

# Rezolvarea problemelor de asociere cu rețele neuronale

octombrie 2004

Problemele care presupun descoperirea unei asocieri între datele de intrare și un răspuns pot fi văzute din punct de vedere formal ca și probleme de aproximare a funcțiilor. De regulă se cunosc câteva exemple de asociere care reprezintă valori ale funcției de aproximat pentru un set de argumente.

## 1 Problematika aproximării funcțiilor

Problema aproximării unei funcții  $\phi : R^N \rightarrow R^M$  poate fi descrisă în mod simplificat astfel:

Presupunem că se cunosc valorile funcției  $\phi$  doar pentru o mulțime de valori ale argumentelor. De exemplu se știe:  $\{(X^1, \phi(X^1)), \dots, (X^L, \phi(X^L))\}$  și se dorește estimarea valorilor lui  $\phi$  pentru orice argument  $X$  din domeniul său de definiție.

Despre o funcție  $\psi : D \subset R^N \rightarrow R^M$  se poate spune că aproximează pe  $\phi$  cu precizia  $\epsilon$  pe  $D$  dacă are loc, de exemplu,  $\|\phi(X) - \psi(X)\| < \epsilon$  pentru orice  $X \in D$ ,  $\|\cdot\|$  fiind o normă în  $R^M$ .

Problematika aproximării funcțiilor este una generală cuprinzând multe dintre problemele concrete ce pot fi rezolvate cu rețele neuronale. Astfel de exemple sunt:

1. Determinarea unei relații funcționale între două mărimi pornind de la date obținute prin măsurători (fitarea datelor experimentale).
2. Efectuarea unei predicții în cadrul unei serii temporale cunoscând evoluția anterioară a acesteia. În acest caz se încearcă aproximarea unei funcții  $\phi : D \subset R^k \rightarrow R$  care să surprindă legătura dintre  $k$  valori anterioare din serie și valoarea curentă.
3. Codificarea/ compresia datelor. Se caută o aplicație  $\phi_1 : R^N \rightarrow R^M$ ,  $M < N$  (folosită pentru etapa de codificare) și o aplicație  $\phi_2 : R^M \rightarrow R^N$  (folosită pentru etapa de decodificare) astfel încât  $\phi_2 \circ \phi_1 : R^N \rightarrow R^N$  să fie cât mai apropiată de aplicația identică,  $\phi_2(\phi_1(X)) = X$ .
4. Clasificarea unor vectori din  $R^N$  în  $M$  clase poate fi văzută ca o asociere între elementul  $X \in R^N$  și un vector indicator al clasei. De exemplu, clasa  $i$  poate fi reprezentată de vectorul  $(0, \dots, 1, \dots, 0)$  singura valoare 1 aflându-se pe poziția  $i$ . Vectorul de ieșire poate conține și valori reale. Dacă vectorul este normalizat, în sensul ca toate valorile sunt din  $[0, 1]$  și suma lor este 1 atunci fiecare componentă poate fi interpretată ca probabilitatea ca vectorul de intrare să facă parte din clasa asociată componentei respective.

Dificultatea problemei rezidă în faptul că în aplicațiile practice exemplele pot fi afectate de zgomot sau prin formalizare nu se ajunge la o funcție care să aibă proprietăți matematice suficient de bune pentru a aplica metodele clasice de aproximare a funcțiilor.

## 2 Rețele neuronale feedforward

În proiectarea unei rețele neuronale,  $\mathcal{R}$ , destinate rezolvării unei probleme de asociere (de exemplu, aproximarea funcției  $\phi$ ) trebuie rezolvate următoarele probleme:

- *Problema reprezentării.* Poate rețeaua  $\mathcal{R}$  (caracterizată de o anumită arhitectură, un anumit mod de funcționare și de un anumit set de parametri) să aproximeze funcția  $\phi$ ? Această problemă este importantă întrucât nu orice rețea poate să aproximeze orice funcție (de exemplu funcția XOR nu poate fi aproximată printr-o rețea cu un singur nivel, funcții liniare de integrare și funcții de transfer de tip Heaviside).
- *Problema învățării.* Presupunem că  $\mathcal{R}$  este o rețea a cărei arhitectură îi permite, teoretic, să aproximeze funcția  $\phi$ . Se pune problema *cum* să se determine valorile parametrilor rețelei astfel încât aceasta să aproximeze efectiv pe  $\phi$ .
- *Problema generalizării.* Fie  $\mathcal{R}$  o rețea antrenată să aproximeze pe  $\phi$ . Se pune problema *cât de bine* aproximează  $\mathcal{R}$  pe  $\phi$  pentru date de intrare care nu fac parte din setul de antrenare.

### 2.1 Suport teoretic

Problema capacității de reprezentare a rețelelor feedforward a fost rezolvată demonstrându-se că rețelele feedforward sunt *aproximatori universal*, adică pentru orice funcție continuă (sau doar măsurabilă),  $\phi$ , există o rețea feedforward cu cel puțin un nivel ascuns, care aproximează pe  $\phi$ .

Sunt cel puțin două direcții pe care s-a avansat în această problematică:

1. Specificarea funcțiilor de integrare și transfer, fără a furniza, însă, suficiente informații despre numărul de unități.
2. Specificarea structurii fiecărui nivel, fără a indica tipul funcțiilor de integrare și transfer.

**Specificarea funcțiilor de integrare și transfer.** Principalele rezultate pe această direcție sunt [Hornik, 1991], [Cybenko, 1989], [Funahashi, 1989]. Ideea centrală a acestor rezultate este:

**Propoziție.** Orice funcție măsurabilă  $\phi : D \subset \mathbf{R}^N \rightarrow \mathbf{R}^M$  (cu  $D$  o mulțime compactă) poate fi aproximată cu orice acuratețe de către o rețea feedforward caracterizată prin:

- (i)  $N$  unități de intrare,  $M$  unități de ieșire și un nivel de unități ascunse (acuratețea aproximării depinde de numărul unităților ascunse).
- (ii) funcții de integrare liniare ( $g_{W_i}(X) = (W_i)^T X$ ) și funcții de transfer,  $f : \mathbf{R} \rightarrow [a, b]$ , continue (sau măsurabile) și crescătoare cu  $\lim_{u \rightarrow \infty} f(u) = b$ ,  $\lim_{u \rightarrow -\infty} f(u) = a$ .

