ANTRENAREA SUPERVIZATĂ A REȚELELOR FEEDFORWARD MULTISTRAT ÎN PROBLEME DE APROXIMARE NELINIARĂ

1. Considerații generale, motivație şi obiect

Rețelele neuronale feed-forward multistrat cu noduri de intrare sigmoidale pot fi utilizate pentru a realiza aproximarea unor funcții neliniare, definite cu ajutorul unei mulțimi de vectori prototip şi respectiv a unei mulțimi de vectori țintă. Numărul de intrări şi ieşiri ai rețelei sunt impuse de dimensiunea vectorilor țintă şi respectiv - dimensiunea vectorilor prototip. Numărul de neuroni din stratul de intrare (şi din stratul ascuns - în cazul rețelelor cu trei straturi) este stabilit de utilizator. Aceste arhitecturi neuronale sunt denumite şi rețele MLP (eng. *MultiLayer Perceptron*).

Prin parcurgerea acestei şedințe de aplicații, studentul îşi va însuşi cunoştințele necesare antrenării supervizate a acestor rețele cu ajutorul algoritmului de propagare inversă. Pentru desfăşurarea practică a experimentelor se va face apel la funcții disponibile în ***Neural Network Toolbox***. Un accent deosebit se va pune pe vizualizarea grafică a progresului antrenării:

- vizualizare grafică tridimensională a suprafeței de eroare şi a căutării minimului în cazul unui neuron cu o singură intrare şi funcție de activare sigmoidală;
- vizualizare grafică bidimensională a evoluției procesului de aproximare în cazul unei rețele cu două straturi

2. Cunoştințe prealabile necesare

- Capitolul "Antrenarea supervizată a rețelelor neuronale", secțiunea "Rețele feed-forward multistrat" din Cursul "Aplicații ale rețelelor neuronale în automatică".
- Experiență de programare în MATLAB.

3. Breviar teoretic

Utilizarea rețelelor neuronale feed-forward multistrat în problemele de aproximare neliniare se bazează pe antrenarea rețelelor prin algoritmul de propagare inversă. Denumirea algoritmului provine din faptul că pentru antrenare informațiile sunt procesate in sens invers, de la ieşirea rețelei, către intrare.

3.1. Proprietatea de aproximare a rețelelor feed-forward cu două straturi, având funcții de activare sigmoidale în primul strat şi liniare în al doilea

O rețea cu două straturi cu noduri sigmoidale în stratul de intrare şi liniare în stratul de ieşire realizează transferul intrare-ieşire prin funcția:

$$\boldsymbol{f}:\mathbb{R}^n\to\mathbb{R}^p, \tag{1-a}$$

$$\boldsymbol{y}=\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{\sigma}^2\left(\boldsymbol{W}^2\boldsymbol{\sigma}^1(\boldsymbol{W}^1\boldsymbol{x}+\boldsymbol{b}^1)+\boldsymbol{b}^2\right), \tag{1-b}$$

unde notațiile au următoarea semnificație, în conformitate cu arhitectura reprezentată grafic în figura 1:

$\boldsymbol{x}\in\mathbb{R}^n$ – vectorul de intrare;

$\boldsymbol{y}\in\mathbb{R}^p$ – vectorul de ieşire;

$\boldsymbol{W}^1\in\mathbb{R}^{m\times n}$ – matricea ponderilor stratului de intrare;

$\boldsymbol{b}^1\in\mathbb{R}^m$ – vectorul deplasărilor stratului de intrare;

$\boldsymbol{W}^2\in\mathbb{R}^{p\times m}$ – matricea ponderilor stratului de ieşire;

$\boldsymbol{b}^2\in\mathbb{R}^p$ – vectorul deplasărilor stratului de ieşire;

$\boldsymbol{\sigma}^1:\mathbb{R}^m\to\mathbb{R}^m$, $\boldsymbol{\sigma}^1(\boldsymbol{u}^1)=\left[\sigma^1(u_1^1)\ \ \ldots\ \ \sigma^1(u_m^1)\right]^T$ – funcția vectorială de activare a stratului de intrare, unde $\boldsymbol{u}^1=\boldsymbol{W}^1\boldsymbol{x}+\boldsymbol{b}^1\overset{not}{=}\left[u_1^1\ldots u_m^1\right]^T$ şi σ^1 are expresia analitică:

$\sigma^1(u)=1/(1+\mathrm{e}^u)$ în cazul sigmoidului unipolar şi, respectiv,

$\sigma^1(u)=(\mathrm{e}^u-\mathrm{e}^{-u})/(\mathrm{e}^u+\mathrm{e}^{-u})$ în cazul sigmoidului bipolar;

$\boldsymbol{\sigma}^2:\mathbb{R}^p\to\mathbb{R}^p$, $\boldsymbol{\sigma}^2(\boldsymbol{u}^2)=\boldsymbol{u}^2$ (echivalent cu $\boldsymbol{\sigma}^2(\boldsymbol{u}^2)=\left[\sigma^2(u_1^2)\ldots\sigma^2(u_p^2)\right]^T$ şi $\sigma^2(u)=u$) – funcția de activare (liniară) a stratului de ieşire, unde $\boldsymbol{u}^2=\boldsymbol{W}^2\boldsymbol{x}^2+\boldsymbol{b}^2\overset{not}{=}\left[u_1^1\ldots u_p^1\right]^T\in\mathbb{R}^p$, $\boldsymbol{x}^2\equiv\boldsymbol{y}^1=\boldsymbol{\sigma}^1(\boldsymbol{u}^1)$.

