Unitatea de învăţare 9.

INSTRUIREA REŢELELOR NEURONALE UNIDIRECŢIONALE CU METODA PROPAGĂRII ÎNAPOI A ERORII

Cuprins

Introducere

Algoritmul de propagare înapoi a erorii (Back-Propagation) este cosiderat în mod uzual ca fiind cel mai important şi mai utilizat algoritm de instruire a reţelelor neuronale. După unele estimări, acest algoritm este utilizat în prezent în aproximativ 90% din aplicaţiile de reţele neuronale.

În esenţă algoritmul de propagare înapoi este o *metodă de instruire supervizată* a reţelelor neuronale multistrat cu transmitere înainte (reţele unidirecţionale) în care se urmăreşte minimizarea erorii medii pătratice printr-o metodă de gradient.

Competenţele unităţii de învăţare

- *Cunoaşterea structurii reţelelor neuronale unidirecţionale*
- *Cunoştarea algoritmului de antrenare cu propagarea înapoi a erorii pentru reţelele unidirecţionale*

Durata medie de parcurgere a unităţii de învăţare este de 2 ore.

9.1. Arhitectura multistrat cu transmitere înainte

Reţelele neuronale studiate până acum au fost, cu puţine excepţii, reţele simple cu un singur neuron. Pentru rezolvarea unor probleme mai complicate este nevoie de o reţea cu mai multe straturi, din care cel puţin unul trebuie să conţină unităţi ascunse, care nu sunt conectate direct la ieşirile reţelei.

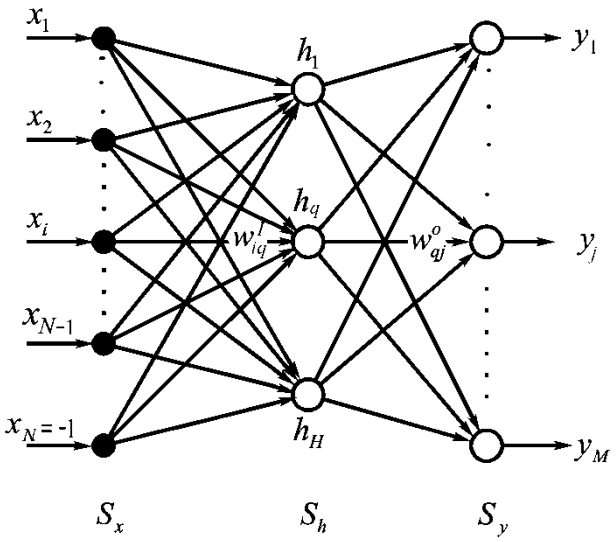

Figura 9.1. Arhitectura multistrat a reţelelor neuronale cu transmitere înainte.

În figura 9.1 se prezintă arhitectura unei reţele neuronale unidirecţionale cu două straturi de neuroni, care se compune dintr-un câmp S_x de ramificare (redirecţionare) a mărimilor de intrare, un strat cu H neuroni conectaţi la semnalele de intrare (denumit *strat ascuns*) şi un alt strat cu M neuroni având conectate la intrare ieşirile neuronilor din stratul ascuns.

Se presupune că intrarea x_N este asociată pragului pentru neuronii din stratul ascuns fiind egală cu -1. Valorile de prag ale acestor neuroni vor fi stabilite de ponderile conexiunilor w^I_{Nq} cu $q = \overline{1,H}$. Ponderile conexiunilor sinaptice ale mărimilor de intrare cu stratul ascuns se notează cu w^I_{iq}, unde $i = \overline{1,N}$ şi $q = \overline{1,H}$, iar a celor dintre stratul ascuns şi cel de ieşire se va nota cu w^o_{qj}, unde $j = \overline{1,M}$.

