PROCESAREA SEMNALELOR ÎN REȚELE NEURONALE LINIARE

Nicoleta Liviana Tudor

Departamentul Tehnologia Informației, Matematică, Fizică

Universitatea Petrol-Gaze din Ploieşti, România

ltudor@upg-ploiesti.ro

Rezumat: Articolul tratează problema procesării semnalelor în rețele neuronale cu unități de calcul liniare şi include o analiză a metodelor de reprezentare a funcțiilor booleene.

ADALINE (Adaptive Linear Neuron or later Adaptive Linear Element) reprezintă un tmodel de rețea neuronală liniară cu un strat de neuroni, dezvoltat de Bernard Widrow şi Ted Hoff la Universitatea Stanford, în 1960. Modelul foloseşte neuroni McCulloch–Pitts şi include ponderi, valori pentru polarizare şi funcție de calcul al sumei ponderate a intrărilor. Diferența dintre Adaline şi perceptronul standard, al lui McCulloch-Pitts constă în modul de realizare a învățării, în care ponderile sunt ajustate conform sumei ponderate a intrărilor. Există, de asemenea, o extensie a rețelelor neuronale liniare, şi anume rețeaua Madaline.

În timpul antrenării, parametrii pondere şi polarizare ai unei rețele neuronale sunt actualizați într-un proces continuu de simulare a mediului în care rețeaua este generată.

O unitate de calcul generică conține două părți: o funcție de procesare care reduce argumentele la o singură valoare şi ieşirea unității dată de funcția de activare, având un singur argument de intrare.

Cuvinte cheie: procesarea semnalelor, rețea neuronală, unități liniare, reprezentarea funcțiilor booleene.

Abstract: This paper addresses the problem of signal processing in neural network with linear units and includes an analysis of the representation of Boolean functions.

ADALINE (Adaptive Linear Neuron or later Adaptive Linear Element) is a single layer neural network. It was developed by Bernard Widrow and Ted Hoff at Stanford University in 1960. It is based on the McCulloch–Pitts neuron and consists of a weight, a bias and a summation function. The difference between Adaline and the standard perceptron (McCulloch-Pitts) is that in the learning phase the weights are adjusted according to the weighted sum of the inputs. There also exists an extension of linear neural network known as Madaline.

Learning is a process when free parameters (weights and bias levels) of a neural network are adapted and adjusted through a continuing process of stimulation by the environment in which the network is embedded.

Generic computing units are split into two functional parts: an integration function reduces the arguments to a single value and the output or activation function produces the output of this node taking that single value as its argument.

Keywords: signal processing, neural network, linear units, representation of Boolean functions.

1. Introducere

Rețelele neuronale liniare sunt asemănătoare cu rețelele de perceptroni, diferența fiind dată de funcția de activare, care este liniară. Acest fapt permite ca ieşirile unei rețele liniare să poată lua orice valoare, în timp ce ieşirea unui perceptron este limitată la două valori: 0 sau 1 [1].

Principalele diferențe între rețelele liniare şi perceptroni constau în modul de interpretare a ieşirii rețelei şi în algoritmul de învățare. Perceptronii pot rezolva probleme de decizie (de exemplu probleme de clasificare), iar rețelele liniare sunt destinate în special rezolvării problemelor de aproximare [2].

O funcție $\boldsymbol{f: R \rightarrow R}$: este liniară dacă:

$x = a * x1 + b * x2 \Rightarrow f(x) = a * f(x1) + b * f(x2)$, pentru orice x, x1, x2.

Rețelele neuronale liniare pot rezolva numai probleme liniar separabile.

În categoria rețelelor uninivel, cu funcții de transfer liniare, se încadrează rețelele ADALINE (ADAptive LINear Element), respectiv MADALINE (Multiple ADAptive LINear Element), introduse de Widrow şi Hoff. Astfel de rețele pot fi utilizate pentru rezolvarea problemelor de algebră booleană, pentru reprezentarea funcțiilor liniare continue, principalele aplicații fiind regresia liniară şi filtrarea liniară a semnalelor, etc.

