii din vecinătatea neuronului câştigător proporțional cu distanța. Procedeul este inspirat din fenomenul inhibiției laterale din creier.

$$w_j(t+1) = w_j(t) + \eta.h(j,k).[x(t) - w_j(t)]$$

Funcția h(j,k) este o funcție proporțională cu distanța dintre neuronii j şi k, cunoscută sub denumirea dată de formă, ca "pălăria mexicană". H(j,j) este 1.

O funcție posibilă este $h(j,k) = e^{-(j-k)^2}$

Vecinătățile descresc în timp ca mărime spre o limita predefinită, localizând zona de maximă activitate. Algoritmul va produce grupări pentru toate tipurile de clase găsite în datele de antrenament Rețeaua devine astfel o hartă topografica de caracteristici. Ordonarea pe hartă a grupărilor si timpul de convergenta sunt dependente, de modul în care, datele de antrenament sunt prezentate rețelei. Odată antrenata rețeaua, detectoarele de caracteristici trebuie insă etichetate manual.

Kohonen si colaboratorii au utilizat aceste rețele în recunoaşterea vorbirii, realizând un procesor hard, ca parte integranta a unui sistem mai mare.

2. Scopul lucrării

Scopul lucrării este studiul principiului de funcționare competitiv şi al unor rețele competitive:

- Rețeaua cu autoorganizare de tip Kohonen;
- Cuantizarea vectorială;
- Câteva dintre modelele competitive ale lui Grossberg : Instar şi Outstar ;
- Testarea unor aplicații ce utilizează rețele competitive;

3. Desfăşurarea lucrării

Se lansează programul MATLAB. Se selectează din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks, şi apoi Other Neural Networks Design Textbook Demos

3.1 Din Chapter 14 demos încărcați programul “Competitive learning”.

Modelele de intrare sunt organizate în trei categorii reprezentate prin culori, pe un cerc. Plasați vectorii pondere ai celor trei neuroni şi selectând Train observați cum se orientează aceşti vectori înspre centrele grupărilor pe care le reprezintă.

Selectați modele de intrare disparate pe cerc şi verificați din nou dacă cei trei neuroni vor învăța şi aceste clase. Alegeți valorile inițiale ale ponderilor departe de modelele de intrare. Vor învăța neuronii cele trei categorii în toate situațiile?

3.2 Din Chapter 14 încărcați programul “Competitive classification”.

Cum a fost formularizată clasificarea modelelor de intrare, aici nişte fructe în funcție de greutate, textură şi formă? Câte straturi are rețeaua neuronală? Câți neuroni are rețeaua în fiecare strat?

3.3 Selectați din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks, programul “Competitive learning”.

Neuronii rețelei competitive învață să reprezinte diferite regiuni ale spațiului de intrare, acolo unde apar modelele de intrare, reprezentate prin puncte.

Tastați help NEWC pentru a obține informații referitoare la modul de funcționare al rețelei competitive.

Vectorii pondere ai celor 8 neuroni se vor orienta spre centrele de greutate ale grupărilor modelelor de intrare, realizând o clasificare a acestora.

La aplicarea unui model de intrare va fi activ doar neuronul corespunzător clasei de apartenență .

3.4 Încărcați din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks programul "One dimensional self organising map"

O RN competitivă de tip Kohonen învață să reprezinte diferite regiuni ale spațiului de intrare, răspunzând la modele similare (care aparțin unei regiuni) prin activarea neuronului corespunzător clasei. RN învață topologia spațiului de intrare.

Modelele de intrare sunt 100 de punct pe un cerc de rază unitate. Rețeaua implementată are 10 neuroni. După 1000 de iterații fiecare neuron va învăța să răspundă (să fie activ) pentru unul dintre cele 10 intervale ale segmentului de cerc de pe care au fost selectate modelele de intrare. Vectorii pondere marcați prin puncte roşii sunt vectorii prototip pentru cele 10 clase .