Vom specifica mărimile de intrare, prin vectorul

$$\mathbf{x} = [x_1, \quad ..., \quad x_N]^{\mathrm{T}}, \tag{9.1}$$

mărimile de ieşire ale neuronilor din câmpul ascuns S_h cu notaţia vectorială

$$\mathbf{h} = [h_1, \quad ..., \quad h_H]^{\mathrm{T}}, \tag{9.2}$$

iar cele ale neuronilor din stratul de ieşire cu

$$\mathbf{y} = [y_1, \quad ..., \quad y_M]^{\mathrm{T}}. \tag{9.3}$$

Funcţiile de activare ale neuronilor din S_h şi S_y se adoptă de tip sigmoidal sau de tip identitate

$$\mathrm{f}(s)=\frac{1}{1+e^{-s}} \text{ respectiv } \mathrm{f}(s)=s\ . \tag{9.4}$$

Vom nota funcţiile de activare ale neuronilor din stratul de ieşire cu f şi activarea totală cu s, iar pentru mărimile similare din câmpul ascuns folosim notaţiile φ şi σ.

Reţelele neuronale cu transmitere înainte cu două straturi au o arie foarte largă de aplicaţii fapt pentru care în literatura de specialitate ele sunt considerate funcţii universale de aproximare ale proceselor studiate.

9.2. Metoda propagării înapoi a erorii

Aplicarea algoritmului propagarii înapoi a erorii presupune două etape. În prima etapă se prezintă reţelei un vector de instruire $\mathbf{x}^k$ şi ieşirea dorită $\mathbf{d}^k$. Semnalul de intrare se propagă înainte prin reţea producând pentru neuronul q din stratul ascuns mărimea:

$$h_q^k=\varphi_q\left(\sum_{i=1}^{N} w_{iq}^I x_i^k\right),\ q=1,2,...,H\ . \tag{9.5}$$

Pentru neuronul j din câmpul de ieşire se obţine ieşirea reală

$$y_j^k=\mathrm{f}_j\left(\sum_{q=1}^{H} w_{q\,j}^o\ h_q^k\right)=\mathrm{f}_j(s_j^k)\,. \tag{9.6}$$

Acest răspuns este comparat cu răspunsul dorit d_j^k. Rezultă astfel eroarea dinamică δ_j^k, care se calculează pentru fiecare neuron de ieşire:

$$\delta_j^k=(d_j^k-y_j^k)\,\mathrm{f}'_j(s_j^k),\ j=1,2,...,M\ . \tag{9.7}$$

(Notaţia ‘ se referă la operaţia de derivare.)

Se calculează apoi schimbările ponderilor pentru toate conexiunile ce intră în stratul final, folosind relaţia de calcul recurent

$$w_{q\,j}^{o\,(k+1)}=w_{q\,j}^{o\,k}+c\delta_j^k\varphi_q(\sigma_q^k)\,. \tag{9.8}$$

Faza a doua corespunde propagării înapoi prin reţea a semnalului de eroare. Se calculează recursiv eroarea dinamică pentru neuronii din stratul ascuns folosind relaţia

$$\delta_{q(h)}^k=\varphi'_q(\sigma_q^k)\sum_{j=1}^{M}\delta_j^k\,w_{q\,j}^{o\,k}\ . \tag{9.9}$$

Calculul de mai sus se face pentru fiecare neuron q din stratul ascuns. Se poate constata că propagarea înapoi implică un calcul de aceeaşi complexitate ca şi propagarea înainte a semnalelor prin reţea.

Descrierea algoritmului pentru o reţea cu două straturi

P_0 Iniţializare

Se iniţializează ponderile reţelei cu valori mici generate aleator. Se adoptă valoarea constantă a ratei de învăţare $c \in (0,1)$.

P_1 Un ciclu de învăţare

Se prezintă reţelei o mulţime de forme de instruire. Pentru fiecare exemplu se execută secvenţa **P_2**…**P_{12}**. Pentru evitarea minimelor locale se recomandă să se schimbe ordinea de prezentare a formelor de instruire, de la un ciclu la altul.

P_2 Se prezintă reţelei vectorul $\boldsymbol{x}^k$ şi vectorul de ieşire dorit $\boldsymbol{d}^k$.