Rețelele liniare cu un singur nivel au aplicabilitate limitată deoarece [3]:

- pot rezolva doar probleme simple de clasificare (probleme liniar separabile);

- pot aproxima doar dependențe simple.

Aceste limitări, ale rețelelor liniare, pot fi depăşite prin includerea nivelelor ascunse, care trebuie să aibă funcții neliniare de transfer.

Funcțiile de activare liniare sunt utile la implementarea rețelelor cu un strat şi, în special, la implementarea rețelelor feed-forward, pentru procesarea semnalelor de activare ale ultimului strat.

În rețele liniare, se foloseşte regula de învățare LMS (Least Mean Squares), care este mai puternică decât regula de învățare a perceptronului. Regula Widrow-Hoff se poate aplica numai rețelelor liniare cu un singur strat, fiind o metodă de optimizare locală.

Modelul rețelelor neuronale artificiale presupune existența a două procese importante:

1. formarea unui semnal de activare a rețelei, prin procesarea valorilor semnalelor de intrare;

2. transformarea semnalului de activare într-un semnal de ieşire, folosind funcții de transfer liniare.

Să considerăm că pentru o unitate singulară, semnalele de intrare şi ponderile sunt reprezentate prin vectorul coloană $\underline{i}$ de dimensiune d, respectiv vectorul ponderilor notat $\underline{w}$. Procesarea semnalelor într-o rețea neuronală cu o unitate singulară este ilustrată în figura 1.

$$\underline{i} = \begin{pmatrix} i_1 \\ i_2 \\ . \\ . \\ i_d \end{pmatrix}, \ \underline{w} = \begin{pmatrix} w_1 \\ w_2 \\ . \\ . \\ w_d \end{pmatrix}$$

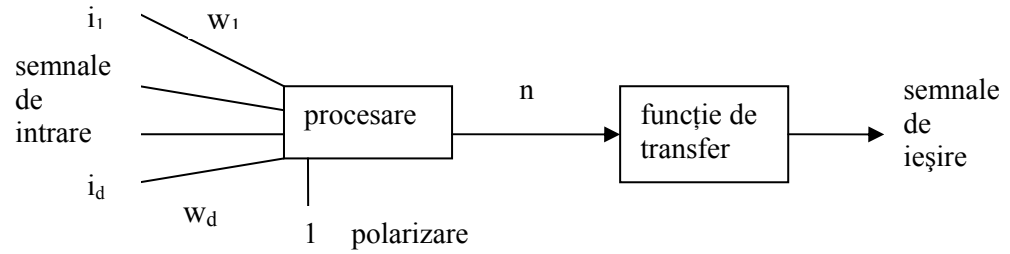


Figura 1. Procesarea semnalelor pentru o unitate singulară

2. Procesarea semnalelor de intrare

Activarea unei rețele neuronale (n), cu o unitate singulară, se poate realiza astfel [2]:

- folosind suma ponderată a intrărilor (şi a polarizării b)

$$n = w_1 * i_1 + w_2 * i_2 + \ldots + w_d * i_d + b =$$

$$= \sum_{j=1}^{d} w_j \, i_j + b = \underline{w}^T * \underline{i} + b \qquad (1)$$

- o funcție aditivă fără ponderi (suma intrărilor)

$$n = \sum_{j=1}^{d} i_j + b = i + b \tag{2}$$

- o funcție multiplicativă

$$n = \prod_{j=1,d} i_j \, b \tag{3}$$

- o funcție relațională $n = \max\{i_j b\}, j = 1,...,= \max\{i_j b\}, j = 1,...d$
- o funcție j polinomială, etc.

2.1 Unități McCulloch-Pitts

Un tip particular de unități elementare este reprezentat de modelul unităților McCulloch-Pitts (M-P), propus în 1943. În conformitate cu modelul M-P, intrările neuronilor pot fi [4]:

- inhibitoare;
- excitatoare.

Procesarea semnalelor de intrare pentru modelul M-P al unui neuron artificial, este reprezentată în figura 2.

