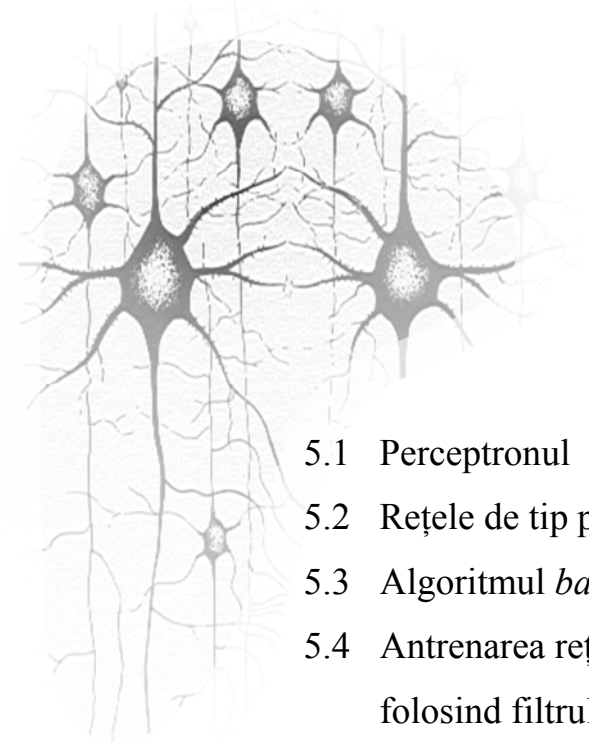
Capitolul **5**

Rețele neurale de tip perceptron multistrat



5.1 Perceptronul

5.2 Rețele de tip perceptron multistrat

5.3 Algoritmul *backpropagation*

5.4 Antrenarea rețelelor MLP folosind filtrul Kalman

În următoarele două capitole prezentăm pe larg rețelele neurale de tip *forward* (cu propagare înainte). Vom începe cu introducerea unui model elementar de rețea cu un singur strat, denumit *perceptron*, utilizat cu precădere în aplicații de clasificare şi vom continua cu prezentarea rețelelor multistrat denumite *Multilayer Perceptron* (MLP), respectiv *Radial Basis Functions* (RBF). Ambele tipuri de sisteme reprezintă în prezent soluțiile folosite cel mai des în rezolvarea unor categorii extrem de largi de aplicații practice, care vor fi ilustrate prin exemple specifice. Motivația fundamentală o reprezintă aşa-numita **capacitate de aproximare universală** a acestora potrivit căreia, în anumite condiții, orice funcție neliniară multidimensională se poate aproxima în limitele unei toleranțe oricât de mici. Vor fi trecute în revistă o serie de aspecte teoretice care justifică afirmația anterioară, precum şi elemente de ordin practic, utile în proiectarea unor rețele neurale de acest tip şi configurarea corectă a unor experimente concrete. Nucleul prezentării din acest capitol îl constituie familia de algoritmi de învățare specifică rețelelor MLP, denumită generic *backpropagation* (cu propagare inversă a erorii). Mai mult, vom insista asupra aspectelor legate de asigurarea capacității de generalizare, fundamentală pentru utilizarea cu succes a oricărui tip de rețea neurală.

5.1 Perceptronul

Majoritatea specialiştilor sunt de părere că anul 1943 constituie momentul de început al cercetării moderne în domeniul rețelelor neurale, marcat de apariția câtorva lucrări de referință. Un loc aparte îl ocupă lucrarea *"A logical calculus of the ideas immanent in nervous activity"* a cercetătorilor americani McCulloch şi Pitts în care este introdus un model simplu al neuronului, privit ca element de procesare individual. Inspirat în mare măsură de aspectul biologic real, dar caracterizat de simplificări majore, modelul va sta la baza multora dintre structurile complexe studiate mai târziu. În Fig. 5.1 se prezintă schema-bloc a modelului, descris de relația de legătură intrare-ieşire [76]:

$$y(n+1) = f\left(\sum_{i=1}^{N} w_i x_i[n] + \theta\right) = \begin{cases} 1, \ pentru \ \sum_{i=1}^{N} w_i x_i[n] + \theta \geq 0 \\ -1, \ pentru \ \sum_{i=1}^{N} w_i x_i[n] + \theta \leq 0 \end{cases} \quad (5.1)$$

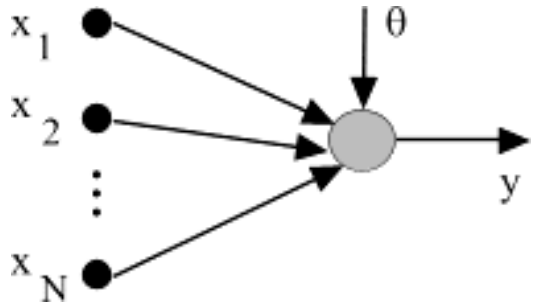


Fig. 5.1: Rețea neurală de tip perceptron

După cum se observă, modelul propus este *discret*, cu intrări *binare* şi cu o funcție de activare de tip comparator cu prag (Fig. 2.2). În plus, ponderile w_i pot lua numai una dintre valorile ± 1, după cum efectul intrării corespunzătoare este excitator sau inhibitor. Interesant de observat este faptul că, deşi extrem de simplu, acest model permite totuşi implementarea funcțiilor logice elementare NAND şi NOR, pe baza cărora se pot sintetiza funcții logice combinaționale, iar folosind posibilitatea de a obține celule de întârziere cu un tact se pot construi şi circuite secvențiale.
Restricția la valori binare ale intrării şi mai ales ale ponderilor (care sunt fixate la valori constante, fără posibilitatea de modificare în funcție de performanțele rețelei), precum şi tipul funcției de activare, alături de necesitatea de *funcționare sincronă* a rețelelor construite din astfel de neuroni elementari constituie limitări majore ale modelului McCulloch-Pitts, care au impus apariția unor soluții superioare. Un exemplu semnificativ în acest sens îl constituie *perceptronul*, introdus în 1958 de către Rosenblatt, denumire atribuită unei rețele neurale elementare utilizate în aplicații de clasificare [155]. Modelul McCulloch-Pitts suferă o modificare esențială, atât datele de intrare cât şi ponderile putând căpăta orice valoare reală. Cu toate acestea, gama de aplicații a unor rețele formate dintr-un *singur strat* de astfel de neuroni elementari este în continuare restrânsă numai la probleme de clasificare în care datele de intrare sunt *liniar separabile*[1].
Există două modalități de obținere a valorilor vectorului de ponderi **W** care să asigure o funcționare corectă a clasificatorului de tip perceptron:

[1] Clasele distincte sunt separate prin aşa-numite *suprafețe de decizie.* Pentru determinarea acestora rețeaua evaluează un set de *funcții de discriminare* $g_i(\mathbf{X})$. Clasele se numesc *liniar separabile* dacă aceste funcții sunt de forma $g_i(\mathbf{X}) = \mathbf{WX}$.

• calculul direct al *suprafețelor de decizie* (care separă clasele adiacente) pe baza unor informații disponibile *a priori* referitoare la natura şi apartenența datelor de intrare
• determinarea suprafețelor de decizie în urma unui *proces de învățare supravegheată*, conform unui algoritm adecvat.

Fără îndoială că această din urmă variantă este de preferat în cazul unui număr sporit de clase distincte şi/sau a unui număr mare de date de intrare. În acest context, meritul istoric al rețelei neurale de tip perceptron constă tocmai în elaborarea unui mecanism de adaptare a valorilor parametrilor sistemului (interconexiunile dintre neuroni şi valorile de prag ale funcției de activare) a cărui convergență a putut fi demonstrată riguros.În varianta inițială, funcția de activare aleasă pentru neuronul individual a fost aceeaşi ca în modelul McCulloch-Pitts. Justificarea teoretică a posibilității de antrenare a unui simplu perceptron (discret) în vederea clasificării corecte a unui set de date în 2 clase distincte este asigurată de o binecunoscută teoremă de convergență elaborată de Rosenblatt [155], bazată pe utilizarea algoritmului din Tabelul 5.1.

Tabelul 5.1: Algoritmul de învățare al perceptronului

Variabile şi parametri:	
Vectorul de coeficienți la iterația n:	$\mathbf{W}[n] = [\ \theta\ w_1[n]\ w_2[n]\ ...w_{M-1}[n]\ ]^T$
Vectorul de intrare la iterația n:	$\mathbf{X}[n] = [-1\ x_1[n]\ x_2[n]\ \dots\ x_{M-1}[n]^T$
Răspunsul dorit la iterația n:	$D[n] \in \{1, -1\}$
Valoarea ieşirii rețelei la iterația n:	$Y[n] \in \{1, -1\}$
Constanta de învățare:	$0 \le \eta \le 1$
Inițializare:	
Valoarea inițală a coeficienților:	$\mathbf{W}[0]$ (tipic = **0)**
Calcul iterativ: n = 1, 2, ...	
Se calculează ieşirea la iterația n:	$Y[n] = \text{sgn}\ \{\mathbf{W}[n]^T\mathbf{X}[n]\}$
Se adaptează valorile ponderilor:	$\mathbf{W}[n+1] = \mathbf{W}[n] + \eta\{d[n] - y[n]\}\mathbf{X}[n]$

Se pot face o serie de observații în legătură cu algoritmul prezentat:

• numărul de iteraţii necesar asigurării unei clasificări corecte depinde de constanta de adaptare şi de succesiunea datelor folosite în etapa de antrenare
• ponderile se modifică numai dacă apar clasificări greşite
• coeficientul de învăţare este constant

Perceptronul cu funcţie de activare de tip comparator s-a dovedit însă incapabil să rezolve probleme de clasificare relativ simple, în cazul în care clasele erau *neseparabile liniar*. În 1969, Minsky şi Papert publicau lucrarea *Perceptrons*, în care demonstrau eşecul algoritmului de mai sus în rezolvarea aşa-numitei "probleme XOR" [2] (problemă de clasificare în 2 clase distincte definită pe baza tabelei de adevăr corespunzătoare funcţiei logice SAU-EXCLUSIV, prin care se cere ca perechile de date (-1,-1), (1,1) să fie asociate unei categorii, iar perechile (1,-1), (-1,1) celeilalte categorii). Mai mult, autorii îşi exprimau convingerea (dovedită mai târziu nejustificată) că această lipsă de eficacitate se regăseşte şi în cazul reţelelor de tip perceptron multistrat.

O modificare importantă a modelului prezentat mai sus o constituie considerarea unei funcţii de activare derivabile, monoton crescătoare, de tip sigmoidal (Fig. 2.2):

$$f(x) = \frac{1}{1+e^{-ax}} \tag{5. 2}$$

O asemenea opţiune este importantă nu numai pentru că sugerează utilizarea unor tehnici de adaptare binecunoscute în vederea obţinerii valorilor parametrilor reţelelor neurale cu un singur strat (din categoria celor prezentate în Capitolul 2), dar mai ales pentru că a permis apariţia unei clase de algoritmi foarte puternici pentru antrenarea reţelelor de tip perceptron multistrat, capabile să rezolve un număr însemnat de aplicaţii extrem de interesante, printre care şi clasificarea corectă a unor date care fac parte din clase neseparabile liniar.

