REŢELE NEURONALE ARTIFICIALE
GENERALITĂŢI. NEURONUL FORMAL. PERCEPTRONUL MULTISTRAT

La originea dezvoltării cercetărilor în domeniul reţelelor neuronale artificiale (RNA) se găseşte o constatare simplă : există sarcini cărora calculatoarele numerice convenţionale le pot face faţă cu dificultate, în timp ce sistemul nervos al celor mai simple organisme găseşte un răspuns fără a face eforturi evidente. Acestea sunt, de exemplu, cazurile recunoaşterii formelor sau coordonării mişcărilor. Performanţele remarcabile ale creierului uman au lăsat să se întrevadă unele avantaje ce s-ar putea obţine folosind modele de inspiraţie biologică.

Primul model al neuronului formal, acceptat în linii generale şi astăzi, este propus în anul 1943 de către W.S. McCulloch şi W. Pitts. Cercetările întreprinse ulterior în neurobiologie şi psihologie, conduc la primul model de învăţare, propus de către D.O. Hebb în 1949. Un impact deosebit asupra cercetărilor întreprinse în direcţia realizării primelor reţele neuronale artificiale, l-a reprezentat publicarea de către F. Rosenblatt, în anul 1958, a primelor sale lucrări despre *perceptron*. Timp de un deceniu, oamenii de ştiinţă au fost aproape unanimi în a considera că noua paradigmă poate fi aplicată cu succes pentru numeroase probleme practice, în care sunt implicate inteligenţa şi memoria umană.

În anul 1969, însă, M. Minsky şi S. Papert demonstrează imposibilitatea principială a modelelor de reţele neuronale artificiale propuse până atunci de a modela probleme relativ simple, aşa cum este cazul funcţiei logice SAU EXCLUSIV (XOR). Aceste concluzii au condus la o scădere dramatică a interesului pentru noi cercetări în această direcţie.

Evenimentul care a relansat cercetările comunităţii ştiinţifice mondiale înspre realizarea unor modele conexioniste performante l-a reprezentat apariţia în 1986 a cărţii *Parallel Distributed Processing, Explorations in the Microstructure of Cognition* de D. Rumelhart şi J. McClelland, care introduce noţiunea de *perceptron multistrat*. În prezent, prin rezolvarea unor probleme de complexitate ridicată, cum sunt cele de estimare, identificare şi predicţie sau optimizare, reţelele neuronale artificiale capătă o pondere şi un impact tot mai mari în numeroase sectoare ale ştiinţei, tehnologiei sau vieţii sociale.

Neuronul formal

Neuronul formal este un element de procesare a informaţiei care modelează, de o manieră simplificată, neuronul real. Cea mai simplă versiune de neuron formal este un automat binar cu două stări : activ (+1) şi inactiv (-1) (Fig. 1).

Starea neuronului se actualizează periodic după următorul mecanism: se determină *potenţialul neuronal* v_i , calculând suma ponderată a intrărilor x_j (care reprezintă ieşirile altor neuroni din reţea sau informaţii provenind de la neuronii de intrare); acest potenţial este comparat cu un prag θ_i, neuronul activându-se ($y_i = +1$) dacă $v_i \geq \theta_i$ sau devenind pasiv ($y_i = -1$) dacă $v_i < \theta_i$. Această prelucrare a informaţiei în interiorul neuronului corespunde unei funcţii de transfer de tip treaptă (Heviside), denumită frecvent *funcţie de activare*.

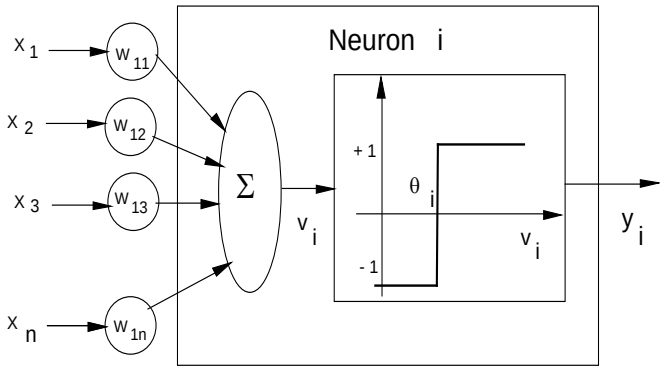

Fig.1 Neuronul formal

Variante îmbunătăţite ale neuronului formal utilizate în present folosesc diverse funcţii de activare. Cele mai utilizate funcţii de activare sunt:

Funcţia de activare liniară, (Fig. 2a) de forma :

$$f(x) = a \cdot x + b$$

unde coeficientul *b* joacă rolul unui prag. Pentru *a = 1* şi *b = 0* se obţine funcţia de activare *identitate*, iar pentru *a = 1* şi *b ≠ 0* – funcţia de activare *identitate plus prag*.

Funcţia de activare treaptă (Fig. 2b)
În acest caz, dacă mărimea de intrare a neuronului, x, atinge sau depăşeşte un prag predefinit θ, funcţia produce la ieşirea neuronului valoarea β; în caz contrar, la ieşire se obţine valoarea –δ. Mărimile β şi δ sunt scalari pozitivi, iar funcţia treaptă este definită formal cu relaţia:

$$f(x) = \begin{cases} \beta \ dacă & x \geq \theta \\ -\delta \ dacă & x \prec \theta \end{cases}$$

De regulă, pentru constantele β şi δ se folosesc valori corespunzătoare unor reprezentări binare, de exemplu β = 1 şi δ = 0, respectiv β = δ =1.

Funcţia de activare rampă (Fig. 2c), combinaţie între funcţiile liniară şi treaptă.
Această funcţie de activare stabileşte limite maximă şi minimă pentru ieşirea neuronului (± *γ*), asigurând totodată o variaţie liniară între aceste limite. De regulă, punctele de saturare sunt simetrice în raport cu originea, iar expresia funciei rampă este:

$$f(x) = \begin{cases} \gamma \ dacă & x \geq \gamma \\ x \ dacă & |x| \prec \gamma \\ -\gamma \ dacă & x \prec -\gamma \end{cases}$$

Funcţii de adaptare tip sigmoid (varianta continuă a funcţiei rampă).
Acestea sunt fucţii mărginite, monotone şi nedescrescătoare, care asigură o variaţie continuă a mărimii de ieşire a neuronului, într-un domeniu predefinit. Cele mai răspândite funcţii sigmoid sunt:

- *sigmoidul logistic* sau *unipolar* (Fig. 2d) $f(x) = \dfrac{1}{1+e^{-\alpha x}}$

- *sigmoidul tangent hiperbolic* sau *bipolar* (Fig. 2e) $f(x) = \dfrac{1-e^{-\alpha x}}{1+e^{-\alpha x}}$

- *Funcţii de adaptare de tip gaussian* (Fig 2f).

Aceste funcţii au formă radială, simetrică în raport cu o valoare medie x_m şi sunt caracterizate de un anumit grad de împrăştiere, descries de varianţa σ:

$$f(x) = e^{-(x-x_m)^2/\sigma}$$

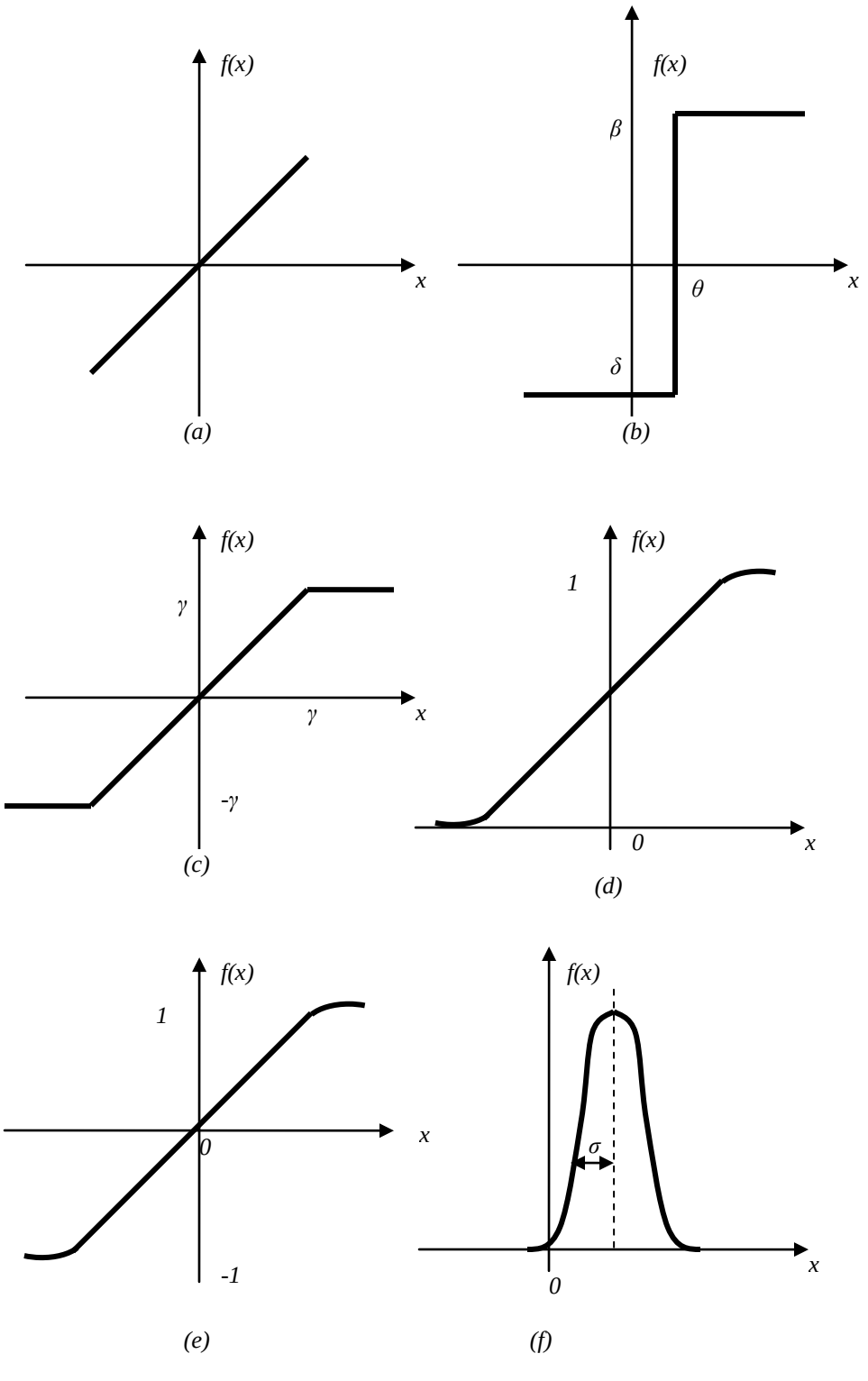

Fig. 2 Funcţii de activare utilizate de RNA

Reţele neuronale artificiale

Neuronul formal calculează suma ponderată a intrărilor, care este trecută apoi prin funcţia de activare, cu o formă în general neliniară. Îndeplinirea unor funcţii care să facă posibilă abordarea unor probleme complexe, de interes practic, este posibilă numai dacă neuronii sunt asociaţi într-un sistem numit *reţea neuronală*.

Reţelele neuronale artificiale (RNA) sunt formate din neuroni (elementele de procesare), legaţi prin conexiuni sinaptice (căile de transmitere a informaţiei între neuroni), caracterizate de anumite ponderi. Fig. 3 ilustrează structura tipică a unei RNA.

RNA sunt organizate pe straturi. Reţeaua din Fig. 3 conţine trei straturi de neuroni, dintre care primul este stratul de *intrare*, iar ultimul stratul de *ieşire*, stratul intermediar numindu-se strat *ascuns*.

Singurele straturi care realizează procesarea propriu-zisă a informaţiei sunt stratul ascuns şi cel de ieşire. Neuronii din primul strat au numai rolul de a prelua mărimile de intrare în reţea; de aceea, aceşti neuroni, fie nu au funcţii de activare, fie folosesc funcţii de activare *identitate* (**$y_i = v_i$**). Din acelaşi motiv, neuronii de intrare nu sunt consideraţi ca formând un strat şi, de multe ori, o reţea ca cea din Fig. 3 este desemnată ca reţea cu două straturi.

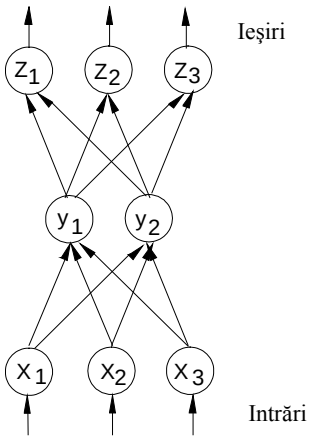

Fig. 3 Arhitectura tipică a unei reţele neuronale artificiale

Legăturile dintre straturile RNA se realizează prin conexiuni sinaptice ponderate. Pentru RNA din Fig. 3, fiecare neuron dintr-un strat este legat cu toţi neuronii din stratul următor şi nu există legături între straturile neconsecutive (de exemplu, intrare-ieşire). O asemenea structură este cunoscută sub numele de RNA *complet conectată*. Structura RNA este descrisă complet de matricea ponderilor conexiunilor dintre două straturi consecutive ***[W]***, ale cărei elemente w_{ij} indică influenţa pe care ieşirea neuronului ***j*** din stratul inferior o are asupra activării neuronului ***i*** din stratul superior. Astfel, ponderile pozitive au caracter excitatoriu, ponderile negative - caracter inhibitoriu, iar ponderile nule indică absenţa conexiunii între cei doi neuroni. Totodată, cu cât valoarea absolută a ponderii w_{ij} este mai mare, cu atât influenţa excitatorie / inhibitorie a neuronului ***j*** asupra neuronului ***i*** este mai pregnantă.

RNA din Fig. 3 are anumite proprietăţi care se aplică unei categorii largi de reţele neuronale:

- fiecare neuron acţionează *independent* de ceilalţi neuroni din acelaşi strat; ieşirea unui neuron depinde numai de semnalele ce se aplică pe conexiunile sinaptice de intrare;
- activarea fiecărui neuron depinde numai de *informaţii cu caracter local***;** informaţia ce este prelucrată de neuron provine numai de pe conexiunile adiacente, nefiind necesară cunoaşterea stării altor neuroni cu care neuronul considerat nu are legături directe;
- *numărul mare de conexiuni* existente asigură un grad ridicat de rezervare şi uşurează reprezentarea distribuită a informaţiei.

Primele două proprietăţi permit funcţionarea eficientă a RNA în paralel, iar ultima proprietate le conferă o sensibilitate redusă faţă de posibilele perturbaţii şi calităţi de generalizare greu de obţinut cu sistemele clasice de calcul.

Aceste proprietăţi sugerează următoarele trei cazuri importante, în care utilizarea RNA se poate dovedi avantajoasă:

- situaţii care impun luarea unui număr mic de decizii pe baza unei cantităţi mari sau foarte mari de informaţii;
- situaţii care necesită realizarea automată a unor clasificări neliniare;
- situaţii în care se doreşte obţinerea în timp real a unei soluţii optimale sau suboptimale pentru o problemă de optimizare combinatorie.

Există mai multe criterii de clasificare a RNA. În continuare, vom prezenta două dintre aceste criterii, care au în vedere arhitectura acestora.

Un prim criteriu se referă la existenţa sau absenţa legăturilor de reacţie inversă între neuronii din diversele straturi ale reţelei. Astfel, se disting două tipuri:

- RNA nebuclate (reţele *feedforward*). În asemenea reţele informaţia circulă într-un singur sens, de la intrare către ieşire, deci la un moment dat, starea unui neuron depinde numai de starea de la acelaşi moment a neuronilor de la care primeşte semnale. Prin urmare, RNA nebuclate sunt structuri statice, folosite cu precădere pentru rezolvarea unor probleme de clasificare sau de identificare a proceselor statice. Din această categorie fac parte reţelele de tipul *perceptronului multistrat***,** a cărui arhitectură a fost prezentată în Fig. 3.
- RNA buclate (reţele *feedback*) sunt acele reţele ale căror grafuri de conexiuni conţin cicluri; circulaţia informaţiei are loc de această dată în ambele sensuri (intrare-ieşire, respectiv ieşire-intrare), astfel încât starea neuronilor la un moment dat este determinată de starea curentă şi de starea la momentul anterior. Prin urmare, RNA buclate au proprietăţile unor sisteme dinamice; ele sunt utilizate ca memorii asociative şi pentru identificarea sau controlul sistemelor dinamice. Dintre reţelele buclate folosite mai des în aplicaţiile practice, menţionăm reţelele *Hopfield* şi *Kohonen*, ale căror arhitecturi sunt prezentate în Fig. 4 şi Fig. 5.

O reţea hibridă, care foloseşte atât legaturi *feedforward*, cât şi legături *feedback*, este reţeaua Hamming (Fig. 6), numită şi *clasificator după similaritate maximă*.

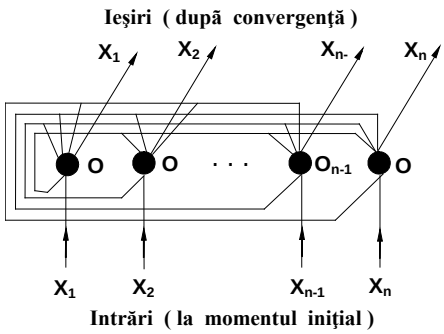

Fig. 4 Reţea Hopfield

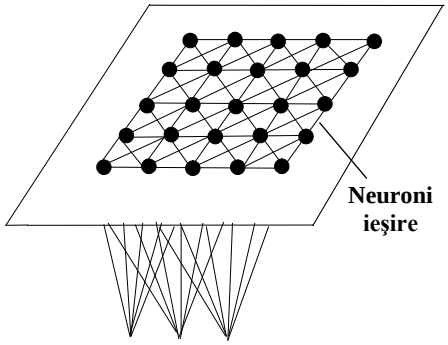

Fig. 5 Reţea Kohonen

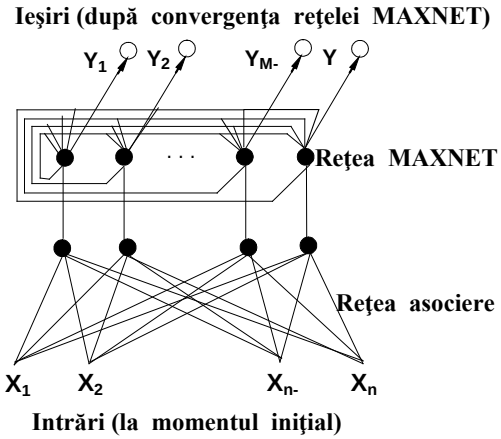

Fig. 6 Reţea Hamming

Cel de-al doilea criteriu de clasificare a RNA are în vedere numărul straturilor de neuroni din reţea. Din acest punct de vedere se disting:

- RNA cu un *singur strat*. În acest caz stratul unic joacă rol dublu *intrare-ieşire*. Totodată, absenţa altor straturi impune ca aceste RNA să aibă o topologie buclată . În această categorie se înscriu reţelele Hopfield (Fig. 4), precum şi variante ale acestora, care se deosebesc în funcţie de modul de conectare a neuronilor. Reţelele cu un singur strat sunt folosite pentru completarea modelelor, filtrarea unor semnale sau rezolvarea unor probleme de optimizare.
- RNA cu *două straturi*. În acest caz, primul strat este stratul de intrare, iar al doilea cel de ieşire, neexistând un strat ascuns. Reţelele din această categorie sunt folosite cu precădere ca reţele clasificatoare. În funcţie de topologia lor, se disting RNA *feedforward* (Fig. 7) sau RNA hibride *feedforward-feedback* (Fig. 8).
- RNA *multistrat*. Reţelele din această categorie pot avea, în principiu, un număr nelimitat de straturi. Toate straturile, cu excepţia primului şi ultimului, sunt straturi ascunse. Structura generală a unei asemenea RNA este indicată în Fig. 9. Majoritatea RNA multistrat utilizate în diverse aplicaţii practice fac parte din categoria reţelelor *feedforward* (nebuclate), iar răspândirea cea mai mare o are *perceptronul multistrat*. Principalele aplicaţii ale acestui tip de reţea au în vedere clasificarea şi aproximarea funcţiilor.

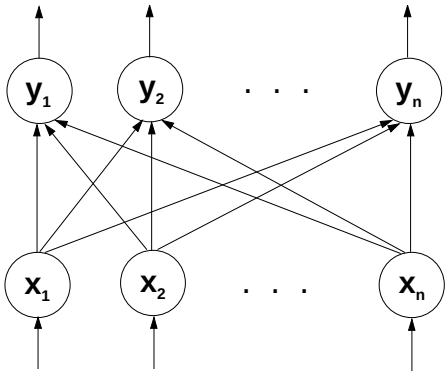

Fig. 7 RNA cu două straturi de tip feedforward

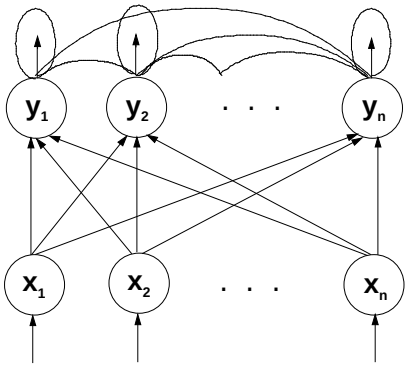

Fig. 8 RNA cu două straturi de tip hibrid

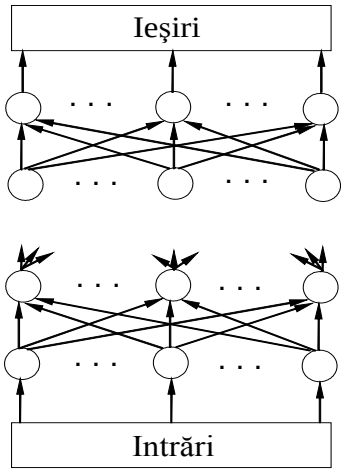

Fig. 9 Perceptronul multistrat

Învăţarea reţelelor neuronale artificiale

Utilizarea RNA în cadrul unei anumite aplicaţii necesită parcurgerea, în prealabil, a unei etape esenţiale - etapa de *învăţare* sau *antrenare*. În majoritatea cazurilor, antrenarea unei RNA constă în determinarea ponderilor conexiunilor sinaptice dintre neuroni w_{ij} şi a pragurilor fiecărui neuron θ_i , care asigură performanţele optime ale reţelei, în sensul în care, pentru un set de date aplicat la intrare, reţeaua oferă la ieşire răspunsul cel mai apropiat de "realitatea" problemei analizate.

Metodele de antrenare pot fi clasificate în două mari categorii: antrenare *supravegheată* şi antrenare *nesupravegheată*.

Antrenarea supravegheată, numită uneori şi învăţare *neadaptivă*, prezintă următoarele particularităţi:

- informaţia prelucrată are caracter global;
- setul de date de învăţare conţine un număr finit de modele de învăţare, organizate în perechi intrare-ieşire cunoscute în prealabil;
- adaptarea ponderilor conexiunilor neuronilor se face calculul abaterilor între ieşirile reale şi cele dorite, pentru fiecare model din setul de antrenare;
- necesită supravegherea din exterior a procesului de adaptare a ponderilor şi pragurilor, prin calculul abaterii între ieşirile reţelei şi cele dorite;
- încheierea etapei de antrenare se face la hotărârea "supraveghetorului", care decide unilateral dacă, în acel moment, performanţele reţelei sunt sau nu satisfăcătoare.

În această categorie intră majoritatea reţelelor neuronale artificiale utilizate în present, cel mai utilizat şi cunoscut tip fiind perceptronul multistrat.

Antrenarea nesupravegheată, numită şi învăţare *adaptivă* sau *auto-organizare*, prezintă următoarele particularităţi:

- informaţia prelucrată are caracter local;
- setul de date de învăţare poate conţine un număr nelimitat de exemplare, care se adaugă la setul iniţial pe măsura rafinării performanţelor reţelei;
- procesul de adaptare a ponderilor şi pragurilor conexiunilor sinaptice şi neuronilor din reţea nu necesită supravegherea din exterior, deoarece reţeaua îşi organizează singură informaţia, asigurând simultan adaptarea ponderilor conexiunilor sinaptice;
- etapa de antrenare se încheie în momentul în care se definitivează organizarea datelor din setul de învăţare iniţial; ea poate fi reluată însă oricând, dacă apar noi caracteristici ale datelor din acest set.

În categoria metodelor de învăţare nesupravegheată intră metodele asociate reţelelor de tip Hopfield şi Kohonen.

Aplicaţii ale reţelelor neuronale artificiale

Câteva din operaţiile pe care le pot executa cu succes RNA sunt următoarele:

- ***clasificarea*** - reţelei i se furnizează un model de intrare , la ieşire obţinându-se clasa sau categoria cărora le aparţine modelul respectiv;
- ***asocierea modelelor*** - reţelei i se furnizează un model de intrare, la ieşire obţinându-se un model asociat primului pe baza unor reguli transmise reţelei în etapa de învăţare;
- ***completarea modelelor*** - reţelei i se furnizează un model de intrare incomplet (din care lipsesc o serie de informaţii), la ieşire obţinându-se modelul reconstituit;
- ***filtrarea semnalelor*** - la intrarea reţelei se aplică un semnal afectat de "zgomot", iar la ieşire se obţine un semnal filtrat, din care s-a eliminat total sau parţial componenta "zgomot";
- ***optimizare*** **-** reţelei i se prezintă un model de intrare care reprezintă valorile iniţiale ale unei probleme de optimizare, la ieşire obţinându-se un set de variabile, care reprezintă soluţia optimală sau suboptimală a problemei;
- ***control*** - modelul de intrare care se transmite reţelei reprezintă starea curentă a unui element de control şi răspunsul care se doreşte de la acesta, iar la ieşire se obţine secvenţa de comenzi care conduce la acest răspuns.

APLICAŢIE

În cadrul şedinţei de laborator, se va utiliza aplicaţia sofware care permite studiul funcţionării neuronului elementar. Se va studia funcţionarea neuronului cu o intrare (Fig.10), a neuronului cu două intrări (Fig. 11) şi a unei reţele neuronale simple de tip feedforward cu două straturi (fig. 12). Cu ajutorul unor cursoare, se pot alege diferite valori pentru mărimile de intrare şi pragul de activare al fiecărui neuron. Programul permite utilizarea unei funcţii de activare identitate, sigmoid logistic sau tangent hiperbolic. Rezultatele sunt prezentate sub formă numerică şi grafică.

Se vor realiza combinaţii mărimi de intrare – praguri – funcţii de activare, analizând şi comparând rezultatele obţinute.