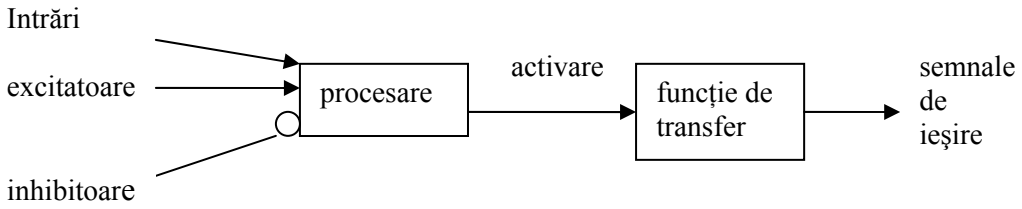


Figura 2. Modelul M-P pentru o unitate elementară

Se poate utiliza o unitate M-P pentru realizarea unor funcții logice (cu ieşiri 0 şi 1).

Parametrii modelului M-P pot fi considerați [4]:

- valoarea de prag (notată Prag);
- suma intrărilor excitatoare activate (notată Iex);
- suma intrărilor inhibitoare activate (notată Iin).

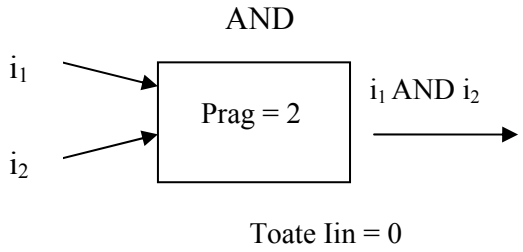
Neuronul se activează (se aprinde) când ieşirea = 1 sau poate să nu se activeze, când ieşirea = 0. Ieşirile unei unități M-P se pot determina astfel (Tabelul 1):

Tabelul 1

Iex	Iin	Ieşire
Iex ≥ Prag	Iin = 0	1 → aprindere (activare)
Iex ≥ Prag	Iin > 0	0 → neactivare
Iex < Prag	Iin = 0	0 → neactivare
Iex < Prag	Iin > 0	0 → neactivare

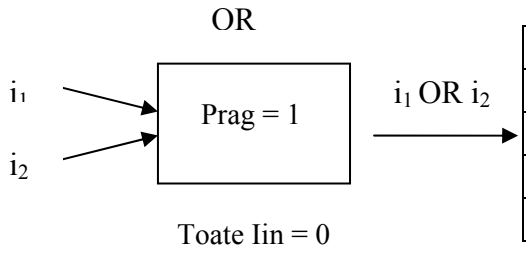
În concluzie, neuronul se aprinde numai dacă suma intrărilor excitatoare activate este ≥ valoarea Prag şi nu există intrări inhibitoare activate.

Exemplu: Reprezentarea funcțiilor logice AND şi OR, cu ajutorul modelului M-P (figura 3a şi figura 3b).



i_1	i_2	Iex	Ieşire
1	1	Iex = Prag = 2	1
1	0	Iex = 1 < 2	0
0	0	Iex = 0 < 2	0
0	1	Iex = 1 < 2	0

Figura 3a. Realizarea funcției AND folosind modelul M-P



i_1	i_2	Iex	Ieşire
1	1	Iex = Prag = 1	1
1	0	Iex = 1	1
0	0	Iex = 0 < 1	0
0	1	Iex = 1	1

Figura 3b. Realizarea funcției OR folosind modelul M-P

O altă variantă a modelului M-P modificat [4], în care rolul intrărilor inhibitoare este diminuat, funcționează astfel (Tabelul 2):

Tabelul 2

Intrări	Ieşire
Iex - Iin ≥ Prag	1 → aprindere (activare)
Iex – Iin < Prag	0 → neactivare

Un exemplu de neuron care realizează o sumă ponderată a intrărilor, urmată de o detecție de prag neliniară este WLIC-T (Weighted Linear Input Combination with Threshold = combinație liniară a intrărilor ponderate cu prag: Figura 4).