Determinarea valorilor optime ale parametrilor reţelelor cu funcţie de activare continuă şi derivabilă (în particular a celor liniare, analizate în Capitolul 3) urmăreşte în general minimizarea unei funcţii de eroare convenabil aleasă J(**W**). Dintre metodele posibil de adoptat, majoritatea cercetătorilor optează pentru tehnica scăderii după gradient (*gradient descent*), potrivit căreia modificarea la un moment dat a unei ponderi oarecare w_i se efectuează pe direcţia după care funcţia de eroare prezintă scăderea cea mai accentuată:

[2] Generalizarea o reprezintă *problema parităţii*, în care ieşirea reţelei trebuie să fie 1 sau -1 după cum numărul de biţi egali cu 1 din vectorul binar aplicat la intrare este par sau impar.

$$\Delta w_i[n] = -\eta \frac{\partial J[n]}{\partial w_i[n]} \tag{5. 3}$$

unde η este o constantă reală şi pozitivă. În cazul rețelei neurale de tip perceptron cu funcție de activare sigmoidală, utilizând o funcție de eroare pătratică se poate scrie:

$$J[n] = \frac{1}{2}(d[n] - y[n])^2 = \frac{1}{2}\{d[n] - f(\mathbf{W}^T[n]\mathbf{X}[n])\}^2 \tag{5. 4}$$

în care mărimile care intervin au aceeaşi semnificație ca în Tabelul 5.1 şi, în plus, s-a presupus că a fost luată în considerație în calculul erorii prezentarea unei singure perechi de date intrare-ieşire (în mod normal, eroarea totală rezultă prin sumarea erorilor individuale corespunzătoare prezentării întregii baze de date avute la dispoziție). Din relațiile anterioare rezultă:

$$\frac{\partial J}{\partial w_i} = \{d[n] - y[n]\} f'(a)\mathbf{X}[n] = \delta[n] f'(a)\mathbf{X}[n] \tag{5. 5}$$

în care am notat prin $a = \sum_{i=0}^{M-1} w_i x_i$ semnalul total aplicat la intrarea neuronului de ieşire (activarea neuronului), iar prin δ[n] eroarea instantanee (această notație o vom regăsi mai târziu şi în cazul rețelelor multistrat, în particular pentru exprimarea corecției aplicate ponderilor aferente neuronilor plasați în straturile ascunse ale rețelelor multistrat).

> *Observație:* spre deosebire de algoritmul descris în Tabelul 5.1, în această situație ajustarea ponderilor se produce chiar şi în cazul unei decizii *corecte* de clasificare. Ca urmare, se poate arăta că perceptronul cu funcție de activare continuă *nu* garantează întotdeauna clasificarea corectă chiar pentru cazul simplu al unor clase liniar separabile [180].

5.2 Rețele de tip perceptron multistrat

Pentru a elimina neajunsurile specifice modelului de tip perceptron putem acționa în două direcții distincte, anume: a) a lăsa nemodificată arhitectura cu un singur

strat, dar a folosi funcții de activare mai complicate pentru neuronul elementar; b) a modifica arhitectura sistemului, prin introducerea unor straturi suplimentare, păstrând modelul de perceptron standard pentru neuronii individuali. Posibile soluții pentru prima categorie pot fi considerate aşa-numitele modele de ordin superior, printre care neuronii de tip *sigma-pi* [157], sau cele inspirate de descompunerea de tip Volterra [24]. A doua variantă reprezintă de departe soluția folosită în majoritatea aplicațiilor practice, sub forma unor arhitecturi cu 2 sau 3 straturi, ca în Fig. 5.2.

După cum s-a mai arătat, succesul unor asemenea sisteme se datorează în principal capacității demonstrate de a aproxima în limitele unei toleranțe oricât de mici orice funcție neliniară, oricât de complicată, fără a impune constrângeri de modelare. Înainte de a trece în revistă principalele rezultate teoretice referitoare la acest aspect, este interesant de a oferi o imagine intuitivă asupra modalității de reprezentare a informației în rețele neurale multistrat. Astfel, să considerăm o rețea cu 3 straturi, având neuroni cu funcție de activare de tip sigmoidal în cele două straturi ascunse, respectiv funcție de activare liniară în stratul de ieşire. Ieşirea unuia dintre neuronii plasați în primul strat ascuns (considerat ca având numai 2 intrări, pentru a permite reprezentarea grafică) arată ca în Fig. 5.3a. La intrarea neuronilor plasați în stratul al doilea se formează combinații liniare ale unor astfel de suprafețe. Considerând două dintre acestea, cu aspect asemănător dar uşor deplasate una față de alta, obținem o suprafață ca în Fig. 5.3b.

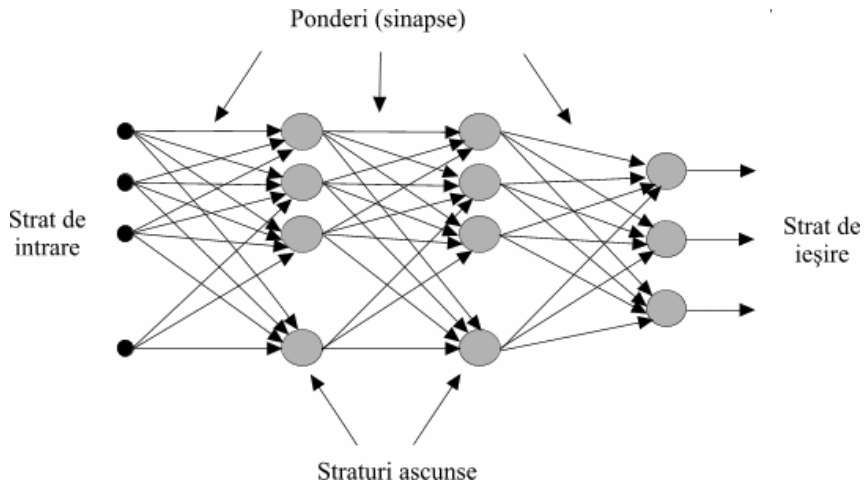


Fig. 5.2: Rețea de tip perceptron multistrat (MLP) cu 3 straturi

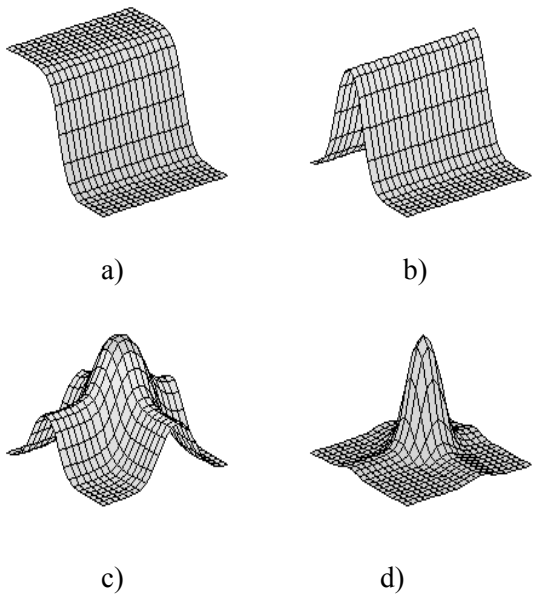


Fig. 5.3: Reprezentări interne ale neuronilor din straturile rețelei MLP

În continuare ne putem imagina combinații ale suprafețelor obținute, având orientări diferite, astfel încât ajungem la reprezentarea din Fig. 5.3c. Reaminitim că o reprezentare din această categorie corespunde intrării în neuronii din stratul al doilea, dotați de asemenea cu funcție de activare sigmoidală, astfel încât ieşirea unui astfel de neuron se va prezenta ca în Fig. 5.3d. Nu mai rămâne decât să sumăm un număr suficient de mare de astfel de "vârfuri" la nivelul stratului de ieşire pentru a obține în final aproximația funcției dorite. Sigur că acest mod de operare este unul intuitiv, însă ilustrează sugestiv posibilitatea ca prin *combinația liniară a unor răspunsuri individuale bine localizate* să putem modela funcții neliniare foarte complicate. Acest principiu va fi regăsit mai târziu în cazul rețelelor neurale de tip *Radial Basis Functions*.
Interesant de remarcat este că pe baza unor considerente tot de natură geometrică se poate justifica capacitatea sporită de modelare a unor rețele multistrat nu numai din perspectiva aplicațiilor de regresie, ci şi a celor de clasificare. Astfel, în Fig. 5.4 se prezintă tipurile de suprefețe de decizie construite cu rețele multistrat având funcție de activare de tip comparator, respectiv de tip sigmoidal.

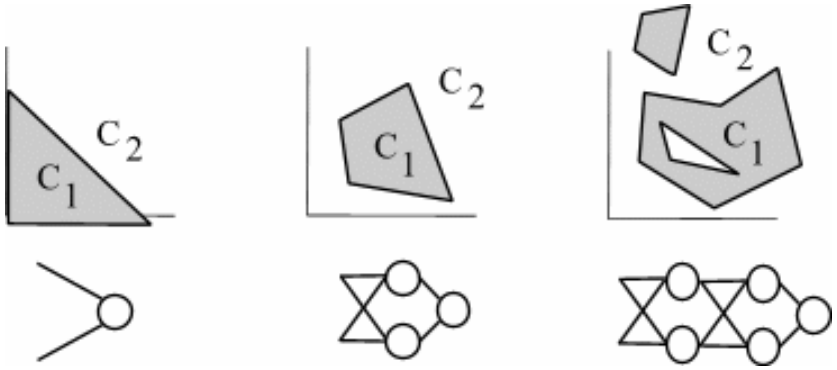


Fig. 5.4: Tipuri de suprafețe de decizie construite cu rețele MLP

5.2.1 Teoreme de aproximare universală

La sfârşitul secolului al XIX-lea celebrul matematician german D. Hilbert a reunit un set de peste 20 de probleme încă nerezolvate, lăsându-le drept "moştenire" viitoarelor generații de matematicieni. Una dintre acestea, cea de a treisprezecea, se referea la posibilitatea ca o funcție de mai multe variabile să poată fi exprimată prin combinația unor funcții având mai puține variabile (în particular, Hilbert enunța părerea că există funcții de 3 variabile care nu se pot exprima sub forma unei combinații de funcții de numai 2 variabile). Un răspuns cu caracter general a fost dat de către matematicianul rus Kolmogorov în anii '50, care a demonstrat că orice funcție continuă de mai multe variabile cu domeniu de definiție închis şi mărginit poate fi reprezentată ca o combinație finită de funcții de o singură variabilă [76]. Cu toate că importanța practică a acestui rezultat în contextul teoriei rețelelor neurale este limitată[3], în literatură se face deseori sublinierea potrivit căreia metodologia de aproximare a unei funcții neliniare oarecare prin intermediul unor rețele de tip *feedforward* multistrat reprezintă "traducerea în practică" a teoremei elaborate de către Kolmogorov.

[3] Limitările teoremei formulate de către Kolmogorov se referă pe de o parte la necesitatea de a utiliza un număr *fix* de funcții de o singură variabilă, pe de altă parte la dependența acestei funcții de natura particulară a funcției multivariabile aproximate.