Transferul intrare-ieşire realizat este parametrizat de ponderile $\boldsymbol{W}^1\in\mathbb{R}^{m\times n}$, $\boldsymbol{W}^2\in\mathbb{R}^{p\times m}$ şi deplasările $\boldsymbol{b}^1\in\mathbb{R}^m$, $\boldsymbol{b}^2\in\mathbb{R}^p$ rețelei.

Fie funcția:

$$\varphi:\mathbb{R}^n\to\mathbb{R}^p, \tag{2}$$

netedă pe o mulțime compactă $S\subseteq\mathbb{R}^n$.

Este demonstrat faptul că, dacă rețeaua neuronală posedă *un număr adecvat de noduri* m în stratul de intrare, atunci parametrii rețelei pot fi determinați de aşa manieră încât $\boldsymbol{f}$ definită prin (1) *să aproximeze* pe S funcția φ din (2) *cu o precizie impusă* (în sensul de „oricât de bună"). Într-o formulare matematică riguroasă, acest rezultat remarcabil poate fi scris sub forma:

$$\begin{aligned}&\forall\varepsilon>0\quad\exists m\in\mathbb{N}\text{ şi }\boldsymbol{W}^1\in\mathbb{R}^{m\times n},\boldsymbol{b}^1\in\mathbb{R}^m,\boldsymbol{W}^2\in\mathbb{R}^{p\times m},\boldsymbol{b}^2\in\mathbb{R}^p\text{ a.î.}\\&\qquad\left\|\boldsymbol{f}(\boldsymbol{x})-\varphi(\boldsymbol{x})\right\|<\varepsilon\quad\forall\boldsymbol{x}\in S.\end{aligned} \tag{3}$$

Observație: Proprietatea de aproximare prezentată mai sus *nu furnizează informații concrete* privind alegerea lui $m\in\mathbb{N}$ şi determinarea lui $\boldsymbol{W}^1\in\mathbb{R}^{m\times n}$, $\boldsymbol{b}^1\in\mathbb{R}^m$, $\boldsymbol{W}^2\in\mathbb{R}^{p\times m}$, $\boldsymbol{b}^2\in\mathbb{R}^p$, ci *garantează numai existența acestor valori*. ♦

Din punct de vedere practic, exploatarea rezultatului discutat anterior are loc în următorul context. Funcția φ din (2) este cunoscută într-un *număr finit* N de puncte $\boldsymbol{x}_i \in S \subseteq \mathbb{R}^n$, pentru care avem:

$$\varphi(\boldsymbol{x}_i) = \boldsymbol{z}_i\ ,\ \ i = 1,...,N\ . \tag{4}$$

La intrarea rețelei se prezintă *vectorii prototip*:

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N \in \mathbb{R}^n, \tag{5}$$

pentru care se consideră *vectorii țintă*:

$$\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N \in \mathbb{R}^p \tag{6}$$

definiți conform (4). Notând prin $\boldsymbol{y}_i$, $i = 1,\ldots,N$, vectorii de ieşire ai MLP, adică:

$$\boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i) \in \mathbb{R}^p,\quad i = 1,\ldots,N, \tag{7}$$

se definesc vectorii eroare prin:

$$\boldsymbol{e}_i = \boldsymbol{z}_i - \boldsymbol{y}_i \in \mathbb{R}^p,\ \ i = 1,\ldots,N\ . \tag{8}$$

Pentru orice număr de neuroni m în stratul de intrare al rețelei, există valori ale parametrilor rețelei notate prin $\boldsymbol{W}^{1*} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b}^{1*} \in \mathbb{R}^m$, $\boldsymbol{W}^{2*} \in \mathbb{R}^{p\times m}$, $\boldsymbol{b}^{2*} \in \mathbb{R}^p$, care asigură *minimizarea erorii pătratice medii* (*Mean Squared Error* – MSE):

$$\sum_{i=1}^{N} \boldsymbol{e}_i^T \boldsymbol{e}_i \Big|_{\boldsymbol{W}^{1*},\boldsymbol{b}^{1*},\boldsymbol{W}^{2*},\boldsymbol{b}^{2*}} \le \sum_{i=1}^{N} \boldsymbol{e}_i^T \boldsymbol{e}_i \Big|_{\boldsymbol{W}^{1},\boldsymbol{b}^{1},\boldsymbol{W}^{2},\boldsymbol{b}^{2}\text{oarecare}}\ . \tag{9}$$

Observație

Parametrii $\boldsymbol{W}^{1*} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b}^{1*} \in \mathbb{R}^m$, $\boldsymbol{W}^{2*} \in \mathbb{R}^{p\times m}$, $\boldsymbol{b}^{2*} \in \mathbb{R}^p$ din (9) nu asigură şi satisfacerea condiției (3) pentru tot compactul $S \subseteq \mathbb{R}^n$. În schimb, inegalitatea (9) posedă o semnificație numerică concretă, furnizând astfel un obiectiv consistent pentru antrenarea rețelei neuronale. ♦