P_3 Se calculează activarea neuronilor din stratul ascuns

$$\sigma_q^k = \sum_{i=1}^{N} w_{i\,q}^{I\,k} x_i^k, \quad q = \overline{1, H}.$$

P_4 Se calculează ieşirile neuronilor din stratul ascuns

$$h_q^k = \varphi_q(\sigma_q^k), \quad q = \overline{1, H}.$$

P_5 Se determină activările neuronilor din câmpul de ieşire

$$s_j^k = \sum_{q=1}^{H} w_{q\,j}^{o\,k} h_q^k, \quad j = \overline{1, M}.$$

P_6 Se calculează ieşirile reale ale reţelei

$$y_j^k = f_j(s_j^k), \quad j = \overline{1, M}.$$

P_7 Se determină eroarea dinamică pentru neuronii din câmpul de ieşire

$$\delta_j^k = (d_j^k - y_j^k)\,\mathrm{f}'_j(s_j^k), \quad j = \overline{1, M}.$$

P_8 Se calculează eroarea dinamică pentru neuronii din stratului ascuns

$$\delta_{q(h)}^k = \varphi'_q(\sigma_q^k) \sum_{j=1}^{M} \delta_j^k w_{q\,j}^{o\,k}, \quad q = \overline{1, H}.$$

Remarcăm calculul acestui semnal înainte de actualizarea ponderilor neuronilor din stratul de ieşire.

P_9 Se actualizează ponderile neuronilor din stratul de ieşire

$$w_{qj}^{o(k+1)} = w_{qj}^{ok} + c\delta_j^k \varphi_q\left(\sigma_q^k\right)$$

P_{10} Se corecteză ponderile neuronilor din stratul ascuns

$$w_{iq}^{I(k+1)} = w_{iq}^{Ik} + c\,\delta_{q(h)}^{k}\,x_i^k \ .$$

Observaţie: Nu prezintă importanţă ordinea în care se actualizează ponderile la paşii **P_9** şi **P_{10}**.

P_{11} Se calculează eroarea

$$E_k = \frac{1}{2}\sum_{j=1}^{M}(d_j^k - y_j^k)^2 \ .$$

P_{12} Dacă $E_k < \varepsilon$ dat, pentru toate perechile $\left(\mathbf{x}^k, \mathbf{d}^k\right)$ din mulţimea de instruire atunci instruirea este încheiată. În caz contrar se se reia secvenţa **P_1-P_{12}** prezentându-se reţelei un alt ciclu de forme de instruire

Observaţie: Ponderile iniţiale pot fi generate folosind un algoritm adecvat care să se bazeze pe o filosofie diferită de cea a propagării înapoi, de exemplu algoritmul de călire simulată sau algoritmi genetici.

9.3. Consideraţii privind performanţele reţelelor neuronale instruite cu metoda propagării înapoi

Performanţele reţelelor neuronale instruite cu metoda propagării înapoi a erorii sunt determinate de următoarele aspecte:

- topologia (arhitectura) reţelei;
- rata de instruire;
- evitarea minimelor locale;
- blocarea algoritmului.

Arhitectura reţelei

Cele mai multe aplicaţii ale algoritmului de propagare înapoi utilizează o arhitectură de reţea neuronală cu un singur strat ascuns. Motivul principal al acestei abordări este acela că neuronii intermediari care nu sunt conectaţi direct cu neuronii de ieşire vor avea variaţii foarte mici ale ponderilor şi, deci, ei vor învăţa foarte lent. Desigur, un alt motiv este simplitatea acesteia în raport cu complexitatea problemelor pe care le poate rezolva. Dacă se alege această structură, mai rămâne o singură decizie de luat în ceea ce priveşte topologia şi anume câţi neuroni trebuie să conţină stratul ascuns.

Rata de instruire. Metoda momentului

Alegerea unor valori adecvate pentru rata de instruire c are un efect semnificativ asupra performanţelor reţelei. În mod obişnuit se consideră $c \in (0,1)$, dar în unele aplicaţii este posibil ca acest interval să fie prea larg. De regulă alegând pentru c o valoare mai mică, de exemplu în intervalul $(0.05, 0.25)$, se asigură în mod cert obţinerea unei soluţii, deşi reţeaua va necesita un număr mai mare de iteraţii pentru instruire. Este, de asemenea, posibil ca valoarea parametrului de învăţare să crească în timpul instruirii, pe măsură ce eroarea reţelei descreşte. Aceasta conduce, în general, la mărirea vitezei de convergenţă. În acest caz există pericolul ca reţeaua să depăşească minimul global, oprindu-se într-un punct de minim local. Creşterea lui c în timpul instruirii este justificată dacă suprafaţa funcţiei criteriu este relativ netedă în zona bazinului de atracţie spre punctul de minim. În acest caz, pe măsura apropierii de punctul corespunzător soluţiei derivata erorii devine tot mai mică, ceea ce produce schimbări neînsemnate ale ponderilor. Din cele spuse se poate concluziona că, în general, se justifică încercarea de a mări valoarea ratei de instruire pe măsură ce eroare reţelei se reduce.