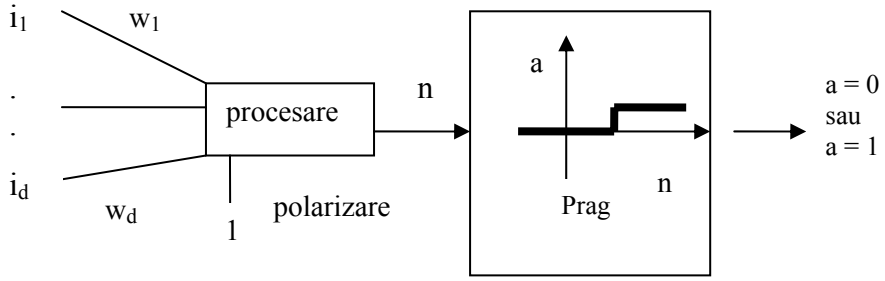


Figura 4. Unitate WLIC-T

Caracteristicile unei unități WLIC-T sunt următoarele [10]:

- realizează o combinație liniară a intrărilor ponderate;
- realizează o detecție de prag;
- ieşirea se formează printr-un proces neliniar;
- funcționarea unui astfel de neuron se exprimă astfel:

$$n = \sum_{j=1}^{d} w_j i_j + b \geq \text{Prag} \Rightarrow \text{ieşirea } a = 1 \tag{4}$$

$$n = \sum_{j=1}^{d} w_j i_j + b < \text{Prag} \Rightarrow \text{ieşirea } a = 0 \tag{5}$$

3. Funcţii de activare

Funcţiile de activare (transfer) transformă semnalul de activare a reţelei în semnal de ieşire. În funcţie de caracteristicile dorite ale semnalelor de ieşire, se pot folosi anumite tipuri de funcţii de activare [1].

Funcţiile squash sunt funcţii periodice (precum sin şi cos), care se formează prin adăugarea sau multiplicarea valorii funcţiei sau variabilei cu o constantă şi pot restrânge valorile semnalelor de ieşire la anumite intervale sau mulţimi.

Funcţia sigmoid (logistica) şi-a împrumutat denumirea de la litera greceasca ‘sigma’:

- se foloseşte în diverse domenii precum reţele neuronale artificiale, biologie, calcul probabilistic;
- este utilă, ca funcţie de distribuţie cumulativă, în studiile demografice;
- este folosită în modelarea funcţiilor de creştere [7].

$$f: R \to (0,1),\ f(n) = \frac{1}{1 + e^{-n}}, \tag{6}$$

unde n este activarea reţelei neuronale, iar ieşirea reţelei este $f(n)$.

O generalizare a funcţiei sigmoid este funcţia:

$$f: R \to R,\ f(n) = \frac{1}{1 + a * e^{-b * n}} \tag{7}$$

unde parametrul b = câştigul (se multiplică derivata funcţiei f cu b).

Derivata funcţiei f este:

$$f'(n) = \frac{\partial f(n)}{\partial n} = \frac{a * b * e^{-b*n}}{(1 + a * e^{-b*n})^2} = \frac{a * b * e^{-b*n + b - b}}{(1 + a * e^{-b*n})^2} =$$

$$= \frac{b}{1 + a * e^{-b*n}} - \frac{b}{(1 + a * e^{-b*n})^2} = b * f(n) - b * f^2(n) =$$

$$= b * f(n)\ (1 - f(n)) \tag{8}$$

Dacă ieşirea reţelei este $f(n)$, atunci ea poate fi exprimată în funcţie de derivata funcţiei sigmoid, în relaţia următoare:

$$\frac{\partial f(n)}{\partial n} = b * f(n) * (1 - f(n)) \tag{9}$$

Studiul funcţiilor de activare folosite în reţele neuronale vizează determinarea aproximativă a domeniului de valori ale ieşirilor reţelei, în cazul de faţă, determinarea valorilor derivatei funcţiei sigmoid generalizate.

4. Reţele liniare

Reţeaua liniară cu un singur neuron foloseşte o funcţie de activare liniară f (în Matlab se foloseşte funcţia purelin, în mod implicit).

Intrările sunt reprezentate prin vectorul coloană $\underline{i} = (i_j)_{j=\overline{1,d}}$, iar vectorul ponderilor este $\underline{w} = (w_j)_{j=\overline{1,d}}$ (figura 5).