**Specificarea structurii fiecărui nivel.** La baza rezultatelor de acest tip stă o teoremă a lui A.K. Kolmogorov [1957] de reprezentare a funcțiilor continue de mai multe variabile prin funcții continue de o variabilă:

**Teoremă.** Există  $\lambda_1, \dots, \lambda_N \in \mathbf{R}$  și  $\psi_1, \dots, \psi_{2N+1} \in \mathcal{C}(I)$  (funcții continue pe  $I$ ) cu proprietatea că pentru orice funcție continuă  $\phi : I^N \rightarrow \mathbf{R}$ , există o funcție continuă  $h : \mathbf{R} \rightarrow \mathbf{R}$  astfel încât:

$$\phi(x_1, \dots, x_N) = \sum_{k=1}^{2N+1} h(\lambda_1 \psi_k(x_1) + \dots + \lambda_N \psi_k(x_N)).$$

Pe baza acestei teoreme, Hecht-Nielsen [1989] a enunțat:

**Propoziție.** Orice funcție continuă  $\phi : [0, 1]^N \rightarrow \mathbf{R}^M$  poate fi aproximată printr-o rețea feedforward caracterizată prin:

(i)  $N$  unități de intrare,  $(2N + 1)$  unități ascunse,  $M$  unități de ieșire.

(ii) Unitățile ascunse au funcția de integrare de forma:

$$g^k(X) = \sum_{j=1}^N \lambda_k \varphi(x_j + k\epsilon) + k,$$

iar funcția de transfer  $f^k(u) = u$ ,  $k \in \{1, \dots, 2N + 1\}$ . Funcția  $\varphi$  este continuă monoton crescătoare, nu depinde de  $\phi$ , dar depinde de  $N$ , iar  $\epsilon$  este un număr rațional.

(iii) Unitățile de ieșire au funcția de integrare:

$$g^i(X) = \sum_{k=1}^{2N+1} h^i(x_k),$$

iar funcțiile de transfer  $f^i(u) = u$ ,  $i \in \{1, \dots, M\}$ .  $h^i$  sunt funcții continue care depind de  $\phi$  și  $\epsilon$ .

Ambele abordări prezintă dezavantaje, principalul fiind acela că sunt rezultate existențiale ce nu indică o metodă efectivă de construire a rețelei. Pe de altă parte pentru o problemă concretă o rețea cu două sau mai multe nivele ascunse se poate comporta mai bine decât una cu un singur nivel ascuns, deși rezultatele teoretice asigură faptul că rețelele cu un nivel ascuns sunt aproximatori universali.

O altă categorie de rezultate teoretice sunt cele referitoare la problema regularizării. Acestea reprezintă bazele dezvoltării rețelelor cu funcții de transfer de tip radial ("radial basis functions"). Scopul regularizării este de a transforma problema aproximării funcțiilor bazată un set finit de date dintr-o problemă rău pusă într-una bine-pusă prin impunerea unor proprietăți pe care ar trebui să le satisfacă funcția (de obicei condiții de netezime - în sensul că unor argumente apropiate le corespund valori apropiate ale funcției). Din punct de vedere intuitiv se poate remarca că proprietățile de netezime au efect benefic asupra capacității de generalizare. Poggio și Girosi au arătat că principiile regularizării conduc la scheme de aproximare echivalente cu rețele cu un singur nivel ascuns, numite *rețele de regularizare*. Un caz particular al acestora sunt rețelele de tipul celor cu funcții radiale de transfer.

Problema învățării poate fi văzută ca o problemă de optimizare ce constă în minimizarea unei funcții de eroare. În cazul rețelelor proiectate prin tehnica regularizării se poate determina în procesul de învățare atât parametrii cât și forma funcțiilor de transfer. În acest caz funcționala de eroare conține doi termeni: unul prin care se urmărește obținerea unei fidelități cât mai bune și unul prin care se urmărește satisfacerea cerințelor regularizării.

## 2.2 Alegerea arhitecturii

Dacă specificul problemei nu impune utilizarea unei anumite arhitecturi atunci se poate opta pentru o rețea constând din: un nivel de intrare, un nivel ascuns și unul de ieșire. În ceea ce privește modul de conectare a unităților, varianta standard este de a conecta total nivelele vecine. În aceste condiții singura problemă ce mai trebuie rezolvată este aceea a stabilirii numărului de unități de pe fiecare nivel. La stabilirea dimensiunii (complexității) rețelei se ține cont de următoarele lucruri:

- Nivelele de intrare, respectiv de ieșire trebuie să aibă atâtea unități câte sunt necesare pentru a reprezenta datele de intrare, respectiv răspunsul rețelei. De exemplu, pentru o rețea proiectată să reprezinte o funcție  $\phi : R^N \rightarrow R^M$  se utilizează  $N$  unități pe nivelul de intrare, respectiv  $M$  pe nivelul de ieșire.

- Numărul de unități ascunse se stabilesc astfel încât rețeaua să fie suficient de complexă pentru a rezolva problema dar nu mai mult decât este necesar. Stabilirea numărului de unități ascunse se bazează fie pe rezultatele teoretice referitoare la capacitatea de reprezentare (rezolvare) a unei anumite arhitecturi, fie pe reguli euristice (de exemplu pentru o rețea cu  $N$  unități de intrare,  $M$  unități de ieșire și un nivel ascuns se poate alege pentru acesta dimensiunea:  $\sqrt{MN}$ ) fie pe tehnici de adaptare a acestora la problema de rezolvat.

Alegerea arhitecturii nu este o problemă simplă întrucât pentru aceeași problemă pot exista mai multe arhitecturi care permit rezolvarea ei. O arhitectură adecvată poate simplifica procesul de învățare și poate asigura o buna capacitate de generalizare.

*Exemplu.* Pentru reprezentarea funcției XOR există cel puțin două arhitecturi diferite (figura 1).