Două sunt tipurile de rezultate de interes în privința capacității diverselor tehnici de aproximare: a) aproximarea universală, care se referă la posibilitatea de a aproxima orice funcție continuă cu un grad de acuratețe oricât de ridicat; b) capacitatea de a oferi cea mai bună aproximare posibilă (*best aproximation capability*), în sensul că din totalitatea de funcții posibil de furnizat de către o metodă dată, există una situată la cea mai mică distanță față de funcția supusă procesului de aproximare.
Există un număr apreciabil de articole care demonstrează capacitatea de aproximare universală a rețelelor multistrat [52], [85], [86]. Astfel, în [52] se demonstrează că orice funcție continuă poate fi aproximată folosind o rețea neurală de tip MLP cu un singur strat ascuns, ai cărui neuroni au funcție de activare sigmoidală. Mai mult, în [85] se extinde valabilitatea rezultatului anterior la funcții de activare mult mai generale, care trebuie doar să îndeplinească condițiile de a fi continue şi mărginite. De asemenea, în [33] se demonstrează că proprietatea de aproximare universală este îndeplinită pentru toate funcțiile de activare diferite de cele polinomiale.
În privința posibilității de a oferi cea mai bună aproximare pentru o funcție dată, se demonstrează că rețelele MLP standard nu posedă această capacitate, pe când rețelele de tip *Radial Basis Functions* (RBF) – care implementează tot arhitecturi *feedforward,* cu un singur strat ascuns, dar având funcții de activare diferite de cele sigmoidale (de exemplu, de tip gaussian) – prezintă această proprietate.
Mai mult, există aplicații practice precum controlul roboților sau rezolvarea ecuațiilor diferențiale, în care este necesară aproximarea simultană atât a unei funcții neliniare oarecare cât şi a derivatei de ordinul I a acesteia. În [86] este abordată această problemă şi sunt indicate condițiile ce trebuie satisfăcute pentru ca acest fapt să fie posibil.
Există o serie de aspecte importante ce trebuie subliniate în contextul subiectului abordat:

- proprietatea de aproximare universală implică, printre altele, posibilitatea ca rețelele multistrat să poată fi utilizate în aplicații de clasificare, în care rolul funcției ce trebuie aproximate este jucat de aşa-numitele *suprafețe de discriminare* (care separă elementele aparținând unor clase distincte). De asemenea, se pot utiliza în scopul modelării *densităților de probabilitate* în raport cu care se efectuează operația de clasificare [24].
- calitatea aproximării, numărul de parametri ai rețelei neurale şi dimensiunea bazei de date avute la dispoziție sunt interdependente. În [18] se formulează analitic această dependență, iar în [19] se demonstrează că eroarea de aproximare scade invers proporțional cu numărul de neuroni din stratul ascuns.
- calitatea aproximării furnizate de către o metodă oarecare, în particular de către o rețea neurală, depined de precizia cu care se efectuează calculele. Datorită

caracterului finit al acesteia, o serie de proprietăți demonstrate teoretic în condițiile unei precizii de lucru infinite îşi pierd relevanța [182].

- rezultatele teoretice enumerate anterior nu oferă de regulă şi soluții constructive care să indice care este arhitectura adecvată (număr de straturi, număr de neuroni pe fiecare strat) rezolvării cu suces a unei probleme de aproximare dată. În practică se folosesc de multe ori metode empirice, bazate pe simulări extensive folosind arhitecturi de dimensiuni variabile sau se face apel la una dintre cele 2 categorii de tehnici prezentate în Capitolul 4 capabile să confere capacitate de generalizare sporită.
- tipul particular al algoritmului de învățare influențează natura concretă a dependenței funcționale intare-ieşire implementate de către o rețea neurală în decursul procesului de antrenare. O imagine intuitivă putem căpăta închipuindu-ne mecanismul de învățare ca pe un traseu parcurs pe un relief reprezentând funcția de eroare definită prin compararea răspunsului dorit cu cel real al rețelei neurale. Relieful este reprezentat într-un sistem de coordonate definit de parametrii rețelei, iar obiectivul urmărit este atingerea valorii minime a acestuia. În orice punct de pe traseu coordonatele acestuia definesc de fapt o relație intrare-ieşire particulară, care aproximează mai bine sau mai rău funcția dorită (precum o bucată de tablă care, după fiecare lovitură aplicată cu un ciocan, se apropie progresiv de forma dorită). Cele mai multe tehnici de căutare (în particular, binecunoscuta familie de algoritmi denumită *backpropagation*) urmează direcția definită de gradientul funcției de eroare, astfel încât, în realitate, ieşirea rețelei neurale nu va putea defini chiar orice funcție posibilă (cu alte cuvinte, relieful avut la dispoziție nu va fi explorat în întregime, ci vor fi urmate numai anumite "cărări", dependente de punctul de pornire). Intuitiv, ca o bucată de tablă să capete forma unei portiere auto nu este necesar ca aceasta să fi căpătat *temporar* forma unei capote!

Există o serie de probleme concrete care apar în studiile teoretice referitoare la capacitatea de aproximare universală a rețelelor neurale, pe care le vom trece în revistă în cele ce urmează. Deoarece justificarea (inclusiv definițiile!) acestora presupun utilizarea unui aparat matematic specializat şi, de regulă, dificil de urmărit de către un nespecialist, prezentarea va fi sumară şi concentrată pe concluziile utile din punct de vedere practic.

Pentru simplitate şi în acord cu majoritatea lucrărilor referitoare la acest subiect, ne vom referi numai la rețele de tip *feedforward* cu un singur strat ascuns şi având un singur neuron în stratul de ieşire, cu funcție de activare liniară. Astfel, ieşirea unei rețele neurale având n neuroni în stratul ascuns cu funcția de activare $\phi(.)$ şi care primeşte la intrare vectorul $\mathbf{X} \in R^p$ va fi:

$$y(\mathbf{X}) = \sum_{k=1}^{n} a_k \phi(\mathbf{w}_k * \mathbf{X} + b_k) \tag{5. 6}$$

unde $\{\mathbf{w}_k \in R^p\}$ şi $\{a_k \in R\}$ desemnează seturile de ponderi corespunzătoare stratului ascuns, respectiv celui de ieşire, iar $\{b_k \in R\}$ reprezintă valorile de prag asociate funcțiilor de activare ale neuronilor din stratul ascuns. Clasa funcțiilor de tipul (5. 6) o vom nota compact sub forma $N_{\phi, p, n}$. Este necesară introducerea prealabilă a unor definiții:

- *Distanța:* Această noțiune este fundamentală pentru a aprecia acuratețea aproximării. În cele mai multe aplicații suntem interesați ca modelul folosit să asigure o calitate bună a aproximării în fiecare punct **X** aparținând unui **domeniu compact** $K \subset R^p$, astfel încât putem folosi aşa-numita distanță uniformă dintre două funcții:

$$d(f,g) = \sup_{X \in K} \left| f(x) - g(x) \right| \tag{5. 7}$$

  Alteori suntem interesați de aprecierea calității medii a procesului de aproximare, astfel încât putem folosi distanțe de tip L^p:

$$d(f,g) = \left[\int_{R^p} \left| f(x) - g(x) \right|^p d\mu(x) \right]^{\frac{1}{p}} = \left\| f - g \right\| \tag{5. 8}$$

  unde $1 \leq p < \infty$, $\mu(x)$ este o măsură asociată domeniului de intrare, iar norma unei funcții este definită prin relația:

$$\left\| f \right\| = \left[\int_{R^p} \left| f(x) \right|^p d\mu(x) \right]^{\frac{1}{p}} \tag{5. 9}$$

- *Densitate:* Aspectul fundamental referitor la caracterul dens al reprezentării pe un **domeniu compact** $K \subset R^p$ folosind clasa $N_{\phi, p, n}$ reprezintă suportul mai multor demonstrații raportate în literatură în acest context. În principiu, folosind diverse instrumente teoretice, se identifică natura proprietăților pe care

trebuie să le aibă funcția de activare $\phi(.)$ astfel încât, pentru orice funcție continuă $f \in C(R^p)$ şi orice $\varepsilon > 0$, să existe $g \in N_{\phi, p, n}$ cu proprietatea:

$$\max_{\mathbf{X} \in K} \left| f(\mathbf{X}) - g(\mathbf{X}) \right| < \varepsilon \tag{5. 10}$$

Ținând cont de definițiile anterioare au fost demonstrate o serie de teoreme riguroase, dintre care amintim:

Teorema 1 ([85]): *Dacă $\phi(.)$ este continuă, mărginită şi nu este constantă, atunci clasa de funcții $N_{\phi, p, n}$ este densă în C(**X**) pentru domenii de definiție compacte $K \subset R^p$ (C(**X**) desemnează spațiul funcțiilor continue pe **X**).*

Teorema 2 ([52]): *Dacă $\phi(.)$ este continuă (dar nu neapărat monotonă), atunci clasa de funcții $N_{\phi, p, n}$ este densă în C(**X**) pentru domenii de definiție compacte $K \subset R^p$.*

Rezultatul având caracterul cel mai general este formulat astfel:

Teorema 3 ([33])**:** Setul $N_{\phi, p, n}$ este dens în $C(R^p)$ (din punctul de vedere al aproximării pe domenii compacte $K \subset R^p$) dacă şi numai dacă funcția ϕ *nu este polinomială*.

O explicație imediată a condiției din cuprinsul teoremei anterioare este legată de caracterul **finit** al gradului unui polinom care ar juca rolul funcției $\phi(.)$. Ca urmare, o reprezentare de tipul (5. 6) ar corespunde la rândul ei tot unui polinom cu grad finit, iar mulțimea tuturor polinoamelor de grad finit **nu este densă** pe un domeniu compact $K \subset R^p$.

Aceste rezultate reprezintă pur şi simplu expresii teoretice ale capacității de aproximare pe care o posedă modelul descris de relația (5. 6), însă nu indică şi modalitatea practică de a obține aproximația dorită.

Deşi cele mai multe analize teoretice conduc la concluzia că funcția de activare $\phi(.)$ trebuie să fie mărginită (iar unele demonstrații îngustează şi mai mult clasa funcțiilor posibil de utilizat la funcții continue şi monotone) se poate arăta că există şi funcții nemărginite pentru care proprietatea referitoare la densitate se menține

(de exemplu, funcția $\phi(t) = e^t$). În același context, merită menționată şi justificarea riguroasă a observației potrivit căreia este suficient ca aspectul densității de reprezentare să fie abordat numai pentru aproximarea funcțiilor de o singură variabilă [33].

- *Complexitate:* acest aspect se referă la estimarea numărului de neuroni din stratul ascuns capabil să ofere aproximația unei funcții date $f \in C(R^p)$ în limitele unei toleranțe predefinite $\varepsilon > 0$, în particular la modalitatea în care dimensiunea necesară a rețelei este influențată de alegerea funcției de activare $\phi(.)$. Un rezultat des citat în literatură a fost prezentat în [18], potrivit căruia *eroarea pătratică* (generată de norma L^2) este mărginită inferior de valoarea:

$$O\left(\frac{1}{n}\right) + O\left(\frac{np\log N}{N}\right) \qquad (5.\ 11)$$

în care n este numărul de neuroni din stratul ascuns, p este dimensiunea vectorilor de intrare şi N este numărul de exemplare care formează baza de date disponibilă. Acest rezultat important poate fi pus în corespondență cu formula fundamentală prezentată în Capitolul 4 referitoare la aşa-numita descompunere *bias-variance* [65]: primul termen (de tip *bias*) defineşte eroarea de aproximare şi exprimă imposibilitatea ca o rețea neurală cu un număr *finit* de parametri să asigure modelarea perfectă a unei funcții necunoscute oarecare, iar al doilea termen (de tip *variance*) reprezintă efectul utilizării unei *baze de date* de dimensiune finită. Un aspect interesant se referă la indicarea unei maniere constructive de a sintetiza rețeaua neurală, adăugând rând pe rând câte un singur neuron care să ofere cea mai bună aproximare nu pentru funcția originală, ci pentru funcția care defineşte eroarea de aproximare (obținută ca diferența dintre funcția originală şi aproximația oferită de arhitectura curentă a rețelei).
Dintr-un alt punct de vedere, aspectul legat de complexitatea procesului de aproximare se referă la relația dintre volumul de calcul necesar şi numărul total de parametri ai modelului utilizat. Într-o serie de lucrări se demonstrează că ajustarea parametrilor unei rețele neurale cu arhitectură fixă nu poate fi realizată într-un interval de timp care să depindă polinomial de numărul total de parametri ai modelului (ca terminologie, se spune că o astfel de problemă este *NP-completă*).
În finalul acestui paragraf trecem în revistă sumar o serie de alte aspecte de interes referitoare la capacitatea de aproximare a rețelelor neurale, unele dintre acestea având şi o importantă conotație practică:

- după cum s-a menționat anterior, multe dintre demonstrațiile întâlnite în literatură au caracter existențial, în sensul că justifică riguros capacitatea de aproximare universală fără a indica şi modalitatea concretă prin care se poate construi o rețea adecvată unei aplicații practice concrete. O excepție în acest sens întâlnim în [33], [34], studii care identifică şi importanța caracterului **mărginit** (nu neapărat şi continuu şi/sau monoton!) al funcției de activare a neuronilor de pe stratul ascuns. Studiile conduc şi la concluzia importantă că este suficientă demonstrarea capacității de aproximare universală a unei funcții de o singură variabilă pe baza modelului oferit de o rețea neurală cu un singur strat ascuns pentru a putea extinde valabilitatea acestei proprietăți şi la cazul funcțiilor de mai multe variabile!
- efortul cercetătorilor s-a concentrat în special în direcția identificării condițiilor în care o rețea neurală multistrat asigură aproximarea satisfăcătoare a unei funcții oarecare. La mijlocul anilor '90 o serie de articole au făcut câțiva paşi mai departe, ocupându-se de posibilitatea ca rețelele multistrat să asigure aproximarea universală a unor funcționale, respectiv operatori[4]. Din punctul de vedere al teoriei sistemelor, funcționalele pot fi privite ca reprezentând răspunsul unui sistem la un semnal de intrare oarecare, valabil la **un moment dat precizat**. De asemenea, un operator (în cazul cel mai simplu, liniar) defineşte răspunsul sistemului la un semnal de intrare **la orice moment de timp**. Ca urmare, existența unor justificări riguroase referitoare la capacitatea de aproximare universală a funcționalelor şi operatorilor permite ca rețelele neurale să poată fi utilizate în aplicații de identificare de sistem (în care intervine în mod nemijlocit caracterul dinamic), după cum capacitatea de aproximare a unor funcții permite rezolvarea problemelor de recunoaştere de forme (*pattern recognition*), care presupun de regulă un caracter static.
- un aspect care survine în mod natural în discuția referitoare la posibilitatea aproximării unor funcții oarecare cu ajutorul modelelor oferite de rețelele neurale de tip *feedforward* este următoarea: există motive pentru a utiliza rețele care includ mai mult de un singur strat ascuns de neuroni? O justificare teoretică general valabilă a răspunsului la o asemenea întrebare nu a fost formulată. Cum atât rețelele cu 2 straturi ascunse cât şi cele cu 3 straturi posedă capacitate universală de aproximare rezultă că trebuie avute în vedere alte criterii pentru a justifica opțiunea pentru una sau alta dintre aceste variante, cum ar fi influența dimensiunilor finite ale arhitecturii şi bazei de date asupra

[4] O *funcție* reprezintă legătura dintre o pereche de numere. O *funcțională* reprezintă legătura dintre o funcție şi un număr, după cum un *operator* reprezintă legătura dintre două funcții.

erorii de aproximare, capacitatea de generalizare sau complexitatea calculelor. Utilizând rețele cu 2 şi 3 straturi având acelaşi număr *total* de parametri, în [149] se prezintă o analiză comparativă efectuată pe un set limitat de probleme de clasificare. Concluziile studiului indică faptul că, *în medie*, performanţe mai bune se obțin folosind rețele cu 2 straturi, însă există întotdeauna şi rețele cu 3 straturi capabile să ofere o calitate a aproximării comparabilă cu cel mai bun rezultat obținut cu rețele având 2 straturi. Pe de altă parte, în [164] se demonstrează că rezolvarea unor *probleme de inversiune* necesită în mod obligatoriu 2 straturi ascunse. În aplicațiile practice rareori se folosesc rețele având mai mult de 2 straturi ascunse, excepția reprezentând-o anumite variante ale sistemelor autoasociative [49]. Un punct suplimentar în favoarea rețelelor utilizând 3 straturi îl constituie posibilitatea ca eroarea de aproximare folosind un număr *finit* de neuroni pe straturile ascunse să poată fi făcută oricât de mică, spre deosebire de cazul rețelelor cu 2 straturi pentru care, aşa cum rezultă din relația (5. 11), există o limită inferioară nenulă.

5.3 Algoritmul *backpropagation*

Celebra lucrare a cercetătorilor americani Minsky şi Papert amintită deseori în paragrafele anterioare referitoare la performanțele limitate ale modelului de tip perceptron a "înghețat" interesul pentru domeniul rețelelor neurale pentru aproape două decenii. În ciuda elaborării unor algoritmi adaptivi valabili în cazul filtrelor discrete liniare (văzute ca cele mai simple exemple de rețele neurale), precum şi a existenței unei palete largi de tehnici de optimizare a unor funcții multidimensionale, lipsa unei posibilități practice de ajustare a ponderilor aferente neuronilor localizați în straturile ascunse ale rețelelor multistrat – pentru care nu sunt disponibile valori ale răspunsurilor dorite, precum în cazul celor de la ieşire! – a întârziat mult dezvoltarea acestui domeniu. De-abia în 1986 Rumelhart, Hinton şi Williams au propus o soluție practică de rezolvare a acestei probleme, "reinventând" de fapt un algoritm prezentat încă din 1974 în teza de doctorat a lui Werbos (într-un context care nu avea legătură cu domeniul rețelelor neurale). Algoritmul este denumit *backpropagation* (cu propagare inversă a erorii) şi a declanşat apariția unei adevărate familii de tehnici de calcul atât a valorilor interconexiunilor din rețelele neurale cât şi a altor mărimi de interes. Ideea de bază este de a specula caracterul derivabil al funcției de activare a neuronilor elementari (în particular, de tip sigmoidal), respectiv al funcției de eroare (în particular, de tip eroare pătratică), pentru a evalua valorile derivatei funcției de eroare în raport cu oricare dintre ponderile rețelei, utilizând intensiv mecanismul derivărilor succesive.

În mod concret, răspunsul unui neuron oarecare din structura unei rețele neurale de tip *feedforward* se poate scrie [76]:

$$z_j = f(a_j) = f(\sum_i w_{ji} z_i) \tag{5. 12}$$

în care funcția f(.) se presupune derivabilă iar a_j desemnează activarea neuronului. În cazul în care neuronii care livrează semnal către neuronul cu indicele j sunt plasați chiar în stratul de intrare atunci variabilele z_i vor fi notate cu x_i, iar dacă neuronul j este plasat în stratul de ieşire atunci vom nota $z_j = y_j$.
Algoritmul urmăreşte minimizarea unei funcții de eroare presupuse la rândul ei derivabile, care poate fi scrisă sub forma:

$$J = \sum_n J[n] \tag{5. 13}$$

unde indicele n contorizează numărul de perechi intrare-ieşire dorită existente în baza de date avută la dispoziție. Să presupunem că pentru fiecare dintre aceste perechi au fost calculate activările, respectiv răspunsurile tuturor neuronilor din rețea, pe baza relațiilor anterioare. Observând că eroarea instantanee J[n] depinde de o pondere oarecare w_{ji} numai prin intermediul activării a_j aferente neuronului destinație j, se poate scrie:

$$\frac{\partial J[n]}{\partial w_{ji}} = \frac{\partial J[n]}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \tag{5. 14}$$

Vom introduce notația auxiliară:

$$\delta j = \frac{\partial J[n]}{\partial a_j} \tag{5. 15}$$

Din relația (5. 12) rezultă:

$$\frac{\partial a_j}{\partial w_{ji}} = z_i \tag{5. 16}$$

astfel încât se poate scrie:

$$\frac{\partial J[n]}{\partial wji} = \delta_j z_i \tag{5. 17}$$

Nu rămâne de făcut decât să identificăm modalitatea de calcul a variabilelor δ_j asociate fiecărui neuron din structura rețelei. Pentru neuronii plasați în stratul de ieşire rezultă imediat:

$$\delta_k = \frac{\partial J[n]}{\partial a_k} = g'(a_k)\frac{\partial J[n]}{\partial y_k} \tag{5. 18}$$

Valorile variabilelor δ_k vor rezulta fără dificultate în urma precizării unor forme concrete ale funcției de activare a neuronilor, respectiv ale funcției de eroare. În Tabelul 5.2 sunt indicate formulele valabile în cazul funcției de activare de tip sigmoidal, respectiv al funcției de eroare pătratice.
Pentru evaluarea variabilelor δ_j corespunzătoare neuronilor din straturile ascunse utilizăm metoda derivărilor înlănțuite pentru a scrie:

$$\delta_j = \frac{\partial J[n]}{\partial a_j} = \sum_k \frac{\partial J[n]}{\partial a_k}\frac{\partial a_k}{\partial a_j} = \sum_k \frac{\partial J[n]}{\partial a_k}\frac{\partial a_k}{\partial z_j}\frac{\partial z_j}{\partial a_j} \tag{5. 19}$$

unde indicele k se referă la neuronii *către care* neuronul j trimite semnal (Fig. 5.5). Ținând cont de relațiile (5. 12) şi (5. 15) rezultă:

$$\delta_j = g'(a_j)\sum_k w_{kj}\delta_k \tag{5. 20}$$

adică valoarea variabilei δ corespunzătoare unui neuron situat într-unul dintre straturile ascunse depinde de valorile aceleeaşi variabile aferente neuronilor plasați "după acesta", adică înspre ieşirea rețelei. Ca urmare, se pot calcula în mod iterativ aceste valori pornind dintre ieşirea spre intrarea reşelei, de unde şi denumirea dată algoritmului – cu propagare inversă a erorii (*backpropagation*).
Nu mai rămâne decât să aplicăm relația (5. 17) pentru a obține valorile derivatelor parțiale ale erorii instantanee J[n] în raport cu ponderea w_{ji} şi a utiliza în final această informație pentru ajustarea efectivă a ponderii prin intermediul unui algoritm de optimizare adecvat, în particular de tip scădere după gradient.

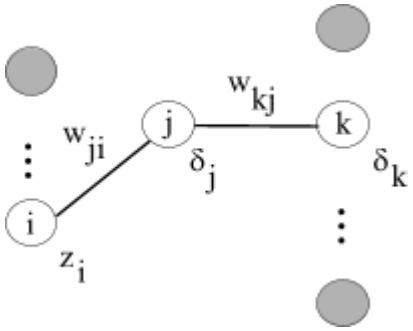


Fig. 5.5: Detaliu din arhitectura unei rețele MLP

În legătură cu algoritmul descris anterior se pot face o serie de observații:

- în practică se preferă acumularea valorilor derivatelor parțiale $\frac{\partial J[n]}{\partial w_{ji}}$ corespunzătoare prezentării unei singure perechi de date intrare-ieşire dorită prin repetarea procedurii de mai sus pentru un număr mai mare de astfel de perechi, în particular pentru "trecerea în revistă" a întregii baze de date avute la dispoziție şi de-abia apoi să se recurgă la ajustarea propriu-zisă a valorilor ponderilor rețelei. Ca terminologie, distingem din acest punct de vedere antrenarea de tip *pattern-by-pattern*, de tip *block*, respectiv de tip *batch*. Derivata parțială a erorii totale acumulate se scrie sub forma:

$$\frac{\partial J}{\partial w_{ji}} = \sum_n \frac{\partial J[n]}{\partial w_{ji}} \tag{5. 21}$$

- în relațiile anterioare s-a presupus că pentru simplitate că toți neuronii din rețea au aceeaşi funcție de activare. Această cerință nu este obligatorie, formulele putând fi modificate cu uşurință în acord cu particularitățile rețelei considerate (de exemplu, în multe situații funcția de activare a neuronilor din stratul de ieşire este liniară).
- subliniem încă o dată caracterul foarte general al analizei anterioare, care permite obținerea după acelaşi principiu şi a valorilor altor informații de interes exprimate sub formă de derivate parțiale (de exemplu, a valorilor matricii

Jacobian $J_{ki} = \frac{\partial y_k}{\partial x_i}$ utilizate pentru evaluarea *sensitivității* ieşirilor rețelei în raport cu fiecare dintre intrări, respectiv ale matricii Hessian $\frac{\partial^2 J}{\partial w_{ji} \partial w_{lk}}$, ambele fiind folosite în definirea unor tehnici specifice de îmbunătățire a performanțelor de generalizare ale unei rețele neurale din categoria celor prezentate în Capitolul 4).