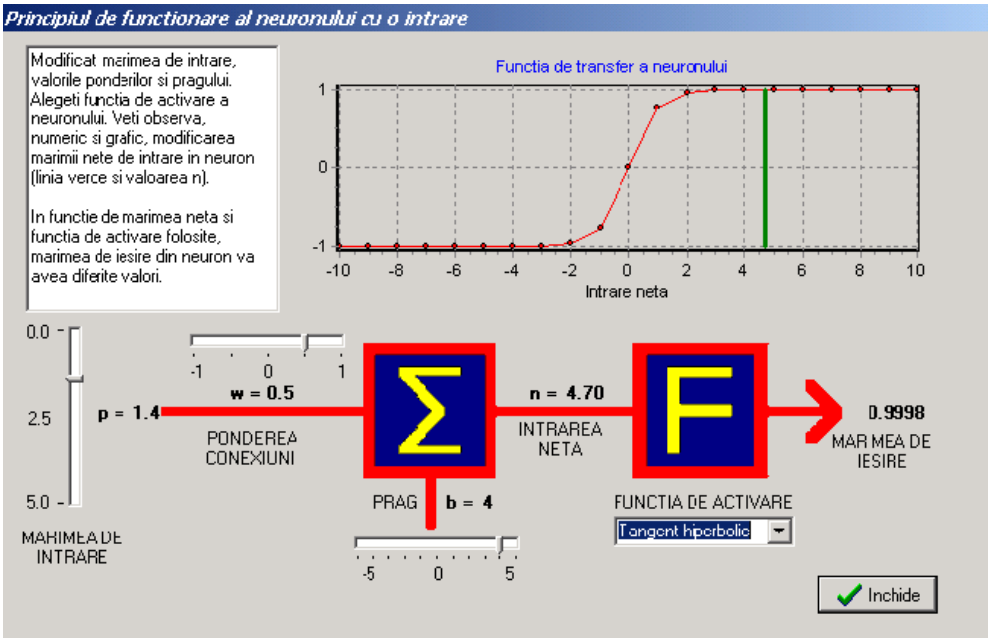

Fig. 10 Neuronul cu o intrare

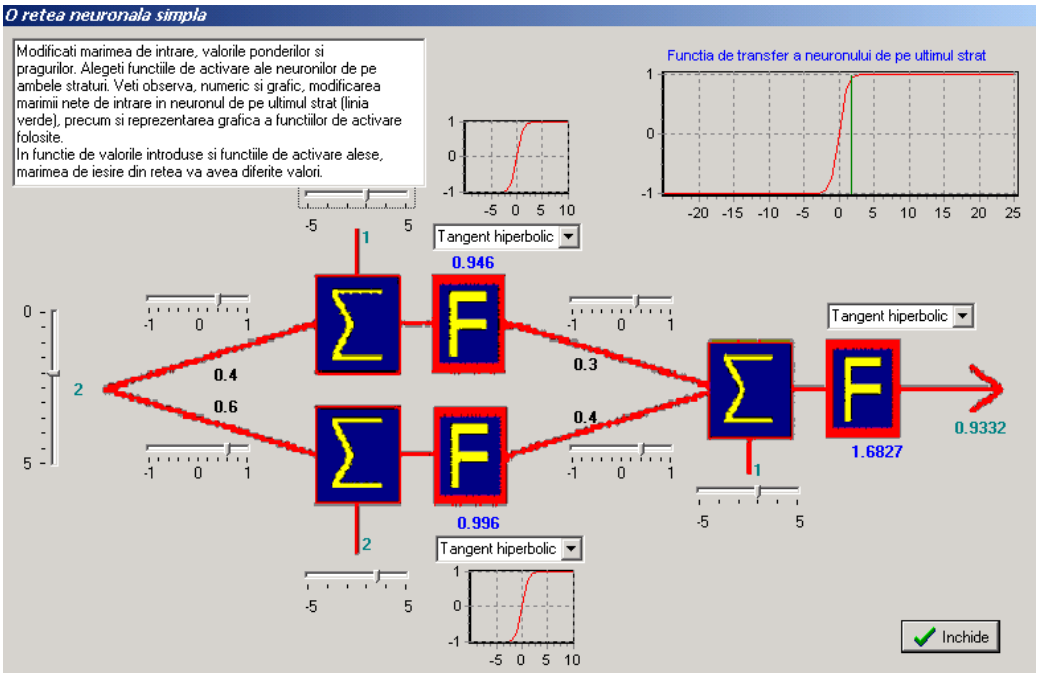

Fig. 11 Reţea neuronală simplă

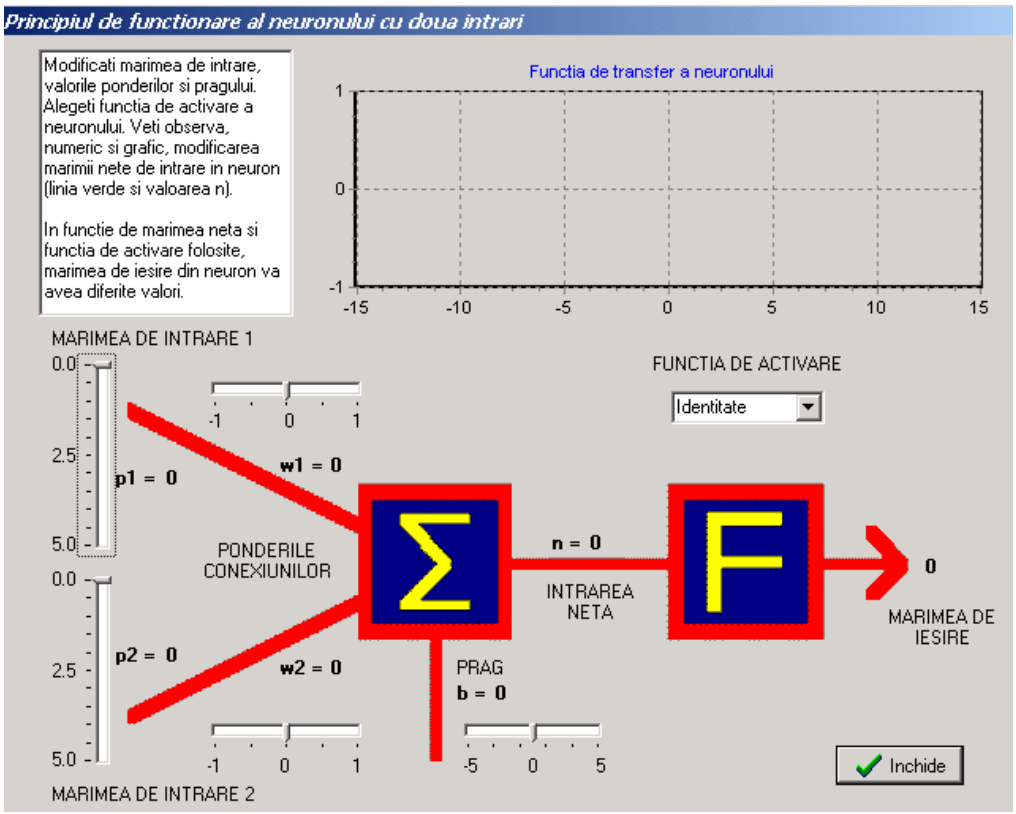

Fig. 12 Neuronul cu două intrări

PERCEPTRONUL MULTISTRAT

Lucrarea publicată de M. Minsky şi S. Papert în anul 1969 demonstra fără nici un dubiu că reţelele dezvoltate pe baza perceptronului lui Rosenblatt nu permit rezolvarea unor probleme simple, , aşa cum este cazul funcţiei logice SAU EXCLUSIV (XOR) . Peste aproape 20 de ani, s-a sugerat că s-ar putea încerca introducerea unor neuroni ascunşi – pentru care nu există intrări şi ieşiri cunoscute apriori – fiind astfel posibilă abordarea unor probleme de genul celor amintite de Minski şi Papert. Ca urmare, D. Rumelhart şi J. McClelland au propus în 1986 ca, într-o reţea neuronală, care conţine şi neuroni ascunşi, să se aplice aşa numita *învăţare* sau *antrenare* după principiul *propagării înapoi* sau *retropropagării* erorii..

Învăţarea prin retropropagarea erorii

Algoritmul de retropropagare a erorii propus de Rumelhart şi McClelland este denumit uneori şi *forma generalizată a regulii Δ.* Acest algoritm, porneşte de la un *set de date de antrenare* format din perechi *intrare – ieşire dorită* foarte asemănător modului de definire tabelară a funcţiilor în vederea aproximării. De exemplu, pentru o funcţie *f*, care depinde de trei variabile *x*, *y* şi *z*, tabelul de definiţie are forma din Fig. 13. De această dată însă variabilele *x*, *y* şi *z* de care depinde funcţia sunt tratate ca mărimi de intrare ale reţelei neuronale (notate, în general, cu x_1, x_2, … , x_n), în timp ce funcţia însăşi *f* reprezintă ieşirea reţelei (notată, în general, cu *d*). Ponderile reţelei neuronale de tip PMS se iniţializează cu valori aleatorii, alese de obicei în intervalul (-1, 1).

Aplicarea algoritmului de retropropagare se face în următoarele ipoteze: (*i*) se consideră cazul unei reţele de tip PMS care foloseşte neuroni ascunşi; (*ii*) funcţiile de activare ale neuronilor ascunşi şi ale celor de ieşire se consideră continue şi derivabile; (*iii*) dacă este cazul, mărimile de ieşire se scalează în intervale corespunzătoare funcţiei de activare folosite.

Variabile			***f(x,y,z)***
x	***y***	***z***	
……	……	……	……
……	……	……	……
……			
……	……	……	……

Fig. 13 Tabelul de definiţie pentru setul de date de antrenare al perceptronului multistrat, în cazul a trei intrări – *x, y* şi *z* – şi a unei ieşiri – *f(x,y,z)*.

Funcţionarea algoritmului de retropropagare se desfăşoară în două etape:

(a) se consideră un model *m* din setul de date de antrenare (o linie din tabelul de definiţie), din care se extrag mărimile de intrare – vectorul $x^{(m)}$ – care se aplică pe intrarea reţelei şi, folosind valorile curente ale ponderilor, se face propagarea înainte a informaţiei de intrare, calculându-se ieşirea reală furnizată de reţea, $o^{(m)}$.

(b) ieşirea reală $o^{(m)}$ se compară cu valoarea dorită $d^{(m)}$ corespunzătoare setului de antrenare şi eroarea astfel calculată se propagă înapoi în reţea – de la stratul de ieşire, spre stratul de intrare – pentru modificarea ponderilor.