3.5 Selectați din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks "Two dimensional self organising map"

O RN competitivă de tip Kohonen de dimensiune 5x6 neuroni învață să clasifice 1000 de vectori bidimensionali de coordonate x_1, x_2, cu valori în intervalul

[-1,+1].

Ponderile sunt inițializate la 0 astfel încât ele vor fi reprezentate în planul ponderilor printr-un punct. În timp ponderile se vor autoorganiza într-o rețea regulată fiecare neuron încercând să reprezinte una dintre cele 5x6 zone ale spațiului modelelor de intrare

3.6 Din "Other Neural Networks Design Textbook Demos", Chapter 14 demos încărcați programul "2 D Feature Map". Pentru diferite constante de învățare şi diferite vecinătăți antrenați rețeaua pentru a reprezenta spațiul modelelor de intrare.

Notați experimentele şi concluziile trase.

Capitolul 6
Rețele neuronale autoorganizatoare

Rețelele neuronale abordate în capitolele precedente învață să implementeze o transformare $F: R^N \to R^M$, din perechile de modele intrare $\{x^p\}$- modele de ieşire dorite $\{o^p\}$. Există însă probleme în care nu dispunem de setul de modele dorite ci doar de modelele de intrare. Rețeaua neuronală trebuie să găsească singură informația relevantă din exemplele$\{x^p\}$ care i se aplică la intrare, pe baza similarității acestora. Câteva probleme din această categorie sunt următoarele:

- Gruparea în categorii RN trebuie să găsească singură criteriul de clasificare şi să realizeze gruparea modelelor de intrare.
- Cuantizarea vectorială RN trebuie să determine discretizarea optimă a spațiului continuu de intrare. Intrarea în sistem este modelul x, n dimensional, iar ieşirea este o reprezentare discretă a spațiului de intrare.
- Reducerea dimensiunii Modelele de intrare sunt grupate într-un subspațiu care are dimensiune mai redusă decât dimensiunea spațiului de intrare. Sistemul neuronal trebuie să învețe transformarea optimă astfel încât cea mai mare parte din distribuția modelelor de intrare să se regăsească la ieşire.
- Extragerea de caracteristici. RN trebuie să extragă trăsăturile caracteristice esențiale ale datelor de intrare. Adesea aceasta implică şi o reducere a dimensiunii.

Dacă există şi modelele de ieşire dorite, acestea pot fi folosite ulterior la o rafinare a parametrilor rețelei autoorganizatoare.

6.1 Principiul învățării competitive

Într-o rețea competitivă toți neuronii unui strat sunt complet conectați. Adică primesc intrări excitatorii de la toți neuronii stratului anterior, transmit conexiuni excitatorii către toți neuronii stratului următor şi conexiuni inhibitorii către toți neuronii stratului din care fac parte. Vectorii pondere sunt inițializați aleator, de obicei la valorile unui subset de modele de intrare. În majoritatea rețelelor autoorganizatoare atât modelele de intrare cât şi vectorii pondere sunt normalizați, având acelaşi număr de N elemente. Astfel atât modelele de intrare cât şi vectorii pondere au aceeaşi lungime şi pot fi interpretate ca şi puncte pe o sferă N dimensională (paragraful 6.3). La aplicarea unui model de intrare fiecare neuron procesează intrarea netă:

$$net_j = \sum_{i=1}^{N} w_{ji} x_i = x^T . w_j \qquad (6.1)$$

Se determină starea de activare a neuronilor prin trecerea intrării nete prin funcția de activare. Se selectează apoi neuronul câştigător printr-una dintre cele două modalități posibile:

1. Neuronul câştigător este declarat neuronul cu cea mai mare stare de activare a_c. $a_j \langle a_c \ \forall j = c$

2. Neuronul câştigător este declarat neuronul cu cea mai mică intensitate de intrare I_j definită de relația:

$$I_J = D(w_j, x) \qquad (6.2)$$

unde D este o distanță metrică oarecare.

