APLICAȚII ALE REȚELELOR NEURALE LA PREDICȚIA UNOR FENOMENE NATURALE

geog. Costel Barbălată
mat. Laurențiu Leuștean
Institutul de Cercetări în Informatică

Rezumat: Rețelele neurale prezintă o abordare alternativă promițătoare la predicția seriilor de timp. În acest articol, se aplică rețelele neurale la predicția unor serii de timp, reprezentând măsurătorile debitelor lichide medii lunare, efectuate la cinci posturi hidrometrice reprezentative.
Cuvinte cheie: rețele neurale,feed-forward, predicție, serii de timp, învățare supervizată constructivă.

1. Introducere

Predicția fenomenelor naturale, legate de scurgerea râurilor - debitele de diferite asigurări, inundațiile, fenomenele de secare - constituie una din prioritățile hidrologiei în momentul de față. Pentru realizarea unor predicții adecvate este necesară cunoașterea modului de manifestare a fenomenului respectiv pe o perioadă îndelungată de timp, de regulă mai mare de 30 de ani. În acest fel, este favorizată dimensionarea folosințelor de apă pe baza cărora se poate trece la amenajarea complexă a râurilor pentru a putea preveni diferite evenimente nedorite sau pentru a le atenua. De asemenea, se pot semnala anumite tendințe în modul de manifestare a râurilor, care pot servi la evitarea efectelor nefaste prin aplicarea unor măsuri adecvate.

Articolul de față prezintă o aplicație a rețelelor neurale la predicția debitelor medii lunare lichide pe baza măsurătorilor efectuate la cinci posturi hidrometrice reprezentative. În acest scop, a fost folosit algoritmul *Cascade-Correlation*, un algoritm de învățare supervizată constructivă. Avantajele acestui algoritm constau în convergența mai rapidă și eliminarea necesității de a determina, a priori, topologia rețelei.

În al doilea capitol sunt prezentate pe scurt rețelele neurale *feed-forward* și algoritmul *Backpropagation*. De asemenea, este prezentat algoritmul de învățare *Quickpropagation*, folosit pentru antrenarea unităților de ieșire și candidat din algoritmul *Cascade-Correlation*. Capitolul următor tratează algoritmul de învățare *Cascade-Correlation*. Al patrulea capitol prezintă modelul experimental. Sunt analizați parametrii folosiți și sunt prezentate cele mai bune rezultate obținute pentru cele cinci serii de timp, precum și valorile parametrilor, corespunzătoare acestor rezultate.

2. Rețelele neurale

Rețelele neurale sunt structuri de procesare a informației, bazate pe modul de funcționare a creierului uman și sunt aplicate, în general, la recunoașterea formelor, analiza datelor, sisteme suport de decizie, predicția seriilor de timp etc.

O rețea neurală este o structură de procesare a informației paralelă, distribuită, care constă în unități de procesare (care pot avea o memorie locală și pot efectua operații locale de procesare a informației) interconectate prin conexiuni. Fiecare unitate de procesare are o singură conexiune de ieșire care se ramifică în conexiuni colaterale; fiecare conexiune colaterală propagă același semnal: semnalul de ieșire al unității de procesare. În fiecare unitate de procesare are loc o prelucrare a informației, care trebuie să fie complet locală; adica, trebuie să depindă numai de valorile curente ale semnalelor de intrare, ce sosesc la unitatea de procesare prin conexiuni, și de valorile înregistrate în memoria locală a unității de procesare.

Rețelele neurale își dezvoltă capacitățile de prelucrare a informației prin învățarea din exemple.Tehnicile de învățare pot fi împărțite în două categorii: învățarea supervizată și învățarea nesupervizată.

Învățarea supervizată necesită un set de exemple pentru care se știe răspunsul dorit al rețelei. Procesul de învățare constă în adaptarea rețelei astfel încât să producă răspunsul corect pentru setul de exemple. Rețeaua obținută ar trebui să fie capabilă să generalizeze adică să dea un răspuns bun atunci când sunt prezentate alte exemple.

La învățarea nesupervizată, rețeaua neurală este autonomă: prelucrează datele care îi sunt prezentate, găsește anumite proprietăți ale lor și învată să reflecte aceste proprietăți la ieșire.

Multe din tehnicile de învățare sunt legate de anumite topologii de rețele neurale.