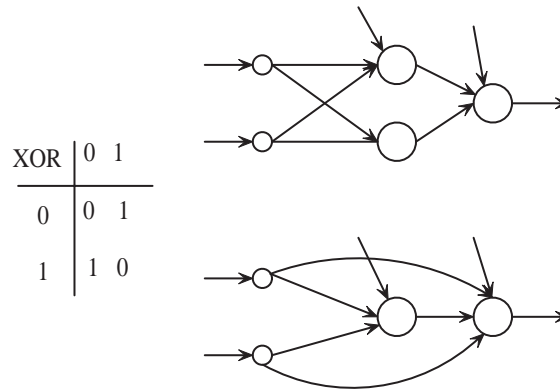


Figura 1: Arhitecturi pentru reprezentarea funcției XOR

### 2.3 Funcționare

Modul în care rețeaua funcționează depinde funcțiile de integrare și de transfer ale unităților funcționale precum și de modul de conectare. Pentru o rețea cu  $K$  nivele funcționale (dintre care  $K - 1$  sunt ascunse), completate cu unitate fictivă (cu excepția ultimului) pentru modelarea pragului de activare (vezi fig. 2) semnalul,  $Y^K$ , de ieșire pentru un semnal de intrare,  $X^0$ , este:

$$Y^K = F^K(W^K F^{K-1}(W^{K-1} F^{K-2}(\dots F^1(W^1 X^0) \dots)))$$

unde  $F^k(U) = (-1, f_1^k(u), \dots, f_{N_k}^k(u))^T$  pentru  $k = \overline{1, K-1}$  iar  $F^K(U) = (f_1^K(u), \dots, f_{N_K}^K(u))^T$ . Operațiile din relația de mai sus se efectuează la nivel vectorial, pentru un nivel  $k$ ,  $W^k$  reprezentând matricea ( $N_k$  linii și  $N_{k-1}$  coloane) ponderilor conexiunilor către acel nivel.

Din punct de vedere matematic funcționarea rețelei constă în compunerea unor transformări iar din punct de vedere algoritmic poate fi implementată printr-un algoritm iterativ simplu:

#### Algoritmul de funcționare (FORWARD).

- Pas 1. Se stabilește semnalul de intrare  $X^0$  ( $x_0^0 = -1$  iar  $x_j^0 = x_j$  pentru  $j = \overline{1, N_0}$ ) și se determină:
- $$x_i^1 = \sum_{j=0}^{N_0} w_{ij}^1 \cdot x_j^0, y_i^1 = f^1(x_i^1) \text{ pentru } i = \overline{1, N_1}.$$

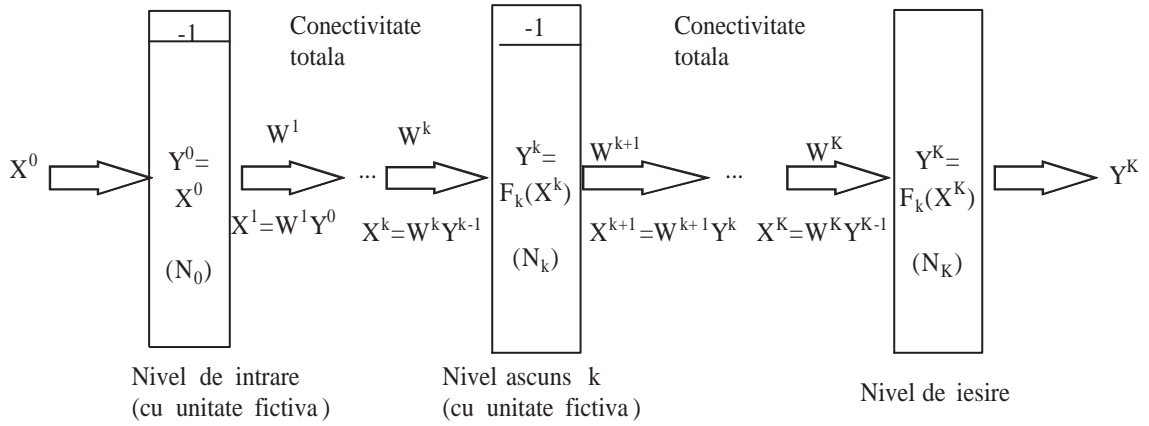


Figura 2: Structura unei rețele multinivel

Pas 2. Pentru fiecare  $k$  de la 1 la  $K - 1$  se efectuează:  $x_i^{k+1} = \sum_{j=0}^{N_k} w_{ij}^{k+1} y_j^k$ ,  $y_i^{k+1} = f^{k+1}(x_i^{k+1})$  pentru  $i = \overline{1, N_1}$  (pentru  $k < K$  componenta 0 a vectorului  $Y^k$  este -1).

Rezultatul obținut de rețea este interpretat în funcție natura problemei. Astfel în cazul unei probleme de clasificare rețeaua se completează cu un nivel care determină maximul valorilor obținute obținându-se, astfel, indicatorul clasei căreia îi aparține vectorul de intrare.

## 2.4 Procesul de învățare

Învățarea presupune adaptarea rețelei la problema de rezolvat prin utilizarea informațiilor de care se dispune despre problemă. Se poate desfășura la unul dintre nivelele:

- *Ponderile conexiunilor.* Este varianta cea mai frecvent folosită. Se presupune că arhitectura este fixată și rămân de stabilit doar valorile ponderilor.
- *Conexiuni.* Prin învățare se ajustează nu numai ponderile conexiunilor ci și conexiunile în sine care pot fi eliminate sau introduse. Partea fixă a rețelei este reprezentată de topologie (numărul unităților și modul de amplasare) iar restul (conexiuni și ponderi) sunt adaptabile.
- *Arhitectura.* Întreaga arhitectură este ajustabilă, prin procesul de învățare adăugându-se sau eliminându-se unități și/sau conexiuni și în același timp determinându-se ponderile asociate. Este cea mai flexibilă variantă dar conduce la un algoritm complex de învățare.

În cazul problemelor de asociere se dispune de un set de antrenare de forma:  $\{(X^1, d^1), \dots, (X^L, d^L)\}$  unde  $X^l \in R^N$  reprezintă data de intrare iar  $d^l$  reprezintă răspunsul corect asociat. În aceste condiții comportarea rețelei poate fi apreciată prin intermediul erorii asociate setului de antrenare. Cel mai frecvent se optează pentru utilizarea erorii medii pătratice:

$$E(W^1, \dots, W^K) = \frac{1}{L} \sum_{l=1}^L E_l(W^1, \dots, W^K) \text{ unde } E_l(W^1, \dots, W^K) = \frac{1}{2} \sum_{i=1}^{N_K} (d_i^l - y_i^{K,l})^2 \quad (1)$$

cu  $Y^{K,l}$  vectorul de ieșire corespunzător intrării  $X^l$ .

Scopul procesului de învățare este determinarea rețelei care minimizează funcția de eroare (1). În cazul în care adaptarea se referă doar la determinarea ponderilor problema învățării constă în a determina parametrii care minimizează pe  $E$ .

**Determinarea ponderilor prin minimizarea funcției de eroare.** În cazul în care funcțiile de transfer asociate unităților au proprietăți matematice suficient de bune (de exemplu sunt continuu diferentiabile) pentru rezolvarea problemei de minimizare poate fi folosită o metodă de tip gradient.