Regula sau algoritmul de antrenare folosit stabileşte tocmai metoda de ajustare a ponderilor din reţea. În cazul formei generalizate a regulii Δ, ajustarea ponderilor se face în sensul minimizării abaterii între valorile reală $o^{(m)}$ şi dorită $d^{(m)}$ de pe ieşirea reţelei. Dacă paşi (a) şi (b) de mai sus se reiau pentru următorul model din setul de antrenare ($m \leftarrow m + 1$), ponderilor li se va aplica o nouă corecţie în raport cu ieşirile $o^{(m+1)}$ şi $d^{(m+1)}$. După epuizarea tuturor modelelor din setul de antrenare, se spune că s-a efectuat un *ciclu de antrenare*. Este de aşteptat ca, pe măsura considerării de noi modele din setul de antrenare, abaterile $o^{(m)} - d^{(m)}$ să se micşoreze, în general, pentru toate modelele. De cele mai multe ori, însă, un singur ciclu de antrenare nu este suficient pentru aproximarea cu suficientă precizie a tuturor valorilor de ieşire indicate în setul de antrenare. Ca urmare, algoritmul se reia pentru un nou ciclu şi procesul continuă, până la satisfacerea unui anumit criteriu de oprire.

Forma generalizată a regulii Δ propusă de Rumelhart este descrisă de următoarele trei propoziţii:

(1) Pentru fiecare model de intrare – ieşire *m* din setul de antrenare, corecţia unei ponderi w_{ij} – notată $\Delta^{(m)}w_{ij}$ – pentru conexiunea dintre neuronul *j* şi neuronul *i* din stratul inferior (vezi Fig. 14.a) este proporţională cu un termen de eroare $\delta_j^{(m)}$ asociat neuronului *j*:

$$\Delta^{(m)}w_{ij} = \eta \cdot \delta_j{}^{(m)} \cdot o_i{}^{(m)} \tag{1}$$

unde $o_i^{(m)}$ este ieşirea neuronului *i* din stratul inferior, pentru modelul *m*, iar η este un factor de proporţionalitate, numit *rată de învăţare*.

(2) Dacă neuronul *j* se află în stratul de ieşire (vezi Fig. 14.a), termenul de eroare $\delta_j^{(m)}$ se calculează în funcţie de abaterea între valoarea reală $o_j^{(m)}$ şi cea dorită $d_j^{(m)}$ şi derivata funcţiei de activare *f* a neuronului *j* în raport cu intrarea netă corespunzătoare modelului *m*, notată $net_j^{(m)}$:

$$\delta_j{}^{(m)} = \left(d_j{}^{(m)} - o_j{}^{(m)}\right) \cdot f'\left(net_j{}^{(m)}\right) \tag{2}$$

(3) Dacă neuronul *j* se află în stratul ascuns (Fig. 14.b), fiind legat prin conexiuni sinaptice cu neuronii *k* din stratul de ieşire, termenul de eroare $\delta_j^{(m)}$ este proporţional cu suma tuturor termenilor de eroare asociaţi neuronilor de ieşire *k*, modificaţi de ponderile conexiunilor respective w_{jk} şi cu derivata funcţiei de activare în raport cu intrarea netă $net_j^{(m)}$:

$$\delta_j{}^{(m)} = \left(\sum_k \delta_k{}^{(m)} \cdot w_{jk}\right) \cdot f'\left(net_j{}^{(m)}\right) \tag{3}$$

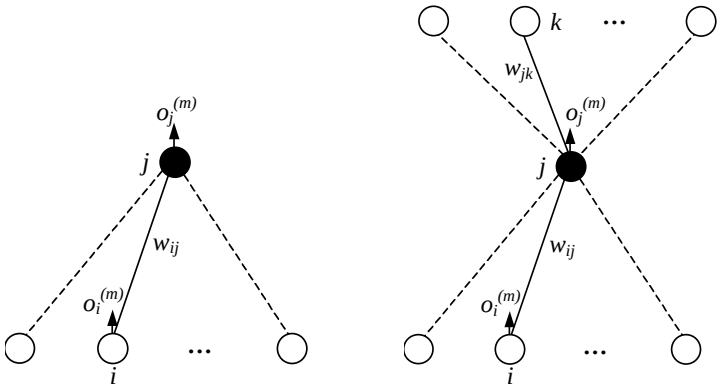

Fig. 14 Cazuri particulare de aplicare a regulii Δ, în funcţie de poziţia conexiunilor sinaptice w_{ij}: (a) neuronul *j* se află în stratul de ieşire sau (b) neuronul *j* se află în stratul ascuns.

Propoziţiile (2) şi (3) arată că ponderile asociate unui anumit neuron sunt ajustate cu termeni direct proporţionali cu abaterile dintre mărimile *reale* şi cele *dorite* corespunzătoare neuronilor cu care primul este legat.

Regulile prezentate pentru aplicarea algoritmului de retropropagare se referă strict la ponderi, fără a aminti nimic de pragurile θ_j asociate fiecărui neuron. Această formalizare nu exclude însă posibilitatea folosirii pragurilor θ_j, care pot fi modelate ca ponderi w_{ij} ale conexiunilor cu un neuron *i* din stratul imediat inferior, a cărui ieşire are întotdeauna valoare unitară.

Cele trei propoziţii care stau la baza algoritmului de retropropagare au fost doar enunţate, fără a aduce în sprijinul lor nici un suport matematic. S-a considerat însă utilă descrierea prealabilă a principiilor formei generalizate a regulii Δ, urmând ca aparatul matematic necesar să fie descris mai târziu. Deocamdată, ca o scurtă introducere, vom menţiona că, dacă pentru estimarea performanţelor reţelei neuronale se foloseşte ca metrică jumătate din abaterea pătratică totală pe stratul de ieşire (pentru un model *m* şi *J* neuroni de ieşire):

$$E^{(m)} = \frac{1}{2}\sum_{j=1}^{J}\left(d_j^{(m)} - o_j^{(m)}\right)^2 \tag{4}$$

algoritmul de retropropagare descris asigură minimizarea abaterii $E^{(m)}$.

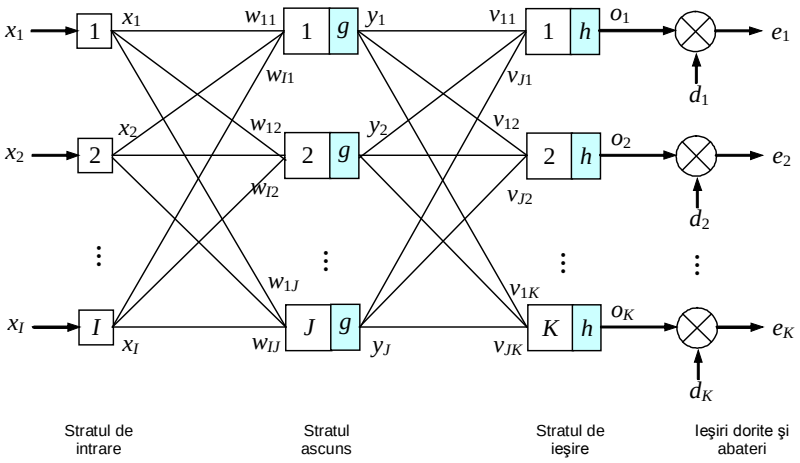

Fig. 15 Arhitectura perceptronului multistrat.

Principiul erorii pătratice minime

În cele ce urmează se consideră o reţea neuronală de tip PMS cu o arhitectură ca cea descrisă în Fig. 15. Reţeaua din Fig. 15 conţine I neuroni de intrare, J neuroni ascunşi şi K neuroni de ieşire. Ponderile conexiunilor dintre straturile de intrare şi cel ascuns, respectiv stratul ascuns şi cel de ieşire se notează cu $\mathbf{w} = \{w_{ij}\}$, respectiv $\mathbf{v} = \{v_{jk}\}$. Funcţiile de activare ale neuronilor din stratul ascuns şi cel de ieşire se notează cu $g(\bullet)$, respectiv $h(\bullet)$. În general, se recomandă ca într-o reţea de tip PMS funcţiile de activare ale neuronilor ascunşi şi celor de ieşire să fie identice. Totuşi, din considerente de claritate se preferă notarea distinctă a celor două funcţii de activare.

Antrenarea unei reţele de tipul celei din Fig. 15 se face folosind un set de date de antrenare ce foloseşte M perechi *intrare-ieşire dorită* de forma $\mathbf{x}^{(m)} = \{x_1^{(m)}, x_2^{(m)}, \ldots, x_I^{(m)}\} \div \mathbf{d}^{(m)} = \{d_1^{(m)}, d_2^{(m)}, \ldots, d_K^{(m)}\}$, $m = 1, \ldots, M$.

În aceste condiţii, pentru aproximarea cât mai corectă a ieşirilor dorite $\mathbf{d}^{(m)}$ prin ieşirile reale $\mathbf{o}^{(m)}$, se va aplica o tehnică de ajustare a ponderilor din reţea folosind ca funcţie obiectiv o estimare a erorilor de aproximare, care poate fi oricare dintre funcţiile abatere care urmează:

- *Abaterea pătratică totală*:

$$APT = \sum_{m=1}^{M} \|\mathbf{d}^{(m)} - \mathbf{o}^{(m)}\|^2 = \sum_{m=1}^{M}\sum_{k=1}^{K}\left(d_k^{(m)} - o_k^{(m)}\right)^2 \tag{5}$$

  unde $\|\bullet\|$ reprezintă norma euclidiană.