2.1 Rețelele neurale *feed-forward* și algoritmul de învățare *Backpropagation*

Rețelele neurale *feed-forward* [1] sunt rețele în care unitățile de procesare (neuronii) sunt, în general, aranjate pe niveluri. O rețea *feed-forward*

se notează
$N_I \times N_1 \times \ldots \times N_i \times \ldots \times N_L \times N_0$, unde:

N_I reprezintă numărul unităților de intrare;

L reprezintă numărul nodurilor din nivelul ascuns $i, i = \overline{1, L}$;

N_i reprezintă numărul unităților de ieșire;

N_0 reprezintă numărul unităților de ieșire.

Figura 1 prezintă o rețea *feed-forward* tipică cu 2 niveluri, în întregime conectată, cu o structură 3x4x3.

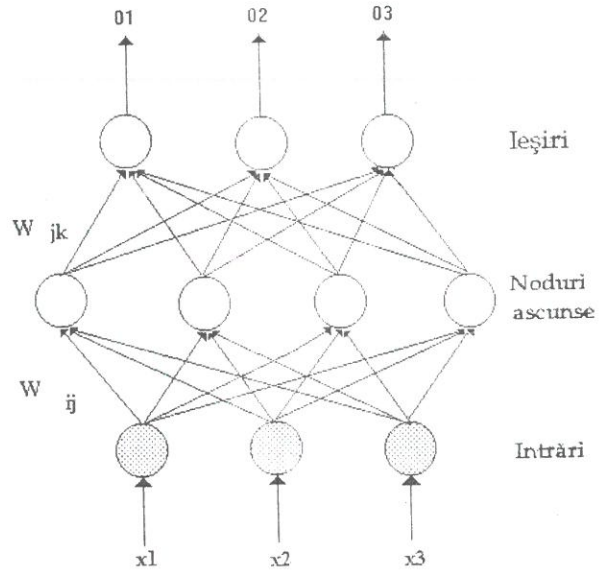

Figura 1: Rețea neurală *feed-forward* cu structura 3x4x3

Prin convenție, nu se numără nivelul de intrare, deoarece unitățile de intrare nu sunt unități de prelucrare, ele doar transmițând mai departe vectorul de intrare x. Unitățile din nivelurile ascunse și din nivelul de ieșire sunt unități de prelucrare. Fiecare unitate de prelucrare are o funcție de activare. Cea mai folosită funcție de activare este funcția sigmoidă:

$$f(x) = \frac{1}{1+e^{-x}}$$

Intrarea rețelei în unitatea de procesare j este dată de:

$$net_j = \sum_i w_{ij} x_i + \theta_j .$$

unde x_i sunt ieșirile unităților din nivelul anterior, w_{ij} este ponderea conexiunii de la unitatea i la unitatea j, iar θ_j este bias-ul unității j.

Valoarea de activare (ieșirea) unității j este dată de:

$$a_j = f(net_j) = \frac{1}{1+e^{-net_j}}$$

Se observă că:

$$f'(\text{net}_j) = \frac{e^{-\text{net}_j}}{(1+e^{-\text{net}_j})^2} = f(\text{net}_j)$$
$$(1 - f(\text{net}_j)) = a_j(1 - a_j)$$

deci derivata este o funcție simplă de ieșire.

Algoritmul de învățare *Backpropagation* este un algoritm de învățare supervizată, promovat în 1986 de D. Rumelhart și J. McClelland [1]. Acest algoritm constă în optimizarea iterativă a unei funcții de eroare, reprezentând o măsură a performanței rețelei. Funcția de eroare folosită este suma erorilor pătratice medii la ieșirile rețelei, pentru toate exemplele din setul de antrenament. Eroarea pentru un exemplu p se definește ca fiind:

$$E_p = \sum_{j=1}^{N_0} (d_{pj} - a_{pj})^2 ,$$

unde d_{pj} este valoarea dorită la ieșirea j, pentru exemplul p, iar a_{pj} este valoarea obținută la ieșirea j, pentru exemplul p.

Eroarea totală este:

$$E = \sum_{p=1}^{P} \frac{1}{2} E_p = \sum_{p=1}^{P} \frac{1}{2} \sum_{j=1}^{N_0} (d_{pj} - a_{pj})^2 ,$$

unde P este numărul de exemple de antrenament.