3.2. Obiectivul antrenării rețelei

Pornind de la *mulțimea vectorilor prototip* (5) şi *mulțimea vectorilor țintă* (6), prin antrenarea unei rețele cu m noduri în stratul de intrare trebuie să se determine parametrii acesteia $\boldsymbol{W}^{1*} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b}^{1*} \in \mathbb{R}^m$, $\boldsymbol{W}^{2*} \in \mathbb{R}^{p\times m}$, $\boldsymbol{b}^{2*} \in \mathbb{R}^p$ care să asigure *minimizarea erorii pătratice globale*, adică satisfacerea inegalității (9). Cu aceste valori ale parametrilor, funcția $\boldsymbol{f}$ definită prin (1) realizează o aproximare (neliniară) in sensul celor mai mici pătrate a funcției φ, cunoscută prin intermediul eşantioanelor (4). Valoarea minimă obținută în (9) depinde (în general) de numărul de noduri m din stratul de intrare al rețelei şi deci m va influența calitatea aproximării realizate de rețea pentru funcția φ.

3.3. Algoritmul de antrenare prin propagare inversă

Algoritmul de antrenare prin *propagare inversă* (eng. *backpropagation*) se bazează pe *minimizarea erorii pătratice medii* (MSE) prin *metoda gradientului*. Să notăm prin

$\boldsymbol{v} \in \mathbb{R}^{m(n+1)+p(m+1)}$ vectorul parametrilor rețelei construit cu elementele matricelor $\boldsymbol{W}^1 \in \mathbb{R}^{m\times n}$, $\boldsymbol{W}^2 \in \mathbb{R}^{p\times m}$ şi ale vectorilor $\boldsymbol{b}^1 \in \mathbb{R}^m$, $\boldsymbol{b}^2 \in \mathbb{R}^p$, elemente ce sunt dispuse în vectorul $\boldsymbol{v}$ conform unei reguli prestabilite (de exemplu, întâi liniile matricei $\left[\boldsymbol{W}^1\ \boldsymbol{b}^1\right]$, urmând apoi liniile matricei $\left[\boldsymbol{W}^2\ \boldsymbol{b}^2\right]$). Considerând vectorii eroare $\boldsymbol{e}_i$ definiți prin (8), ca funcții de parametrii rețelei (adică de elementele vectorului $\boldsymbol{v}$) se definește funcția obiectiv:

$$J(\boldsymbol{v}) = \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{e}_i^T(\boldsymbol{v})\,\boldsymbol{e}_i(\boldsymbol{v}). \tag{10}$$

Minimizarea prin metoda gradientului a lui $J(\boldsymbol{v})$ (care înseamnă satisfacerea condiției (9)) constă în modificarea vectorului parametrilor $\boldsymbol{v} \in \mathbb{R}^{m(n+1)+p(m+1)}$ la fiecare iterație, în conformitate cu formula:

$$\boldsymbol{v}_{\text{new}} = \boldsymbol{v}_{\text{old}} - \eta \nabla J(\boldsymbol{v}_{\text{old}}), \tag{11}$$

unde $\eta > 0$ notează *pasul metodei de optimizare*, iar $\nabla J(\boldsymbol{v}_{\text{old}}) = \left[\partial J/\partial \boldsymbol{v}\right]^T\Big|_{\boldsymbol{v}_{\text{old}}}$ notează vectorul gradient (normal la suprafețele de nivel constant ale lui J (10), care trec prin v_{old}). Printr-o exprimare adecvată a vectorului gradient, relația (11) poate fi formulată *direct în termenii parametrilor rețelei* $\boldsymbol{W}^1 \in \mathbb{R}^{m\times n}$, $\boldsymbol{b}^1 \in \mathbb{R}^m$, $\boldsymbol{W}^2 \in \mathbb{R}^{p\times m}$, $\boldsymbol{b}^2 \in \mathbb{R}^p$, furnizând astfel *mecanismul de efectuare a unei iterații* pentru *algoritmul de antrenare prin propagare inversă*. Termenul de "propagare inversă" se referă la maniera de calcul a valorilor curente ale vectorului gradient care se bazează pe o strategie de propagare a erorilor, definite prin (8), de la stratul de ieşire, înapoi către stratul de intrare.

Pentru algoritmul de antrenare se stabilesc, de către utilizator, un *număr maxim de iterații* sau *epoci* şi o valoare ε care se doreşte a fi atinsă în minimizarea funcției obiectiv (10) (adică *un prag pentru eroarea pătratică globală*). Atingerea unui prag *oarecare* de eroare ε impus de utilizator *nu este întotdeauna posibilă*, deoarece valoarea minimă a lui $J(\boldsymbol{v})$ definită prin (10) poate fi strict pozitivă.

De asemenea, se fixează (de către utilizator) o valoare pentru pasul metodei de optimizare $\eta > 0$, care, în contextul problemei de optimizare, este referită sub denumirea de *rată de învățare* (eng. *learning rate*).