În Tabelul 5.2 sunt prezentate relațiile deduse anterior pentru cazul particular al unei funcții de activare de tip sigmoidal, respectiv al unei funcții de eroare de tip pătratic. Este important de observat că utilizarea unei asemenea funcții de activare aduce cu sine avantajul că valoarea derivatei se poate exprima comod sub forma:

$$f(x) = \frac{1}{1+e^{-x}}; \quad f'(x) = f(x)[1-f(x)] \tag{5. 22}$$

5.3.1 Analiza algoritmului backpropagation

Vom enumera în continuare elementele care influențează semnificativ performanțele algoritmului *backpropagation*, ilustrând cauzele unor posibile rezultate nesatisfăcătoare şi oferind indicații utile în aplicațiile practice.

- *Funcția de activare a neuronului elementar*

În mod uzual, aceasta se alege dintre următoarele două variante, prima fiind bipolară (Fig. 2.2e) şi a doua unipolară (Fig. 2.2f):

$$f_1(x) = \tanh(ax) = \frac{1-e^{-ax}}{1+e^{-ax}}; \qquad f_2(x) = \frac{1}{1+e^{-ax}} \tag{5. 23}$$

Au fost raportate rezultate care indică creşteri semnificative ale vitezei de convergență în cazul utilizării unei funcții de activare bipolare [57]. În plus, ajustarea parametrului a (care controlează valoarea derivatei în origine a funcției) conform aceleeaşi reguli de scădere după gradient are uneori efecte benefice, în special în faza inițială a procesului de învățare. O motivație elegantă a utilității

unor astfel de funcţii de activare sigmoidale (în afară de plauzibilitatea biologică) se prezintă în [70]. Din punctul de vedere al algoritmului prezentat în Tabelul 5.2, un avantaj major îl constituie relaţia simplă dintre funcţiile f(.) şi derivatele de ordinul I ale acestora (de exemplu, $df_2(x)/dx = af_2(1-f_2)$), care conduce la evaluarea rapidă a variabilelor **δ**.
În aplicaţii de clasificare se utilizează deseori funcţia denumită *softmax*, care permite interpretarea ieşirilor reţelei neurale ca *probabilităţi condiţionate* [27]:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{5. 24}$$

Derivata funcţiei de activare are un efect semnificativ asupra vitezei de convergenţă. Observaţia elementară potrivit căreia modificarea ponderilor unui neuron (direct proporţională cu mărimea derivatei) este neglijabilă pentru semnale de intrare de amplitudine mare (pentru care derivata practic se anulează) a impus creşterea artificială a valorii derivatei. Astfel, simpla adăugare a unei constante pozitive mici la valoarea reală a derivatei a permis în unele situaţii reducerea la jumătate a timpului de antrenare a reţelei [57].

Tabelul 5.2: Algoritmul *backpropagation*

Variabile şi parametri:	
Numărul de straturi:	L
Numărul de neuroni pe stratul l*:	$C^{(l)}$
Vectorul de intrare la iteraţia n:	$\mathbf{X}[n] = \{-1\ x_1[n]\ x_2[n]\ \ldots\ x_P[n]\}^T$
Vectorul ponderilor unui neuron din stratul l la iteraţia n:	$\mathbf{W}^{(l)}[n]$
Răspunsul dorit la iteraţia n:	$\mathbf{d}[n]$
Vectorul nivelelor de activare ale neuronilor din stratul l la iteraţia n:	$\mathbf{a}^{(l)}[n]$

Vectorul ieşirilor neuronilor din stratul l la iterația n:	$\mathbf{z}^{(l)}[n]$
Vectorul valorilor locale ale gradientului funcției de eroare corespunzător neuronilor din stratul l la iterația n:	$\boldsymbol{\delta}^{(l)}[n]$
Vectorul ieşirilor rețelei la iterația n:	$\mathbf{y}[n] = [y_1[n]\ y_2[n]\ \ldots\ y_Q[n]\}^T$
Vectorul erorilor rețelei la iterația n:	$\mathbf{e}[n] = [e_1[n]\ e_2[n]\ \ldots\ e_Q[n]\}^T$
Constanta de învățare:	$0 \le \eta \le 1$
Inițializare:	
Valori inițale ale ponderilor:	Valori aleatoare mici
Calcul iterativ (n = 1, 2,…) pentru propagarea înainte: l = 1, 2, ... L	
Nivelul de activare al neuronului j din stratul l la iterația n:	$a_j^{(l)}[n] = \sum_{i=0}^{C^{(l-1)}} w_{ji}^{(l)}[n] z_i[n]$
Ieşirea neuronului j din stratul l la iterația n:	$z_j^{(l)}[n] = \frac{1}{1+e^{-a_j^{(l)}[n]}}$
Eroarea corespunzătoare ieşirii neuronului j din stratul l la iterația n:	$e_j[n] = d_j[n] - y_j[n]$
Calcul iterativ (n = 1, 2,…) pentru propagarea inversă: l = 1, 2, ... L	
Se calculează valorile locale ale gradienților funcției de eroare la iterația n**:	$\delta_j^{(L)}[n] = y_j[n](1-y_j[n])e_j[n]$ $\delta_j^{(l)}[n] = z_j^{(l)}[n](1-z_j^{(l)}[n]) \times$ $\times \sum_k \delta_k^{(l+1)}[n] w_{kj}^{(l+1)}[n]$
Se ajustează valorile ponderilor care unesc neuronii din stratul l cu cei din stratul (l-1):	$w_{ji}^{(l)}[n+1] = w_{ji}^{(l)}[n] +$ $+ \eta \delta_j^{(l)} z_i^{(l-1)}[n]$

* numărul intrărilor în rețea este $P = C^{(0)}$, iar dimensiunea stratului de ieşire este $Q = C^{(L)}$

** limita superioară a sumei după k poate fi cel mult $C^{(l+1)}$, însă nu este obligatoriu ca un neuron să fie conectat la toți neuronii din stratul următor

• *Funcția de eroare*

După cum s-a mai amintit, scopul urmărit prin antrenarea unei rețele neurale este de a *modela procesul* care a generat perechile de date intrare-ieşire disponibile şi nu de a memora aceste date. Descrierea cea mai completă a acestui proces se face în termenii densităților de probabilitate. După cum s-a arătat în Capitolul 2, motivația utilizării celor mai multe dintre funcțiile de eroare uzuale este oferită de binecunoscutul principiu al plauzibilității maxime (*maximum likelihood*), care reprezintă o metodă de estimare parametrică a densității de probabilitate a unui proces aleator pe baza unui set finit de date [24].
În formă generală, funcția de eroare se poate scrie sub forma:

$$J = \sum_i \sum_j \left| d_{ji} - y_j(\mathbf{X}_i, \mathbf{W}) \right|^R \qquad (5.\ 25)$$

unde y_j desemnează vectorul valorilor de ieşire din rețea, **W** reuneşte totalitatea parametrilor rețelei (ponderi şi valori de prag), iar i indexează seturile de perechi intrare-ieşire din baza de date disponibilă. Pentru R=2 se obține eroarea pătratică.

> *Observație:* a) eroarea pătratică este extrem de sensibilă la prezența unor erori individuale mari (*outliers*). Performanțe superioare se obțin folosind tipuri de distanțe cu R<2 (de exemplu distanța Manhattan, pentru care R=1).
> b) funcția de eroare definită anterior este specifică modului de antrenare de tip *batch* (ajustarea parametrilor rețelei se face după fiecare prezentare integrală a bazei de date disponibile).

Este important de subliniat că aspectul funcției de eroare – suprafață multidimensională cu numeroase minime locale – este **independent de algoritmul de învățare!** Ca urmare, vom întâlni aplicații ce pot fi caracterizate ca fiind simple sau complicate în funcție de natura concretă a dependenței intrare-ieşire ce trebuie modelată şi de baza de date disponibilă, iar gradul de dificultate va fi acelaşi îndiferent de natura algoritmului de antrenare utilizat. De altfel, după cum indică aşa-numitele teoreme de tip *No Free Lunch* [181] amintite în Capitolul 4, în lipsa unor constrângeri predefinite şi calculând în medie performanțele pentru totalitatea sarcinilor ce ar putea fi rezolvate de către o rețea neurală, nici nu există diferențe sistematice între diverşii algoritmi de învățare.
Există mai multe surse de apariție a minimelor locale ale unei funcții de eroare. Una dintre acestea este legată de tipul funcției de activare utilizate. Astfel,

utilizarea unei funcții sigmoidale care tinde asimptotic dar nu atinge niciodată limitele uzuale {-1, 1} în care se plasează gama dinamică a valorilor *target* poate conduce la apariția un rminime locale. O altă sursă o reprezintă natura concretă a bazei de date: exemplare neseparabile liniar nu pot fi învățate perfect folosind rețele fără straturi ascunse. În acest context merită amintite rezultatele riguroase care indică absența minimelor locale în cazul rețelelor capabile să clasifice corect date aparținînd unor categorii separabile liniar [69].

• *Inițializarea valorilor ponderilor*

După cum s-a arătat anterior, funcția de eroare pătratică asociată unei rețele neurale de tip multistrat – văzută ca funcție dependentă de ansamblul parametrilor rețelei – se prezintă sub forma unei hipersuprafețe cu aspect complicat, caracterizată de regulă prin prezența unui ansamblu de valori minime locale care însoțesc valoarea minimă absolută. Deşi în unele situații cantonarea într-o astfel de soluție suboptimală poate fi justificată prin compromisul dintre performanțele obținute şi viteza de convergență, de cele mai multe ori sunt necesare teste repetate pornind din condiții inițiale diferite pentru a ajunge la rezultate satisfăcătoare. Este adevărat că există posibilitatea ca, prin alegerea adecvată a arhitecturii rețelei, să diminuăm uneori şansa apariției unor astfel de minime locale [69], însă problema alegerii valorilor inițiale ale ponderilor rămâne una foarte importantă. În literatură se întâlnesc numeroase soluții, de cele mai multe ori nejustificate teoretic şi testate numai pe cazuri particulare de aplicații. Cea mai des întâlnită indicație este aceea de a alege valori inițiale aleatoare ale ponderilor rețelei, justificată de pricipiul "spargerii simetriei" (*breaking the symmetry* [157]), bazat pe observația că inițializarea cu valori *identice* nu poate conduce la "dezvoltarea" unor parametri către valori finale diferite (şi cerute ca atare de aplicația considerată). Un alt indiciu este furnizat de necesitatea de a preîntâmpina *saturarea prematură* a ieşirilor neuronilor şi deci scăderea vitezei de convergență în cazul utilizării algoritmilor de tip scădere după gradient, de tipul *backpropagation* (în zonele de saturație ale funcției de transfer de tip sigmoidal valoarea derivatei este foarte redusă şi, în consecință, ajustarea valorilor ponderilor este practic nesemnificativă).
Cele mai multe studii experimentale conduc la concluzia comună că valorile inițiale ale parametrilor rețelei neurale trebuie alese în mod aleator, dintr-o gamă de valori repartizate uniform în intervalul $[-a/\sqrt{F_i}\ ,\ a/\sqrt{F_i}\ ]$, unde F_i desemnează numărul total de intrări într-un neuron (*fan-in*), iar a este o constantă pozitivă, aleasă tipic în intervalul [2, 3]. De asemenea, în mod orientativ, valoarea semnalului total de intrare într-un neuron cu funcție de activare sigmoidală se poate lua în jurul lui 1 [25], pentru a asigura funcționarea acestuia într-o zonă în care amplitudinea derivatei este semnificativă.