În timpul procesului de antrenare, este folosit un set de exemple, fiecare exemplu fiind o pereche formată din intrarea și ieșirea corespunzătoare dorită. Exemplele sunt prezentate rețelei secvențial, într-o manieră iterativă. Se realizează corecțiile ponderilor pentru a adapta rețeaua la comportarea dorită. Această iterare continuă până când ponderile conexiunilor permit rețelei să realizeze transformarea dorită. Fiecare prezentare a întregului set de antrenament se numește epocă.

Pentru a minimiza eroarea totală, se folosește metoda gradienților. Modificarea fiecărei ponderi este proporțională cu derivata măsurii erorii în raport cu ponderea respectivă.

$$\Delta_p w_{ij} = -\eta \frac{\partial E_p}{\partial w_{ij}} ,$$

unde η este un parametru care determină dimensiunea pasului, numit rata de învățare.

Derivata erorii este:

$$\frac{\partial E_p}{\partial w_{ij}} = -\delta_{pj} a_{pi},$$

unde, pentru o unitate de ieşire:

$$\delta_{pj} = f'(net_{pj})(d_{pj} - a_{pj}),$$

iar pentru o unitate ascunsă:

$$\delta_{pj} = f'(net_{pj}) \sum_k \delta_{pk} w_{jk},$$

unde δ_{pk} sunt semnalele de eroare de la nivelul următor nivelului în care se află unitatea j.

2.2. Algoritmul de învăţare *Quickpropagation*

Algoritmul *Quickpropagation* [3] diferă de *Backpropagation* prin modul în care sunt adaptate ponderile.

Regula de actualizare a ponderilor este:

$$\Delta w_{ij}(t) = \frac{S_{ij}(t)}{S_{ij}(t-1) - S_{ij}(t)} \Delta w_{ij}(t-1),$$

unde $S_{ij}(t)$ şi $S_{ij}(t-1)$ sunt panta curentă, respectiv precedentă, a erorii în raport cu ponderea $_{ij}(t)$. $S_{ij}(t)$ este definită astfel:

$$S_{ij}(t) = \frac{\partial E(t)}{\partial w_{ij}(t)} = \frac{1}{2} \sum_{p=1}^{P} \frac{\partial E_p(t)}{\partial w_{ij}(t)} =$$

$$\frac{1}{2} \sum_{p=1}^{P} -\delta_{pj} \cdot a_{pi},$$

unde $E(t)$ şi $_{pj}$ au aceleaşi valori ca în algoritmul *Backpropagation*, iar P este numărul de exemple de antrenament.

Panta erorii este derivata erorii totale E, deci semnalele de eroare $_{pj}$ trebuie să fie acumulate pentru toate cele P exemple de antrenament. Aceasta înseamnă că ponderile sunt ajustate numai după prezentarea tuturor exemplelor din setul de antrenament.

Pentru a începe regula de actualizare sau pentru a continua în cazul în care panta precedentă a erorii $S_{ij}(t-1)$ este 0, regula de actualizare a ponderilor se schimbă astfel:

$$\Delta w_{ij}(t) = \begin{cases} \dfrac{S_{ij}(t)}{S_{ij}(t-1) - S_{ij}(t)} \Delta w_{ij}(t-1) \\ \text{daca } S_{ij}(t) S_{ij}(t-1) < 0 \\ \dfrac{S_{ij}(t)}{S_{ij}(t-1) - S_{ij}(t)} \Delta w_{ij}(t-1) + \varepsilon S_{ij}(t) \\ \text{daca } S_{ij}(t) S_{ij}(t-1) \geq 0 \end{cases}$$

Prin urmare, se adaugă la pondere panta curentă înmulţită cu o rată de învăţare ε dacă panta precedentă este 0, sau dacă panta precedentă şi cea curentă au aceeaşi direcţie.

Pentru a evita paşii prea mari, în regula de actualizare a ponderilor se introduce un parametru μ:

$$\Delta w_{ij}(t) = \min(\Delta w_{ij}(t), \mu \Delta w_{ij}(t-1)),$$

unde S. Fahlmann sugereaza o valoare μ=1.75.

2.3. Învăţarea supervizată constructivă

Ceea ce caracterizează algoritmii de învăţare supervizată neconstructivi (*Backpropagation*, *Quickpropagation*) este necesitatea de a determina topologia reţelei (numărul de niveluri şi numărul neuronilor de pe fiecare nivel) înainte de aplicarea algoritmului. Acurateţea învăţării şi buna generalizare depind foarte mult de topologia reţelei alese. Determinarea celei mai bune topologii pentru o problemă şi un set de antrenament se bazează pe criterii empirice, find necesar un proces destul de lung.