- *Abaterea pătratică parţială* (asociată unui model m):

$$APP^{(m)} = \sum_{k=1}^{K}\left(d_k^{(m)} - o_k^{(m)}\right)^2 \tag{6}$$

- *Abaterea pătratică medie totală*:

$$APMT = \frac{1}{M * K} \sum_{m=1}^{M} \|\mathbf{d}^{(m)} - \mathbf{o}^{(m)}\|^2 = \frac{1}{M * K} \sum_{m=1}^{M} \sum_{k=1}^{K} \left(d_k^{(m)} - o_k^{(m)}\right)^2 \quad (7)$$

- *Abaterea pătratică medie parţială*:

$$APMP = \frac{1}{K} \sum_{k=1}^{K} \left(d_k^{(m)} - o_k^{(m)}\right)^2 \quad (8)$$

Se verifică imediat că între cele patru tipuri de abateri există relaţiile:

$$APMT = \frac{APT}{M * K} \quad ; \qquad APMP = \frac{APP}{K} \quad (9)$$

De asemenea, la estimarea acestor abateri se pot folosi mărimi relative, raportate la valoarea datorită a ieşirii. Astfel, în relaţiile (5)-(8), termenii ($d_k^{(m)} - o_k^{(m)}$) vor fi înlocuiţi cu ($d_k^{(m)} - o_k^{(m)}$) / $d_k^{(m)}$.

În cele ce urmează, la implementarea diferitelor algoritme de retropropagare se va folosi abaterea pătratică totală *APT* care va fi înmulţită, din considerente de simplificare ulterioară a expresiilor de calcul, cu factorul ½ şi care, pentru comoditate va fi notată cu *E*:

$$E = \frac{1}{2} \sum_{m=1}^{M} \left\| d^{(m)} - o^{(m)} \right\|^2 \quad (10)$$

Abaterea *E* se consideră ca o funcţie de ponderile **w** şi **v**, care se exprimă, succesiv, sub formele:

$$E(\mathbf{w}, \mathbf{v}) = \frac{1}{2} \sum_{m=1}^{M} \|\mathbf{d}^{(m)} - \mathbf{o}^{(m)}\|^2 = = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left(d_k^{(m)} - o_k^{(m)}\right)^2 = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left[d_k^{(m)} - h\left(q_k^{(m)}\right)\right]^2 =$$

$$= \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left[d_k^{(m)} - h\left(\sum_{j=1}^{J} v_{jk} \cdot y_j^{(m)} \right) \right]^2 = = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left[d_k^{(m)} - h\left(\sum_{j=1}^{J} v_{jk} \cdot g\left(r_j^{(m)} \right) \right) \right]^2 = \quad (11)$$

$$= \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left[d_k^{(m)} - h\left(\sum_{j=1}^{J} v_{jk} \cdot g\left(\sum_{i=1}^{I} w_{ij} \cdot x_i^{(m)} \right) \right) \right]^2$$

În aceste din urmă relaţii s-au mai folosit notaţiile:

(a) *intrarea netă* a neuronului *k* din stratul de ieşire, pentru modelul *m*:

$$q_k^{(m)} = \sum_{j=1}^{J} v_{jk} \cdot y_j^{(m)} \quad (12)$$

(b) *intrarea netă* a neuronului *j* din stratul ascuns, pentru modelul *m*:

$$r_j^{(m)} = \sum_{i=1}^{I} w_{ij} \cdot x_i^{(m)} \quad (13)$$

unde $y_j^{(m)}$ este ieşirea neuronului ascuns *j*, pentru modelul *m*, iar $x_i^{(m)}$ este intrarea reţelei pe neuronul de intrare *i*, pentru modelul *m*.

Minimizarea funcţiei abatere *E* şi determinarea punctului (**w* , v***) care asigură valoarea minimă a funcţiei *E* se face folosind procedeul cunoscut al anulării derivatelor lui *E* în raport cu necunoscutele:

$$\frac{\partial E}{\partial w_{ij}} = 0 \qquad \frac{\partial E}{\partial v_{jk}} = 0 \qquad \left(i = 1,\ldots,I;\ j = 1,\ldots,J;\ k = 1,\ldots,K\right) \tag{14}$$

Relaţiile (11) şi (14) indică un sistem de ecuaţii neliniare , a cărui rezolvare se face pe cale iterativă, folosind o metodă de tip gradient, ale cărei principii de aplicare sunt prezentate în continuare.

Metode de gradient

Cea mai simplă metodă de gradient folosită în problemele de optimizare – în cazul de faţă minimizarea funcţiei abatere *E* – permite determinarea unui minim local *x** al funcţiei *f* (*x*), impunând anularea derivatei:

$$f'(x) = \frac{df}{dx} = 0 \tag{15}$$

În cazul unei funcţii *f* (*x*) neliniare, cu o formă complexă, rezolvarea directă a ecuaţiei (15) nu este posibilă; se poate aplica însă o metodă iterativă care permite determinarea unei aproximaţii a punctului de minim *x**. Pornind de la o aproximaţie iniţială x^0 şi aplicând formula de recurenţă:

$$x^{t+1} = x^t - \eta \cdot \frac{df}{dx}\Big|_{x^t} \tag{16}$$

se determină un şir de aproximaţii succesive care – în anumite condiţii – tinde către minimul local *x**. În relaţia (16) η reprezintă un factor pozitiv folosit pentru amplificarea sau atenuarea deplasării în lungul direcţiei *df / dx*. În cazul unor valori prea mari ale factorului η, punctul de minim poate fi depăşit, în timp ce valori prea mici ale lui η, pot determina o apropiere foarte lentă de punctul de minim. Relaţia (16) descrie aşa-numita *metodă a gradientului*.

În cazul algoritmului de retropropagare, pentru care funcţia obiectiv *E* depinde de mai multe variabile (vectorii **w** şi **v**), relaţia de recurenţă (16) este adusă la forma vectorială, în care derivata *df* / *dx* este înlocuită prin gradientul ∇E, ajungându-se la:

$$\begin{aligned} \mathbf{w}^{t+1} &= \mathbf{w}^t - \eta \cdot \nabla E(\mathbf{w}^t) \\ \mathbf{v}^{t+1} &= \mathbf{v}^t - \eta \cdot \nabla E(\mathbf{v}^t) \end{aligned} \tag{17}$$

iar factorul η se numeşte *rată de învăţare*.

Propagarea înapoi a erorii

Se va prezenta în continuare deducerea expresiilor pentru termenii de eroare $\delta_j^{(m)}$ din relaţiile (2) şi (3) indicate pentru forma generalizată a regulii Δ. În acest scop se va folosi arhitectura reţelei neuronale de tip PMS din Fig. 5, pentru care semnificaţia notaţiilor folosite este: *I*, *J* şi *K* – numărul de neuroni din straturile de intrare, ascuns şi de ieşire; x_i (*i* = 1,…,*I*) - intrările reţelei neuronale; y_j (*j* = 1,…, *J*) - ieşirile produse de neuronii ascunşi; o_k (*k* = 1,…,*K*) - ieşirile reţelei neuronale; d_k (*k* = 1,…,*K*) - valorile dorite pentru ieşirile reţelei neuronale; w_{ij} (*i* = 1,…,*I*; *j* = 1,…,*J*) - ponderile conexiunilor dintre straturile de intrare şi ascuns; v_{jk} (*j* = 1,…,*J*; *k* = 1,…,*K*) - ponderile conexiunilor dintre straturile ascuns şi de ieşire; *g* şi *h* – funcţiile de activare ale neuronilor din straturile ascuns şi de ieşire. Se mai folosesc, de asemenea, notaţiile: $r_j^{(m)}$ – intrarea netă a neuronului *j* din stratul ascuns, pentru modelul *m* (vezi rel. (13)); $q_k^{(m)}$ – intrarea netă a neuronului *k* din stratul de ieşire, pentru modelul *m* (vezi rel. (12)).

Dintre funcţiile obiectiv menţionate, pentru ajustarea ponderilor din reţeaua neuronală, se va folosi într-o primă etapă abaterea pătratică parţială $APP^{(m)}$, corespunzătoare modelului *m*. Cu alte cuvinte, ajustarea ponderilor se face după prezentarea fiecărui model din setul de antrenare. În finalul acestui paragraf, se va prezenta şi modul în care are loc propagarea înapoi a erorii în cazul în care antrenarea se face pe întregul lot, adică ajustarea ponderilor se face o singură dată, la fiecare ciclu, după prezentarea tuturor modelelor. În continuare, pentru simplificarea notaţiilor,

se renunţă la indicele care precizează modelul *m*, iar abaterea se notează simplu cu *E*; astfel, până la precizări contrare, în cadrul acestui paragraf notaţia *E* ţine locul lui $APP^{(m)}$.