Se organizează vectorii prototip $\boldsymbol{x}_i \in \mathbb{R}^n$, $i = 1,\ldots,N$, şi cei țintă $\boldsymbol{z}_i \in \mathbb{R}^p$, $i = 1,\ldots,N$, sub forma matricelor:

$$\boldsymbol{X} = [\boldsymbol{x}_1 \ldots \boldsymbol{x}_N] \in \mathbb{R}^{n\times N}, \tag{12}$$

respectiv:

$$\boldsymbol{Z} = [\boldsymbol{z}_1 \ldots \boldsymbol{z}_N] \in \mathbb{R}^{p\times N}. \tag{13}$$

La *fiecare iterație* a algoritmului de propagare inversă se parcurg etapele descrise mai jos. Valorile parametrilor rețelei la începerea unei iterații sunt notate prin $\boldsymbol{W}^1_{\text{old}} \in \mathbb{R}^{m\times n}$, $\boldsymbol{W}^2_{\text{old}} \in \mathbb{R}^{p\times m}$ (pentru ponderi) şi, respectiv, $\boldsymbol{b}^1_{\text{old}} \in \mathbb{R}^m$, $\boldsymbol{b}^2_{\text{old}} \in \mathbb{R}^p$ (pentru deplasări).

Etapa I (Etapa de prezentare)

Se prezintă vectorii prototip $\boldsymbol{x}_i$, $i=1,...,N$, la intrarea rețelei neurale, adică pentru valorile curente ale parametrilor rețelei ($\boldsymbol{W}^1_{\text{old}}$, $\boldsymbol{W}^2_{\text{old}}$, $\boldsymbol{b}^1_{\text{old}}$, $\boldsymbol{b}^2_{\text{old}}$) se calculează, conform (1-b), ieşirile rețelei, obținând:

$$\boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i) = \boldsymbol{\sigma}^2\left(\boldsymbol{W}^2_{\text{old}}\boldsymbol{\sigma}^1(\boldsymbol{W}^1_{\text{old}}\boldsymbol{x}_i + \boldsymbol{b}^1_{\text{old}}) + \boldsymbol{b}^2_{\text{old}}\right),\ i=1,...,N, \tag{14}$$

care se organizează sub forma matricei:

$$\boldsymbol{Y} = [\boldsymbol{y}_1 \dots \boldsymbol{y}_N] \in \mathbb{R}^{p\times N}. \tag{15}$$

Etapa II (Etapa de verificare)

Se calculează vectorii erorilor $\boldsymbol{e}_i$, $i=1,...,N$, definiți prin (8) ca diferența dintre vectorii țintă $\boldsymbol{z}_i \in \mathbb{R}^p$ şi vectorii ieşirii la iterația curentă $\boldsymbol{y}_i \in \mathbb{R}^p$, construind matricea

$$\boldsymbol{E} = [\boldsymbol{e}_1 \dots \boldsymbol{e}_N] = \boldsymbol{Z} - \boldsymbol{Y} \in \mathbb{R}^{p\times N}. \tag{16}$$

Algoritmul se opreşte dacă este îndeplinită una din următoarele două condiții:
(χ1) eroarea pătratică medie este inferioară pragului ε stabilit la startarea algoritmului, adică:

$$\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{e}_i^T\boldsymbol{e}_i < \varepsilon \tag{17}$$

sau:
(χ2) a fost atins numărul maxim de iterații stabilit la startarea algoritmului.

La oprire, algoritmul va furniza valorile parametrilor rețelei rezultate din antrenare, care vor fi notate prin $\boldsymbol{W}^1_f \in \mathbb{R}^{m\times n}$, $\boldsymbol{W}^2_f \in \mathbb{R}^{p\times m}$ (pentru ponderi) şi, respectiv, $\boldsymbol{b}^1_f \in \mathbb{R}^m$, $\boldsymbol{b}^2_f \in \mathbb{R}^p$ (pentru deplasări).

Dacă nici una din condițiile (χ1) sau (χ2) nu este îndeplinită, se continuă cu etapa următoare a iterației curente.

Etapa III (Etapa de calcul a vectorilor delta prin propagare inversă)

Pentru valorile curente ale parametrilor rețelei $\boldsymbol{W}^1_{\text{old}} \in \mathbb{R}^{m\times n}$, $\boldsymbol{W}^2_{\text{old}} \in \mathbb{R}^{p\times m}$, $\boldsymbol{b}^1_{\text{old}} \in \mathbb{R}^m$, $\boldsymbol{b}^2_{\text{old}} \in \mathbb{R}^p$, pentru fiecare vector prototip $\boldsymbol{x}_i$, $i=1,\dots,N$, se calculează *matricele Jacobian ale funcțiilor de activare* (vectoriale) ale celor două straturi, notate prin:

$$\boldsymbol{Q}^1_i \in \mathbb{R}^{m\times m},\ \boldsymbol{Q}^2_i \in \mathbb{R}^{p\times p},\ i=1,\dots,N.$$

- pentru stratul de intrare:

$$\boldsymbol{Q}^1_i = \operatorname{diag}\left\{q^1_i(1),...,q^1_i(m)\right\}, \tag{18-a}$$

unde:

$$q^1_i(k) = (\sigma^1)(u^1_i(k)) = \frac{1}{(\operatorname{Inv}\sigma^1)(y^1_i(k))},\quad k=1,...,m; \tag{18-b}$$