• *Numărul de neuroni din straturile ascunse*

Lipsa unor teoreme constructive care să indice în mod clar dimensiunile (număr de straturi şi număr de neuroni pe fiecare strat) unei reţele neurale utilizate într-o aplicaţie dată constituie în continuare una dintre limitările majore ale acestora şi sursa a numeroase critici îndreptăţite. După cum s-a arătat într-unul dintre paragrafele anterioare, rezultate teoretice riguroase demonstrează că reţelele *feedforward* cu 3 straturi, respectiv cele cu numai 2 straturi şi cu număr foarte mare de neuroni în stratul ascuns posedă capacitate de aproximare universală. Problema aproximării simultane atât a unei funcţii multidimensionale oarecare cât şi a derivatelor parţiale ale acesteia a fost de asemenea abordată [86], datorită aplicaţiilor posibile (controlul mişcării roboţilor, analiza seriilor de timp haotice). În paragraful (5.2) a fost prezentată o explicaţie intuitivă bazată pe considerente geometrice a rolului jucat de neuronii plasaţi în straturile ascunse ale unei reţele multistrat, justificându-se în acest fel capacitatea de modelare superioară a celor având 2 astfel de straturi. Un rezultat interesant a fost prezentat în [164] în legătură cu aşa-numita problemă de modelare inversă, formulată în felul următor:

> Dându-se o funcţie continuă $f : R^m \to R^M$, o submulţime compactă $\mathbf{U} \subseteq R^M$ şi valoarea reală ε, să se găsească funcţia $\varphi : R^M \to R^m$ astfel încât să fie îndeplinită relaţia:

$$\left\| \varphi(f(\mathbf{u})) - \mathbf{u} \right\| < \varepsilon , \quad \mathbf{u} \in \mathbf{U} \tag{5. 26}$$

Această problemă apare în aplicaţii de mişcare mecanică, sub forma concretă a unei ecuaţii de tipul **x**[n] = f{**x**[n-1], **u**[n]}, în care este necesară deducerea semnalului **u**[n] în funcţie de variabila **x**[n], pentru o valoare dată a vectorului **x**[n-1]. Se poate demonstra că o reţea neurală statică având un singur strat ascuns nu garantează rezolvarea unor astfel de probleme, pe când cele cu 2 straturi ascunse asigură o soluţie indiferent de natura particulară a parametrilor implicaţi.
Un alt rezultat interesant indică faptul că un set de N perechi intrare-ieşire pot fi învăţate perfect dacă folosim o reţea cu un singur strat ascuns în care sunt plasaţi (N-1) neuroni [88]. Rezultatul este dificil de folosit în aplicaţiile practice realiste, deoarece dimensiunile arhitecturii ar deveni exagerate. În [69] se demonstrează că reţelele cu structură "piramidală" (la care numărul de neuroni descreşte dinspre stratul de intrare spre cel de ieşire) sunt lipsite de minime locale ale funcţiei de eroare. Ca o indicaţie practică, sunt de preferat reţelele care includ un număr de intrări mult mai mare decât cel al neuronilor de pe primul strat ascuns.

• *Tehnici de optimizare*

Funcţia de eroare în cazul unei rețele neurale de tip perceptron multistrat este o funcție multidimensională neliniară ai cărei parametri sunt valorile interconexiunilor şi pragurilor neuronilor elementari. Un algoritm de antrenare oarecare urmăreşte modificarea acestor parametri în sensul evoluției erorii către valoarea minimă. Reprezentarea geometrică a acesteia pune în evidență, de regulă, prezența unui minim *global* şi a unui număr de minime locale ca în Fig. 5.6. Elaborarea unui algoritm de antrenare adecvat se bazează pe tehnici de optimizare neliniară. Variantele utilizate în prezent se încadrează într-una din următoarele 2 categorii:
a) metode în care funcția de eroare descreşte sau rămâne constantă de la o iterație la alta, fără posibilitatea de a creşte temporar. Dezavantajul acestora constă în imposibilitatea de a "evada" din minimele locale. Exemple din această categorie sunt algoritmul denumit *conjugate gradients*, cele de tip quasi-Newton, precum şi varianta bloc (*batch*) a algoritmului de scădere după gradient [76].
b) metode în care eroarea evoluează în medie către valoarea minimă, permițând creşteri temporare ale valorii acesteia. Din această categorie amintim varianta secvențială (*pattern-by-pattern*) a algoritmului de scădere după gradient şi varianta *backpropagation* cu moment [76].
Multe dintre metodele performante de optimizare neliniară fac apel la aproximări locale pătratice ale funcției de eroare, în care alături de matricea Jacobian, care reuneşte derivatele parțiale de ordinul I ale acesteia, intervine şi matricea Hessian, formată din derivatele parțiale de ordinul II: algoritmul *conjugate gradients* şi varianta sa scalată, precum şi algoritmul Levenberg-Marquardt [24].

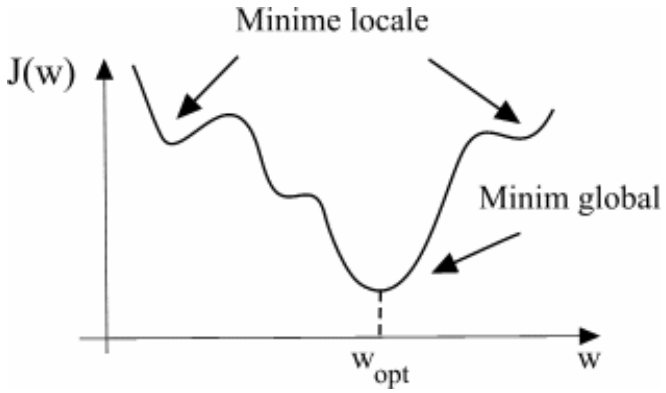


Fig.5.6: Aspect tipic al funcției de eroare

• *Dimensiunea setului de date şi modalitatea de antrenare*

Există 2 modalităţi distincte de antrenare a unei reţele de tip perceptron multistrat cu ajutorul algoritmului *backpropagation*. Prima constă în modificarea setului de ponderi după fiecare prezentare a câte unei singure perechi de date intrare-ieşire (*pattern mode*). Alternativ, ponderile pot fi ajustate după prezentarea întregului set de date avute la dispoziţie (*batch mode*), o iteraţie de acest tipfiind denumită **epocă de antrenare**. Varianta optimă depinde de aplicaţia dată, însă se recomandă ca în prima variantă perechile să fie prezentate **într-o ordine aleatoare**, pentru a evita ca, în caz contrar, reţeaua să "considere" în mod fals că datele de lucru prezintă un caracter periodic intrinsec. Din punct de vedere practic, este de dorit ca perechile intrare-ieşire folosite în procesul de antrenare să fie cât mai diferite, pentru ca reţeaua să aibă la dispoziţie un număr de "scenarii" cât mai reprezentativ pentru problema concretă considerată. În acest scop, în setul de antrenare se inserează câteodată (mai ales atunci când baza de date este redusă sau puternic redundantă) şi date provenind din suprapunerea unor nivele de zgomot peste valorile originale, capacitatea de generalizare a reţelei îmbunătăţindu-se.
Dimensiunea bazei de date D folosite în procesul de învăţare este de asemenea extrem de importantă. Un rezultat teoretic binecunoscut exprimă legătura dintre acest parametru, numărul de ponderi ale reţelei N_W şi valoarea finală a erorii pătratice medii $J(\infty)$ sub forma $D = N_W / J(\infty)$ [19]. Mai mult, împărţirea judicioasă a bazei de date iniţiale într-un set de antrenare şi altul de validare este hotărâtoare în obţinerea unor performanţe de generalizare satisfăcătoare. Există rezultate teoretice care justifică alegerea dimensiunii setului de validare la aproximativ 10% din baza de date originală [7].
În [89] este prezentată o analiză referitoare la influenţa dimensiunilor bazei de date asupra funcţiei de eroare ataşate reţelei neurale. Se constată că în cazul unor seturi reduse de perechi intrare-ieşire aspectul acestei funcţii este caracterizat de prezenţa unor "paliere" extinse, întrerupte din când în când de pante de scădere foarte abruptă. Pe măsură ce dimensiunea bazei de date creşte, aspectul suprafeţei de eroare devine mai regulat.

5.3.2 Variante ale algoritmului *backpropagation*

După cum s-a arătat anterior, reţelele statice multistrat antrenate cu varianta standard a algoritmului *backpropagation* prezintă 2 dezavantaje majore, anume viteza redusă de convergenţă şi posibilitatea cantonării în soluţii suboptimale, corespunzătoare unor minime locale ale funcţiei de eroare. Pentru a le diminua au

fost propuse numeroase soluții practice, cu un grad de eficiență şi de generalitate variabil, concentrate atât pe folosirea unor arhitecturi diferite de cea multistrat tradițională, cât şi pe algoritmi de învățare mai performanți. Din punctul de vedere al arhitecturii sistemului, menționăm varianta denumită *cascade-correlation* propusă de Fahlman şi Lebiere [57] (amintită şi în contextul tehnicilor constructive prezentate în Capitolul 4), precum şi rețelele modulare [x], subliniind totodată indicația de a utiliza arhitecturi "piramidale" (în sensul unui număr de neuroni descrescător dinspre intrarea spre ieşirea rețelei) pentru a evita apariția minimelor locale în funcția de eroare [69].
În ceea ce priveşte algoritmii de învățare, merită menționate o serie de indicații rezultate din aplicațiile practice care oferă posibilitatea de a creşte viteza de convergență [76]:

- parametrii adaptivi ai rețelei (ponderi şi valori de prag) trebuie să aibă constante de adaptare individuale
- valorile constantelor de adaptare trebuie să poată fi modificate pe parcursul procesului de învățare
- dacă gradientul funcției de eroare în raport cu un parametru al rețelei are acelaşi semn pentru câteva iterații sucesive atunci valoarea constantei de adaptare corespunzătoare acelui parametru trebuie mărită
- în mod asemănător, dacă valoarea gradientului alternează ca semn pentru câteva iterații succesive atunci constanta de adaptare corespunzătoare trebuie să fie micşorată

În literatură au fost propuse câteva variante îmbunătățite ale algoritmului *backpropagation* care țin cont de observațiile anterioare, dintre care menționăm:

A. Algoritmul backpropagation cu moment

O serie de rezultate experimentale au demonstrat că viteza de convergență a algoritmului clasic de tip scădere după gradient poate fi crescută considerabil prin includerea unui termen suplimentar în relația care defineşte regula de ajustare a parametrilor unui astfel de sistem adaptiv, sub forma [152]:

$$\Delta \mathbf{W}[n] = -\eta \frac{\partial J[n]}{\partial \mathbf{W}[n]} + \alpha \Delta \mathbf{W}[n-1] \tag{5. 27}$$

unde constanta pozitivă α este denumită constantă de moment. După cum s-a arătat în Capitolul 3, necesitatea practică de a lucra cu valori *estimate* şi nu cele exacte ale gradientului funcției de eroare face ca procesul de convergență al parametrilor

unui sistem adaptiv bazat pe algoritmul standard să fie caracterizat prin apariția așa-numitului *zgomot de gradient* (cu atât mai accentuat cu cât valoarea constantei de adaptare η este mai mare), manifestat prin oscilații de o parte şi de alta a direcției reale a acestuia. Includerea termenului de moment are rolul de a filtra într-o oarecare măsură aceste oscilații, ghidând mecanismul de adaptare în direcția cea bună.