Câteva distanțe metrice, uzual utilizate, sunt următoarele:

- Norma euclidiană, dată de amplitudinea vectorului diferență :

$$d = \|x-v\| = \|\delta\| = (\delta_1^2 + \ldots + \delta_p^2)^{1/2} = (\delta^T \delta)^{1/2} \tag{6.3}$$

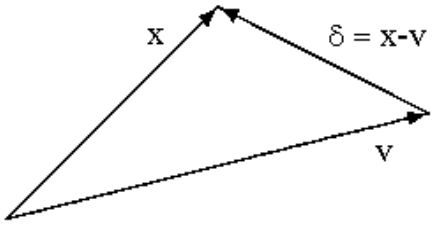


Fig.2.2 Reprezentarea vectorului diferență

- Pătratul amplitudinii vectorului diferență:

$$d = \|x - v\|^2 = \|\delta\|^2 = \delta^T \cdot \delta \tag{6.4}$$

Relația (6.4) reprezintă o simplificare față de cazul anterior.

- Distanța Manhattan, care este o sumă a valorilor absolute ale coordonatelor vectorului diferență:

$$d = \sum_{j=1}^{p} |\delta_i| \tag{6.5}$$

- Proiecția lui x pe v. Aceasta este cea mai simplă măsură a asemănării vectorilor normalizați:

$$d = v^T \cdot x = \|v\| \cdot \|x\| \cdot \cos\alpha \tag{6.6}$$

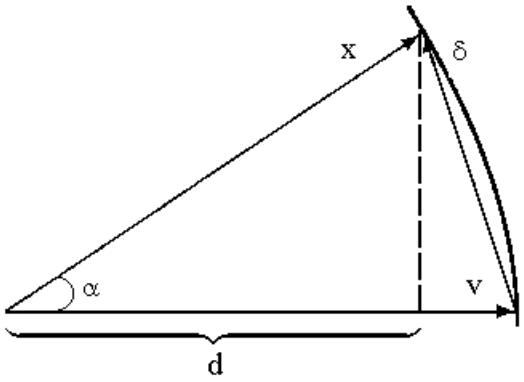


Fig.2.3 Proiecția vectorului x pe v

- Distanța măsurată ca produs:

$$d = x \cdot v^T \tag{6.7}$$

Se recomandă ca cei doi vectori să fie normalizați înainte de măsurare: || x ||=|| v ||=1.

- Distanța Hamming:

$$d = \sum_j |x_j == v| = \begin{cases} 0 & \text{pentru } x = v \\ 1 & \text{pentru } x \neq v \end{cases} \tag{6.8}$$

Exemplu:
Se calculează distanțele prezentate anterior pentru vectorii x = [1 1 -1 1] şi v = [1 -1 -1 -1].

- distanța eulidiană = sqrt($0^2 + 2^2 + 0^2 + 2^2$) = 2.83
- distanța Manhattan = 0 + 2 + 0 + 2 = 4
- distanța Hamming = 0 + 1 + 0 + 1 = 2
- distanța ca produs = [1 1 -1 1]·[1 -1 -1 -1]^T = 0

Odată selectat neuronul câştigător învățarea are loc prin modificarea ponderilor, conform unei strategii de tip competiție, cunoscute în literatura sub numele de "câştigătorul ia totul ".Din acest motiv RN autoorganizatoare se numesc şi RN competitive. Neuronul câştigător tinde către valoarea de activare maximă, adică 1, în timp ce toți ceilalți tind către valoarea minimă, zero, printr-un proces iterativ de inhibiție laterală.

În decursul anilor, mai mulți cercetători au elaborat RN competitive, având la baza diferite reguli de învățare: Kohonen, von der Malsburg (1973), Grossberg (1972, 1976), Fukushima (1975), Bienenstock , Cooper şi Munro (1980), Rumelhart şi Ziepser (1985).

6.3 Interpretarea geometrică

Modelele de intrare şi vectorii pondere normalizaţi pot fi reprezentaţi prin puncte pe o sferă N dimensională. Conform regulii de învăţare de fiecare dată când un neuron câştigă competiţia, vectorul său pondere se îndreaptă înspre modelul de intrare x, mişcare ilustrată în Fig.6.1.