Algoritmii de învăţare constructivi [4] se caracterizează prin faptul că determinarea topologiei reţelei este parte a algoritmului. De obicei, algoritmii constructivi pornesc cu o reţea minimală alcătuită numai din unităţi de intrare şi de ieşire interconectate. Apoi se adaugă treptat unităţi între unităţile de intrare şi cele de ieşire, aceste noi unităţi formând unul sau mai multe niveluri. Algoritmii constructivi existenţi diferă prin: momentul când se adaugă noi unităţi, locul unde vor fi plasate aceste unităţi şi numărul de unităţi adăugate la un moment dat. Procesul de creştere a reţelei se incheie când reţeaua a învăţat setul de antrenament la un nivel acceptabil. Algoritmii constructivi pot avea o viteză de convergenţă mai mare decât cei neconstructivi, deoarece antrenarea se face pe o reţea care creşte şi de aceea ponderile actualizate la fiecare pas sunt mai puţine decât în cazul antrenării întregii reţele la fiecare pas.

O problemă nerezolvată încă este faptul că topologia finală obţinută de algoritmii constructivi nu este neapărat cea optimă. Topologia finală depinde de metoda de construcţie a algoritmului, deci de modul în care unităţile sunt adăugate la reţea. O metodă de construcţie care să obţină

topologia optimă pentru orice problemă dată nu se cunoaşte încă. O abordare posibilă este considerarea mai multor metode de construcţie şi selectarea aceleia care obţine cele mai bune rezultate pentru problema dată, criteriul de comparaţie fiind, în general, acurateţea generalizării.

3. Algoritmul de învăţare *Cascade-Correlation*

Algoritmul de învăţare *Cascade-Correlation* este un algoritm constructiv de învăţare supervizată introdus în [2].

Arhitectura iniţială a algoritmului este ilustrată în figura 2. La începutul algoritmului, reţeaua este un perceptron cu un singur nivel, cu fiecare unitate de intrare conectată la fiecare unitate de ieşire.

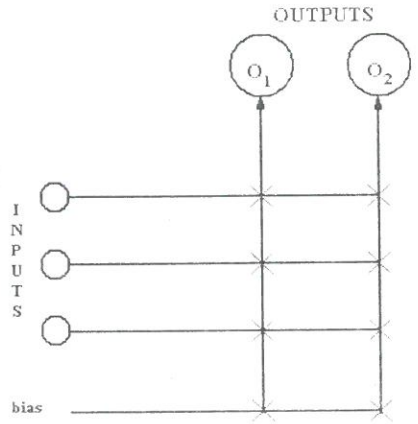

Figura 2: Arhitectura iniţială a algoritmului *Cascade-Correlation*

Algoritmul de învăţare constă din două etape: antrenarea unităţilor de ieşire şi crearea unei unităţi noi.

În prima etapă, conexiunile directe intrare-ieşire sunt antrenate cât se poate de bine pe întregul set de antrenament folosind *Quickpropagation*. Deoarece este antrenat un singur nivel, nu este necesară o propagare înapoi a erorilor. Dacă nu a apărut o reducere semnificativă a erorii după un anumit număr de epoci, întregul set de antrenament este rulat prin reţea încă o dată pentru a măsura eroarea. Dacă eroarea este mai mică decât toleranţa specificată, algoritmul se opreşte. Dacă este necesară reducerea erorii, începe etapa de creare a unei unităţi noi.

În această etapă, o nouă unitate ascunsă, conectată la toate unităţile de intrare şi la unităţile ascunse existente, este antrenată cât se poate de bine pe întregul set de antrenament, folosind *Quickpropagation*. Scopul acestei antrenări este de a maximiza corelaţia dintre ieşirea acestei unităţi ascunse şi suma erorilor obţinute la unităţile de ieşire. Dacă nu a apărut o reducere semnificativă a erorii după un anumit număr de epoci de antrenament, ieşirea noii unităţi ascunse este conectată la toate unităţile de ieşire ale reţelei.

