CONSIDERAŢII PRIVIND UTILIZAREA REŢELELOR NEURONALE ÎN IDENTIFICAREA SISTEMELOR NELINIARE

ing. Constantin Voloşencu

Universitatea "Politehnica", Timişoara

Rezumat: Lucrarea prezintă un studiu privind posibilitate utilizării reţelelor neuronale de tip feedforward în identificarea sistemelor neliniare. Se introduc câţiva indicatori pentru analiza calităţiireţelelor neuronale. Ca exemplu de calcul se alege identificarea unui regulator fuzzy cu metoda identificării neuronale directe. Se prezintă şi se interpretează valorile indicatorilor de calitate ai reţeleineuronale, determinaţi pentru diferite metode de antrenare şi pentru diferite structuri de reţea neuronală. Calculele s-au efectuat cu ajutorul bibliotecii de reţele neuronale a mediului de programare Matlab.

Cuvinte cheie: identificare neuronală, reţele neuronale feedforward, metode de antrenare a reţelelor neuronale, propagarea înapoi a erorii, metoda gradientului conjugat, regulatoare fuzzy.

1. Introducere

Reţelele neuronale reprezintă o provocare pentru oamenii de ştiinţă, o soluţie care se speră că va putea fi folosită cu mai mult succes în viitor în problemele de conducere inteligentă, la procese complexe şi cu parametrii variabili în timp. Pe baza lor s-ar putea dezvolta noi metode de tratare a problemelor de conducere şi luare a deciziei, bazate în special pe principalul avantaj al reţelelor neuronale şi anume posibilitatea antrenării lor pentru recunoaşterea formelor [4], [5], [8], [13]. Aceste forme pot fi modele ale proceselor complexe sau legi de conducere. În ultimele decenii, s-au dezvoltat diverse structuri de reţele neuronale şi diverse metode de antrenare a lor. În revistele de specialitate au apărut lucrări care prezintă aplicaţii ale reţelelor neuronale în domeniul identificării şi conducerii proceselor neliniare [10], [12], [14]. În practică însă, problema aplicării pe scară largă a reţelelor neuronale în sisteme de conducere este frânată de o serie de impedimente, cum ar fi: inexistenţa unor metode precise de proiectare a sistemelor de conducere bazate pe reţele neuronale în care să existe încrederea că ele pot asigura fără dubii o calitate certă, complexitatea sporită a sistemelor de dezvoltare a unui echipament de conducere bazat pe reţele neuronale, inexistenţa unor echipamente dedicate, ieftine şi uşor de folosit şi, nu în ultimul timp, neîncrederea în repetabilitatea obţinerii unor rezultate asemănătoare, chiar pentru aceeaşi structură de reţea neuronală.

Dintre avantajele utilizării reţelelor neuronale în probleme de conducere se pot aminti posibilitatea proiectării de regulatoare fără o modelare complexă a procesului condus şi posibilitatea învăţării modelului procesului şi a legii de reglare. Pe lângă aceste avantaje de ordin general, reţelele neuronale permit obţinerea unor avantaje specifice caracterului de calcul matematic al unui regulator neuronal şi anume interpolarea, ca şi la sistemele fuzzy, dar şi extrapolarea [14].

Pentru a utiliza în practică reţele neuronale, pentru fiecare caz particular trebuie să se răspundă la câteva chestiuni şi anume: -ce fel de reţea neuronală trebuie utilizată? -câte epoci de antrenare trebuie efectuate? -ce metodă de antrenare trebuie utilizată? -câte straturi ascunse trebuie să fie folosite în structura reţelei? -câţi neuroni trebuie utilizaţi pe fiecare strat? -ce fel de funcţii de activare trebuie utilizate pentru neuronii din reţea? -ce fel de model trebuie utilizat pentru antrenarea reţelei? -câte conexiuni trebuiesc realizate în reţea? -care sunt valorile ponderilor pentru fiecare conexiune? -care sunt valorile polarizărilor pentru fiecare neuron din reţea? -cum se implementează reţeaua rezultantă? - ce indicatori de calitate asigură reţeaua neuronală obţinută?

Mulţi cercetători au încercat să dea răspuns la aceste întrebări, pentru diverse aplicaţii concrete dar, nu s-a reuşit încă să se elimine complet caracterul empiric şi nesistematic de sinteză a structurilor de conducere bazate pe reţele neuronale. Răspunsurile la aceste întrebări trebuie date raportat la valorile indicilor de calitate doriţi şi la valorile care pot fi asigurate de fiecare variantă de conducere bazată pe reţea neuronală în parte. În dezvoltarea reţelelor neuronale se pot introduce nişte indicatori de calitate specifici acestora, care vor fi prezentaţi în lucrare.

Pentru efectuarea calculelor de antrenare a reţelelor neuronale s-au utilizat bibliotecile de sisteme fuzzy şi de reţele neuronale din cadrul mediului de programare Matlab 4.2.

2. Dezvoltarea rețelei neuronale

2.1. Principiul identificării neuronale

Utilizarea rețelelor neuronale necesită un model după care acestea pot fi antrenate. În prezenta lucrare se ia ca model un regulator bazat pe logică fuzzy. După acest model se caută să se obțină un regulator bazat pe rețele neuronale sau mai simplu numit "regulator neuronal". Pentru aceasta se folosește metoda identificării neuronale directe, prezentată în [10], [13], [14].

Principiul identificării neuronale directe se prezintă, în continuare, cu referire la figura 1. O rețea neuronală primește aceleași intrări x ca și regulatorul fuzzy *RF*, iar ieșirea regulatorului fuzzy u constituie ieșirea dorită y^d pe timpul antrenării. Scopul identificării este de a gsi rețeaua neuronală al cărui răspuns y_R să fie identic cu cel al regulatorului fuzzy u pentru o mulțime dată de intrări x. În timpul identificării se minimizează norma vectorială $||u-y_R||$ printr-un număr de ajustări ale ponderilor, utilizând o tehnică de învățare (antrenare). În figura 1 se poate observa că pentru antrenarea rețelei neuronalese folosește ca și model un regulator bazat pe logică fuzzy propriu-zis *RF*. Din structura din figura 1 rezultă că numărul de neuroni de pe stratul de intrare al rețelei neuronaletrebuie să fie egal cu numărul de intrări al regulatorului fuzzy *RF*. Antrenarea se face după eroarea ε dintre ieșirea dorită și ieșirea momentană a rețelei.

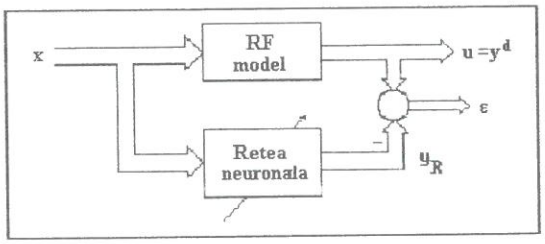

Figura 1. Schema identificării neuronale

2.2. Regulatorul fuzzy

Ca model de antrenare se utilizează un regulator fuzzy cu două mărimi de intrare *e* și *de* și o mărime de ieșire *u* (figura 2).

Baza de reguli a regulatorului conține numai 9 reguli, după cum se prezintă în tabelul 1. Ca metodă de inferență se utilizează metoda maxim-minim. Ca metodă de defuzificare, se utilizează metoda centrului de greutate [8], [13].