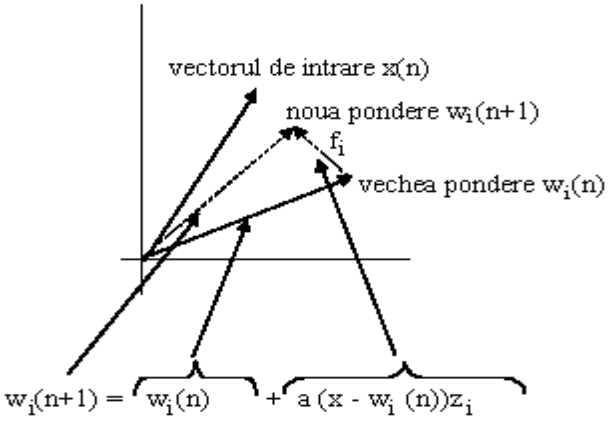
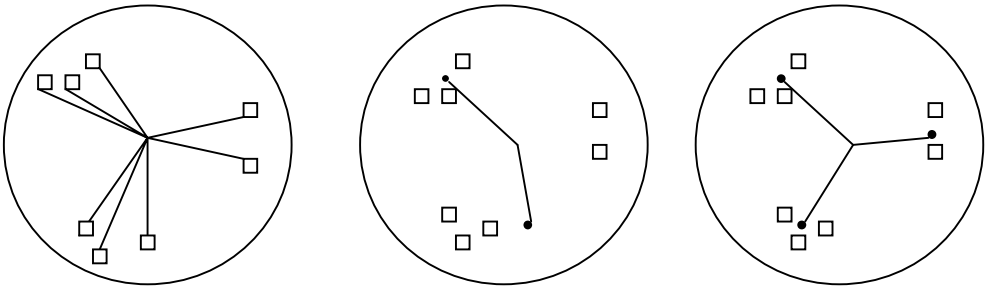


Fig.6.1 Deplasarea vectorului pondere a neuronului declarat câştigător



2a 2b 2c

Fig.6.2 prezintă învăţarea grupelor de modele în cazul unei RN formate din trei neuroni .

Fig.6.2.a Modele similare de intrare sunt situate în puncte apropiate pe sferă.
Fig.6.2.b Când se aplică un model de intrare câştigă competiţia neuronul al cărui vector pondere este cel mai apropiat de modelul de intrare câştigă competiţia. Vectorul său pondere se roteşte înspre modelul de intrare.
Fig.6.2c Dacă există trei neuroni şi trei grupări de modele de intrare, fiecare dintre neuroni va câştiga competiţia pentru una dintre cele trei grupări.

Dacă există în RN mai mulţi neuroni de ieşire decât numărul grupărilor modelelor de intrare, pe măsură ce RN învaţă, vectorii pondere devin mai deşi acolo unde modelele de intrare sunt mai dese şi mai rari, sau chiar absenţi acolo unde modelele de intrare sunt mai puţine. Cu alte cuvinte RN se adaptează pentru a măsura funcţia densitate de probabilitate a modelelor de intrare.

6.4 Estimarea funcţiei densitate de probabilitate

Se doreşte ca vectorii w_j să se aranjeze în spaţiul R^N astfel încât să înveţe funcţia densitate de probabilitate a modelelor de intrare. Dar regula de învăţare

Kohonen nu asigură, în general, un set de vectori pondere echiprobabili. Cu alte cuvinte, fiind dat un model de intrare x din spațiul R^N, în conformitate cu funcția densitate de probabilitate ρ, probabilitatea ca x să fie cel mai aproape de w_j să fie 1/N $\forall$ j=1,2,...N.