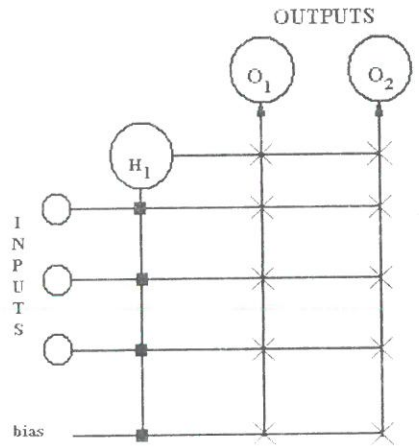

Figura 3: Arhitectura *Cascade-Correlation* după adăugarea unei unităţi ascunse

Figura 3 arată reţeaua după adăugarea unei unităţi ascunse, iar figura 4 ilustrează reţeaua după adăugarea a două unităţi ascunse.

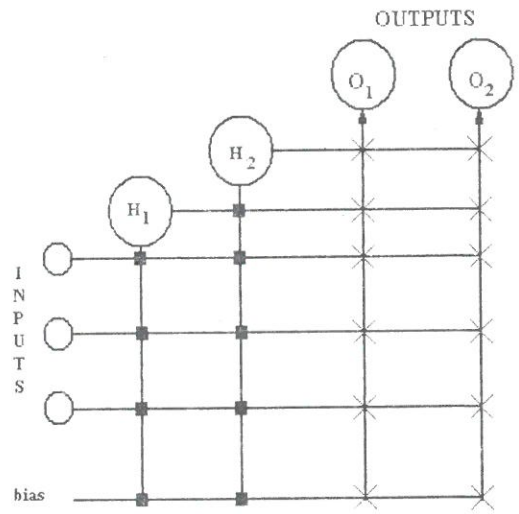

Figura 4: Arhitectura *Cascade-Correlation* după adăugarea a două unităţi ascunse

În continuare, explicăm modul în care noua unitate ascunsă este antrenată folosind *Quickpropagation*. Subliniem faptul că noua unitate este antrenată înainte de a fi adăugată la reţea. Scopul antrenării este de a maximiza S, suma peste toate unităţile de ieşire (o) a magnitudinii corelaţiei între o'', valoarea de ieşire a noii unităţi ascunse şi

E_o, eroarea reziduală observată la unitatea de ieşire o. S este definită astfel:

$$S = \sum_o \left| \sum_p \left(o_p^h - \overline{o^h} \right) \left(E_{p,o} - \overline{E_o} \right) \right|,$$

unde $E_{p,o}$ este eroarea la ieşirea o pentru exemplul de antrenament p, o_p^h este ieşirea noii unităţi ascunse pentru exemplul p, iar cantităţile $\overline{o^h}$ şi $\overline{E_o}$ sunt valorile lui o^h şi E_o împărţite la numărul exemplelor de antrenament. Pentru a maximiza S, se calculează derivata parţială a lui S în raport cu ponderile $_i^h$ ale conexiunilor care intră în noua unitate.

Obţinem:

$$\frac{\partial S}{\partial w_i^h} = \sum_{p,o} \sigma_o \left(E_{p,o} - \overline{E_o} \right) f_p' I_{i,p},$$

unde σ_o este semnul corelaţiei între valoarea de ieşire a noii unităţi, o_p^h şi eroarea ieşirii o, $E_{p,o}$; f_p' este derivata pentru exemplul p a funcţiei de activare a noii unităţi în raport cu suma intrărilor sale; $I_{i,p}$ este intrarea pe care noua unitate o primeşte de la unitatea i, pentru exemplul p, unde unitatea i este fie o unitate de intrare a reţelei, fie o unitate ascunsă adăugată anterior.

Singurele ponderi actualizate sunt ponderile $_i^h$ ale conexiunilor care intră în noua unitate, de unde rezultă că algoritmul *Quickpropagation* este folosit pentru antrenarea unui singur nivel. În acest fel nu este necesară o propagare înapoi a erorilor. Ponderile conexiunilor care intră în unităţile ascunse anterior adăugate rămân îngheţate. Când nu se mai imbunătăţeşte, ieşirea noii unităţi este conectată la fiecare unitate de ieşire a reţelei şi ponderile conexiunilor care intră în noua unitate rămân îngheţate pentru perioada de antrenare rămasă.

Acum începe din nou antrenarea unităţilor de ieşire ale reţelei, de această dată şi cu ponderile pornind de la noua unitate ascunsă adăugată.