- pentru stratul de ieşire:

$$\boldsymbol{Q}_i^2 = \operatorname{diag}\left\{q_i^2(1),...,q_i^2(m)\right\}, \tag{19-a}$$

unde:

$$q_i^2(k) = (\sigma^2)(u_i^2(k)) = \frac{1}{(\operatorname{Inv}\sigma^2)(y_i^2(k))} = 1, \quad k = 1,...,m. \tag{19-b}$$

În relațiile (18-b) şi (19-b) care definesc elementele matricelor Jacobian, notațiile $(\operatorname{Inv}\sigma^1)$ şi $(\operatorname{Inv}\sigma^2)$ au semnificația de inverse ale funcțiilor de activare σ^1 şi respectiv σ^2 corespunzătoare celor două straturi. Deoarece funcțiile de activare ale celui de al doilea strat sunt liniare, $\boldsymbol{Q}_i^2$ este *matricea unitate* pentru $i = 1,\ldots,N$.

Cu ajutorul matricelor Jacobian $\boldsymbol{Q}_i^1 \in \mathbb{R}^{m\times m}$, $\boldsymbol{Q}_i^2 \in \mathbb{R}^{p\times p}$ şi a vectorilor erorilor $\boldsymbol{e}_i \in \mathbb{R}^p$, $i = 1,\ldots,N$, $i = 1,\ldots,N$, se definesc *vectorii delta* (prin procesarea informației de la ieşire, către intrare, procesare numită *propagare inversă*):

- pentru stratul de ieşire:

$$\boldsymbol{\delta}_i^2 = \boldsymbol{Q}_i^2\boldsymbol{e}_i = \boldsymbol{e}_i \in \mathbb{R}^p, \quad i = 1,\ldots,N; \tag{20}$$

- pentru stratul de intrare:

$$\boldsymbol{\delta}_i^1 = \boldsymbol{Q}_i^1\left((\boldsymbol{W}^2)^T\boldsymbol{\delta}_i^2\right) \in \mathbb{R}^m, \quad i = 1,\ldots,N. \tag{21}$$

Astfel, suntem în măsură să construim *matricele delta* ale celor două straturi:

- pentru stratul de ieşire

$$\Delta^2 = [\boldsymbol{\delta}_i^2 ....\boldsymbol{\delta}_N^2] \in \mathbb{R}^{p\times N}, \tag{22}$$

- pentru stratul de intrare

$$\Delta^1 = [\boldsymbol{\delta}_i^1 ....\boldsymbol{\delta}_N^1] \in \mathbb{R}^{m\times N}. \tag{23}$$

O reprezentare sub forma de schemă bloc a modului de calcul al *vectorilor delta* prin propagare inversă într-o rețea cu două straturi este dată în fig. 2.

Etapa IV (Etapa de actualizare a parametrilor)

Cu ajutorul matricelor delta se pot calcula *matricele de actualizare* a parametrilor aplicând *regula delta* pe fiecare strat:

- pentru stratul de ieşire

- ponderi

$$\boldsymbol{D}_{\boldsymbol{W}^2} = \Delta^2(\boldsymbol{X}^2)^T \in \mathbb{R}^{p\times m}, \tag{24-a}$$

- deplasări

$$\boldsymbol{D}_{\boldsymbol{b}^2} = \Delta^2 \underbrace{[1\ \ldots\ 1]^T}_{N \text{ elemente}} \in \mathbb{R}^p; \tag{24-b}$$

- pentru stratul de intrare

- ponderi

$$\boldsymbol{D}_{\boldsymbol{W}^1} = \Delta^1\boldsymbol{X}^T \in \mathbb{R}^{m\times n}, \tag{25-a}$$

- deplasări

$$\boldsymbol{D}_{\boldsymbol{b}^1} = \Delta^1 \underbrace{[1 \ldots 1]}_{N \text{ elemente}}^T \in \mathbb{R}^m . \tag{25-b}$$

În relația (21-a), matricea $\boldsymbol{X}^2 \in \mathbb{R}^{m \times N}$ colectează vectorii $\boldsymbol{x}_i^2 = \boldsymbol{y}_i^1 \in \mathbb{R}^m$, $i = 1, \ldots, N$, care se prezintă la intrarea celui de-al doilea strat, adică:

$$\boldsymbol{X}^2 = [\boldsymbol{x}_1^2 \ldots \boldsymbol{x}_N^2] \in \mathbb{R}^{m \times N} . \tag{26}$$

Pe baza matricelor de actualizare se determină noile valori ale parametrilor rețelei:

- pentru stratul de ieşire

- ponderi

$$\boldsymbol{W}_{\text{new}}^2 = \boldsymbol{W}_{\text{old}}^2 + \eta \boldsymbol{D}_{\boldsymbol{W}^2} , \tag{27-a}$$

- deplasări

$$\boldsymbol{b}_{\text{new}}^2 = \boldsymbol{b}_{\text{old}}^2 + \eta \boldsymbol{D}_{\boldsymbol{b}^2} ; \tag{27-b}$$

- pentru stratul de intrare

- ponderi

$$\boldsymbol{W}_{\text{new}}^1 = \boldsymbol{W}_{\text{old}}^1 + \eta \boldsymbol{D}_{\boldsymbol{W}^1} , \tag{28-a}$$