Există însă o serie de metode sistematice de mărire a vitezei de convergenţă a algoritmului propagarii înapoi a erorii. Una dintre acestea este cunoscută sub denumirea de *metoda momentului*. Tehnica metodei constă în adăugarea unui termen, numit moment, la corecţia ponderii. Acest termen este proporţional cu variaţia precedentă a ponderii, ceea ce face ca această modificare să se facă pe aceeaşi direcţie cu cea a corecţiei precedente. Se poate afirma că în acest fel reţeaua a devenit "conservatoare", în sensul că direcţiile noi nu sunt favorizate.

Regula de corecţie pentru neuronii din stratul de ieşire se scrie acum

$$w_{qj}^{o(k+1)} = w_{qj}^{ok} + c\delta_j^k h_q^h + b\Delta w_{qj}^{ok}, \tag{9.10}$$

unde

$$\Delta w_{qj}^{ok} = w_{qj}^{ok} - w_{qj}^{o(k-1)}. \tag{9.11}$$

Coeficientul b se alege, de obicei, în jurul valorii 0.9. Este interesant de relevat efectul introducerii termenului suplimentar corespunzător momentului, aşa cum rezultă din literatură. Acest efect a fost studiat pentru diverse aluri ale suprafeţei corespunzătoare funcţiei criteriu (de eroare). De exemplu dacă suprafaţa de eroare prezintă văi adânci şi înguste, care conţin mai multe puncte de minim, când se foloseşte metoda momentului există tendinţa ca reţeaua să urmeze văile adânci în loc să treacă dintr-o parte în alta a acestora. Cercetările experimentale pun în evidenţă faptul că se poate obţine acelaşi rezultat luându-se $b = 0$ şi reducând valoarea lui c, ceea ce implică o viteză de convergenţă mult mai lentă. De asemenea s-a constatat că metoda momentului nu reprezintă un panaceu, aceasta funcţionând foarte bine pentru unele probleme dar are un efect redus, sau chiar negativ, în cazul altor aplicaţii.

O generalizare a metodei momentului, care are pentru anumite aplicaţii rezultate foarte bune este *metoda de netezire exponenţială.* Regula de corecţie folosită de metoda de netezire este

$$w_{qj}^{o(k+1)} = w_{qj}^{ok} + c\,(1-b)\,\delta_j^k h_q^k + b\,\Delta w_{qj}^{ok}, \tag{9.12}$$

unde

$$\Delta w_{qj}^{ok} = w_{qj}^{ok} - w_{qj}^{o(k-1)}. \tag{9.13}$$

În această formulă, $b \in [0,1]$, şi are semnificaţia unui coeficient de netezire. Se observă că dacă $b = 0$, regula se reduce la forma standard a algoritmului propagării înapoi a erorii. Dacă $b = 1$ obţinem

$$w_{q\,j}^{o\,(k+1)} = w_{q\,j}^{o\,k} + \Delta w_{q\,j}^{o\,k}, \tag{9.14}$$

deci ajustarea la acest pas este identică cu cea de la pasul precedent, deci

$$\Delta w_{q\,j}^{o\,(k+1)} = \Delta w_{q\,j}^{o\,k} \tag{9.15}$$

Dacă $b \in (0,1)$, atunci corecţia ponderii este netezită cu o cantitate proporţională cu b.

Regulile de corecţie pentru neuronii din stratul intermediar vor avea expresii formal asemenea cu cele folosite pentru neuronii câmpului de ieşire.