Pentru a mări viteza de antrenare a noii unităţi, un "bazin" de unităţi candidat sunt antrenate în paralel, fiecare începând cu ponderi aleatoare diferite. Unitatea candidat, care obţine cel mai mare S, este aleasă ca noua unitate ce va fi adăugată la reţea.

Corelând valoarea noii unităţi ascunse cu eroarea obţinută la unităţile de ieşire, unităţile de ieşire pot reduce eroarea în următoarea epocă deoarece pentru exemplele de antrenanent care produc erori mari la ieşire, unitatea ascunsă va produce o valoare mare.

4. Modelul experimental

O serie de timp este o secvenţă de date ordonate în timp. Problemele de predicţie a seriilor de timp pot fi uşor asociate reţelelor neurale *feed-forward* [5, 6]. Numărul unităţilor de intrare corespunde numărului de date de intrare. Numărul unităţilor de ieşire reprezintă orizontul predicţiei. Predicţia unui pas înainte poate fi realizată cu o reţea cu o unitate de ieşire, iar predicţia a k paşi înainte poate fi asociată unei reţele cu k unităţi de ieşire. În figura 5 este prezentat un model de reţea neurală în care 8 termeni ai seriei de timp sunt folosiţi pentru a prezice următorii 2 termeni.

În acest articol, s-a folosit algoritmul de învăţare *Cascade-Correlation* pentru predicţia a cincii serii de timp reprezentând măsuratorile debitelor medii lunare lichide efectuate la Orşova (Dunăre), Ismail (Dunăre), Podari (Jiu), Faţa Motrului (Motru) şi Turburea (Gilort). Datele folosite pentru prima serie de timp sunt cele din perioada 1959-1992, pentru a

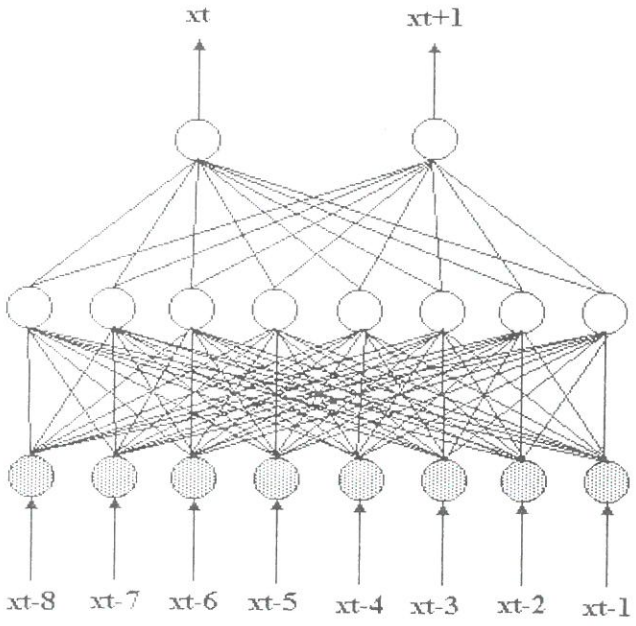

Figura 5: Reţea neurală pentru predicţia a 2 paşi înainte

doua serie de timp cele din perioada 1946- 1986, iar pentru ultimele trei serii de timp cele din perioada 1956- 1989.

Debitul mediu lunar lichid reprezintă volumul de apă scurs prin secțiunea unui râu în decursul unei luni calendaristice raportat la unitatea de timp și se exprimă în m^3 /sec.

O predicție este considerată corectă dacă valoarea abaterii este de cel mult $\pm 10\%$ din valoarea măsurată.

Ponderile au fost inițializate cu valori aleatoare, uniform distribuite în intervalul [-1,1]. Datele au fost normalizate în intervalul [0,1] înainte de a intra în rețea :

$$data_{[0,1]} = \frac{data\_maxima - data}{data\_maxima - data\_minima}$$

Predicțiile de la ieșirea rețelei au fost transformate la scala originală înainte de a se calcula procentul predicțiilor corecte.

Algoritmul *Quickpropa-gation*, folosit pentru antrenarea unităților de ieșire și a unităților candidat, este îmbunătățit cu tehnica de eliminare a porțiunilor netede (*"flat-spot"*). Problema porțiunilor netede apare când ieșirea a_j a funcției de activare sigmoidă a unui neuron j se apropie foarte mult de 0 sau 1. În acest caz, derivata $a_j(1-a_j)$ devine prea apropiată de zero, iar semnalul de eroare δ_j are o valoare prea mică. Din această cauză, ponderile se modifică foarte puțin. Tehnica constă în adăugarea întotdeauna a constantei 0,1 la derivata lui a_j, obținând $a_j(1-a_j)+0,1$.