- deplasări

$$\boldsymbol{b}_{\text{new}}^1 = \boldsymbol{b}_{\text{old}}^1 + \eta \boldsymbol{D}_{\boldsymbol{b}^1} . \tag{28-b}$$

Cu aceste calcule, iterația curentă se încheie şi se trece la o nouă iterație, efectuând atribuirile:

$$\begin{aligned} &\boldsymbol{W}_{\text{new}}^2 \to \boldsymbol{W}_{\text{old}}^2, \quad \boldsymbol{b}_{\text{new}}^2 \to \boldsymbol{b}_{\text{old}}^2, \\ &\boldsymbol{W}_{\text{new}}^1 \to \boldsymbol{W}_{\text{old}}^1, \quad \boldsymbol{b}_{\text{new}}^1 \to \boldsymbol{b}_{\text{old}}^1. \end{aligned} \tag{29}$$

Observații

1° Inițializarea parametrilor rețelei (adică $\boldsymbol{W}_{\text{old}}^1 \in \mathbb{R}^{m \times n}$, $\boldsymbol{W}_{\text{old}}^2 \in \mathbb{R}^{p \times m}$, $\boldsymbol{b}_{\text{old}}^1 \in \mathbb{R}^m$, $\boldsymbol{b}_{\text{old}}^2 \in \mathbb{R}^p$ la prima iterație a algoritmului) se face cu valori *subunitare arbitrare*; în acest mod se asigură senzitivitate maximă pentru funcțiile sigmoidale din primul strat al rețelei.

2° Convergența algoritmului de propagare inversă se realizează în condițiile specifice metodei gradientului (ca tehnică de optimizare), fiind puternic dependentă de valoarea ratei de învățare η şi anume:

- pentru η inadecvat mic, deplasarea către optim, în spațiul parametrilor, este foarte lentă,
- pentru η inadecvat mare, deplasarea către optim, în spațiul parametrilor, se realizează cu oscilații (eventual cu pierderea stabilității algoritmului).

3° În cazul când algoritmul se încheie prin satisfacerea condiției (χ1), acuratețea parametrilor furnizați $\boldsymbol{W}_f^1 \in \mathbb{R}^{m \times n}$, $\boldsymbol{W}_f^2 \in \mathbb{R}^{p \times m}$ (ponderi) şi $\boldsymbol{b}_f^1 \in \mathbb{R}^m$, $\boldsymbol{b}_f^2 \in \mathbb{R}^p$ (deplasări) depinde de valoarea pragului ε stabilit pentru eroarea pătratică globală.

4° Suprafața erorii $J(\boldsymbol{v})$ definită prin (10) uzual posedă *minime locale* în care căutarea minimului poate eşua, dacă rata de învățare are o valoare prea mică. Astfel de minime locale

nu pot fi părăsite decât prin schimbarea valorii ratei de învățare, sau prin restartarea algoritmului cu alte valori inițiale ale parametrilor.

5^0 Numărul de neuroni din stratul de intrare se alege în funcție de "gradul de neliniaritate" care se constată la funcția φ din examinarea eşantioanelor (4).

3.4. Utilizarea rețelelor feedforward cu trei straturi

Rețelele feedforward cu trei straturi (primele două cu noduri sigmoidale şi cel de ieşire cu noduri liniare) posedă proprietatea de aproximare discutată în paragraful 3.1. Uzual se face apel la astfel de rețele atunci când funcția ce trebuie aproximată φ din (2) prezintă moduri de variație mai complicate, puse în evidență de eşantioanele (4).

Antrenarea se realizează prin metoda propagării inverse organizată pe cele trei straturi prin generalizarea algoritmului prezentat în paragraful 3.3 aplicând *regula delta* succesiv celor trei straturi. În acest caz, vor trebui calculați *vectorii şi matricele delta* pentru toate cele trei straturi, într-o manieră analogă cu (17) – (20).

4. Facilități software oferite de Neural Network Toolbox

Funcția **newff** creează o rețea feed-forward, numărul de straturi şi de neuroni din stratul de intrare şi din straturile ascunse, precum şi funcția de activare a neuronilor fiecărui strat fiind specificate de către utilizator în apelul funcției. După crearea rețelei utilizatorul trebuie să furnizeze parametrii de antrenare doriți.

Pentru inițializarea cu valori arbitrare a parametrilor unei rețele se poate utiliza funcția **rands**. De asemenea, pentru inițializarea parametrilor se poate apela funcția **init**.

Funcția **train** antrenează rețeaua conform rutinei de antrenare specificate la creare şi a parametrilor de antrenare aleşi. Algoritmul de antrenare prin propagare inversă descris mai sus este implementat de rutina '**traingd**'. Un algoritm de antrenare mai rapid este algoritmul Levenberg-Marquardt, implementat în rutina '**trainlm**'.

În cazul unui neuron cu o singură intrare, pentru reprezentarea grafică tridimensională a suprafeței erorii pătratice globale (ca funcție de ponderea şi deplasarea neuronului) se pot utiliza funcțiile **errsurf** şi **plotes** (apelate în această ordine).

Pentru simularea rețelei antrenate se poate folosi funcția **sim**.