Pentru a ilustra ideea metodelor de tip gradient considerăm problema determinării lui  $X^* \in D \subset R^N$  cu proprietatea că  $F(X^*) \leq F(X)$  pentru orice  $X \in D$ ,  $F$  fiind o funcție definită pe  $D$  și cu valori reale. Presupunem că pentru  $F$  se poate calcula gradientul:  $\nabla F(x) = (\frac{\partial F(x)}{\partial x_1}, \dots, \frac{\partial F(x)}{\partial x_N})^T$ . Metoda gradientului simplu este un proces iterativ de aproximare a lui  $X^*$  care pornește de la o aproximație inițială,  $X(0)$ , și care se bazează pe următoarea relație de recurență:  $X(k+1) = X(k) - \eta(k)\nabla F(X(k))$ . Procesul iterativ continuă până când este satisfăcut un criteriu de convergență de tipul  $\|X(k+1) - X(k)\| < \epsilon$  (cu  $\|\cdot\|$  norma euclidiană și  $\epsilon > 0$  o constantă mică care reflectă precizia aproximării).

Acest proces iterativ are proprietatea că satisface  $F(X(k+1)) \leq F(X(k))$  fiind astfel o metodă de descreștere ce utilizează ca direcție de descreștere direcția opusă gradientului, iar ca pas de descreștere o valoare reală  $\eta(k)$  constantă sau ajustabilă.

Structura generală a metodei gradientului este:

*Pas 1. Inițializare.* Se aleg:

aproximația inițială  $X(0) \in D$ ,

pasul de descreștere ( $\eta(0)$ ),

numărul maxim de iterații ( $k_{max}$ )

și se inițializează contorul de iterații ( $k = 0$ ).

*Pas 2. Ajustare.*

REPETĂ

$X(k+1) = X(k) - \eta(k)\nabla F(X(k))$

$k = k + 1$

determină  $\eta(k)$

PÂNĂ CÂND  $\|X(k+1) - X(k)\| < \epsilon$  sau  $k > k_{max}$

Pasul de descreștere poate fi ales constant ( $\eta(k) = \eta$ ) sau poate fi determinat la fiecare iterație. O modalitate de alegere la fiecare iterație este cea prin care  $\eta(k)$  este ales astfel încât să conducă la o descreștere maximă: se alege  $\eta^*(k)$  cu proprietatea că  $F(X(k) - \eta^*(k)\nabla F(X(k))) \leq F(X(k) - \eta(k)\nabla F(X(k)))$  pentru orice  $\eta(k)$ . Această variantă presupune rezolvarea unei probleme de minimizare în raport cu  $\eta(k)$ .

Metoda gradientului simplu prezentată mai sus este o metodă de *minimizare locală* în sensul că determină punctul de minim local aflat în vecinătatea aproximației inițiale.

Majoritatea algoritmilor de învățare supervizată bazați pe minimizarea unei funcții de eroare folosesc o metodă de tip gradient astfel că structura lor generală cuprinde două etape principale: inițializarea parametrilor și procesul iterativ de ajustare. În funcție de metoda de minimizare aleasă se poate construi câte un algoritm de învățare. În continuare vom descrie un algoritm pentru determinarea ponderilor unei rețele feedforward folosind metoda gradientului simplu. Este vorba despre algoritmul BACKPROPAGATION, cel mai răspândit algoritm de antrenare supervizată.

Pentru a deduce relațiile de ajustare vom considera o rețea cu un singur nivel ascuns (fig. 3) și vom indica elementele vectorilor asociați nivelului de intrare cu  $j$ , cele ale vectorilor asociați

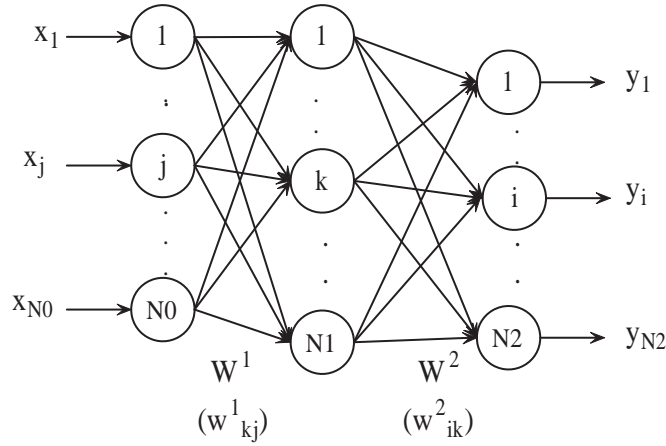


Figura 3: Structura unei rețele multinivel

nivelului ascuns cu  $k$  iar pentru nivelul de ieșire cu  $i$ . În felul acesta vom omite indicele nivelului la vectorii  $X$  și  $Y$ . Pentru a deduce relațiile de ajustare calculăm componentele gradientului funcției  $E_l(W^1, W^2)$  prima dată în raport cu componentele  $w_{ik}^2$  ale matricii  $W^2$  după care în raport cu componentele  $w_{kj}^1$  ale matricii  $W^1$ . Reamintim că  $y_i = f(x_i) = f(\sum_{k=0}^{N_1} w_{ik}^2 y_k)$ ,  $i = \overline{1, N_2}$  iar  $y_k = f(x_k) = f(\sum_{j=0}^{N_0} w_{kj}^1 y_j)$  pentru  $k = \overline{1, N_1}$  și  $-1$  pentru  $k = 0$ . Componentele  $y_j$  sunt determinate de vectorul de intrare. Dacă acesta este  $X^l$ , atunci  $y_0 = -1$  și  $y_j = x_j^l$  pentru  $j = \overline{1, N_0}$ .

Cu aceste notații, derivatele parțiale ale lui  $E_l(w_{kj}^1, w_{ik}^2)$  pot fi scrise:

$$\frac{\partial E_l(w_{kj}^1, w_{ik}^2)}{\partial w_{ik}^2} = -(d_i^l - y_i) \frac{\partial y_i(w_{ik}^2)}{\partial w_{ik}^2} = -f'(x_i)(d_i^l - y_i)y_k = -\delta_i^l y_k \quad (2)$$

cu  $\delta_i^l \stackrel{not}{=} f'(x_i)(d_i^l - y_i)$ .

$$\begin{aligned} \frac{\partial E_l(w_{kj}^1, w_{ik}^2)}{\partial w_{kj}^1} &= -\sum_{i=1}^{N_2} \frac{\partial y_i(w_{kj}^1)}{\partial w_{kj}^1} (d_i^l - y_i) = -\sum_{i=1}^{N_2} f'(x_i) w_{ik}^1 f'(x_k) x_j (d_i^l - y_i) = \\ &= -x_j f'(x_k) \sum_{i=1}^{N_2} w_{ik}^1 f'(x_i) (d_i^l - y_i) = -x_j f'(x_k) \sum_{i=1}^{N_2} w_{ik}^1 \delta_i^l = -x_j \delta_k^l, \end{aligned} \quad (3)$$

cu  $\delta_k^l = f'(x_k) \sum_{i=1}^{N_2} w_{ik}^1 \delta_i^l$ .