Pot apare următoarele probleme:

- ca unele regiuni, acolo unde densitatea de probabilitate este mică, să nu fie reprezentate;
- regiunile cu densitate de probabilitate mare să fie supraeşantionate ;

S-au elaborat o serie de soluții pentru rezolvarea acestor probleme cum sunt:

1. Metoda ***radial sprouting*** este adecvată pentru distanța euclidiană şi alte măsuri similare [Hecht Nielsen].
   Vectorii pondere sunt inițializați la zero şi modelele de intrare x sunt multiplicate cu β (un număr pozitiv mic, $0 < \beta \leq 1$). Procesul de învățare începe cu o valoare scăzută a lui β , aproape de zero. Astfel toți vectorii pondere sunt aproape de vectorii de intrare. Pe măsură ce rețeaua neuronală învață β crește, vectorii pondere sunt forțați să se îndepărteze de zero şi să urmeze modelele de intrare. Câțiva vectori pondere pot rămâne în urmă şi sunt irosiți în procesul de clasificare.
   Dezavantajul metodei constă din faptul că procesul de învățare este încetinit.
2. O altă soluție a fost de a adăuga **vectori de zgomot uniform distribuiți intrărilor,** în scopul pozitivării funcției densitate de probabilitate. Inițial nivelul zgomotului este mult mai mare decât valoarea modelelor de intrare. În timp puterea zgomotului scade. Învățarea în prezența zgomotului este însă şi mai lentă decât în cazul metodei "radial sprouting".
3. Adăugarea unui termen numit "**conştiință**" pentru fiecare neuron , care monitorizează numărul de situații succesive în care acesta a câştigat competiția. Această metodă rezolvă problema echiprobabilității vectorilor pondere [].
   Conceptul de bază al mecanismului de învățare cu conştiință este de a ține o evidență a timpului f_i în care neuronul i a câştigat competiția:

$$\mathbf{f_i[k+1] = f_i[k] + \beta.(o_i - f_i[k])} \tag{6.9}$$

unde:

- o este ieşirea 0 sau 1 a neuronilor după ce s-a terminat competiția;
- β este o constantă pozitivă mică, cu o valoare tipică de 0,0001;
  Se determină apoi curentul de polarizare b_i conform relației:

$$\mathbf{b_i} = \gamma \cdot \left(\frac{\mathbf{1}}{\mathbf{N}} - \mathbf{f_i} \right) \tag{6.10}$$

unde γ este o constantă pozitivă, tipic de valoare 10.

Termenul b_i reprezintă cantitatea prin care frecvența de câştigare a competiției de către neuronul i este sub nivelul de echiprobabilitate $1 / N$. Neuronul cu cea mai mică diferență $\min[D(x,w_i) - c_i]$ este declarat câştigător şi îşi va modifica ponderile conform regulii de învățare, apropiindu-se de modelul de intrare. Spre deosebire de cazul uzual când un singur neuron işi modifică ponderile, şi ceilalți neuroni îşi modifică ponderile îndepărtându-se de intrare. Elementele de procesare care câştigă prea des competiția au valori b_i negative mari. Cele care nu câştigă prea des competiția au valori de polarizare pozitive astfel încât favorizate de relația de declarare a neuronului câştigător.

În final vectorii pondere se vor distribui într-o configurație aproape echiprobabilă

Metoda este cunoscută şi sub denumirea de "frequency competitive learning" .

Aproape toate informațiile referitoare la date din unele domenii ca de exemplu teoria informației, recunoaşterea formelor , statistică se regăsesc în funcția distribuție de probabilitate.

6.5 Rețeaua MAXNET

În rețeaua MAXNET fiecare neuron este cuplat cu el însuți excitator şi îi inhibă pe toți ceilalți:

$$w_{ij} = \begin{cases} -\alpha & \text{pentru} \quad i \neq j \\ 1 & \text{pentru} \quad i = j \end{cases} \tag{6.11}$$

unde $\alpha = \frac{1}{N} < 1$ este o constantă pozitivă mică, iar N numărul de neuroni din RN

Relația (6.11) se poate scrie ca o matrice de dimensiune NxN:

$$W_N = \begin{bmatrix} 1 & -\alpha & ... & -\alpha \\ -\alpha & 1 & ... & -\alpha \\ & ... & ... & \\ -\alpha & -\alpha & ... & 1 \end{bmatrix} \tag{6.12}$$