O formă mai generală a algoritmului de propagare înapoi a erorii se obţine dacă se admite o rată de instruire variabilă care se modifică la fiecare pas. De obicei se adoptă aceeaşi valoare a ratei de instruire atât pentru neuronii din câmpul de ieşire cât şi pentru cei din stratul intermediar. Regulile de corecţie corespunzătoare acestei situaţii se vor scrie:

- pentru neuronii stratului de ieşire

$$w_{q\,j}^{o\,(k+1)} = w_{q\,j}^{o\,k} + c_k\ \delta_j^k\ h_q^k. \tag{9.16}$$

- pentru unităţile câmpului ascuns

$$w_{i\,q}^{I\,(k+1)} = w_{i\,q}^{I\,k} + c_k\ \delta_{q(h)}^k\ x_i^k, \tag{9.17}$$

unde

$$\delta_{q(h)}^k = \varphi'_q(\sigma_q^k) \sum_{j=1}^{M} \delta_j^k w_{q\,j}^{o\,k}. \tag{9.18}$$

Pentru a evita oscilaţiile care pot să apară în jurul soluţiei optime, este necesar ca ratele de instruire c_k să descrească în timp. S-a constatat din încercări, că cu cât descreşterea este mai lentă, cu atât mai rapid decurge instruirea reţelei. Pe de altă parte, din studiile efectuate rezultă că dacă viteza de descreştere a ratei de învăţare este mare, atunci vectorul pondere "uită" formele deja învăţate. O soluţie optimă care satisface în ambele situaţii menţionate mai înainte se obţine adoptând rata de instruire conform relaţiei

$$c_k = \frac{1}{k}, \tag{9.19}$$

unde k este indexul pasului.

Cerinţele contradictorii menţionate mai înainte pentru rata de instruire constituie aşa numita dilemă *stabilitate-plasticitate* care apare în procesul de învăţare a unei reţele neuronale. Conform acestei dileme o reţea neuronală trebuie să satisfacă două cerinţe contradictorii:

- i) pe de o parte, reţeaua trebuie să fie suficient de stabilă pentru a-şi aminti formele învăţate anterior şi

- ii) pe de altă parte, reţeaua trebuie să fie suficient de plastică, pentru a putea învăţa noi forme.

Evitarea minimelor locale

În unele situaţii algoritmul de propagare înapoi conduce la stări de minim local. Dacă reţeaua ajunge într-un minim local, este posibil ca eroarea asociată funcţiei criteriu să fie nepermis de mare. Pentru a scoate reţeaua în decursul instruirii dintr-un punct de minim local se pot folosi următoarele tehnici simple: schimbarea constantei de instruire, reluarea procesului cu ponderi iniţiale schimbate, adăugarea unor valori aleatoare mici la vectorul pondere (scuturarea reţelei), schimbarea numărului de neuroni din câmpul ascuns.

Se pot utiliza de asemenea unele metode mai elaborate care se aplică reţelei în faza iniţială a instruirii, metode care asigură aducerea acesteia în apropierea unui minim global (remarcăm în acest context că, în general, o reţea neuronală poate să admită mai multe puncte de minim global perfect echivalente). Dintre aceste metode amintim două: călirea simulată şi algoritmii genetici. Prima metodă poate fi folosită atât pentru evitarea unui minim local, cât şi pentru ieşirea dintr-o astfel de situaţie.

Blocarea algoritmului

Procesul de instruire cu metoda metoda propagării înapoi a erorii se poate bloca în cazul în care se adoptă iniţial valori egale pentru toate ponderile reţelei, sau dacă apare o astfel de situaţie în cursul învăţării. Blocarea se produce deoarece eroarea se propagă înapoi prin semnale proporţionale cu valorile ponderilor, deci toţi neuronii ascunşi conectaţi direct la cei de ieşire vor primi semnale de eroare identice. Având în vedere că schimbarea ponderilor depinde de semnalele de eroare, rezultă că ponderile acestor conexiuni între neuronii din câmpul ascuns şi cel de ieşire vor fi întotdeauna aceleaşi cu consecinţa că reţeaua nu poate ieşi din această stare. Pentru evitarea acestei situaţii trebuie să se aplice o procedură numită de *rupere a simetriei ponderilor*. Inconvenientul se elimină foarte uşor dacă, la momentul iniţial al instruirii, se adoptă valori aleatoare reale şi mici pentru ponderi. De regulă valoarea maximă, în modul, a acestor ponderi depinde de numărul de intrări ale neuronilor din câmpul ascuns. Se recomandă pentru neuronii din acest câmp ca valoarea ponderilor conexiunilor cu mărimile de intrare să fie cuprinse în intervalul $(-2/n, 2/n)$, unde n este numărul intrărilor pentru fiecare neuron ascuns.