Pentru ajustarea ponderilor se va folosi metoda gradientului. Sub formă matriceală, ecuaţiile de iterare ale acestei metode au forma:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \cdot \nabla E(\mathbf{w}^t) = \mathbf{w}^t + \Delta\, \mathbf{w}^t \qquad (18)$$
$$\mathbf{v}^{t+1} = \mathbf{v}^t - \eta \cdot \nabla E(\mathbf{v}^t) = \mathbf{v}^t + \Delta\, \mathbf{v}^t$$

Particularizarea acestor relaţii pentru una din ponderi:

$$w_{ij}^{t+1} = w_{ij}^t - \eta \cdot \left.\frac{\partial E}{\partial w_{ij}}\right|_{w_{ij}^t}$$
$$v_{jk}^{t+1} = v_{jk}^t - \eta \cdot \left.\frac{\partial E}{\partial v_{jk}}\right|_{v_{jk}^t} \qquad (19)$$

arată că ajustarea ponderilor din reţea presupune calculul derivatelor funcţiei-eroare în raport cu fiecare dintre ponderi. În continuare se va indica modul de calcul al acestor derivate pentru cazul reţelei cu un singur strat ascuns. De asemenea, se vor indica particularităţile legate de folosirea diferitelor tipuri de funcţii de activare. În final se descrie o generalizare a formulelor stabilite în cazul folosirii abaterii pătratice parţiale $APP^{(m)}$ pentru cazul antrenării pe întregul lot, când se foloseşte ca funcţie obiectiv abaterea pătratică totală *APT*.

În toate cazurile, deducerea expresiilor derivatelor din relaţia (19) se va face pe baza aşa – numitei reguli a *derivării în lanţ*. Conform acestei reguli, dacă se doreşte calculul derivatei $\partial y / \partial x$ şi există o dependenţă de forma $y = f(a)$, unde mărimea *a* depinde ea însăşi de variabila *x* ($a = f(x)$), atunci se poate scrie:

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial a} \cdot \frac{\partial a}{\partial x} \qquad (20)$$

Dacă se admite că *a* depinde de o altă mărime *b*, care la rândul ei poate fi exprimată în funcţie de variabila *x*, relaţia (20) devine:

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial a} \cdot \frac{\partial a}{\partial b} \cdot \frac{\partial b}{\partial x} \qquad (21)$$

Pe acest principiu, în măsura posibilităţii definirii unor relaţii de interdependenţă corespunzătoare, expresiile de forma (20) şi (21) pot fi extinse, prin derivare în lanţ, oricât.

Reţele neuronale de tip PMS cu un strat ascuns

Perceptronul multistrat cu un singur strat ascuns este probabil tipul de reţea neuronală cu propagare înainte cel mai des folosit în aplicaţiile practice. Arhitectura unei asemenea reţele neuronale coincide cu cea prezentată în Fig. 15.

Intrarea netă a neuronului *i* din reţea se calculează cu o formulă de tipul:

$$net_i = \sum_{j=1}^{n} w_{ji} \cdot x_j \qquad (22)$$

care corespunde intrărilor nete, notate $r_j^{(m)}$ şi $q_k^{(m)}$, pentru reţeaua din Fig. 14. Funcţia de activare *f* (corespunzătoare funcţiilor *g* şi *h* pentru reţeaua din Fig. 14) se aplică acestei intrări nete, pentru a determina ieşirea neuronului *i*:

$$out_i = f(net_i) \qquad (23)$$

Respectând aceste convenţii, în continuare se vor deduce expresiile derivatelor funcţiei-eroare în raport cu ponderile din reţea:

(A) Ponderile dintre stratul ascuns şi stratul de ieşire

$$\frac{\partial E}{\partial v_{jk}} = \frac{\partial E}{\partial o_k}\cdot\frac{\partial o_k}{\partial v_{jk}} = \frac{\partial E}{\partial o_k}\cdot\frac{\partial o_k}{\partial q_k}\cdot\frac{\partial q_k}{\partial v_{jk}} =$$
$$= \frac{\partial}{\partial o_k}\left[\sum_{p=1}^{K}(d_p - o_p)^2\right]\cdot h'(q_k)\cdot\frac{\partial}{\partial v_{jk}}\left[\sum_{p=1}^{J} v_{pk}\cdot y_p\right] = \qquad (24)$$
$$= 2\cdot(d_k - o_k)\cdot(-1)\cdot h'(q_k)\cdot y_j = -2\cdot(d_k - o_k)\cdot h'(q_k)\cdot y_j$$

(B) Ponderile dintre stratul de intrare şi stratul ascuns

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j}\cdot\frac{\partial y_j}{\partial w_{ij}} = \frac{\partial E}{\partial y_j}\cdot\frac{\partial y_j}{\partial r_j}\cdot\frac{\partial r_j}{\partial w_{ij}} =$$
$$= \frac{\partial E}{\partial y_j}\cdot g'(r_j)\cdot\frac{\partial}{\partial w_{ij}}\left[\sum_{p=1}^{I} w_{pj}\cdot x_p\right] = \frac{\partial E}{\partial y_j}\cdot g'(r_j)\cdot x_i =$$
$$= \frac{\partial}{\partial y_j}\left[\sum_{k=1}^{K}(d_k - o_k)^2\right]\cdot g'(r_j)\cdot x_i =$$
$$= \sum_{k=1}^{K}\frac{\partial}{\partial y_j}\left[(d_k - o_k)^2\right]\cdot g'(r_j)\cdot x_i =$$
$$= 2\cdot\left[\sum_{k=1}^{K}(d_k - o_k)\cdot(-1)\cdot\frac{\partial o_k}{\partial y_j}\right]\cdot g'(r_j)\cdot x_i =$$
$$= -2\cdot\left[\sum_{k=1}^{K}(d_k - o_k)\frac{\partial o_k}{\partial q_k}\cdot\frac{\partial q_k}{\partial y_j}\right]\cdot g'(r_j)\cdot x_i =$$
$$= -2\cdot\left[\sum_{k=1}^{K}(d_k - o_k)\cdot h'(q_k)\cdot\frac{\partial}{\partial y_j}\left[\sum_{p=1}^{J} v_{pk}\cdot y_p\right]\right]\cdot g'(r_j)\cdot x_i =$$
$$= -2\cdot\left[\sum_{k=1}^{K}(d_k - o_k)\cdot h'(q_k)\cdot v_{jk}\right]\cdot g'(r_j)\cdot x_i \qquad (25)$$

Expresiile astfel stabilite vor fi modificate pentru a ţine seama de forma funcţiilor de activare *g* şi *h*. Pentru aceste funcţii se vor considera două cazuri posibile: *sigmoidul unipolar* şi *sigmoidul bipolar*. Expresiile acestor două funcţii, considerând – în cazul sigmoidului unipolar – şi un prag *b*, au formele:

- sigmoidul unipolar: $f_1(x) = \frac{1}{1+e^{(-x+b)}}$
- sigmoidul bipolar: $f_2(x) = \frac{1+e^{-x}}{1-e^{-x}}$

Pentru sigmoidul unipolar, expresia derivatei este:

$$\frac{df_1}{dx} = -\frac{(-1)\cdot e^{(-x+b)}}{\left[1+e^{(-x+b)}\right]^2} = \frac{e^{(-x+b)}}{\left[1+e^{(-x+b)}\right]^2} = \frac{1+e^{(-x+b)}-1}{\left[1+e(-x+b)\right]^2}$$

$$= \frac{1}{1+e^{(-x+b)}} - \frac{1}{\left[1+e^{(-x+b)}\right]^2} = f_1(x) - \left[f_1(x)\right]^2 = f_1(x)\cdot\left[1-f_1(x)\right]$$

În mod asemănător, pentru sigmoidul bipolar, se poate scrie:

$$\frac{df_2}{dx} = \frac{(-1)\cdot e^{-x}\left(1-e^{-x}\right) - e^{-x}\cdot\left(1+e^{-x}\right)}{\left(1-e^{-x}\right)^2} = -2\frac{e^{-x}}{\left(1-e^{-x}\right)^2}$$

Dacă în expresia lui $f_2(x)$ se adună, respectiv se scade 1 în ambii termeni, se obţin relaţiile:

$$1+f_2(x) = 1+\frac{1+e^{-x}}{1-e^{-x}} = \frac{2}{1-e^{-x}} \qquad 1-f_2(x) = 1-\frac{1+e^{-x}}{1-e^{-x}} = -\frac{2e^{-x}}{1-e^{-x}}$$

Înmulţind aceste ultime două expresii din relaţia, se obţine:

$$\left[1+f_2(x)\right]\cdot\left[1-f_2(x)\right] = -4\frac{e^{-x}}{\left(1-e^{-x}\right)^2} \tag{26}$$

astfel încât, pentru derivată df_2 / dx din (.55) se obţine expresia echivalentă:

$$\frac{df_2}{dx} = \frac{1}{2}\left[1+f_2(x)\right]\cdot\left[1-f_2(x)\right] \tag{27}$$

Relaţiile (26) şi (27) au fost folosite pentru rescrierea relaţiilor de calcul a derivatelor funcţiei-eroare în raport cu ponderile (vezi Tabelul 1).

Tabelul 1 – Relaţiile de calcul ale derivatelor funcţiei-eroare în raport cu ponderile pentru reţele de tip PMS cu un singur strat ascuns şi funcţii de activare de tip sigmoidal.