Modul de calcul al lui  $\delta_k^l$  sugerează transmiterea prin rețea a semnalului de eroare corespunzător nivelului de ieșire ( $\delta_i^l$ ) în sens invers celui în care circulă semnalele în faza de funcționare. Aceasta analogie stă la baza denumirii algoritmului: ”(error) backpropagation”=”propagare înapoi (a erorii)”. Pe baza relațiilor (2) și (3) se deduc relațiile de ajustare corespunzătoare exemplului  $l$  din setul de antrenare  $(X^l, d^l)$ . Cum eroarea globală,  $E$ , se obține însumând erorile parțiale,  $E_l$ , ajustările corespunzătoare întregului set de antrenare s-ar obține prin însumarea celor corespunzătoare fiecărui exemplu în parte.

Din punctul de vedere al modului de aplicare a ajustărilor în raport cu parcurgerea setului de antrenare există două variante ale algoritmului Backpropagation:

---

*Pas 1. Inițializări.* Se inițializează:  
matricile  $W^1, \dots, W^K$  cu valori aleatoare mici (uniform repartizate în  $(-1,1)$ )  
rata de învățare  $\eta(0)$   
toleranța la învățare  $E^*$   
numărul maxim de epoci de antrenare  $p_{max}$   
contorul de epoci  $p = 0$

*Pas 2. Proces iterativ de ajustare.*  
**REPETĂ**  
**PENTRU**  $l = \overline{1, L}$   
*Etapa FORWARD:* pentru  $X^0 = X^l$  se aplică algoritmul FORWARD  
*Etapa BACKWARD:*  
 $\delta_i^{K,l} = f'_K(x_i^{K,l})(d_i^l - y_i^{K,l}), i = \overline{1, N_K}$   
**PENTRU**  $k = \overline{K-1, 1}$  (propagarea erorii înapoi în rețea)  
 $\delta_j^{k,l} = f'_k(x_j^{k,l}) \sum_{i=1}^{N_{k+1}} w_{ij}^{k+1} \delta_i^{k+1,l}, j = \overline{1, N_k}$   
*Etapa ajustării propriu-zise:*  
 $w_{ij}^k = w_{ij}^k + \eta(p) \delta_i^{k,l} y_j^{k-1}, i = \overline{1, N_k}, j = \overline{0, N_{k-1}}, k = \overline{1, K}$   
*Recalcularea erorii:*  
 $E = 0$   
**PENTRU**  $l = \overline{1, L}$   
Calculează  $Y^{K,l}$  aplicând din nou algoritmul FORWARD  
 $E = E + \|d^l - Y^{K,l}\|^2$   
 $E = E/L$   
 $p=p+1$   
**PÂNĂ CÂND** ( $E \leq E^*$ ) sau ( $p > p_{max}$ ).

---

Figura 4: Algoritmul backpropagation în varianta serială

- *Serială.* (fig. 4) La fiecare iterație se parcurge întreg setul de antrenare și ajustările se operează după întâlnirea fiecărui exemplu. Nu corespunde unei implementări fidele a metodei gradientului însă prezintă avantajul că nu trebuie cumulate toate ajustările înainte de a fi aplicate.
- *Pe blocuri.* (fig. 5) La fiecare iterație se parcurge întreg setul de antrenare, se cumulează toate ajustările după care se aplică efectiv. Provine din aplicarea directă a algoritmului de minimizare asupra funcției  $E$ . Prezintă avantajul că este mai robust în raport cu eventualele erori din datele de intrare.

### 3 Rețele cu funcții radiale

#### 3.1 Arhitectură și funcționare.

Rețelele cu funcții de transfer radiale, numite și rețele de tip RBF ("Radial Basis Functions") sunt constituite din  $N$  unități de intrare,  $K$  unități ascunse și  $M$  unități de ieșire. Diferența

---

*Pas 1. Inițializări.* Se inițializează:

matricile  $W^1, \dots, W^K$  cu valori aleatoare mici (uniform repartizate în (-1,1))

rata de învățare  $\eta(0)$

toleranța la învățare  $E^*$

numărul maxim de epoci de antrenare  $p_{max}$

contorul de epoci  $p = 0$

*Pas 2. Proces iterativ de ajustare.*

REPETĂ

$\Delta_{ij}^k = 0$  pentru  $k = \overline{1, K}, i = \overline{1, N_k}, j = \overline{0, N_{k-1}}$

PENTRU  $l = \overline{1, L}$

*Etapa FORWARD:* pentru  $X^0 = X^l$  se aplică algoritmul FORWARD

*Etapa BACKWARD:*

$\delta_i^{K,l} = f'_K(x_i^{K,l})(d_i^l - y_i^{K,l}), i = \overline{1, N_K}$

PENTRU  $k = \overline{K-1, 1}$  (propagarea erorii înapoi în rețea)

$\delta_j^{k,l} = f'_k(x_j^{k,l}) \sum_{i=1}^{N_{k+1}} w_{ij}^{k+1} \delta_i^{k+1,l}, j = \overline{1, N_k}$

*Cumularea ajustărilor*

$\Delta_{ij}^k = \Delta_{ij}^k + \delta_i^{k,l} y_j^{k-1}, i = \overline{1, N_k}, j = \overline{0, N_{k-1}}, k = \overline{1, K}$

*Etapa ajustării propriu-zise:*

$w_{ij}^k = w_{ij}^k + \eta(p) \Delta_{i,j}^k, i = \overline{1, N_k}, j = \overline{0, N_{k-1}}, k = \overline{1, K}$

*Recalcularea erorii:*

$E = 0$

PENTRU  $l = \overline{1, L}$

Calculează  $Y^{K,l}$  aplicând din nou algoritmul FORWARD

$E = E + \|d^l - Y^{K,l}\|^2$

$E = E/L$

$p=p+1$

PÂNĂ CÂND ( $E \leq E^*$ ) sau ( $p > p_{max}$ ).