5. Problematica propusă pentru studiu

Problema 1

Se va studia şi lansa în execuție programul ***arna_bp1*** elaborat în MATLAB, care antrenează (prin aplicarea regulii delta) un neuron cu funcție sigmoidală bipolară, pentru a aproxima o funcție neliniară φ definită prin vectorii prototip:

$$x_i = \frac{(i-1)\pi}{12}, \quad i = 1,...,7,$$

şi vectorii țintă:

$$z_i = \sin(x_i), \quad i = 1,...,7\,.$$

Să se opereze modificările necesare în program asupra ratei de învățare şi a numărului de iterații pentru a realiza antrenarea, pornind de la condițiile inițiale situate în următoarele domenii:

1. $(w_0, b_0) \in [-4.5, -1.5] \times [-4.5, -1.5]$
2. $(w_0, b_0) \in [-1.5, 1.5] \times [-4.5, -1.5]$
3. $(w_0, b_0) \in [1.5, 4.5] \times [-4.5, -1.5]$
4. $(w_0, b_0) \in [-4.5, -1.5] \times [-1.5, 1.5]$
5. $(w_0, b_0) \in [-1.5, 1.5] \times [-1.5, 1.5]$
6. $(w_0, b_0) \in [1.5, 4.5] \times [-1.5, 1.5]$
7. $(w_0, b_0) \in [-4.5, -1.5] \times [1.5, 4.5]$
8. $(w_0, b_0) \in [-1.5, 1.5] \times [1.5, 4.5]$
9. $(w_0, b_0) \in [1.5, 4.5] \times [1.5, 4.5]$

În fiecare din situațiile (1) - (9) se vor preciza:

a) valorile inițiale ale parametrilor, reprezentarea grafică a aproximării realizate de rețea pe baza acestor valori şi eroarea corespunzătoare;
b) valorile finale ale parametrilor (rezultate din antrenare), reprezentarea grafică a aproximării realizate de rețea pe baza acestor valori şi eroarea corespunzătoare;
c) numărul total de iterații cât a durat antrenarea şi reprezentarea grafică a evoluției erorii ca funcție de numărul de iterații;
d) traiectoria în planul parametrilor (w, b) – reprezentare grafică;
e) valorile utilizate pentru parametrii de antrenare comentate prin prisma convergenței algoritmului şi a formei suprafeței erorii. Se vor raporta şi explica eventualele eşecuri în antrenare, datorate alegerii neadecvate a parametrilor de antrenare.

Problema 2

Să se utilizeze programul ***arna_bp1*** (cu modificările necesare considerate) pentru a selecta o regiune din planul parametrilor (w, b) şi o gamă de valori a ratei de învățare care să asigure antrenarea eficientă pentru perechile de vectori (prototip, țintă):

$$x_i = \frac{\pi}{2} + \frac{(i-1)\pi}{12}; \;\; z_i = \sin(x_i); \;\; i = 1,...,7.$$

Să se comenteze criteriile care au condus la selectarea respectivă, prin referire la convergența algoritmului şi forma suprafeței erorii.

Problema 3

Se va studia şi lansa în execuție programul ***arna_bp2*** elaborat în MATLAB care antrenează o rețea cu două straturi (cu noduri tansigmoidale pe primul strat şi liniare pe al doilea) pentru a aproxima o funcție neliniară $\varphi : \mathbb{R} \to \mathbb{R}$ cunoscută prin intermediul perechilor de vectori prototip-țintă:

$$(x_i, z_i), \quad z_i = \varphi(x_i), \quad i = 1,...,N\,.$$

Se consideră vectorii ţintă:

$$x_i = (i-1)/50, \quad i = 1,...,51,$$

şi vectorii prototip:

$$z_i = \varphi(x_i) = 1 + \exp(-x_i)\sin(2\pi x_i - \frac{\pi}{2}), \quad i = 1,...,51.$$

Antrenarea reţelei se va face folosind mai întâi metoda gradientului (rutina '**traingd**'), iar apoi algoritmul Levenberg-Marquardt (rutina '**trainlm**'). În fiecare din cele două cazuri se vor studia următoarele:

a) Operând modificările necesare în programul ***arna_bp2*** să se studieze rolul următorilor factori asupra eficienţei antrenării reţelei:

- numărul de neuroni în stratul de intrare;
- rata de învăţare (doar în cazul folosirii rutinei '**traingd**').

Aprecierea eficienţei se va face în termenii erorii pătratice globale obţinute în urma antrenării şi a numărului de iteraţii necesare pentru atingerea unui prag impus pentru eroare.
Se va comenta modul de alegere a valorilor iniţiale pentru parametrii reţelei.

b) Pentru testele de antrenare eficiente realizate, se vor preciza valorile parametrilor reţelei rezultaţi în urma antrenării.