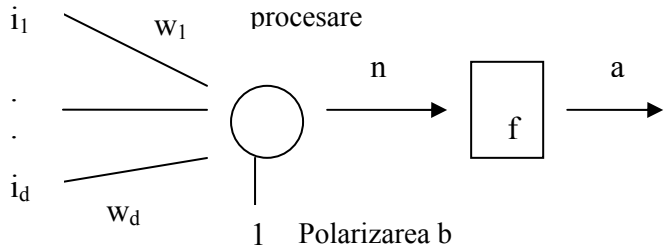


Figura 5. Reţea liniară cu un neuron

Activarea neuronului este $n = \sum_{j=1}^{d} w_j i_j + b = \underline{w}^T \underline{i} + b$, iar ieşirea este dată de relaţia

$$a = f(n) = f(\underline{w}^T \underline{i} + b) \tag{10}$$

4.1 Generalizare

Fie o reţea liniară cu un strat cu *S* neuroni şi *d* intrări (figura 6). Atunci vectorul de activare al stratului respectiv se poate determina conform relaţiei:

$$n_k = \sum_{j=1}^{d} w_{kj} i_j + b_k, \ \forall \ k = 1,..., S \tag{11}$$

$$\underline{n} = W \underline{i} + \underline{b} \tag{12}$$

unde $\underline{n} = (n_k)_{k=\overline{1,S}}$, $W = (w_{kj})_{k=\overline{1,S}, j=\overline{1,d}}$, $\underline{i} = (i_j)_{j=\overline{1,d}}$, $\underline{b} = (b_k)_{k=\overline{1,S}}$.

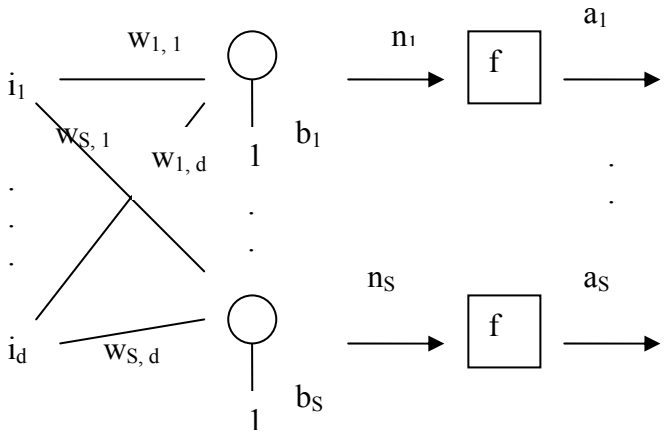


Figura 6. Rețea liniară cu d intrări şi S neuroni

Vectorul semnalelor de ieşire este $\underline{a}$:

$$\underline{a} = f(\underline{n}) = f(W\underline{i} + \underline{b}), \text{ unde } \underline{a} = (a_k)_{k=\overline{1,S}}. \quad (13)$$

4.2 Dreapta de decizie

Rețelele liniare au o regiune de decizie care este determinată de vectorii de intrare, pentru care activarea n este zero.

Pentru o rețea liniară cu un singur neuron, ecuația $\underline{w}^T \underline{i} + b = 0$ indică dreapta de decizie, unde $\underline{i} = (i_j)_{j=\overline{1,d}}$, $\underline{w} = (w_j)_{j=\overline{1,d}}$.

Pentru d =2, această ecuație reprezintă o dreaptă:

$$w_1 * i_1 + w_2 * i_2 + b = 0 \quad (14)$$

Pentru d = 3, ecuația $\underline{w}^T \underline{i} + b = 0$ reprezintă un plan, iar pentru mai multe intrări, ecuația reprezintă un hiperplan.

Pentru cazul d =2, dreapta de decizie $w_1 * i_1 + w_2 * i_2 + b = 0$ delimitează cele două regiuni de clasificare, corespunzătoare semiplanelor pozitive ($a > 0$) şi negative ($a < 0$) (figura 7).