Modelul de intrare este activ doar în momentul inițial. Fiecare neuron procesează intrarea sa netă conform relației (6.1), adică sub formă matricială:

$$net[k] = W_N . x[k]^T \tag{6.13}$$

Apoi se determină ieşirea aplicându-se funcția de activare intrării nete:

$$o[k+1] = f(net[k]) \tag{6.14}$$

Funcția de activare este definită de relația:

$$f(net[k]) = \begin{cases} net[k] & \text{pentru} \quad net[k] \geq \theta \\ 0 & \text{pentru} \quad net[k] < \theta \end{cases} \tag{6.15}$$

Ieşirile tuturor neuronilor la momentul k+1 se folosesc pentru a determina intrarea netă în neuroni la momentul următor de timp k+2. Se poate demonstra că aplicându-se în mod recursiv relațiile (6.13) şi (6.14) rețeaua MAXNET va converge înspre o situație în care doar neuronul cu cea mai mare intrare netă inițială va rămâne activ în timp ce toți ceilalți vor converge spre activarea zero. Din acest motiv rețeaua MAXNET se numeşte şi rețea de tipul (" câştigătorul ia totul " în engleză winner-takes-all).

O rețea similară este MINNET care la ieşire va avea un singur neuron activ, acela cu cea mai mică stare de activare inițială.

Exemplu

Tipic o rețea neuronală competitivă este alcătuită din două straturi de neuroni:
-stratul de măsurare al distanței;
-stratul competitiv, de tip MAXNET;
Structura unei rețele neuronale competitive este reprezentată în figura 6.3:

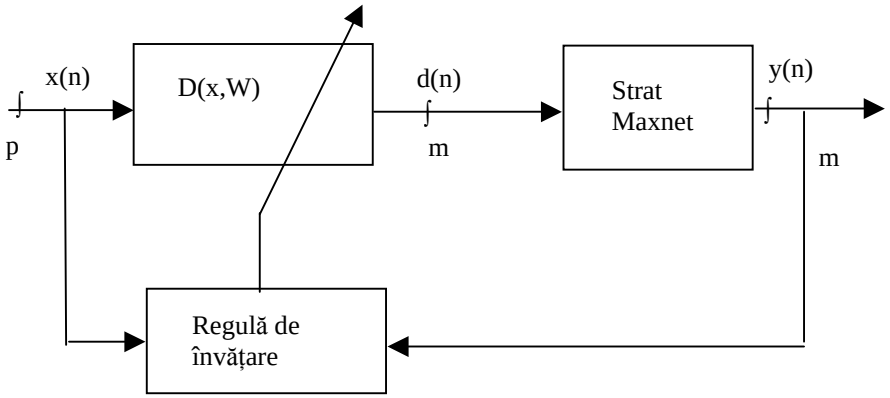


Fig.6.3 Structura rețelei neuronale competitive

Fie un clasificator neuronal de caractere, implementat cu o rețea Hamming ca prim strat şi o rețea Maxnet ca al doilea strat. Literele C, I, T sunt modelele prototip . RN va selecta clasa căreia îi aparține modelul aplicat la intrare, respectiv clasa la distanța Hamming cea mai mică față de acesta.

Stratul Hamming va avea la ieşire un neuron cu cea mai mare stare de activare , dacă distanța Hamming dintre modelul de intrare şi categoria reprezentată de neuron este minimă. Stratul MAXNET suprimă ieşirile tuturor neuronilor cu excepția celui care a avut cea mai mare stare de activare inițială.

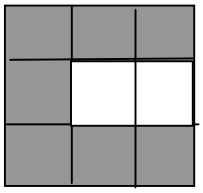
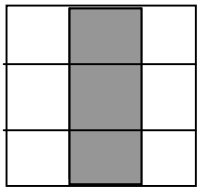
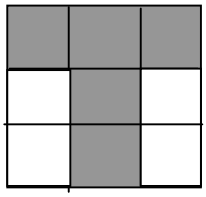
Fie modelul prototip pentru o clasă m, $s^m = [s_1^m s_2^m \quad ... \quad s_N^m]$.