O analiză elegantă a acestui algoritm se prezintă în [152] şi se bazează pe o analogie mecanică. Astfel, să considerăm un corp de masă m, care se mişcă într-un mediu vâscos caracterizat de constanta de frecare μ sub acțiunea unui câmp de forțe conservative[5] având energia potențială J. Ecuația care descrie mişcarea este:

$$m\frac{d^2\mathbf{x}}{dt^2}+\mu\frac{d\mathbf{x}}{dt}=-\frac{\partial J}{\partial \mathbf{x}} \tag{5. 28}$$

unde **x** desemnează coordonatele spațiale. Se observă uşor că relația precedentă devine identică în cazul particular m = 0 cu versiunea *analogică* a algoritmului standard de tip scădere după gradient:

$$\frac{d\mathbf{W}}{dt}=-\eta\frac{\partial J}{\partial \mathbf{W}} \tag{5. 29}$$

Pentru a stabili legătura dintre relațiile (5. 27) şi (5. 28) apelăm la discretizarea acesteia din urmă, sub forma:

$$m\frac{\mathbf{x}(t+\Delta t)+\mathbf{x}(t-\Delta t)-2\mathbf{x}(t)}{(\Delta t)^2}+\mu\frac{\mathbf{x}(t+\Delta t)-\mathbf{x}(t)}{\Delta t}=-\frac{\partial J}{\partial \mathbf{x}} \tag{5. 30}$$

care în urma unui calcul elementar se poate rescrie sub forma:

$$\mathbf{x}(t+\Delta t)-\mathbf{x}(t)=-\frac{(\Delta t)^2}{m+\mu\Delta t}\frac{\partial J}{\partial \mathbf{x}}+\frac{m}{m+\mu\Delta t}\left[\mathbf{x}(t)-\mathbf{x}(t-\Delta t)\right] \tag{5. 31}$$

Introducând notațiile:

[5] Un câmp de forțe conservative F este descris de relația F(**x**) = - grad (J(**x**))

$$\eta = \frac{(\Delta t)^2}{m + \mu \Delta t}; \quad \alpha = \frac{m}{m + \mu \Delta t} \tag{5. 32}$$

rezultă că relațile (5. 27) şi (5. 28) coincid, constanta de moment α jucând rolul masei m. Fără a prezenta demonstrațiile riguroase din [152], menționăm numai concluziile rezultate, care oferă indicații practice importante:

- în cazul analogiei anterioare bazate pe abordarea analogică, se poate demonstra că viteza de convergență creşte dacă este îndeplinită condiția:

$$k_i < \frac{\mu^2}{2m} \tag{5. 33}$$

unde k_i sunt valorile proprii ale matricii Hessian $\left(\frac{\partial J}{\partial x_i \partial x_j} \right)$ (matricea este pozitiv definită şi, ca urmare, $k_i > 0$). Mai mult, viteza maximă de convergență către valoarea minimă a funcției de cost corespunzătoare (în cazul mecanic, suma energilor cinetică şi potențială, iar în cazul sistemului adaptiv funcția de eroare) se obține pentru $k_i = \frac{\mu^2}{4m}$. Este important de subliniat că procesul de convergență către o valoare minimă (nu neapărat unică!) a funcției de cost se petrece indiferent de valorile (pozitive) ale parametrilor m şi μ, respectiv η şi α.

- prin discretizarea ecuațiilor diferențiale analizate anterior în jurul unui punct de echilibru se obțin ecuații cu diferențe cu o formă principial asemănătoare. Există însă o deosebire fundamentală: procesul de convergență este garantat numai dacă sunt îndeplinite condițiile $|\alpha| < 1$ şi $0 < \eta k_i < 2+2\alpha$. În plus, valoarea optimă a constantei de moment este:

$$\alpha = \left(1 - \sqrt{\eta k_i}\right)^2 \tag{5. 34}$$

În aplicațiile practice valoarea parametrului α se alege în limitele 0.7-0.9.

B. Algoritmul delta-bar-delta

O modalitate concretă de a satisface cerințele enumerate la începutul paragrafului constă în modificarea valorii constantei de adaptare asociate unei ponderi w_{ij} conform relației [76]:

$$\Delta\eta_{ij}[n+1] = \gamma \frac{\partial J[n]}{\partial \eta_{ij}[n]} \frac{\partial J[n-1]}{\partial \eta_{ij}[n-1]} \tag{5. 35}$$

unde constanta pozitivă γ controlează amplitudinea ajustării valorii curente. Este evident că în situațiile în care semnul derivatei parțiale a erorii se menține pentru 2 iterații consecutive constanta de adaptare se măreşte, iar în cazul alternanței semnelor valoarea se diminuează. Această variantă este denumită *delta-delta* şi intuitiv poate fi considerată atractivă, însă prezintă propriile sale neajunsuri: valori succesive cu acelaşi semn dar de amplitudine mică ale gradientului funcției de eroare vor conduce la ajustări nesemnificative ale constantelor de adaptare, respectiv valori succesive ale gradientului cu semne diferite şi de amplitudine mare vor micşora drastic aceleaşi constante de adaptare. O soluție superioară este oferită de algoritmul denumit *delta-bar-delta*, al cărui mod de operare este descris prin relațiile:

$$\Delta\eta_{ij}[n+1] = \begin{cases} k\,, & \text{dacă } S_{ij}[n-1]\dfrac{\partial J[n]}{\partial w_{ij}[n]} > 0 \\ -\beta\eta_{ij}[n]\,, & \text{dacă } S_{ij}[n-1]\dfrac{\partial J[n]}{\partial w_{ij}[n]} < 0 \\ 0\,, & \text{în celelalte cazuri} \end{cases} \tag{5. 36}$$

în care variabila $S_{ij}[n]$ defineşte o sumă ponderată (într-o manieră exponențială) a valorilor curentă şi anterioare ale gradientului funcției de eroare, conform relației:

$$S_{ij}[n] = (1-\alpha)\frac{\partial J[n-1]}{\partial w_{ij}[n-1]} + \alpha S_{ij}[n-1] \tag{5. 37}$$

Pe baza relațiilor anterioare putem observa că, într-o manieră asemănătoare variantei *delta-delta*, valorile constantei de adaptare cresc sau descresc după cum

semnele gradientului funcției de eroare la momentul curent şi al sumei ponderate S_{ij} de la momentul anterior coincid sau sunt diferite. Deosebirea esențială constă în faptul că procesul de creştere a valorii constantei de adaptare este liniar, cu pas fix, pe când procesul de descreştere este exponențial (şi deci rapid), dependent de valoarea curentă a acestei constante.

C. Algoritmul RPROP

Reamintim că ajustarea unui parametru oarecare w folosind principiul scăderii după gradient se face pe baza relației $\Delta w[n] = -\eta \frac{\partial J[n]}{\partial w[n]}$. Se observă uşor că, în realitate, nu este suficient să controlăm cu atenție numai valorile constantei de adaptare η căci ceea ce contează este modul în care se modifică *produsul* dintre constanta η şi valoarea gradientului. Pe baza acestei constatări a fost introdus algoritmul denumit RPROP (*resilient prapagation*) [154], care porneşte de la definirea unui parametru ataşat fiecărei ponderi a rețelei neurale care indică valoarea cu care se efectuează ajustarea acestei ponderi la un moment dat:

$$\Delta_{ij}[n] = \begin{cases} \eta^{+}\Delta_{ij}[n-1], & dacă\ \dfrac{\partial J[n]}{\partial w_{ij}[n]}\dfrac{\partial J[n-1]}{\partial w_{ij}[n-1]} > 0 \\ \eta^{-}\Delta_{ij}[n-1], & dacă\ \dfrac{\partial J[n]}{\partial w_{ij}[n]}\dfrac{\partial J[n-1]}{\partial w_{ij}[n-1]} < 0 \\ \Delta_{ij}[n-1], & în\ celelalte\ cazuri \end{cases} \tag{5. 38}$$

în care $0 < \eta^{-} < 1 < \eta^{+}$. Modificarea ponderilor rețelei neurale va fi:

$$w_{ij}[n] = \begin{cases} -\Delta_{ij}[n-1], & dacă\ \dfrac{\partial J[n]}{\partial w_{ij}[n]} > 0 \\ \Delta_{ij}[n-1], & dacă\ \dfrac{\partial J[n]}{\partial w_{ij}[n]} < 0 \\ 0, & în\ celelalte\ cazuri \end{cases} \tag{5. 39}$$

cu observația suplimentară că în cazul în care $\frac{\partial J[n]}{\partial w_{ij}[n]}\frac{\partial J[n-1]}{\partial w_{ij}[n-1]} < 0$ (adică a fost "ratată" un minim al funcției de eroare din cauza unei ajustări de amplitudine prea mare) valoarea ponderii revine la cea anterioară $\Delta w_{ij}[n] = -\Delta w_{ij}[n-1]$, după care pasul $\Delta_{ij}[n]$ se micşorează.

În practică algoritmul porneşte de la o valoare inițială Δ_0 (aleasă proporțional cu amplitudinea inițială a ponderilor rețelei) şi permite o variație într-o gamă dinamică predefinită $[\Delta_{min}, \Delta_{max}]$. De cele mai multe ori alegerea parametrilor η^+ şi η^- nu este critică, valorile recomandate în [x] fiind $\eta^- = 0.5$ şi $\eta^+ = 1.2$.

5.4 Antrenarea rețelelor MLP folosind filtrul Kalman

Viteza de convergență scăzută reprezintă, alături de posibilitatea cantonării în valori suboptimale ale parametrilor (corespunzătoare unor minime locale ale funcției de eroare), unul dintre principalele dezavantaje ale algoritmilor de învățare din categoria *backpropagation*. Una dintre posibilitățile cele mai eficiente de a ocoli acest neajuns este de a utiliza pentru antrenare filtrul Kalman, transformând astfel procesul de adaptare într-un proces de *estimare optimală* a setului de parametri ai rețelei. Relațiile care definesc acest algoritm au fost prezentate pe larg în Capitolul 3 referitor la studiul rețelelor neurale liniare, astfel încât în cele ce urmează ne vom concentra asupra aspectelor specifice utilizării acestui instrument în contextul rețelelor statice de tip MLP, menționând încă de la început că prezența neliniarităților va influența modul de operare şi performanțele algoritmului. Astfel, să considerăm un sistem dinamic liniar descris prin ecuațiile de stare:

$$\begin{aligned}\mathbf{x}[n+1] &= \mathbf{\Phi}[n+1,n]\mathbf{x}[n] + \mathbf{v}_1[n] \\ \mathbf{y}[n] &= \mathbf{C}[n]\mathbf{x}[n] + \mathbf{v}_2[n]\end{aligned} \qquad (5.\ 40)$$

în care $\mathbf{\Phi}$[n+1,n] desemnează matricea de tranziție a stărilor, v_1[n] este zgomotul de proces, modelat ca un proces aleator de tip zgomot alb cu valoare medie nulă şi matrice de autocorelație $\mathbf{Q}_1$[n], $\mathbf{C}$[n] este o matrice asociată procesului de măsură, iar $\mathbf{v}_2$[n] este zgomotul de măsură, considerat de asemenea cu valoare medie nulă şi matrice de autocorelație $\mathbf{Q}_2$[n]. Matricile $\mathbf{\Phi}$[n+1,n] şi $\mathbf{C}$[n] se consideră cunoscute, iar procesele aleatoare $\mathbf{v}_1$[n] şi $\mathbf{v}_2$[n] sunt statistic independente.