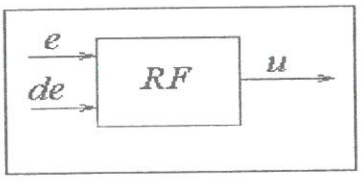

Figura 2. Regulatorul fuzzy

Tabelul 1. Baza de reguli

u		e		
		NB	ZE	PB
de	NB	NB_2	NB_4	ZE_6
	ZE	NB_8	ZE_1	PB_9
	PB	ZE_7	PB_5	PB_3

Ca universuri de discurs ale mărimilor de intrare și de ieșire se aleg universuri scalate în domeniul [1], [1] și extins la [2], [2]. Funcțiile de apartenență ale variabilelor regulatorului se prezintă în figura 3.

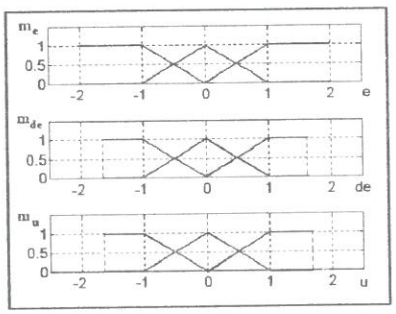

Figura 3. Funcțiile de apartenență

2.3. Încadrarea în direcțiile recente din învățarea supravegheată

Cercetările ale căror rezultate se fac cunoscute în această lucrare, se încadrează în direcțiile prezente în învățarea supravegheată [13], [14]. Se utilizează structuri feedforward multistrat proiectate global, în care toate ponderile influențează ieșirea la un moment dat, spre deosebire de așa numitele regulatoare neuronale articulate, care sunt dezvoltate pe baza unei proiectări locale. În urma antrenării, rezultă rețele neuronale, care aproximează mai mult sau mai puțin regulatorul fuzzy. Acestor rețele neuronale antrenate după un regulator fuzzy li se mai spune și sisteme neuro-fuzzy [8]. Dezvoltarea unui regulator neuro-fuzzy se încadrează în așa numita conducere supervizată [5], [8], [13], [14], [15]. Acțiunile țintă vin în mod obișnuit din înregistrarea acțiunilor umane care se iau pentru rezolvarea problemei de conducere, dar ele pot veni si de la alte sisteme de conducere. Motivația sistemelor supervizate este aceeași ca și

cea a sistemelor expert, să transpună comportarea expertului uman ("clonare"). Deci, se copiază ceea ce se presupune că ar trebuie făcut de un operator uman. După ce s-au determinat cunoştinţele umane de operare se poate dezvolta un program de calculator care să realizeze o sarcină adecvată. Conducerea supervizată are avantajul că este metoda cea mai directă şi mai sigură de succes dintre alte metode de reglare neuronală dezvoltate în teorie. Legea de conducere neuronală se poate implementa prin echipament, cu circuite integrate neuronale sau prin soft cu aşa numitele memorii neuronale. învăţarea supervizată are câteva avantaje, cum ar fi: poate fi utilizată în sisteme de reglare cu mai multe elemente de execuţie, este robustă la zgomote, permite o învăţare într-un timp relativ mai scurt [13], [14]. Ca şi principal dezavantaj se menţionează faptul că nu asigură un optim pe termen lung. Există metode de conducere neuronale mult mai sofisticate, dar pentru dezvoltarea unor structuri de reglare bazate pe astfel de reţele neuronale sunt necesare calculatoare cu o putere de calcul mult mare decât cea a simplelor calculatoare personale [13].

2.4. Niveluri în dezvoltarea structurilor de conducere bazate pe reţele neuronale

Tehnologia implementării conducerii neuronale este relativ complexă, ea putând fi comparată cu tehnologia realizării calculatoarelor. În domeniul calculatoarelor Există cel puţin trei niveluri importante de cercetare şi analiz. La nivelul cel mai scăzut se construiesc circuite. La nivelul de mijloc se combină circuitele pentru a construi calculatoare. La nivelul cel mai înalt se studiază cum să se utilizeze calculatoarele pentru rezolvarea unor probleme practice. La fel se pot evidenţia trei niveluri principale şi în dezvoltarea sistemelor conducerii neuronale [13], [14]. Astfel, în prezenta lucrare, se testează metode pentru antrenarea reţelelor neuronale, care vor avea ca rezultat construirea unui regulator care va fi un sistem cu învăţare supravegheată. Un sistem cu învăţare supravegheată este un sistem care învaţă o funcţie neliniară sau o aplicaţie statică de la un vector la un alt vector. La nivelul mediu, se va obţine o reţea, pe baza metodei de identificare neuronală directă, care să asigure indicatori de calitate specifici reţelelor neuronale cât mai buni. Nivelul superior îl va constitui utilizarea reţelei neuronale, dezvoltată în problema practică de conducere. Fiecare din cele trei niveluri ţine cont unul de altul şi lucrează unul cu altul.

Aplicaţia practică, tratată în lucrare, se încadrează în genul de aplicaţii în care se utilizează noţiunile de conducere neuronală în combinaţie cu alte noţiuni pentru a construi sisteme complexe pentru aplicaţii specifice. Astfel, se utilizează logica fuzzy, pentru a crea aşa numitele sisteme neuro-fuzzy.

Figura 4 dă o imagine a nivelurilor de interes în conducerea neuronală. La nivelul cel mai scăzut, trebuie să construim sisteme cu învăţare supravegheată, bazate pe propagarea înapoi. La nivelul intermediar, se consideră identificarea neuronală, iar nivelul superior îl constituie utilizarea în probleme de conducere.

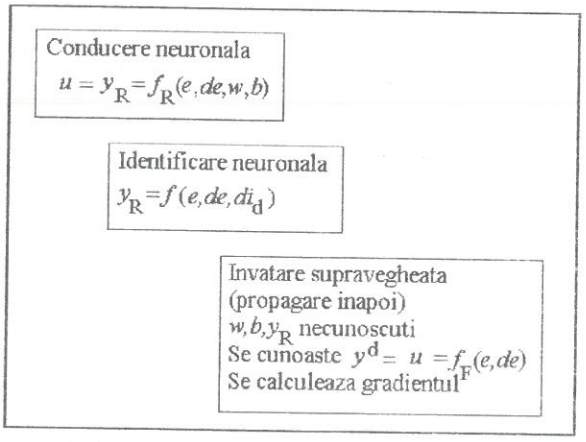

Figura 4. Dezvoltarea reţelelor

2.5. Structura reţelelor neuronale

Ca reţea neuronală, se utilizează o reţea neuronală feedforward, cu polarizări pentru fiecare neuron. Schema de principiu a unei astfel de reţele neuronale feedforward multistrat se prezintă în figura 5. Intrările *e* şi *de* ale regulatorului fuzzy sunt în acelaşi timp intrările x_1 respectiv x_2 pentru reţeaua neuronală. Eroarea de antrenare ε se obţine ca fiind diferenţa dintre ieşirea regulatorului fuzzy y_f, care este ieşirea dorită pentru reţea y^d şi ieşirea reţelei y_R:

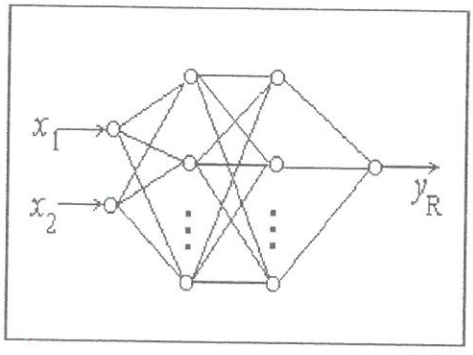

Figura 5. Schema reţelei neuronale

$$\varepsilon = u - y_R \qquad (1)$$

Reţeaua neuronală are două intrări, prima x_1, pentru variabila *e* şi a doua x_2 pentru variabila *de* şi o ieşire y_R pentru variabila *u*. Reţeaua neuronală are polarizări determinate pentru intrări unitare la fiecare neuron din reţea. Pe schema din figura 5 nu s-au mai desenat conexiunile polarizărilor. Funcţia de activare a unui neuron *i* oarecare din reţea este de forma:

$$y_i = f_a\left(\sum_j w_{ij} y_j + b_i\right) \qquad (2)$$

unde y_i este ieşirea neuronului *i*, f_a este Funcţia de activare, w_{ij} sunt ponderile de la neuronul anterior y_j la neuronul *i* şi b_i este polarizarea neuronului y_i. Schema bloc a unui astfel de neuron artificial general se prezintă în figura 6. Straturile ascunse au ca funcţii de activare f_a funcţia tangentă hiperbolică $f_a(x)=thx$. Neuronul de ieşire are ca funcţie de activare funcţia liniar $f_a(x)=x$.

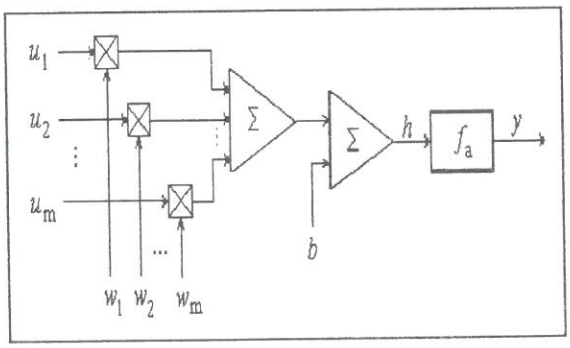

Figura 6. Schema bloc a neuronului artificial

Prin antrenare, se caută să se găsească o funcţie neliniară $u = f_R(e, de)$ cu variabilele regulatorului fuzzy, definite pe universurile de discurs: $e \in [-2, 2]$, $de=[-2, 2]$ şi $u=[-2, 2]$.

2.6. Teorema lui Kolmogorov utilizată în antrenarea reţelelor neuronale

În stabilirea dimensiunilor unei reţele neuronale feedforward multistrat, mulţi autori utilizează teorema lui Kolmogorov de aproximare a unei funcţii neliniare de mai multe variabile cu ajutorul altor funcţii [7]. Această teoremă se referă la reprezentarea funcţiilor continue cu ajutorul sumelor şi a superpoziţiilor de funcţii neliniare. Enunţul original al acestei teoreme se dă în continuare:

Există un set de funcţii continui crescătoare $h_{kj} : I = [0,1] \rightarrow R$ astfel încât orice funcţie f continuă pe I^m poate fi scrisă sub forma:

$$f(x_1,\ldots,x_m) = \sum_{j=1}^{2m+1} g_j \left[\sum_{k=1}^{m} h_{kj}(x_k)\right] \qquad (3)$$

unde g_j sunt funcţii continui de o variabilă, alese într-un mod potrivit.

Teorema lui Kolmogorov arată că orice funcţie continuă de mai multe variabile poate fi reprezentată exact prin intermediul unei superpoziţii de funcţii continue de o singură variabilă şi prin operaţii de adunare.

Utilizând limbajul reţelelor neuronale, se poate interpreta teorema lui Kolmogorov după cum urmează. Orice funcţie continuă, definită pe un cub *m*-dimensional poate fi implementată exact printr-o reţea neuronală feedforward cu 4 straturi, care are $m(2m+1)$ unităţi cu funcţii continui crescătoare $h_{kj} : I \rightarrow R$ pe primul strat ascuns şi $2m+1$ unităţi cu funcţii continue g_j pe al 2-lea strat ascuns [7].

Din păcate, teorema lui Kolmogorov nu dă şi metoda de alegere a funcţiilor de o variabilă *g* şi *h*.

În teoria reţelelor neuronale, aceste funcţii de aproximare se realizează cu ajutorul funcţiilor de activare de pe cele două straturi ascunse f_{a1} şi f_{a2}, stratul de ieşire având ca funcţie de activare funcţia liniar $y=x$. Valorile ponderilor conexiunilor şi ale polarizărilor neuronilor intră în expresiile acestor funcţii continue şi crescătoare. Ele se determină prin metode de învăţare.

În cazul din lucrare, dimensiunea cubului este $m=2$, adică funcţia care se aproximează $f_F(e, de)$ este o funcţie de două variabile. Pe baza acestei teoreme, ar rezulta că numărul neuronilor de pe primul strat trebuie să fie $m(2m+1)=10$ şi numărul de neuroni de pe al doilea strat ascuns trebuie să fie $2m+1=5$.

În alegerea structurilor optime, se va ţine cont şi de aceste considerente.

După cum s-a precizat, teorema lui Kolmogorov nu precizează tipul funcţiilor neliniare care se pot lua pentru a se realiza o bună aproximare. În articolul [7] se dau câteva teoreme referitoare la posibilităţile de aproximare ale reţelelor neuronale feedforward pentru cazul concret în care se utilizează funcţii neliniare de tip sigmoidă. Fără a prezenta aceste teoreme, amintim numai la ce se referă ele. Astfel, pe baza lor se arată că este posibil să se aproximeze orice funcţie neliniară cu ajutorul unei reţele neuronale feedforward cu două straturi ascunse, având ca funcţii de activare funcţii de tip sigmoidă. Pentru aceasta, sunt necesari $mr(r+1)$ neuroni pe primul strat ascuns şi $r^2(r+1)^r$ neuroni pe al doilea strat ascuns, unde *r* este un număr natural, cu proprietatea că $r \geq 2m+1$ (*m* este numărul de

neuroni de pe stratul de intrare) şi valoarea lui r se determină corelat cu valoarea care se doreşte pentru eroarea de aproximare. Din păcate, aceste considerente pot fi aplicate numai în cazul în care există la dispoziţie echipamente de calcul foarte puternice, cum ar fi calculatoare paralele sau circuite neuronale cu sute de neuroni. Această observaţie apare, evident, dintr-un simplu calcul. Dacă pentru aplicaţia din lucrare este necesară o reţea cu doi neuroni pe stratul de intrare m=2, atunci rezultă că o valoare minimă pentru r ar fi 5. Deci, pe primul strat ascuns, sunt necesari 60 de neuroni şi pe al doilea strat ascuns vor fi necesari 900 de neuroni. Or, o astfel de reţea neuronală nu poate fi antrenată pe un simplu calculator de tip PC, oricât de performant ar fi el. Această observaţie va reieşi şi din rezultatele antrenărilor prezentate în lucrare.

Pentru reţeaua neuronală din lucrare s-au ales 3 structuri, cu următoarele caracteristici: 1. un strat ascuns cu 5 neuroni; 2. un strat ascuns cu 10 neuroni şi 3. două straturi ascunse cu respectiv 10 şi 5 neuroni.