---

Figura 5: Algoritmul backpropagation în varianta "pe blocuri"

dintre rețelele feedforward multinivel și cele RBF constă în funcțiile de integrare și cele de transfer specifice nivelului ascuns. Semnalul produs de unitatea  $k$  aflată pe nivelul ascuns, corespunzător unui semnal de intrare  $X$ , este:

$$y_k = g_k(\|X - C^k\|)$$

unde  $C^k = (c_{k1}, \dots, c_{kN})$  este vectorul ponderilor conexiunilor către unitatea ascunsă  $k$  numit și *centrul* sau *prototipul* acesteia. Deci funcția de integrare asociată unităților ascunse se bazează pe calculul unei distanțe dintre vectorul de intrare și centrul corespunzător, distanța cea mai frecvent folosită fiind cea euclidiană ( $\|X - C^k\|^2 = \sum_{j=1}^N (x_j - c_{kj})^2$ ). Funcțiile de transfer,  $g_k$ , sunt diferite de cele sigmoide, fiind similare funcțiilor *nucleu* folosite în teoria aproximării. Cele mai frecvent folosite sunt cele caracterizate prin simetrie radială care au proprietatea de localitate (produc valori semnificative doar pentru valori mici ale argumentului și tind către zero în cazul valorilor mari). Exemple de astfel de funcții sunt cea gaussiană ( $g_k(u) = \exp(-\frac{u^2}{2\sigma_k^2})$ ), cea de tip Cauchy ( $g_k(u) = 1/(u^2 + \sigma^2)$ ) și  $g_k(u) = 1/\sqrt{(u^2 + \sigma^2)}$ . Există însă și variante de funcții radiale a căror valoare crește o dată cu distanța față de centrul de simetrie (de exemplu  $g_k(u) = \frac{\sqrt{r^2 + u^2}}{r}$ ). Spre deosebire de exemplele anterioare, aceste funcții de transfer nu au un caracter local.

Ieșirea produsă de rețea, pentru un semnal de intrare  $X \in R^N$  este vectorul  $Y \in R^M$  având componentele:

$$y_i = \sum_{k=1}^K w_{ik} g_k(\|X - C^k\|) - w_{i0}, \quad i = \overline{1, M}$$

unde  $W = (w_{ik})_{i=\overline{1, M}, k=\overline{0, K}}$  este matricea ponderilor conexiunilor dintre nivelul ascuns și cel de ieșire, iar  $\|\cdot\|$  reprezintă norma euclidiană. În continuare vom nota cu  $z_k$  semnalul produs de către unitatea ascunsă,  $k$ , adică  $z_k = g_k(\|X - C^k\|)$ . Se poate folosi și o variantă normalizată, în care  $z_k$  se calculează după cum urmează:

$$z_k = \frac{g_k(\|X - C^k\|)}{\sum_{l=1}^K g_l(\|X - C^l\|)}$$

Caracterul de localitate al funcției de transfer face ca fiecare unitate ascunsă să producă ieșiri semnificative doar pentru semnale de intrare apropiate de centrul corespunzător unității. Fiecare unitate ascunsă reacționează doar la semnale provenind din zona care le este asociată, zonă numită *câmp receptiv*. O bună comportare a rețelei se obține dacă domeniul datelor de intrare este acoperit de câmpurile receptiv ale unităților ascunse. Dimensiunea câmpurilor receptiv depinde de parametrul  $\sigma_k$  corespunzător. O valoare mare pentru  $\sigma_k$  induce o dimensiune mare a câmpului receptiv pe când o valoare mică a lui  $\sigma_k$  conduce la un câmp receptiv restrâns.

Întrucât semnalul de ieșire este o combinație liniară a funcțiilor de bază, rețelele RBF pot fi încadrate în categoria rețelelor neuronale liniare motiv pentru care pot fi dezvoltati algoritmi de învățare care nu necesită tehnici de optimizare neliniară ci doar tehnici din algebra liniară. În felul acesta rețelele RBF pot beneficia de rezultate din domeniul modelelor statistice liniare. Trebuie remarcat însă că funcțiile de transfer sunt neliniare, deci rețelele RBF nu sunt modele liniare în raport cu semnalul de intrare ci doar în raport cu cel produs de către nivelul ascuns. Aceasta înseamnă că dacă parametrii asociați funcțiilor de bază (de exemplu centrul  $C_k$  și lărgimile câmpurilor receptiv,  $\sigma_k$ ) sunt fixați (nu se modifică în procesul de învățare) modelul este unul liniar pe când dacă sunt ajustabili atunci modelul este neliniar.

Apropierea dintre rețelele neuronale și modelele din statistică este ilustrată de corespondențele existente între termenii utilizați în cele două domenii (fig. 6).

Rețele neuronale	Statistică
rețea	model
învățare	estimare
învățare supervizată	regresie
generalizare	interpolare
set de antrenare	observații
ponderi	parametri
intrări	variabile independente
ieșiri	variabile dependente

Figura 6: Corespondențe terminologice între statistică și rețele neuronale [Orr, 1996]

### 3.2 Capacitatea de reprezentare.

La fel ca și rețelele feedforward cu cel puțin un nivel ascuns de unități sigmoide și rețelele RBF sunt aproximatori universali:

*Fie  $S \subset \mathbb{R}^N$  o mulțime compactă și  $\varphi : S \rightarrow \mathbb{R}^M$  o funcție continuă. Pentru orice  $\epsilon > 0$  există o rețea RBF cu  $N$  unități de intrare,  $M$  unități de ieșire și  $K$  unități ascunse astfel încât  $\sup_{X \in S} \|\varphi(X) - Y\| < \epsilon$ ,  $Y$  fiind vectorul de ieșire corespunzător vectorului de intrare  $X$ .*

Calitatea aproximării depinde de numărul de unități ascunse. Diferența esențială dintre rețelele feedforward multinivel și cele RBF este dată de faptul că primele realizează o aproximare globală iar celelalte o aproximare locală. Rețelele RBF asigură o partiționare a domeniului datelor de intrare caracterizată prin faptul că pentru fiecare zonă există o unitate ascunsă care produce o valoare semnificativă atunci când primește semnale din zona respectivă. Dacă semnalul de intrare se află la frontiera dintre două zone atunci unitățile ascunse corespunzătoare ambelor zone vor produce valori semnificative astfel că răspunsul rețelei va fi o medie ponderată a valorilor asociate acestor unități ascunse. În felul acesta, rețeaua asigură o trecere "netedă" de la o zonă la alta.