Matricea ponderilor pentru stratul Hamming care realiyeayă o clasificare în p categorii este dată de relația:

$$W_H = \frac{1}{2}\begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \cdots & s_N^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \cdots & s_N^{(2)} \\ s_1^{(p)} & s_2^{(p)} & \cdots & s_N^{(p)} \end{bmatrix}$$

Pentru litera C, modelul prototip, conform imaginii de mai jos, are structura s=[1 1 1 1 -1 -1 1 1 1 1].

În mod similar pentru litera I modelul prototip este s=[-1 1 -1 -1 1 -1 1 -1 1] şi pentru litera T modelul prototip este s=[1 1 1 -1 1 -1 -1 1 -1]. Matricea ponderilor pentru rețeaua Hamming este:

$$W_H = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}$$

Intrarea netă pentru rețeaua Hamming este dată de relația

$$net_m = \frac{1}{2} \cdot s^m \cdot x^T + \frac{N}{2} \quad , \qquad \text{pentru } m=1,2, \ldots , p$$

sau de :

$$net_m = N - HD(x, s^{(m)})$$

unde HD este distanța Hamming, numărul de poziții în care cei doi vectori diferă. Practic net ne dă numărul de poziții în care cei doi vectori x şi s^m se aseamănă.

$$f(net_m) = \frac{1}{n} \cdot net_m$$

Intrările rețelei Hamming sunt date de:

$$net_1 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \frac{9}{2} = \frac{5}{2} + \frac{9}{2} = 7$$

$$\text{net}_2 = \frac{1}{2}\begin{bmatrix}-1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1\end{bmatrix}\cdot\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix} + \frac{9}{2} = -\frac{3}{2} + \frac{9}{2} = 3$$

$$\text{net}_3 = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1\end{bmatrix}\cdot\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix} + \frac{9}{2} = \frac{1}{2} + \frac{9}{2} = 5$$

Ieşirile rețelei Hamming sunt intrări pentru rețeaua Maxnet la momentul 0 :

$$f(\text{net}_1) = \frac{1}{9}\cdot\text{net}_1 = \frac{7}{9}$$

$$f(\text{net}_2) = \frac{1}{9}\cdot\text{net}_2 = \frac{3}{9}$$

$$f(\text{net}_3) = \frac{1}{9}\cdot\text{net}_3 = \frac{5}{9}$$

Într-o formă vectorială modelul de intrare în rețeaua Maxnet este:

$$x[0] = \begin{bmatrix}\frac{7}{9} & \frac{3}{9} & \frac{5}{9}\end{bmatrix}$$

Dacă se alege ε=0.2 (care respectă condiția ε<1/3), matricea ponderilor pentru rețeaua Maxnet W_N este :

$$W_N\cdot = \begin{bmatrix}1 & -\frac{1}{5} & -\frac{1}{5}\\ -\frac{1}{5} & 1 & -\frac{1}{5}\\ -\frac{1}{5} & -\frac{1}{5} & 1\end{bmatrix}$$

MAXNET est au moment initial :

$$\text{net}[0] = W_N \cdot o[0] = \begin{bmatrix} 1 & -\frac{1}{5} & -\frac{1}{5} \\ -\frac{1}{5} & 1 & -\frac{1}{5} \\ -\frac{1}{5} & -\frac{1}{5} & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{7}{9} \\ \frac{3}{9} \\ \frac{5}{9} \end{bmatrix} = \begin{bmatrix} 0.599 \\ 0.067 \\ 0.333 \end{bmatrix}$$