2.7. Seturile de antrenare

Se utilizează un număr de n_s=16.641 seturi de antrenare de forma $(x_1, x_2; y_R)_k=(e, de; u)_k$, k=1,..,n_s. Un set $(e, de, u)_k$ se obţine din suprafaţa regulatorului fuzzy $u_k=f_R(e_k, de_k)$. Valorile elementelor unui astfel de set de antrenare se obţin prin eşantionarea universurilor de discurs ale variabilelor e şi de în n_u=128 părţi: $di_{ij} = f_{RI}(e_i; de_j)$, unde $(e_i; de_j)$=(-2+(i-1).2/n_u;-2+(j1).2/n_u), pentru i, j = 1,..., n_u+1, unde f_{RI} este Funcţia neliniară a regulatorului fuzzy. Rezultă n_s seturi $y_{Rij} = f_{Rn}(u_{1i}; u_{2j})$ = (-2+(i-1).2/n_u;-2+(j-1).2/n_u), for i,j = 1,..., n_u+1, unde f_{Rn} este funcţia neliniară a reţelei neuronale. La fiecare epocă de antrenare toate seturile de antrenare $(e, de; du_k)$, k=1,..,n_s sunt trecute prin reţea.

2.8. Metode de antrenare

Se utilizează mai multe metode de antrenare, şi anume: metoda propagării înapoi a erorii de bază, metoda propagării înapoi rapidă şi metoda gradientului conjugat în varianta Levenberg-Marquardt [1], [2], [5], [6], [8], [11], [13], [15].

Metoda gradientului descrescător poate fi foarte lentă, dacă viteza de învăţare este mică şi poate oscila dacă ea este mare. Aceste chestiuni se evită, în special problema oscilaţiilor, utilizând un termen suplimentar numit "moment". Ideea constă în a da fiecărei conexiuni w_{ij} o oarecare inerţie sau moment, astfel încât ea să tindă să-şi schimbe ponderea după o forţă de coborâre medie, pe care o simte, în loc să oscileze la fiecare pas. Atunci, viteza efectivă de învăţare poate fi crescută fără a apărea oscilaţii. Această tehnică este implementată luând o contribuţie de la pasul anterior şi dând-o fiecărei schimbări a ponderii:

$$\Delta w_{ij}(t+1) = -\eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t) \qquad (4)$$

Moment α trebuie sa aibă valori cuprinse între 0 şi 1. Se recomandă o valoare de 0,9 [5].

Nu este uşor să se aleagă valori adecvate pentru parametrii η şi α într-o problemă particulară. Mai mult, valoarea considerată cea mai bună la începutul antrenării poate să nu mai fie atât de bună mai târziu. Mulţi autori sugerează ajustarea automată a parametrilor pe măsură ce învăţarea progresează [5]. Abordarea uzuală este de a verifica dacă o actualizare particulară a ponderilor a scăzut la momentul respectiv funcţia de cost. Dacă nu, atunci procesul trebuie modificat şi viteza η trebuie redusă. Pe de altă parte, dacă mai mulţi paşi consecutivi au dus la scăderea funcţiei cost, atunci probabil am fost prea conservatori şi se poate încerca creşterea vitezei η. Aceste considerente dau următoarea schemă de alegere a vitezei:

$$\Delta\eta = \begin{cases} +a \;\; daca \;\; \Delta E < 0 \;\; consistent; \\ -b\eta \;\; daca \;\; \Delta E > 0; \\ 0 \;\; altfel. \end{cases} \qquad (5)$$

unde ΔE este variaţia sau schimbarea funcţiei de cost şi a şi b sunt constante adecvate. Înţelesul noţiunii de consistent se bazează pe ultimii k paşi.

Acest tip de schemă adaptivă poate fi făcută mai eficace, utilizând diferite viteze de învăţare. Se poate utiliza o viteză anumită η_q pentru fiecare formă q sau pentru fiecare conexiune ij. Chiar şi fără o regulă adaptivă este mai adecvat să se utilizeze diferite viteze η în aceeaşi structură de reţea. Se poate alege η_{ij} invers proporţional cu numărul de conexiuni care intră în unitatea i ("fan-in" al unităţii i) [5], [13].

2.9. Evitarea minimelor locale

Metodele bazate pe gradientul descrescător se pot bloca în minime locale ale funcţiilor de cost [3], [5], [13]. Pentru evitarea acestui lucru este foarte important alegerea dimensiunii iniţiale a ponderilor. Dacă ele sunt prea mari, Funcţia de activare de tip sigmoidă se va satura de la început şi sistemul se va bloca într-un minim local sau pe o zonă plată lângă punctul de plecare. În lucrare alegerea iniţială a ponderilor se face în modul

următor. Se aleg ponderile aleator, astfel încât amplitudinea intrării h_i în unitatea i să fie mai mică, dar nu prea mică, decât unitatea. Aceasta se realizează luând ponderile w_{ij} de ordinul a $1/k_i^{1/2}$ unde k_i este numărul conexiunilor de la unitatea i la unitatea j (fan-in al unității j).

Un tip de minime locale obișnuite sunt acelea în care două sau mai multe erori se compensează între ele. Aceste minime nu sunt prea adânci, astfel că puțin zgomot (fluctuații aleatoare) pot să le ocolească. O abordare simplă și eficientă este de a utiliza actualizarea incrementală, o formă la un moment dat, alegând formele din setul de antrenare într-o ordine aleatoare. Atunci, media realizată peste forme este evitată și alegerea aleatoare generează zgomot [5].

2.10. Compararea inițială a metodelor de antrenare

În cadrul prezentei lucrări se utilizează mai mult metode pentru antrenarea rețelelor neuronale. Se face o comparație inițială pentru un exemplu de antrenare extrem de simplu [1]. Astfel, se ia o problemă de asociere unui vector de intrare p={-1, -0,99, -0,98,..,098, 099, 1}, cu n_p=201 de elemente (sau forme de antrenare) și a unui vector de ieșire $t=\{t_i \mid t_i=\sin p_i,\ p_i \in p,\ i=1,..,n_p\}$. Se alege o rețea neuronală feedforward cu un singur neuron de intrare, un singur strat ascuns, cu 10 neuroni pe stratul ascuns. Funcțiile de activare ale neuronilor sunt respectiv: pe stratul ascuns $f_a(x)=\text{th}(x)$, iar pe stratul de ieșire $f_a(x)=x$. Ponderile și polarizările rețeleiau fost inițializate aleator. S-au folosit consecutiv pentru Această problemă următoarele metode: metoda propagării înapoi a erorii în forma de bază, metoda propagării înapoi rapidă, metoda gradientului conjugat în varianta Levenberg-Marquardt și metoda bazelor radiale. Valoarea impusă pentru eroarea pătratică la antrenare a fost de 0,01. Viteza de antrenare pentru metoda propagării înapoi a fost de 0,01. Fiecare algoritm a fost cronometrat utilizând Funcția intern " *tic-toc*" a calculatorului de calcul a duratei de calcul a unei secvențe de instrucțiuni. Pentru fiecare algoritm s-a determinat, prin cumulare, numărul de operații de calcul în virgulă flotantă cu ajutorul funcției Matlab "*flops*". Astfel de operații în virgulă mobilă sunt considerate: adunările, scăderile, înmulțirile și împărțirile a două numere reale. Fiecare dintre aceste operații sunt considerate câte o operație în virgulă mobilă. Rezultatele obținute în urma utilizării acestor metode pe acest unic caz sunt prezentate în tabelul 2, din Anexa la lucrare.