Modul de operare al filtrului Kalman conduce la obținerea unei valori estimate a vectorului de stare optimă din punctul de vedere al erorii pătratice medii şi se bazează pe utilizarea recursivă a relațiilor de mai jos:

$$\mathbf{P}[n+1]^{-} = \mathbf{\Phi}[n+1,n]\mathbf{P}[n]\mathbf{\Phi}^{H}[n+1,n] + \mathbf{Q}_1[n] \tag{5. 41}$$

$$\mathbf{G}[n] = \mathbf{\Phi}[n+1,n]\mathbf{P}[n]^{-}\mathbf{C}[n]^{H}(\mathbf{C}[n]\mathbf{P}[n]^{-}\mathbf{C}[n]^{H} + \mathbf{Q}_2[n])^{-1} \tag{5. 42}$$

$$\mathbf{P}[n] = \mathbf{P}[n-1]^{-} - \mathbf{\Phi}[n,n+1]\mathbf{G}[n]\mathbf{C}[n]\mathbf{P}[n-1]^{-} \tag{5. 43}$$

$$\hat{\mathbf{x}}[n] = \hat{\mathbf{x}}[n]^{-} + \mathbf{G}[n](y[n] - \mathbf{C}[n]\hat{\mathbf{x}}[n]^{-}) \tag{5. 44}$$

în care G[n] este denumit cîştig Kalman. În cazul în care sistemul dinamic considerat este neliniar, ecuațiile de stare care îl descriu capătă forma:

$$\begin{aligned} \mathbf{x}[n+1] &= f(\mathbf{x}[n]) + \mathbf{v}_1[n] \\ \mathbf{y}[n] &= h(\mathbf{x}[n]) + \mathbf{v}_2[n] \end{aligned} \tag{5. 45}$$

în care funcțiile neliniare f(.) şi h(.) sunt considerate diferențiabile şi se presupune că sunt cunoscute. Utilizarea ecuațiilor care definesc filtrul Kalman devine posibilă în acest context neliniar prin *liniarizarea ecuațiilor de stare în jurul punctului curent de funcționare*, înlocuind practic sistemul neliniar printr-unul liniar variabil în timp. În mod concret, în relațiile (5.22)-(5.25) se efectuează înlocuirile prezentate mai jos, obținându-se în acest mod aşa-numitul **filtru Kalman extins** (*Extended Kalman Filter* – EKF) [77]:

$$\begin{aligned} \mathbf{\Phi}[n+1,n] &\Rightarrow \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}[n]^{-}} \\ \mathbf{C}[n] &\Rightarrow \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}[n]^{-}} \end{aligned} \tag{5. 46}$$

Se pot face o serie de observații importante referitoare la algoritmul EKF:

- dacă versiunea standard a filtrului Kalman, valabilă pentru sisteme liniare, se bucură de performanțe optime din punctul de vedere al valorilor estimate ale vectorului de stare acestea nu se transmit automat şi spre versiunea sa extinsă la cazul neliniar. Algoritmul EKF este pur şi simplu un algoritm de estimare capabil să conducă de cele mai multe ori foarte rapid la soluții de bună calitate într-o serie de aplicații.

- singurii parametri disponibili pentru a controla evoluția filtrului Kalman sunt valoarea inițială a matricii de covarianță a erorii de estimare **P**[0] şi matricile de autocorelație $\mathbf{Q}_1[n]$ şi $\mathbf{Q}_2[n]$. Alegerea unor valori mari ale acestora are efectul micşorării câştigului Kalman, operațiune oarecum asemănătoare micşorării constantei de adaptare din cazul algoritmilor de tip scădere după gradient.

Utilizarea algoritmului EKF pentru ajustarea ponderilor unei rețele statice de tip perceptron multistrat se bazează pe o formulare particulară a ecuațiilor (5. 45) sub forma:

$$\begin{aligned} \mathbf{W}[n+1] &= \mathbf{W}[n] \\ \mathbf{y}[n] &= h\big(\mathbf{x}[n], \mathbf{W}[n]\big) + \mathbf{v}_2[n] \end{aligned} \tag{5. 47}$$

din care rezultă că vectorul **W** care reuneşte întregul set de ponderi ale rețelei joacă rolul vectorului de stare, matricea de tranziție a stărilor este o matrice identitate, iar matricea de autocorelație $\mathbf{Q}_1[n] = \mathbf{0}$. Utilizarea practică a algoritmului trebuie să țină cont de următoarele elemente:

- deşi în formularea originală a filtrului Kalman, respectiv a variantei sale extinse, se presupune că matricile de autocorelație ale zgomotelor $\mathbf{Q}_1[n]$ şi $\mathbf{Q}_2[n]$ sunt cunoscute, în realitate aceste mărimi trebuie estimate pe baza datelor disponibile. În literatură au fost propuse o serie de metode de estimare bazate pe impunerea unor modele statistice (de regulă, de tip autoregresiv – AR) referitoare la procesul aleator **x**[n] [90], [130].
- în multe aplicații practice se dovedeşte totuşi utilă introducerea în expresia primei ecuații din (5.28) a unui termen reprezentând zgomotul de proces, precum în formularea standard din relația (5. 45). Rolul acestui termen este dublu: pe de o parte se elimină problemele de metodă numerică asociate sistemului de ecuații care defineşte filtrul Kalman (legate în special de necesitatea asigurării caracterului pozitiv definit al matricii de covarianță **P**[n] definită prin relația (5.24)), pe de altă parte cresc şansele ca valorile vectorului de stare să "evadeze" din minimele locale ale funcției de eroare (în mod oarecum asemănător utilizării variantei "cu moment" a algoritmului *backpropagation*).

În cele ce urmează prezentăm câteva aspecte suplimentare referitoare la utilizarea algoritmului EKF în antrenarea rețelelor neurale statice:

A. Variante ale algoritmului EKF

Dezavantajul major al filtrului Kalman, transmis şi variantei sale extinse, se referă la volumul mare de memorie necesar stocării variabilelor de lucru, în special a matricii de covarianță a erorii de estimare **P**[n]. Dimensiunile acesteia sunt proporționale cu pătratul numărului de ponderi ale rețelei, care face ca în aplicații practice uzuale aplicarea algoritmului să fie extrem de costisitoare. În literatură au fost propuse o serie de variante simplificatoare, care reduc în mod substanțial volumul de calcul şi de memorie necesar, dintre care enumerăm metodele DEKF (*Decoupled EKF*) [151], MEKA (*Multiple EKF*) [161] şi *Dual EKF* [130].

- *DEKF:* Această variantă se bazează pe partiționarea vectorului care reuneşte ponderile rețelei în subgrupuri care sunt considerate independente ("decuplate"), astfel încît matricea globală de covarianță **P**[n] se poate aranja în formă bloc-diagonală. În acest fel, relațiile recursive care definesc filtrul Kalman la nivel global se pot "sparge" într-un set de ecuații de acelaşi tip, valabile însă pentru vectori de stare care reunesc subseturi de ponderi ale rețelei neurale. Eficiența metodei din punct de vedere al resurselor de calcul necesare va depinde de modalitatea particulară de definire a subgrupelor de parametri (de exemplu, ponderile rețelei se pot grupa pe straturi, pe neuroni individuali sau, la limită, fiecare pondere în parte poate fi considerată ca reprezentând un grup de sine stătător).
- *MEKA:* Varianta se bazează pe considerarea fiecărui neuron din rețea ca un subsistem independent, pentru care se urmăreşte minimizarea unei funcții de eroare definite local (în particular, metoda prezintă avantajul că în ecuația de tipul (5. 42) care defineşte câştigul Kalman nu mai apare o inversare de matrici, ci o împărțire printr-un scalar).
- *Dual EKF:* Cea de a treia variantă, utilizată cu precădere în aplicații de predicție, în particular pentru analiza semnalelor vocale, utilizează o *pereche* de filtre Kalman, unul bazat pe modelarea prin ecuații de stare a procesului aleator supus predicției, iar celălalt responsabil cu modelarea vectorului de ponderi ale rețelei, după metoda descrisă mai sus.

Toate aceste soluții conduc la reduceri substanțiale ale resurselor de calcul necesare şi s-au dovedit exrem de utile în numeroase aplicații practice importante referitoare la prelucrarea semnalelor vocale, control neliniar sau predicție. O serie de aspecte de ordin practic merită subliniate:

- toate aceste variante ale algoritmului standard se bazează pe ipoteza simplificatoare că (grupe de) parametri ai rețelei sunt independente din punct

de vedere statistic. Faptul este valabil pentru rețele având un singur strat, însă este dificil de generalizat în cazul arhitecturilor multistrat.

- datorită mecanismului propriu de operare al filtrului Kalman, este posibil ca rețelele neurale antrenate prin acest procedeu să asigure şi posibilitatea de urmărire a evoluțiilor lente ale dependenței funcționale intrare-ieşire modelate.
- un aspect extrem de important şi care nu este întotdeauna tratat cu suficientă atenție se referă la modalitatea de estimare a valorilor dispersiei zgomotului din relația (5. 47), dependentă de regulă de aplicația practică studiată. În literatură sunt descrise o serie de metode concrete referitoare în special la aplicații de prelucrare a semnalelor vocale, bazate pe luarea în considerație a unor modele statistice consacrate sau pe tehnici de reestimare periodică pe durata procesului de antrenare a rețelei [130].

B. Îmbunătățirea capacității de generalizare

Într-o serie de articole recente a fost abordată şi problema îmbunătățirii capacității de generalizare a rețelelor neurale folosind filtrul Kalman [165], [166]. După cum s-a arătat în Capitolul 3, metodele avute la dispoziție în acest scop se încadrează într-una dintre următoarele 2 variante: a) tehnici constructive, de tip *learn-and-grow* sau *pruning*; b) introducerea unui termen de regularizare în formularea funcției de cost supuse minimizării. Articolele menționate se referă la tehnica de *pruning* şi se bazează pe posibilitatea de a identifica seturi de ponderi nesemnificative (a căror eliminare nu va conduce la o mărire sensibilă a erorii) folosind informațiile puse la dispoziție de metodologia specifică de lucru a filtrului Kalman. În mod concret, notând cu $\mathbf{P}_\infty$ valoarea finală a matricii de covarianță a erorii de estimare (adică $\mathbf{P}_\infty = \lim_{n\to\infty} \mathbf{P}[n]$), considerând matricea de autocorelație a zgomotului de proces $Q_1[n] = q\mathbf{I}$ (**I** este o matrice unitate de dimensiune adecvată şi q o valoare scalară pozitivă) şi notând prin $\hat{\mathbf{W}}_k$ vectorul obținut prin punerea pe zero a ponderii de pe poziția k din vectorul de ponderi rezultat în urma procesului de antrenare folosind filtrul Kalman (restul valorilor rămânând nealterate) se poate arăta că modificarea erorii de aproximare ca efect al eliminării ponderii k este[6]:

$$\Delta J_k \approx q\left(\mathbf{P}_\infty^{-2}\right)\hat{\mathbf{W}}_k \tag{5. 48}$$

[6] Relația este valabilă în condițiile în care valorile proprii ale matricii $\mathbf{P}_\infty >> q$

Relația de mai sus oferă un criteriu de ordonare a ponderilor rețelei în raport cu importanța lor din punctul de vedere al asigurării capacității de aproximare a procesului aleator măsurat, astfel încât se vor putea elimina în mod justificat (grupe) de parametri nesemnificativi. Condițiile în care a fost dedusă relația anterioară se referă la existența unui model de bună calitate şi a unei baze de date suficient de mari.