Rețelele RBF pot fi aplicate pentru rezolvarea problemelor de clasificare, de aproximare și pentru predicție în serii temporale.

### 3.3 Algoritmi de învățare.

Pornind de la un set de antrenare de forma  $\{(x^1, d^1), \dots, (x^L, d^L)\}$  o rețea RBF poate fi proiectată și antrenată în unul dintre modurile:

1. Se utilizează atâtea unități ascunse câte exemple sunt în setul de antrenare și se fixează centrul corespunzător astfel încât valorile lor să fie egale cu componentele  $x^k$  din setul de antrenare. Parametrii funcțiilor de transfer ( $\sigma_k$ ) se aleg egale cu o valoare prestabilită (de exemplu, 1) sau în funcție de distanțele existente între centrul. Singurii parametri ajustabili rămân coeficienții combinației liniare,  $w_{ik}$ ,  $k = \overline{1, M}$ ,  $i = \overline{1, L}$ .
2. Dacă numărul de exemple din setul de antrenare este mare atunci se selectează  $K < L$  exemple care sunt utilizate pentru a stabili valorile celor  $K$  centrul. Ceilalți parametri se pot determina în maniera specificată mai sus. În ce privește selecția celor  $K$  centrul aceasta se poate face în unul dintre modurile:

- Aleator. Este varianta cea mai simplă însă nu ține cont de influența elementelor selectate asupra capacității de generalizare a rețelei.
- Sistematic. Se selectează ca centri acele elemente din setul de antrenare care asigură o descreștere maximă a erorii de generalizare. Pentru a decide câți centri se vor folosi se utilizează metode de selecție a modelelor, specifice statisticii (de exemplu validarea încrucișată). Pentru a evita reantrenarea rețelei pentru fiecare nou centru selectat au fost dezvoltate diferite tehnici de accelerare bazate pe rezultate din algebra liniară. Un exemplu îl reprezintă algoritmul ”celor mai mici pătrate ortogonale” (OLS - orthogonal least squares).

3. Pentru determinarea centrilor se folosește un algoritm dinamic de grupare (capabil să stabilească și numărul centrilor). În aceste condiții centrul nu vor fi nepărat elemente din setul de antrenare ci reprezentanți ai acestora.

În continuare sunt trecute în revistă câteva dintre particularitățile acestor variante.

### 3.3.1 Determinarea ponderilor $W$ .

Considerăm că setul de antrenare este de forma  $\{(X^l, d^l)\}_{l=1, \overline{L}}$  numărul de centri este  $K$  (de regulă  $K \leq L$ ) iar valorile centrilor sunt fixate (obținute de exemplu prin selecția ca centri a unor exemple din setul de antrenare). Rămâne de determinat valorile ponderile astfel încât rețeaua să aibă o comportare cât mai bună pe setul de antrenare. Fără a altera rezultatele se poate considera că rețeaua posedă o singură unitate de ieșire. În acest caz ponderile căutate pot fi organizate sub forma unui vector, iar semnalul de ieșire ar fi:

$$y^l = \sum_{k=1}^K w_k g_{kl} \quad l = \overline{1, L},$$

unde  $g_{kl} = g(\|X^l - C^k\|)$ . Suma pătratelor erorilor pe setul de antrenare este:

$$E(W) = \sum_{l=1}^L (d^l - \sum_{k=1}^K w_k g_{kl})^2$$

iar ponderile ce minimizează această sumă sunt cele care satisfac  $\frac{\partial E(W)}{\partial w_j} = 0$  pentru  $j = \overline{1, K}$ . Dar

$$\frac{\partial E(W)}{\partial w_j} = - \sum_{l=1}^L (d^l - \sum_{k=1}^K w_k g_{kl}) g_{jl}, \quad j = \overline{1, K}$$

Condiția de anulare a derivatelor parțiale este echivalentă cu:

$$\sum_{l=1}^L g_{jl} d^l = \sum_{l=1}^L g_{jl} \left( \sum_{k=1}^K g_{kl} w_k \right), \quad j = \overline{1, K}$$

sau în scriere vectorială:

$$G^T d = (G^T G) w$$

unde

$$G = \begin{bmatrix} g_{11} & g_{21} & \cdots & g_{K1} \\ \vdots & \vdots & & \vdots \\ g_{1L} & g_{2L} & \cdots & g_{KL} \end{bmatrix}$$

Prin urmare vectorul ponderilor se obține ca fiind:

$$W = A^{-1}C^T d \quad \text{unde } A = G^T G$$

iar suma pătratelor erorilor este  $E = d^T P d$  unde  $d$  este vectorul constituit din ieșirile specificate în setul de antrenare, iar  $P$  este așa-numita matrice de proiecție:  $P = I_L - GA^{-1}G^T$  cu  $I_L$  matricea identitate de ordin  $L$ .

### 3.3.2 Selecția sistematică a centrilor.

Dacă  $L$  este mare, utilizarea tuturor exemplilor pentru a stabili valorile centrilor pe lângă faptul că este ineficientă din punct de vedere computațional poate avea influență negativă și asupra capacității de generalizare a rețelei. Ideal ar fi să se selecteze pentru centri acea submulțime din mulțimea tuturor exemplilor care asigură cea mai bună comportare a rețelei. Analiza tuturor submulțimilor posibile este costisitoare așa că se recurge la diverse tehnici euristice. Una dintre acestea este aceea a selecției incrementale a centrilor ("forward selection"). Ideea de bază este că se pornește de la un set vid de centri și se adaugă succesiv câte un centru selectat din setul de antrenare. Selecția fiecărui centru se bazează pe criteriul erorii pe setul de antrenare (se alege acel centru care provoacă cea mai mare descreștere a erorii pe setul de antrenare). Numărul de centri nu trebuie fixat de la început, decizia opririi creșterii nivelului ascuns fiind luată pe baza unui alt criteriu. Acest criteriu oferă informații privind capacitatea de generalizare. Există diverse măsuri ale capacității de generalizare. Una dintre cele mai simple este cea bazată pe tehnica validării încrucișate ("cross-validation"): setul de antrenare se descompune în două subseturi distincte, unul utilizat efectiv pentru antrenare iar celălalt pentru testare. Cu cât eroarea pe setul de testare este mai mică cu atât comportarea rețelei este mai bună. Pentru a evita influența modului de construire a setului de testare se construiesc succesiv mai multe seturi de testare iar erorile corespunzătoare acestora se cumulează. Una dintre cele mai simple variante este de a considera setul de testare constituit dintr-un singur element (tehnica LOO - "leave one out"), generând în felul acesta  $L$  seturi de antrenare/testare distincte. Decizia de a stopa adăugarea de centri se ia în momentul în care eroarea cumulată pe seturile de testare începe să crească (aceasta sugerează că adăugarea mai multor centre ar altera capacitatea de generalizare a rețelei).