Din acest tabel comparativ se pot trage următoarele concluzii:

- Metoda propagării înapoi este metoda cea mai lentă ca și timp de calcul, necesită cele mai multe epoci de antrenare și utilizarea ei duce la numărul maxim de operații în virgulă mobilă.

- Metoda rețelelor baze radiale pare a fi metoda cea mai rapidă din punct de vedere al timpului de calcul și care la numărul minim de operații în virgulă mobilă. Dar, din experimentările efectuate va rezulta că această metodă din urmă este și mai mare consumatoare de memorie calculator. Ea nu a putut fi utilizată în cazul unei probleme mai complexe, cum este cea a identificării unui regulator fuzzy.

Din rezultatele experimentărilor efectuate pentru o problemă relativ mai complexă, cu un număr mare de seturi de antrenare, pentru identificarea regulatorului fuzzy se va vedea că metoda care asigură indicatori de calitate cei mai buni, utilizabilă pe resurse de calcul mici este rămâne metoda Levenberg-Marquardt.

3. Indicatori de calitate specifici rețelelor neuronale

În dezvoltarea sistemelor de conducere bazate pe rețele neuronale se reflectă preocupările pentru reducerea timpului de antrenare, a numărului de epoci de antrenare, a numărului de eșantioane de antrenare necesare, a numărului de straturi și neuroni și a erorii la ieșirea rețelei. Se urmărește obținerea unor înalte performanțe a rețelelor de a răspunde cât mai corect semnalelor aplicate la intrare pentru rechemarea formelor (sau modelelor) memorate. Compararea diverselor soluții de rețele neuronale se face pe baza unor anumiți indicatori de calitate empirici specifici rețelelor neuronale. Pentru a compara din punct de vedere calitativ mai multe rețele neuronale introducem câțiva indicatori de calitate, specifici metodelor de antrenare a rețelelor neuronale.

Indicatorii de calitate definiți în procesul de antrenare al unei rețele neuronale sunt definiți cu ajutorul celor mai frecvent utilizate noțiuni din teoria rețelelor neuronale și anume: eroarea de antrenare la ieșirea rețelei, pentru fiecare neuron de pe stratul de ieșire, numărul de epoci de antrenare și durata de antrenare. Un indicator de calitate important este eroarea medie pătratică la ieșirea rețelei, care dă o informație globală despre rezultatul antrenării rețelei neuronale. Acești indici de calitate se pot determina în urma unor antrenări a unor tipuri diferite de rețele neuronale, în aceleași condiții de antrenare, adică același număr de seturi de antrenare, aplicate într-o epocă de antrenare la

intrarea rețelelor. Ei sunt valabili pentru orice rețea neuronală și orice metodă de antrenare. Dar ei depind de tipul de echipament de calcul utilizat, adică de puterea de calcul utilizată.

- Eroarea de antrenare la ieșirea unui neuron de pe stratul de ieșire este:

$$\varepsilon_i = y_i - y_i^d \tag{6}$$

unde y^d reprezintă valoare dorită a ieșirii, iar y este valoare reală a ieșirii.

- Eroarea pătratică se definește ca și suma pătratelor erorilor de la ieșirea rețelei, rezultate pentru fiecare din formele aplicate la intrare, la sfârșitul antrenării, după ce s-au parcurs un anumit număr de cicluri de antrenare. Se precizează că în cursul unui ciclu de antrenare prin rețea se trece tot setul de antrenare.

$$E = \sum_{i=1}^{n}\sum_{q=1}^{p}\varepsilon_i^q \tag{7}$$

unde n este numărul de neuroni de pe stratul de ieșire și p este numărul de forme existente într-un set de antrenare, iar în acest caz eroarea ε_i^q are următoarea expresie:

$$\varepsilon_i^q = y_i^q - y_i^{qd} \tag{8}$$

unde ε_i^q este eroarea de ieșire a neuronului i de pe stratul de ieșire, obținută la aplicarea formei numărulului q la intrarea rețelei, y_i^q este ieșirea neuronului i de pe stratul de ieșire, la aplicarea formei q, iar y_i^{qd} este ieșirea dorită pentru neuronul i de pe stratul de ieșire la aplicarea formei q la intrare. Acesta este și indicatorul de eroare pentru minimizarea căruia se antrenează rețeaua cu metodele alese în lucrare.

Eroarea medie pătratică se poate calcula însumată pentru toate ieșirile rețelei neuronale $\overline{E}$ sau pentru un neuron de pe stratul de ieșire $\overline{E}_i$, cu următoarele relații:

$$\overline{E} = \frac{1}{p}E, \quad \overline{E}_i = \frac{1}{n}\overline{E} \tag{9}$$

- Eroarea medie în valoare absolută la un neuron de la ieșirea rețelei:

$$\varepsilon_m = \sqrt{\overline{E_i}} \tag{10}$$

- Numărul de epoci de antrenare rezultă în urma terminării procesului de antrenare, atunci când se atinge o valoare dorită pentru eroarea medie pătratică.

- Durata de antrenare se poate exprima și în număr de epoci de antrenare, dar practic, ea rezultă după ce s-a obținut o rețea care asigură recunoașterea formelor cu o eroare medie pătratică de o valoare impusă, dorită. Această durată de antrenare depinde de puterea de calcul a echipamentului de calcul utilizat, de metoda matematică de antrenare utilizată și de numărul de seturi de antrenare utilizate. Ea crește pe măsură ce dorim o eroare pătratică mai mică. Durata de antrenare este în realitate suma duratelor tuturor proceselor care se efectuează asupra unei rețele neuronale din momentul punerii problemei de antrenare și până la obținerea unei rețele care să atingă o eroare pătratică cât mai mică. Prin noțiunea de antrenare a unei rețele se pot înțelege mai multe faze prin care se trece în procesul de dezvoltare a unei rețele neuronale eficace. Astfel, procesul de antrenare per ansamblu al unei rețele neuronale poate fi văzut ca o înșiruire de așa numite antrenări intermediare. Printr-o antrenare intermediară se înțelege trecerea tuturor seturi de antrenare prin rețea, într-un anumit număr de epoci de. În urma unei antrenări primare rezultă o eroare pătratică, care inițial va fi mai mare decât eroarea pătratică impusă. Se mai fac alte antrenări intermediare, modificându-se structura rețelei. Abia după atingerea unei erori pătratice mici, antrenarea se poate considera terminată. Deci, în urma considerațiilor expuse se poate vorbi de durata unei antrenări intermediare, efectuate în anumite condițiiasupra rețeleișide durata de antrenare globală. În unele cazuri o antrenare globală poate dura zile, săptămâni și chiar ani, în funcție de complexitatea rețelei.

Un indicator de calitate foarte important este viteza de scădere a erorii pătratice, definit cu relația:

$$v_E = \frac{\Delta E}{\Delta n} = E_{n+1} - E_n, \quad \Delta n = 1 \tag{11}$$

unde ΔE este diferența de eroare pătratică de la antrenarea n E_n și eroarea pătratică E_{n+1} de la antrenarea n+1.

Un ultim indice de calitate, care mai poate fi luat în considerare, este numărul de operații în virgulă flotantă efectuate de calculator pe perioada unei antrenări parțiale a unei rețele neuronale.