Ieşirile reţelei Maxnet, respectiv intrările nete la iteraţiile succesive sunt:

$$o[1] = f(\text{net}[0]) = \begin{bmatrix} 0.599 \\ 0.067 \\ 0.333 \end{bmatrix}$$

$$\text{net}[1] = W_N \cdot f(\text{net}[0]) = \begin{bmatrix} 1 & -\frac{1}{5} & -\frac{1}{5} \\ -\frac{1}{5} & 1 & -\frac{1}{5} \\ -\frac{1}{5} & -\frac{1}{5} & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.599 \\ 0.067 \\ 0.333 \end{bmatrix} = \begin{bmatrix} 0.520 \\ -0.120 \\ 0.120 \end{bmatrix}$$

$$o[2] = f(\text{net}[1]) = \begin{bmatrix} 0.520 \\ 0 \\ 0.120 \end{bmatrix}$$

$$\text{net}[2] = W_N \cdot f(\text{net}[0]) = \begin{bmatrix} 1 & -\frac{1}{5} & -\frac{1}{5} \\ -\frac{1}{5} & 1 & -\frac{1}{5} \\ -\frac{1}{5} & -\frac{1}{5} & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.520 \\ 0 \\ 0.120 \end{bmatrix} = \begin{bmatrix} 0.480 \\ -0.140 \\ 0.096 \end{bmatrix}$$

$$o[3] = f(\text{net}[2]) = \begin{bmatrix} 0.480 \\ 0 \\ 0.096 \end{bmatrix}$$

$$\text{net}[3] = W_N \cdot f(\text{net}[2]) = \begin{bmatrix} 1 & -\frac{1}{5} & -\frac{1}{5} \\ -\frac{1}{5} & 1 & -\frac{1}{5} \\ -\frac{1}{5} & -\frac{1}{5} & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.520 \\ 0 \\ 0.120 \end{bmatrix} = \begin{bmatrix} 0.461 \\ -0.115 \\ -10^{-7} \end{bmatrix}$$

$$o[4] = f(net[3]) = \begin{bmatrix} 0.461 \\ 0 \\ 0 \end{bmatrix}$$

Ieşirea reţelei MAXNET rămâne pentru toate iteraţiile succesive aceeaşi :

$$o[4] = \begin{bmatrix} 0.461 & 0 & 0 \end{bmatrix}$$

Aşadar modelul de intrare, prototipul afectat de zgomot va fi clasificat ca litera C.

Lucrarea nr.6

2. Scopul lucrării

- Scopul lucrării este studiul algoritmului cuantizării vectoriale (learning vector quantization)

3. Desfăşurarea lucrării

Se lansează programul MATLAB. Se selectează din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks, şi apoi Other Neural Networks Design Textbook Demos

3.1 Din Chapter 14 Demos încărcați programul “LVQ1”.

Modelele de intrare sunt reprezentate prin pătrate ce se pot "trage" cu mouse-ul pentru a defini diferite probleme. Se doreşte antrenarea a patru neuroni ai căror vectori pondere sunt reprezentați prin cruciulițe astfel încât modelele de intrare să fie clasificate în două categorii. Funcția "Learn" vă permite antrenarea o singură dată şi "Train" de cinci ori. Folosiți "Random" pentru a inițializa la diferite valori vectorii pondere. Vor reuşi să clasifice neuronii modelele de intrare în toate situațiile?

3.2 Din Chapter 14 demos încărcați programul “LVQ2”.

Repetați 3.1 observând comparativ cu LVQ1 cum are loc antrenarea neuronilor.

Notați observațiile. Vor reuşi de data aceasta neuronii să clasifice modelele de intrare în toate situațiile?

3.3 Selectați din menu-ul Help, Examples and Demos, apoi din Toolboxes, Neural Networks, programul “Learning vector quantization”.

Neuronii unei rețele competitive cu două straturi învață să clasifice modele de intrare bidimensionale în două categorii, utilizând algoritmul cuantizării vectoriale. Cele 10 modele utilizate pentru antrenare, sunt reprezentate prin cruciulițe de culori diferite corespunzătoare celor două clase. Tastați help NEWLVQ pentru a obține informații referitoare la modul de funcționare al rețelei competitive.

Vectorii pondere ai celor 4 neuroni ascunşi se vor orienta spre "centrele de greutate" ale grupărilor modelelor de intrare, realizând o clasificare a acestora.

La aplicarea unui model de intrare va fi activ doar neuronul corespunzător clasei de apartenență .