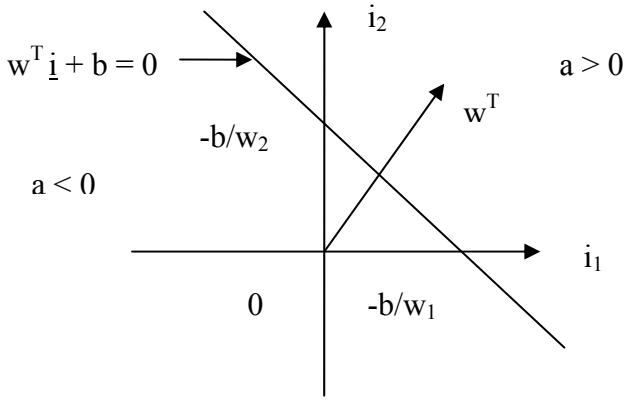


Figura 7. Drepta de decizie pentru o rețea liniară cu 2 intrări

Dreapta de decizie intersectează axa orizontală ($O i_1$) în punctul de coordonate ($-b/w_1$, 0): $i_2 = 0 \Rightarrow w_1 * i_1 + b = 0 \Rightarrow i_1 = -b/w_1$

Intersecția dreptei de decizie cu axa verticală ($O i_2$) este dată de punctul de coordonate ($0, -b/w_2$): $i_1 = 0 \Rightarrow w_2 * i_2 + b = 0 \Rightarrow i_2 = -b/w_2$

4.3 Eroarea pătratică medie

Fie o rețea liniară cu un strat cu *S* neuroni şi *d* intrări.

Ca şi regula de învățare a perceptronului, algoritmul LMS (least mean square) este un exemplu de antrenare supervizată [6], în care există un set de antrenare reprezentat de *m* perechi de intrări cunoscute şi ieşiri dorite (ieşiri target):

$$(i_1, t_1), (i_2, t_2), \ldots, (i_m, t_m),$$

unde $i_j \ (j = 1,...,m) \in R^d$, iar $t_j \ (j = 1,...,m) \in R^S$

Pentru fiecare intrare aplicată, se calculează ieşirea rețelei şi se compară cu ieşirea țintă (dorită). Eroarea se calculează ca o diferență între ieşirea țintă şi ieşirea curentă. De exemplu, eroarea la pasul *j (j = 1,...,m)* reprezintă o măsură a distanței dintre ieşirea produsă de rețea a_j şi ieşirea dorită t_j:

$$e_j = t_j - a_j \tag{15}$$

Antrenarea rețelei vizează minimizarea erorii medii pătratice pentru setul de antrenare (media pătratelor diferențelor):

$$mse = \frac{1}{m}\sum_{i=1}^{m} e_i^{\,2} = \frac{1}{m}\sum_{i=1}^{m}(t_i - a_i)^2, \tag{16}$$

unde $a_i = f(n_i) = f(\sum_{j=1}^{d} w_{ij} p_j + b_i), \forall\, i = 1,...,S, t_j, a_j \ (j = 1,...,m) \in R^S$

În Matlab, funcția de performanță a erorii este mse (mean squared error).

Algoritmul LMS (least mean square) de antrenare a unei rețele liniare ajustează ponderile şi pragurile (polarizările) pentru a minimiza această eroare medie. Scopul antrenării este de a minimiza numărul de puncte care nu sunt clasificate corect.

5. Probleme de algebră booleană

Rețelele liniare pot rezolva probleme liniar separabile, precum reprezentarea funcțiilor booleene.

În Matlab, antrenarea unei rețele liniare poate utiliza funcțiile adapt sau train (trains sau trainb), ambele actualizând ponderile şi polarizările conform regulii de învățare Widrow-Hoff (learnwh) [5].

Exemplu. Se poate proiecta o rețea neuronală liniară pentru realizarea funcției logice NAND (AND Negat), iar pentru antrenare se poate folosi algoritmul de antrenare Widrow-Hoff.

Funcția logică NAND se foloseşte în diverse aplicații:

- în circuitele electronice, ca semnale logice de intrare. Se produc semnale de ieşire conform regulilor funcțiilor logice AND/NOT şi OR/NOT;
- în logica matematică (NAND este cunoscută sub denumirea de funcție Sheffer).