În continuare, prezentăm parametrii folosiți în algoritmul *Cascade-Correlation*:

- **candDecay, outDecay**:

  cantitatea care se scade din panta fiecărei ponderi care intră într-o unitate candidat (de ieșire) la fiecare epocă. Astfel, se previne obținerea unor ponderi cu valori prea mari. În general, acești parametri se setează la 0. Dacă ponderile unităților candidat (unităților de ieșire) devin foarte mari (>100), se setează candDecay (outDecay) la 0,001. În unele probleme care nu au suficiente exemple de antrenament, se poate obține o bună generalizare și cu valori mai mari, ca de exemplu 0,01.

- **candMu, outMu**:
  parametrul μ din algoritmul Quickpropagation, corespunzător antrenării unităților candidat (de ieșire). In general, acești parametri se setează la 2.0. Dacă problema determină oscilații se schimbă la 1,75 sau 1,5. S. Fahlmann sugerează o valoare $\mu = 1,75$.

- **candEpsilon, outEpsilon**
  rata de învățare ε din Quickpropagation, pentru unitățile candidat (de ieșire). Rata de învățare poate varia foarte mult (de la 1000 la 0,01), în funcție de problemă. Dacă observăm o convergență înceată, în special la sfârșitul epocii, rata de învățare se mărește. Dacă suntem într-o regiune haotică (eroarea nu se îmbunătățește mai multe epoci), rata de învățare se micșorează. Pentru antrenarea unităților de ieșire, rata de învățare se împarte la numărul exemplelor de antrenament:

  $$\frac{\text{outEpsilon}}{\text{nr\_exemple\_antrenament}}$$

  Pentru unitățile candidat, *candEpsilon* se scalează astfel:

  $$\frac{\text{candEpsilon}}{\text{nr\_exemple\_antrenament\_*(nr\_intrari + nr\_unitati\_ascunse)}}$$

  În experimente am folosit valori ale lui *candEpsilon* între 10 și 1000 și valori ale lui *outEpsilon* între 0,01 și 10.

- **maxNewUnits:**

  numărul maxim de unități noi ce pot fi adăugate la rețea. Dacă numărul unităților din rețea devine prea mare, rețeaua învață bine exemplele din setul de antrenament, dar produce rezultate slabe la generalizare. Am folosit pentru maxNewUnits o valoare de 15.

- **nCands**:

  numărul unităților candidat care se vor antrena. Cea mai bună din ele va fi adăugată la rețea. Am folosit între 5 și 10 unități candidat.

- **candEpochs**:
  numărul maxim de epoci de antrenare a unităților candidat, înaintea selectării celei mai bune unități și adăugării ei la rețea.

- **outEpochs**:

  numărul maxim de epoci de antrenare a unităților de ieșire ale rețelei, înaintea antrenării unui nou set de unități candidat.

- **candPatience, outPatience**:

  indică după câte epoci fără îmbunătățiri semnificative procesul de antrenare se

opreşte. Am setat aceşti parametri la valori între 12 şi 15.

Pentru fiecare alegere a acestor parametri, am rulat 5 experimente cu ponderi iniţiale diferite. Rezultatele prezentate sunt mediile celor 5 rulări. Ultimii 18 termeni ai seriilor date au fost folosiţi ca exemple de testare a performanţelor reţelei. De asemenea, am folosit 18 termeni ca exemple de validare. Validarea poate ajuta la îmbunătăţirea performanţelor pe setul de date. Programul reţine starea reţelei corespunzătoare celei mai bune performanţe pe exemplele de validare. Dacă această performanţă nu se îmbunătăţeşte după un anumit număr de cicluri de antrenament, reţeaua este restaurată la acea poziţie şi procesul se opreşte.

În continuare, prezentăm cele mai bune rezultate obţinute:

- **Orşova**: procentaj corect 38,89%;
- **Ismail**: procentaj corect 27,78%;
- **Podari**: procentaj corect 11%;
- **Faţa Motrului**: procentaj corect 9,3%;
- **Turburea**: procentaj corect 7,4%.