Sigmoid unipolar	$\frac{\partial E}{\partial v_{jk}} = -2\cdot(d_k - o_k)\cdot o_k\cdot(1-o_k)\cdot y_j$ $\frac{\partial E}{\partial w_{ij}} = -2\cdot\left[\sum_{k=1}^{K}(d_k - o_k)\cdot o_k\cdot(1-o_k)\cdot v_{jk}\right]\cdot y_j\cdot(1-y_j)\cdot x_i$
Sigmoid bipolar	$\frac{\partial E}{\partial v_{jk}} = -(d_k - o_k)\cdot(1+o_k)\cdot(1-o_k)\cdot y_j$ $\frac{\partial E}{\partial w_{ij}} = -\frac{1}{2}\cdot\left[\sum_{k=1}^{K}(d_k - o_k)\cdot(1+o_k)\cdot(1-o_k)\cdot v_{jk}\right]\cdot(1+y_j)\cdot(1-y_j)\cdot x_i$

Propagarea înapoi a erorii în cazul antrenării pe întregul lot

Antrenarea după abaterea pătratică parţială pentru fiecare model *m*, APP^m, determină adaptarea ponderilor astfel încât se asigură reducerea erorii de aproximare pentru modelul *m*. Nimic nu garantează însă că, la prezentarea următorului model *m* + 1, adaptarea ponderilor în sensul micşorării erorii pentru acel model, nu va produce creşterea erorilor de aproximare pentru modelul *m* şi pentru toate celelalte modele prezentate deja reţelei în cadrul ciclului curent. În realitate, de cele mai multe ori chiar aşa se întâmplă şi, în consecinţă, de la un ciclu de antrenare la altul, eroarea globală de aproximare scade foarte lent.

Pentru eliminarea acestei deficienţe se poate folosi o schemă de antrenare care realizează adaptarea ponderilor o singură dată în cadrul unui ciclu de antrenare. În acest caz, după prezentarea fiecărui model *m*, ieşirile reale furnizate de reţea o_k^m, $k = 1,\ldots,K$ sunt memorate, urmând ca la sfârşitul ciclului, după prezentarea ultimului model, să se calculeze abaterea pătratică totală:

$$E = (APT) = \sum_{m=1}^{M} \sum_{k=1}^{K} \left(d_k^{(m)} - o_k^{(m)}\right)^2 \tag{28}$$

iar adaptarea ponderilor să se facă în sensul minimizării acestei funcţii-eroare globală

Caseta 1 Algoritmul de retropropagare pentru reţele neuronale de tip PMS

1. Definirea arhitecturii reţelei PMS: numărul de neuroni de pe fiecare strat (*I*, *J*, *K*) şi setul de date de antrenare: $\{\mathbf{x}^{(m)}, \mathbf{d}^{(m)}\}$ $m = 1,\ldots,M$. Definirea numărului de cicluri de antrenare: C_{max}.

2. Definirea parametrilor reţelei: ratele de învăţare pentru ponderile **v** şi **w**, notate η_1, respectiv η_2.

3. Iniţializarea ponderilor reţelei cu valori aleatorii în intervalul (-1, 1):
   $v_{jk} = 2 \cdot random(\,) - 1;$
   $w_{ij} = 2 \cdot random(\,) - 1$ ($i = 1,\ldots,I$; $j = 1,\ldots, J$; $k = 1,\ldots,K$).

4. Ajustarea poderilor:
   for $c = 1$ to C_{max} do.
   for $m = 1$ to M do.
   // Propagare înainte în primul strat
   for $j = 1$ to J do
   $y_j = 0$;
   for $i = 1$ to I do $y_j = y_j + w_{ji} \cdot x_i$
   // Propagarea înainte în al doilea strat
   for $k = 1$ to K do
   $o_k = 0$;
   for $j = 1$ to J do $o_k = o_k + v_{kj} \cdot y_j$
   for $j = 1$ to J do
   // Adaptarea ponderilor pentru al doilea strat
   for $k = 1$ to K do
   $$v_{jk} = v_{jk} + \eta_1 \cdot \left(d_k^{(m)} - o_k\right) \cdot o_k \cdot (1 - o_k) \cdot y_j$$
   // Adaptarea ponderilor pentru primul strat}
   for $i = 1$ to I do
   $$w_{ij} = w_{ij} + \eta_2 \cdot \sum_{k=1}^{K} \left[\left(d_k^{(m)} - o_k\right) \cdot o_k \cdot (1 - o_k) \cdot v_{jk}\right] \cdot o_k \cdot (1 - o_k) \cdot x_i^{(m)}$$

5. Reţeaua a fost antrenată pe cele *M* modele, în C_{mazx} cicluri, iar caracteristicile sale se găsesc în ponderile v_{jk} şi w_{ij}.

APLICAŢIE

În cadrul şedinţei de laborator se va utiliza aplicaţia specializată pentru studiul reţelelor neuronale de tip perceptron multistrat (Fig. 16). Aceasta permite aproximarea uneia din funcţiile logice elementare ŞI (AND), SAU (OR), SAU exclusiv (XOR).

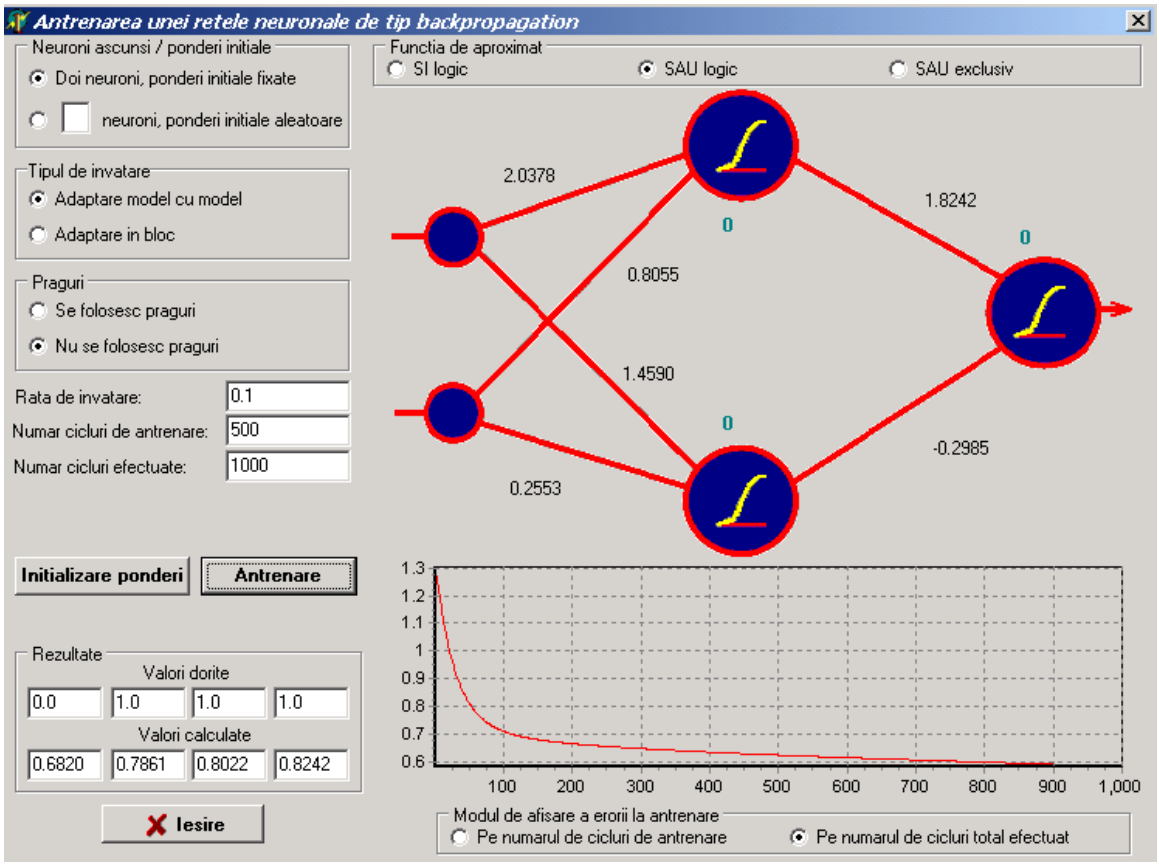

Fig. 16 Aplicaţia pentru studiul reţelelor neuronale de tip PMS

Se parcurg următorii paşi:

1. *Alegerea funcţiei ce se doreşte a fi aproximată (ŞI, SAU, SAU exclusiv)*

2. *Construirea reţelei neuronale*

   În cadrul acestui pas, se aleg numărul de neuroni de pe stratul ascuns, implicit doi, dar numărul acestora poate fi indicat şi de către utilizator. Pentru iniţializarea ponderilor, porgramul dispune, pentru reţeaua cu doi neuroni pe stratul ascuns, de un set de valori implicite, care asigură o convergenţă rapidă a procesului de calcul. Pentru reţelele cu stratul ascuns definit de utilizator, iniţializarea ponderilor se face cu ajutorul unor valori aleatoare.

   Adaptarea ponderilor se poate face model cu model sau în bloc, conform principiilor prezentate în cadrul lucrării. De asemenea, utilizarea pragurilor este opţională.

3. *Definirea parametrilor de calcul*

   În cadrul acestui pas, se indică rata de învăţare utilizată pentru procesul de antrenare şi numărul de cicluri de antrenare de efectuat.

4. Antrenarea reţelei neuronale şi analiza rezultatelor

Înainte de antrenarea reţelei, este necesară iniţializarea ponderilor. Apoi, prin apăsarea repetată a butonului “Antrenare”, se va efectua câte un set de cicluri de antrenare, până când valorile calculate de reţeaua neuronală se vor apropia de valorile dorite.

Pe parcursul procesului de calcul, programul indică: numărul de cicluri de antrenare efectuate, valoarea curentă pentru ponderi şi praguri (numai pentru cazul reţelei cu doi neuroni pe stratul ascuns), valoarea curentă a funcţiei aproximate şi, sub formă grafică, valoarea curentă a erorii.

Pentru fiecare funcţie logică modelată de program, se va efectua antrenarea reţelei neuronale pentru diferite strucuri ale stratului ascuns şi parametri de calcul. Se vor compara rezultatele obţinute şi se vor trage concluzii.