Funcția NAND cu două intrări a şi b se poate reprezenta (tabelul 3):

Tabelul 3. Funcția NAND

a	b	a AND b	a NAND b = NOT (a AND b)
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	1

Algoritmul de antrenare Widrow-Hoff determină [8]:

- valorile modificate ale ponderii (derivata ponderii dW) pentru un neuron, o intrare (p) şi o eroare (e)

>>dw = lr * e * p' % p' este transpusa lui p

- valorile actualizate ale polarizării (derivata biasului db) pentru o anumită rată de învățare (parametrul lr), conform regulii de învățare:

>>db = lr * e

Funcția de performanță a rețelei liniare este mse (mean squared error):

>>t; % ieşirea țintă

>>a = sim(net, p); % ieşirea determinată de rețea

>>e = t - a

>>perf = mse(e)

Algoritmul LMS (Least Mean Squares) - algoritmul Widrow-Hoff se bazează pe minimizarea erorii medii pătratice. Pentru o rețea liniară cu un set de antrenare având *m* perechi (intrări şi ieşiri dorite), antrenarea rețelei cu algoritmul LMS converge la soluția finală, după parcurgerea paşilor (epocilor):

- pentru fiecare vector de intrare p_i (i = 1, …,m) se actualizează ponderile şi pragurile;
- se calculează ieşirile a_i produse de rețea şi se compară cu ieşirile Targets t_i;
- se calculează eroarea instantanee [9]:
  $e_i = t_i - a_i$, unde i = 1, …, m (17)
- se foloseşte regula Widrow-Hoff pentru a calcula variația ponderilor;
- se calculează eroarea pătratică medie pentru valorile e_1, …, e_m;
- dacă eroarea (parametrul goal) este atinsă (goal = met) sau numărul maxim de epoci a fost realizat, antrenarea se opreşte;
- altfel, antrenarea trece la următoarea epocă .

Observație: Dacă se atinge scopul propus (parametrul goal), atunci rețeaua este convergentă şi produce ieşirile dorite pentru toți vectorii de intrare.

6. Concluzii

Acest articol prezintă modul de procesare a semnalelor într-o rețea neuronală liniară şi un studiu de caz pentru rezolvarea problemelor de algebră booleană.

Este prezentat modelul unităților McCulloch-Pitts, conform căruia intrările neuronilor pot fi inhibitoare şi excitatoare. Procesarea semnalelor în rețele neuronale liniare presupune existența a două procese importante: formarea unui semnal de activare a rețelei şi transformarea

semnalului de activare într-un semnal de ieşire, folosind funcţii de transfer liniare.

Studiul de caz prezintă modul de reprezentare a funcţiilor booleene folosind reţele neuronale liniare şi antrenate cu regula de învăţare Widrow-Hoff.

BIBLIOGRAFIE

1. **ANDERSON, JAMES:** Neural models with cognitive implications. In LaBerge & Samuels, Basic Processes in Reading Perception and Comprehension Models. Hillsdale. Erlbaum, 1977, pp. 27-90.
2. **DUMITRESCU, HARITON COSTIN:** Reţele neuronale. Teorie şi aplicaţii. Editura Teora, Bucureşti, 1996, 460 p.
3. **KRÖSE, BEN; PATRICK VAN DER SMAGT:** An introduction to Neural Networks. University of Amsterdam. Netherlands, 1996, 135 p.
4. **MOISE, ADRIAN:** Reţele neuronale pentru recunoaşterea formelor. Editura MATRIX ROM, Bucureşti, 2005, 309 p.
5. **MATLAB** Online Help, version 7.1. 2005
6. **ROJAS, RAÚL:** Neural Networks A Systematic Introduction. Springer-Verlag, Berlin, 1996.
7. **SCHALKOFF, ROBERT:** Pattern Recognition Statistical. Structural and Neural Approaches. John Wiley & Sons, New York, 1992.
8. **WIDROW; STERNS:** Adaptive Signal Processing. New York, Prentice-Hall, 1985.
9. **WIDROW; WINTER:** Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition. Computer 21, 1988, pp. 25-39.
10. **YU HEN HU; JENQ-NENG HWANG**: Handbook of Neural Network Signal Processing. Electrical Engineering & Applied Signal Processing Series. CRC Press. 2001, 408 p.