Problema 4

Să se reia Problema 3 pentru aceiaşi vectori ţintă şi vectorii prototip:

$$z_i = \varphi(x_i) = 1 + \exp(-x_i)\sin(8\pi x_i - \frac{\pi}{2}), \quad i = 1,...,51.$$

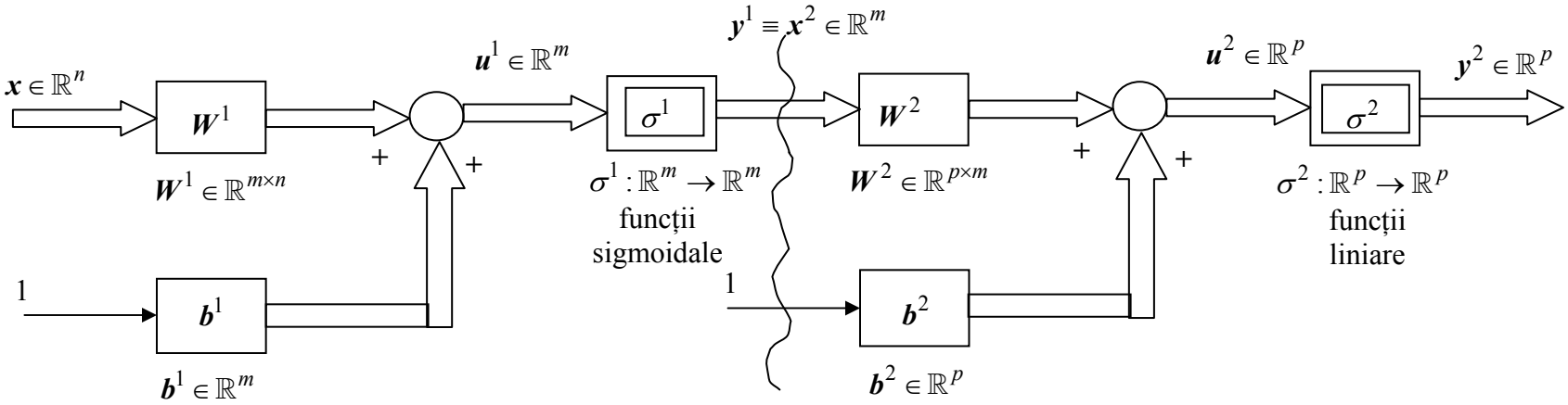

Fig.1. Schema bloc a unei rețele neuronale cu două straturi, utilizată în probleme de aproximare neliniară

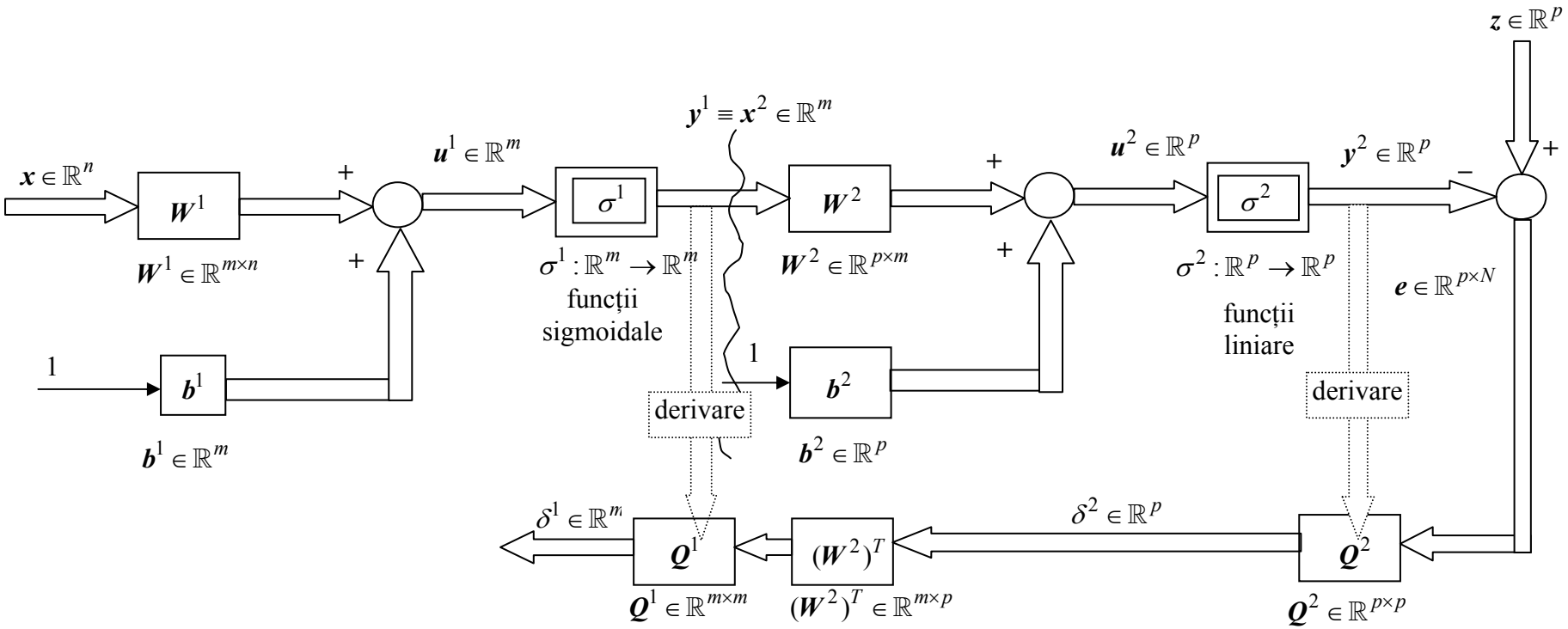

Fig.2. Schema bloc a procesării în sens invers a informației în antrenarea unei rețele cu două straturi prin algoritmul propagării inverse - calculul vectorilor delta: δ^2 și δ^1