Această tehnică incrementală este folosită și pentru rețelele feedforward arbitrare, însă în cazul rețelelor RBF există marele avantaj al faptului că la adăugarea unui centru sau la extragerea unui element din setul de antrenare (pentru a estima capacitatea de generalizare prin LOO) nu este necesară reantrenarea rețelei ci se pot folosi niște relații de recurență care leagă între ele matricile implicate în estimarea parametrilor  $W$  (matricile  $G$ ,  $A$ ,  $P$ ). În felul acesta numărul calculelor efectuate este redus considerabil.

Procedura poate fi accelerată prin descompunerea matricii  $G$  într-un produs între o matrice ortogonală și una triunghiulară evitându-se calculul inversei matricii  $A$  și simplificând astfel calculele. Această metodă este cunoscută sub numele de OLS - "orthogonal least squares".

### 3.3.3 Determinarea centrilor folosind tehnici de grupare.

Problematika grupării este următoarea: se dau  $L$  vectori  $X^1, X^2, \dots, X^L$  și se pune problema amplasării lor în  $K < L$  clase (grupe sau clusteri) astfel încât elementele aflate într-o clasă să fie mai apropiate între ele decât cele aflate în clase diferite. Noțiunea de apropiere se stabilește pornind de la criteriul ce a stat la baza grupării. Un astfel de criteriu îl poate reprezenta distanța euclidiană.

În rezolvarea unei probleme de grupare pot interveni două situații: numărul de clase,  $K$ , este cunoscut, respectiv este necunoscut.

Un algoritm simplu de grupare, bazat pe cunoașterea numărului de clase, este *algoritmul mediilor* bazat pe ideea că centrul (prototipul) unei clase poate fi reprezentat de media aritmetică a elementelor clasei, iar afectarea unui vector la o clasă sau alta se face pe criteriul distanței minime (fig. 7).

---

Se inițializează centrele  $C^1, C^2, \dots, C^K$  cu vectori selectați aleator din setul de antrenare.

REPEAT

*Construirea claselor:* fiecare vector  $X^l$  este afectat clasei  $\omega_k$  care verifică:

$$d(X^l, C^k) \leq d(X^l, C^i), \quad \text{pentru orice } i \in \{1, \dots, K\} \quad d \text{ fiind distanța euclidiană}$$

*Recalcularea centrelor:* pentru fiecare clasă  $\omega_k$  se recalculează centrul:

$$C^k = \frac{1}{\text{card } \omega_k} \sum_{X \in \omega_k} X, \quad k = \overline{1, K}.$$

UNTIL este îndeplinit un criteriu de oprire.

---

Figura 7: Algoritm de grupare. Metoda K-mediilor

*Observații.*

1. Algoritmul este oprit fie când la ultima parcurgere clasele nu s-au modificat fie când s-au efectuat un număr mare de iterații.
2. Prin acest algoritm se pot alege ca prototipuri vectori ce nu fac parte din setul inițial de antrenare.
3. Algoritmul mediilor conduce la determinarea acelor centri care minimizează

$$\sum_{k=1}^K \sum_{X \in \omega_k} d(X, C^k).$$

Dacă numărul de clase nu este cunoscut se poate folosi un algoritm bazat pe principiul competiției (vezi fig. 8 în care numărul claselor este inițializat cu o valoare mică și ulterior este mărit dacă este cazul. De exemplu, dacă un vector de intrare din setul de antrenare nu face parte din câmpul receptiv al nici unei unități ascunse atunci se adaugă o nouă unitate având centrul determinat chiar de vectorul de intrare.

Valoarea parametrului  $\delta$  influențează modul de partiționare a domeniului datelor de intrare. Valori mici ale lui  $\delta$  favorizează partiționarea domeniului de intrare în multe zone, pe când valori mai mari ale lui  $\delta$  conduce la stabilirea a mai puține centre. Valoarea  $\eta_t$  controlează mărimea ajustării parametrilor  $c_{kj}$ . Când devine suficient de mică, ajustarea este neglijabilă și algoritmul poate fi stopat. Descrescerea ratei  $\eta_t$  trebuie să fie lentă pentru a nu provoca oprirea algoritmului înaintea ajustării suficiente a centrilor.

## Referințe

- [1] G. Cybenko; Approximation by superpositions of a sigmoid function, Mathematics of Control, Signals and Systems, 2:303-314, 1989.
- [2] K. Funahashi; On the approximate realization of continuous mappings by neural networks, Neural Networks, 2:183-192, 1989.

---

*Inițializări:*

Se stabilește numărul inițial de centre,  $K$ .

Se inițializează valorile centrelor cu elemente selectate aleator din setul de antrenare.

Se inițializează indicatorul de iterație,  $t = 1$  și rata de învățare,  $\eta_1 \in (0, 1)$ .

REPEAT

FOR  $l = \overline{1, L}$  se efectuează

se determină  $k^* \in \{1, \dots, K\}$  cu  $\|X^l - C^{k^*}\| = \min_k \|X^l - C^k\|$ ;

IF  $\|X^l - C^{k^*}\| < \delta$  THEN  $C^{k^*} = C^{k^*} + \eta_t(X^l - C^{k^*})$

ELSE se adaugă o nouă unitate ascunsă:  $K = K + 1$  și  $C^{K+1} = X^l$

Incrementează indicatorul de iterație,  $t = t + 1$

Determină nouă valoare  $\eta_t$  a ratei de învățare,  $\eta_t = \eta_0 t^{-\alpha}$ ,  $\alpha \in (0, 1]$ .

UNTIL  $t > t_{max}$  sau  $\eta_t < \epsilon$

---

Figura 8: Algoritm de determinare a centrilor bazat pe un proces de competiție

- [3] M. Hagan, H. Demuth; Neural Network Design, PWS Publ. Company, 1995.
- [4] R. Hecht-Nielsen - NeuroComputing, 1991.
- [5] K. Hornik; Approximation Capabilities of multilayer feedforward networks, Neural Networks, 4(2), 251-257, 1991.
- [6] M. Orr; Introduction to Radial Basis Function Networks, Technical Report, Centre for Cognitive Science, Univ. of Edinburgh, 1996.
- [7] H. White; Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings, Neural Networks, 3(5):535-549, 1990.