Valorile parametrilor corespunzătoare acestor rezultate sunt:

candDecay = outDecay = 0.0, candMu = outMu = 2.0, candEpsilon = 100.0, outEpsilon = 0.5, nCands = 8, candEpochs = outEpochs = 200, candPatience = outPatience = 12.

Reţelele neurale cu care au fost obţinute cele mai bune rezultate au avut între 1 şi 4 noduri ascunse. Reţelele cu mai multe noduri ascunse au învăţat mai bine exemplele din setul de antrenament, dar au obţinut performanţe mai slabe pe setul de testare. Variaţii mici ale valorilor parametrilor de mai sus nu duc la schimbări semnificative în performanţele reţelelor neurale.

5. Concluzii

Marea variabilitate a factorilor fizico- geografici (meteorologici, geomorfologici, geologici, pedologici şi biotici), care contribuie la realizarea scurgerii, face ca procentajul predicţiilor corecte să varieze în limite destul de largi în strânsă corelaţie cu extensiunea bazinelor hidrografice.

Astfel, pentru fluviul Dunărea care are un bazin hidrografic ce depăşeşte 800000 kmp are loc o omogenizare a condiţiilor fizico - geografice şi, în consecinţă, variaţia scurgerii de la un an la altul este mult mai atenuată comparativ cu cea înregistrată în bazinele Jiu, Motru şi Gilort, cu suprafeţe bazinale mult mai reduse (sub 10000 kmp).

Varietatea condiţiilor naturale este reliefată şi de raportul dintre debitul lunar mediu minim

şi cel maxim. Astfel, pentru secţiunile studiate, rapoartele sunt mai mari pe Dunăre (între 1:7 şi 1:8) şi substanţial mai mici pentru Jiu (1:39), Gilort (1:63) şi Motru (1:82).

Diferenţele dintre procentajul corect pentru secţiunea Ismail (27,78%) şi cel obţinut pentru secţiunea Orşova (38,89%) se datorează caracterului de torenţialitate pe care îl înregistrează râurile colectate de Dunăre în aval de Orşova.

Scăderea suprafeţei bazinale, în condiţiile unei mari variaţii a factorilor naturali, face ca debitele medii lunare să oscileze, de la o lună la alta şi de la un an la altul, mult mai pronunţat. În consecinţă şi rata de corectitudine a predicţiilor este mult mai scăzută: 11% pentru Jiu la Podari, 9,3% pentru Motru la Faţa Motrului şi 7,4% pentru Gilort la Turburea.

Cunoaşterea modului de evoluţie a debitelor în diferite secţiuni prezintă o importanţă deosebită, întrucât facilitează rezolvarea unor probleme impuse de necesităţile practice.

Rezultatele obţinute în acest articol corelate cu cele prezentate într-o lucrare anterioară [7] conduc la ideea că performanţele modelelor de predicţie cu reţele neurale depind de mulţi factori, cum ar fi structura reţelei, algoritmul de învăţare, alegerea parametrilor reţelei. Se impune continuarea studiului prin folosirea altor arhitecturi de reţele neurale şi a altor algoritmi de învăţare.

Bibliografie

1. **RUMELHART, D., McCLELLAND, J., PDP Research Group 1986**: Parallel Distributed Processing: Explorations. În: The Microstructure of Cognition, MIT Press, Cambridge, Vol. 1, 1986.
2. **FAHLMANN, S., LEBIERE, C.**: The Cascade-Correlation Learning Architecture. În: Technical Report, Carnegie Mellon University, Pittsburgh, 1990.
3. **SIMON, N.**: Constructive Supervised Learning Algorithms for Artificial Neural Networks, Master Thesis, Delft University of Technology, 1991.
4. **FIESLER, E.**: Comparative Bibliography of Ontogenic Neural Networks. În: Technical Report, IDIAP, Switzerland, 1994.
5. **McCLUSKEY, P.**: Feedforward and Recurrent Neural Networks and Genetic Programs for Stock Market and Time Series

Forecasting. În: Technical Report, Brown University, 1992.

6. **TANG, Z., FISHWICK, P.**: Feed-forward Neural Nets as Models for Time Series Forecasting. În: Technical Report , University of Florida, Gainesville, 1993.

7. **LEUȘTEAN, L.**: Applications of Neural Networks To Some Echological Phenomena Forecasting. În: Proc. of the 3rd International Symposium of Economic Informatics, Bucharest, 1997.