Criteriul de comparare a rețelelor neuronale dezvoltate în practică are la bază principiile general acceptate în teoria rețelelor neuronale:

- Se consideră o comportare adecvată a unei rețele neuronale la antrenare, dacă duratele unei antrenări intermediare și ale antrenării globale sunt cât mai reduse.

- Se consideră o comportare adecvată la aplicarea formelor la intrarea rețelei, dacă valorile erorilor de la ieșirile rețelei, valorile erorii pătratice, ale erorii medii pătratice și ale erorii medii pătratice pentru un neuron de ieșire sunt scăzute.

- De asemenea, se consideră că rețeaua neuronală se comportă bine în procesul de reglare, dacă sunt îndeplinite condițiile de calitate empirice, specifice sistemelor de reglare.

Antrenarea se consideră, de obicei, încheiată numai în urma unui compromis între mai mulți indicatori de calitate.

4. Valorile indicilor de calitate obținuți în urma antrenării

În continuare, se prezintă o analiză comparativă a valorilor indicilor de calitate, obținuți în antrenările efectuate, cu referire la tabelul 3, prezentat în Anexă. În acest tabel, s-au trecut valorile următorilor indici de calitate: numărul de epoci utilizate în antrenare, durata antrenării, durata unei epoci, eroarea pătratică E și eroarea medie la neuronul de la ieșirea rețelei e_m.

Valorile indicilor de calitate, rezultați în urma utilizării într-o problemă practică a celor trei metode de antrenare, pe trei cazuri particulare trecute în acest tabel trebuiesc comparate cu valori date în tabelul 2. Valorile din tabelul 2 au fost determinate pentru o problemă simplă, reprezentată printr-o rețea mică, cu un număr redus de neuroni și un set cu puține forme de antrenare, care aproximează o funcție neliniară de o singură variabilă. În tabelul 3, sunt date valori ale indicilor de calitate rezultați într-o problemă mai complexă, care presupune rețele cu un număr mai mare de neuroni și cu un set de antrenare cu foarte multe forme. Se va vedea că, în mare, se păstrează raporturile de calitate dintre primele două metode. În schimb, timpul crește foarte mult la metoda Levenberg-Marquardt care, în plus, mai este și o mare consumatoare de memorie RAM și memorie virtuală. Cu această din urmă metodă, s-a reușit totuși să se obțină o scădere mai semnificativă a erorii la antrenare, după un număr relativ redus de epoci de antrenare. În realitate, dacă se dorește obținerea unei rețele care să aproximeze bine o funcție neliniară de mai multe variabile timpul de calcul pentru o antrenare finală, alcătuită din mai multe antrenări intermediare, indiferent de metoda utilizată este destul de mare, de ordinul orelor și chiar al zilelor. Singurul mijloc de scădere a timpului de antrenare îl constituie utilizarea unor echipamente de calcul mult mai performante.

5. Concluzii rezultate în urma antrenării rețelelor neuronale

În urma experimentărilor efectuate, se pot trage destul de multe concluzii interesante și se poate da răspuns la întrebările formulate la început.

În figura 7, se prezintă comparativ evoluția sumei pătratelor erorilor de la ieșirea rețelelor neuronale antrenate. Notațiile celor 9 cazuri prezentate sunt: *bp1*, *bp2* și *bp3* reprezintă cele trei cazuri de antrenare cu metoda propagării înapoi de bază; *bpx1*,*bpx2* și *bpx3* reprezintă cele trei cazuri de antrenare cu metoda propagării înapoi rapidă iar *lm1*, *lm2* și *lm3* și reprezintă cazurile de antrenare cu metoda Levenberg-Marquardt. Numere de la 1 la 3 reprezintă cele trei structuri de rețea neuronală. Se observă că viteza de scădere a erorii pătratice crește la metodele de antrenare în ordinea enumerată. Numărul de epoci de antrenare necesare scade la metodele utilizate tot în ordinea enumerată mai sus. Eroarea pătratică cea mai mică a putut fi atinsă, așa după cum s-a spus și după cum rezultă și din această figură cu metoda Levenberg-Marquardt, pentru rețeaua cu două straturi ascunse.

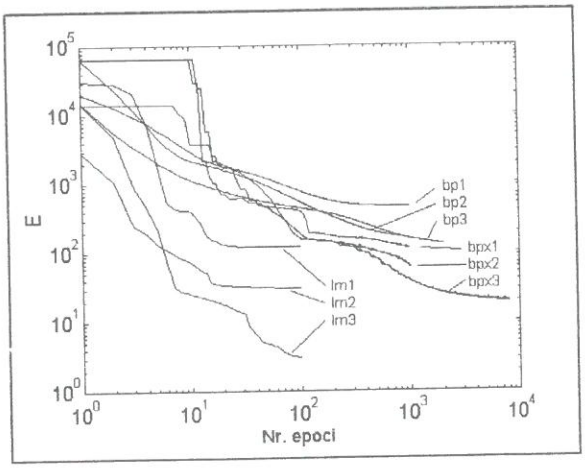

Figura 7. Evoluția sumei pătratelor erorilor pentru cazurile de antrenare

Rețeaua acceptată, în final, în urma unui compromis între indicatorii de calitate, este rețeaua obținută cu metoda de antrenare Levenberg-Marquardt, pentru care a treia structură de rețea are

următoarele ponderi şi polarizări: o matrice de ponderi de la cei doi neuroni de pe stratul de intrare la cei 10 neuroni de pe primul strat ascuns $n_{w1}(10x2)$, o matrice de ponderi de la cei 10 neuroni de pe primul strat ascuns la cei 5 neuroni de pe al doilea strat ascuns $n_{w2}(5x10)$, o matrice de ponderi de la cei 5 neuroni de pe al doilea strat ascuns la neuronul de pe stratul de ieşire $n_{w3}(1x5)$, un vector de polarizări pentru neuronii de pe primul strat ascuns $n_{b1}(10)$, un vector de polarizări pentru neuronii de pe cel de-al doilea strat ascuns $n_{b2}(5)$ şi un vector de polarizori pentru neuronul de pe stratul de ieşire $n_{b3}(1)$.

Valorile ponderilor şi ale polarizărilor sunt prezentate în tabelul 4 din Anexă, aşa cum au fost preluate ele din mediul de lucru Matlab.

În urma simulărilor efectuate, se poate observa că, utilizând metoda Levenberg-Marquardt, numărul de epoci de antrenament creşte o dată cu creşterea numărului de straturi ascunse. De asemenea, durata unei antrenări intermediare creşte foarte mult, pe măsură ce cresc numărul de straturi şi numărul de neuroni din reţea. Pe măsură ce s-a crescut numărul de straturi şi numărul de neuroni, a scăzut valoarea indicelui erorii pătratice. Pentru scăderea şi mai mult a erorii la antrenare, ar trebui dezvoltate reţele mult mai mari, conform considerentelor de extindere a teoremei lui Kolmogorov la reţele cu funcţii de activare de tip sigmoidă. Pentru acest lucru, sunt necesare, însă, echipamente de calcul mult mai puternice decât calculatoarele de tip PC.

Aceste valori ale ponderilor şi polarizărilor pot fi folosite în sistemele de reglare în care se doreşte să se implementeze regulatorul fuzzy printr-o reţea neuronală.