Să ne reamintim ...

Algoritmul de propagare înapoi a erorii (Back-Propagation) este cosiderat în mod uzual ca fiind cel mai important şi mai utilizat algoritm de instruire a reţelelor neuronale. După unele estimări, acest algoritm este utilizat în prezent în aproximativ 90% din aplicaţiile de reţele neuronale.

În esenţă algoritmul de propagare înapoi este o *metodă de instruire supervizată* a reţelelor neuronale multistrat cu transmitere înainte (reţele unidirecţionale) în care se urmăreşte minimizarea erorii medii pătratice printr-o metodă de gradient.

Aplicarea algoritmului propagării înapoi a erorii presupune două etape. În prima etapă se prezintă reţelei un vector de instruire şi ieşirea dorită. Semnalul de intrare se propagă înainte prin reţea producând la ieşirea reţelei o anumită ieşire. Diferenţa dintre ieşirea dorită şi cea obţinută reprezintă o eroare de instruire (dinamică).

Faza a doua corespunde propagării înapoi prin reţea a semnalului de eroare. Pentru fiecare neuron al fiecărui strat, se calculează o mărime de eroare, în baza căreia se ajustează ponderile sinaptice ale neuronului respectiv.

Test de evaluare a cunoştinţelor

1. Descrieţi arhitectura unidirecţională, detaliind notaţiile utilizate.

2. Descrieţi principiul care stă la baza metodei propagării înapoi a erorii.

3. Enumeraţi paşii algoritmului propagării înapoi e erorii.

Teme propuse

1. Realizaţi un program C/C++ pentru implementarea algoritmlui de instruire.

Test de autoevaluare a cunoştinţelor

1. Algoritmul propagării înapoi a erorii este o metodă de instruire:

    a) supervizată

    b) nesupervizată

    c) poate fi supervizată sau nesupervizată în funcţie de datele de antrenare disponibile

2. Algoritmul propagării înapoi a erorii, în forma standard, se aplică:

    a) reţelelor unidirecţionale

    b) reţelelor bidirecţionale

c) exclusiv reţelelor formate din mai multe straturi

3. Un strat de neuroni ale căror ieşiri sunt intrări ale altor neuroni este:

a) un strat ascuns

b) un strat de intrare

c) un strat de ieşire

4. Eroarea de instruire pentru o pereche de date de antrenare este:

a) suma diferenţelor între valorile corecte ale ponderilor de antrenare şi cele obţinute pentru perchea de date

b) suma diferenţelor dintre valorile dorite ale ieşirilor reţelei şi valorile obţinute la ieşirile reţelei, pentru toate datele de antrenare anterioare

c) suma diferenţelor dintre valorile dorite ale ieşirilor reţelei, specificate în perechea de date, şi valorile obţinute la ieşirile reţelei

5. Rata de instruire poate fi:

a) orice valoare reală

b) orice valoare reală, pozitivă şi subunitară

c) orice valoare întreagă pozitivă

6. Rata de instruire:

a) selectează datele de antrenare utile, relevante

b) se referă la viteza cu care se efectuează fiecare pas din algoritmul de instruire

c) influenţează convergenţa algoritmului de antrenare şi performanţele

7. Performanţele algoritmului de instruire sunt influenţate de:

a) topologia reţelei

b) aplicarea unor metode de evitare a minimelor locale

c) situaţiile de blocare a algoritmului

8. Metoda momentului:

a) este o metodă de creştere a vitezei de convergenţă a algoritmului

b) este o metodă de evitare a situaţiilor de blocare a algoritmului

c) este o metodă de evitare a minimelor locale

9. Pentru evitarea minimelor locale se pot folosi următoarele tehnici:

a) metoda momentului

b) schimbarea ratei de instruire

c) reluarea procesului de instruire pornind cu alte valori iniţiale ale ponderilor

10. Majoritatea aplicaţiilor în care se utilizează reţele neuronale antrenate cu metoda propagării înapoi e erorii:

a) nu au nici un strat ascuns

b) au un singur strat ascuns

c) au două straturi ascunse