Se poate spune că, utilizând metoda Levenberg-Marquardt, o antrenare intermediară pentru o reţea cu două straturi se face în cel mult 80 de epoci.

Din analiza valorilor ponderilor şi a polarizărilor de mai sus, se poate trage concluzia că nu se pot curăţa conexiuni şi, cu atât mai puţin, neuroni din reţea, deoarece valorile ponderilor şi ale polarizărilor nu au decăzut într-atât încât să se poată realiza eliminarea unor conexiuni, cu atât mai puţin a unor neuroni.

Se poate spune că reţeaua neuronală finală, descrisă de funcţia neliniară de două variabile, f_{Rn} se potriveşte cu regulatorul fuzzy, descris de funcţia f_{Rf} cu o eroare pătratică de aproximativ E≈2. Aceasta înseamnă o eroare medie în valoare absolută $|\varepsilon_m|$ pentru un singur neuron în jur de:

$$|\varepsilon_m| = \sqrt{E/n_S} = 0{,}011 \qquad (12)$$

În figura 8, se prezintă variaţia erorilor pentru cele 16641 de forme aplicate la intrare, pentru cazul al treilea de antrenare. După cum se vede din această figură, erorile sunt uniform distribuite pentru toate cele n_s seturi de antrenare.

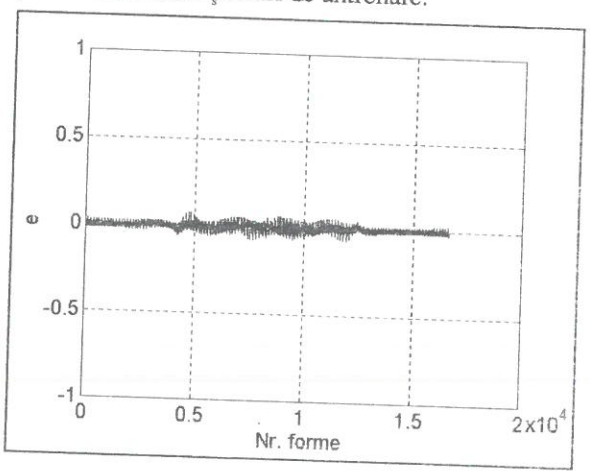

Figura 8. Repartiţia erorilor de la ieşirea reţelei pe formele de antrenare

În figurile 9, 10, 11, 12, 13 şi 14 din Anexă, se prezintă suprafeţele reţelelor neuronale, rezultate în urma antrenărilor efectuate cu metoda Levenberg-Marquardt şi, respectiv, suprafeţele erorilor introduse de reţele.

După cum se vede şi de pe suprafeţele erorilor, valorile erorilor diferă pe anumite porţiuni. Porţiunile unde erorile sunt mai mari se găsesc în jurul originii, la mijlocurile universurilor de discurs şi la capetele intervalelor.

Bibliografie

1. **BEALE, M.:** Neural Network Toolbox for Matlab, Mathworks, Inc. USA, 1995.

2. **DECLERQ, F.; R. DE KEYSER:** Neural Predictive Control of a Continuous Stirred Tank Reactor. Tempus Workshop Automation 2001, Wien, 1997.

3. **GORI, M.; TERI, A.:** On the Problem of Local Minima in Backpropagation. IEEE Trans. on Pattern Analysis and Machine Intelligence, Jan., 1992.

4. **GUPTA, M.M.; RAO, D.H.:** Neuro-Control Systems. Theory and Applications, IEEE Press, 1994.

5. **HERTZ, J., A. KROGH, R.G. PALMER:** Introduction to the Theory of Neural Computation, Addison-Wesley Publishing Co., Redwood Cliffs, USA, 1991.

6. **HOGAN, M.T.; H.B. DEMUTH, M. BEALE:** Neural Network Design, PWS Pub. Co., 1996.

7. **JIN, L.; M.M. GUPTA, P.N. NIKIFORUK:** Approximation Capabilities of Feedforward and Recurrent Neural Networks, Intelligent Control Systems. Theory Applications. IEEE Press, 1996.

8. **KOSKO, B.:** Neural Network and Fuzzy Systems, Prentice Hall, Inc., Englewood Cliffs, N. J., 1992.

9. **LOTFI, A.:** Fuzzy Inference Systems Toolbox for Matlab, Mathworks, Inc., USA, 1995.

10. **NARENDRA.K.S.:** Identification and Control of Dynamical Systems Using Neural Networks. IEEE Trans. on Neural Networks, March, 1990.

11. **SHEWCHUK, J.R.:** An Introduction to the Conjugate Gradient Method that Even an Idiot Can Understand, A Report, School of Computer Science, Carnegie Melon University, USA, 1994.

12. **VOLOŞENCU, C.:** Neural Control-A Brief Survey, Bul. Şt. şi Tehnic, UPT, Timişoara, 1996.

13. **VOLOŞENCU, C.:** Reglare fuzzy si neuronală, cu simulări în Matlab, Ed. Eurobit, Timişoara, 1997.

14. **WERBOS, P.J.:** Neural Networks for Control: An Overview. Proc. of IEEE International Conference on Automatic Control, 1990.

15. **ZURADA, J.M.:** *Introduction to Artificial Neural Systems*, West Pub., USA, 1992.

Anexă

Tabelul 2. Rezultate comparative în antrenarea reţelelor feedforward pe baza a 4 metode

Metoda de antrenare	Timpul de calcul [s]		Număr de epoci		Număr de operaţii în virgulă flotantă	
	[s]	[%]		[%]		[%]
Propagare înapoi	10,8	100	1498	100	7.803.909	100
Propagare înapoi rapidă	2,4	22,2	273	18	1.426.668	18
Metoda Levenberg-Marquardt	0,9	8,3	8	0,53	653.837	8,4
Reţele cu baze radiale	0,4	3,7	6*	-	37.403	0,48

*în cazul metodei cu baze radiale 6 reprezintă numărul de neuroni cu funcţii de activare de tip exponenţial

Tabelul 3. Valorile indicilor de calitate la antrenarea reţelelor neuronale

Metoda de antrenare	Caz de studiu (Nr. de straturi, nr. neuroni	Număr de epoci	Durata antrenării	Durata unei epoci	E	e_m
Propagarea înapoi a erorii	1 str., 5 n.	500	10 min 32 s	1,2 s	248	0,12
	1 str. 10 n.	500	20 min 21 s	2,4 s	216	0,11
	2 str., 10/5 n.	4000	4 h 23 min	4 s	106	0,08
Propagarea înapoi rapidă	1 str., 5 n.	1000	17 min	1 s	153	0,096
	1 str. 10 n.	500	25 min	3 s	124	0,86
	2 str., 10/5 n.	500	35 min	4 s	125	0,866
Metoda Levenberg-Marquardt	1 str., 5 n.	100	60 s	0,6 s	107	0,08
	1 str. 10 n.	100	2 h 30 min	1 min 34 s	30	0,04
	2 str., 10/5 n.	100	15 h 12 min	9 min 8 s	3	0,01

w1:

-2.1415	-0.2337
-2.7108	0.1821
0.0708	-2.3727
0.7387	-2.3441
0.6530	-2.7011
-0.7958	-1.4917
0.4784	2.4372
3.3444	0.0611
-0.3638	2.4951
-0.3268	2.5012

b1:

-0.0548
-2.5959
2.1729
-0.0904
0.4857
1.5210
1.0015
-3.1072
-3.6209
2.0660

b2:

1.6650
1.9095
0.4286
0.8975
-0.1743

b3:

0.8402

Tabelul 4. Ponderile şi polarizările reţelei neuronale

w2:

2.0637	-0.4325	-0.7329	1.6890	-1.7246	-0.1827	-0.8558	-0.7453	-0.4751	1.0169
0.4856	0.9393	-0.1735	-2.9587	2.1208	-0.5780	-0.9068	0.0531	0.2176	-1.3755
0.8671	-0.4014	0.8062	-0.3389	0.0147	-0.3361	-0.3469	0.4727	0.5744	-0.3243
1.8620	0.6427	-0.3738	0.5792	-0.9978	-0.5069	1.1131	0.4620	-0.0949	0.7937
-0.7359	0.5808	-0.3506	0.4656	-0.0678	0.3366	0.5960	-0.2323	-0.7318	0.6340

w3:

-0.6742	-0.5858	-1.3564	0.6184	-1.2901

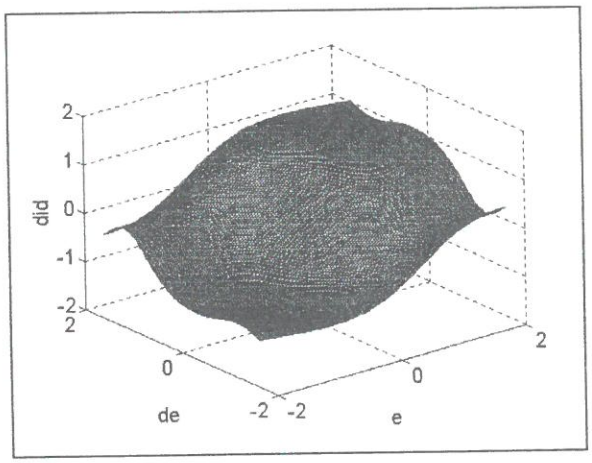

Figura 9. Suprafaţa regulatorului neuro-fuzzy cu 5 neuroni pe un singur strat ascuns

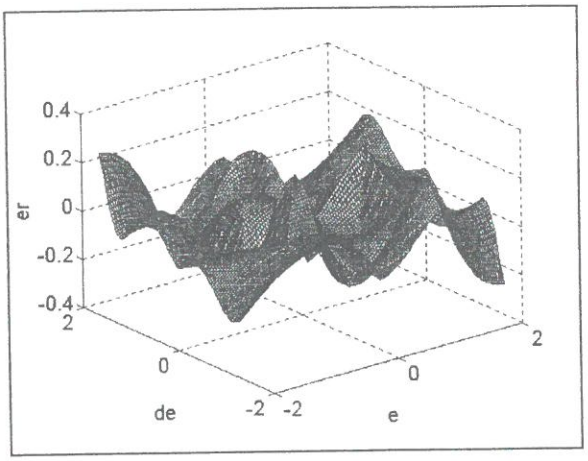

Figura 10. Suprafaţa erorilor de la ieşirea reţelei neuronale cu 5 neuroni pe un singur strat ascuns

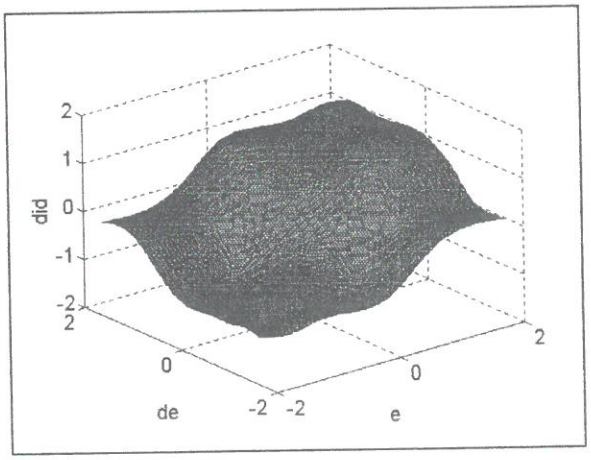

Figura 11. Suprafaţa regulatorului neuro-fuzzy cu 10 neuroni pe un singur strat ascuns

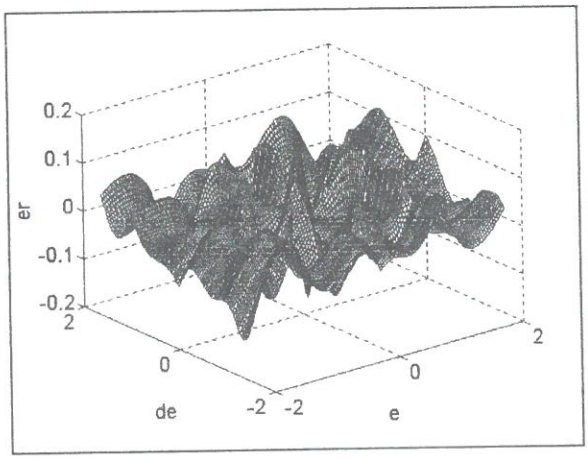

Figura 12. Suprafaţa erorilor de la ieşirea reţelei neuronale cu 10 neuroni pe un singur strat ascuns

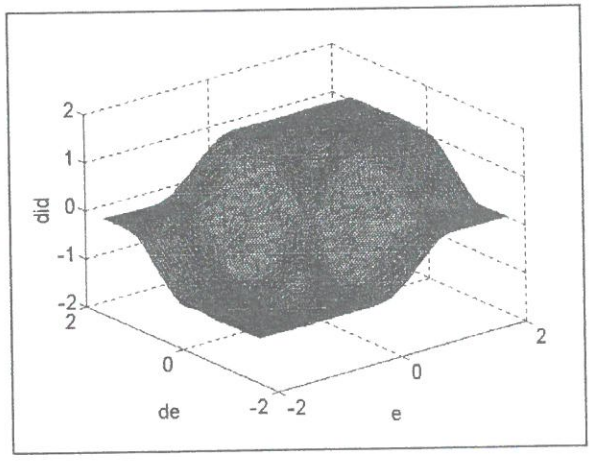

Figura 13. Suprafaţa regulatorului neuro-fuzzy cu 10 respectiv 5 neuroni pe două straturi ascunse

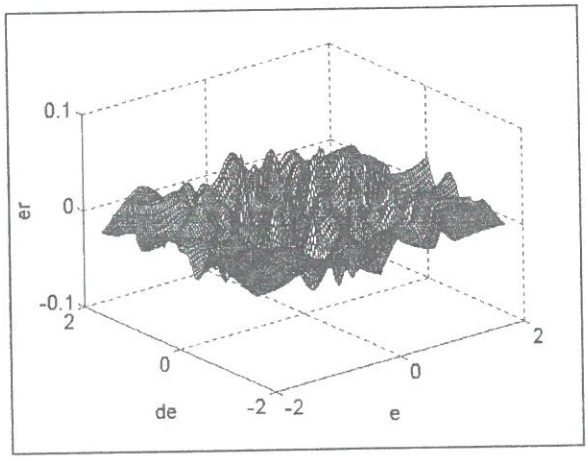

Figura 14. Suprafaţa erorilor de la ieşirea reţelei neuronale cu 10 respectiv 5 neuroni pe două straturi ascunse