Simulator pentru rețele neuronale Hopfield

Conf.dr. Constanţa BODEA, prof.dr. Ion IVAN, prep. Anton CREŢU,
Catedra de Informatică Economică, A.S.E., Bucureşti

Rețelele neuronale Hopfield reprezintă un bun instrument pentru rezolvarea problemelor de optimizare. În prima parte a articolului sunt trecute în revistă principalele caracteristici ale rețelelor Hopfield şi modul de utilizare a acestora în optimizare. Partea a doua a articolului prezintă un simulator de rețele neuronale Hopfield realizat de autori, cu exemplificarea utilizarii sale în rezolvarea unui model de programare liniară.

***Cuvinte cheie**: rețele neuronale recurente, stabilitatea rețelelor neuronale, optimizare combinatorială, simulator de rețele neuronale recurente.*

1. Introducere

Rețelele Hopfield reprezintă rețele neuronale recurente, simetrice, total conectate, fără autoasociere. Arhitectura acestor rețele este prezentată în fig.1.

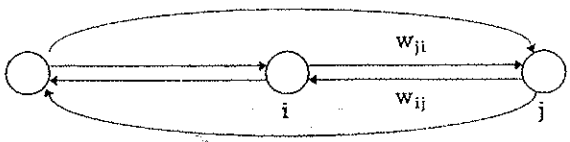

Fig. 1. Rețea Hopfield

Simetria conexiunilor se exprimă prin egalitățile:

$w_{ij}=w_{ji}$, pentru i,j desemnând unități din rețea.

Lipsa auto-asocierii se exprimă prin:

$w_{ii}=0$.

Nivelul de activare al unei unități ia valori binare {0,1} sau bipolare {-1,1}, fiind calculat cu ajutorul unei funcții de activare de forma:

$$O_j = F(\sum_i w_{ji} O_i) = \begin{cases} 1, & \text{dacă } \sum_i w_{ji} O_i > T_j \\ 0/-1, & \text{în rest} \end{cases}$$

Funcționarea rețelei presupune aplicarea inputurilor la toate nodurile din rețea în acelaşi timp. Acest lucru înseamnă setarea unităților pe un nivel de activare inițial. Rețeaua funcționează apoi autonom, trecând printr-o succesiune de stări. Setarea inițială este primul output al rețelei, care serveşte ca următor input, pe baza căruia se produce un nou output ş.a.m.d., până la fixarea rețelei pe o stare stabilă, starea în care nivelul de activare al unităților nu se mai schimbă. Rețeaua poate funcționa:

- sincron, atunci când toți neuronii încearcă să-şi schimbe starea simultan;
- asincron, atunci când fiecare neuron încearcă să-şi schimbe starea la un moment diferit de al ceilalți neuroni. La fiecare moment (tact) de timp se selectează aleator un neuron care încearcă a-şi schimba starea.

Rețeaua neuronală se află într-o stare stabilă atunci când neuronii din rețea acționează asupra celorlalți, fără a determina schimbarea valorilor de activare a niciunuia dintre ei.

Dacă O^S reprezintă o stare stabilă a rețelei, un bazin de atracție B este definit prin relația:

$$B(O^S)=\{O|T^nO=O^S\},$$

unde: T reprezintă transformarea stării, prin intermediul F, iar B reprezintă setul de stări O care evoluează spre O^S într-un număr finit de tranziții.

Rețelele Hopfield sunt utilizate prin punerea în corespondență a stărilor stabile cu soluțiile problemei pe care vrem să o rezolvăm. Rețeaua va ajunge la fixarea pe o stare stabilă, deci la soluție. Sunt două direcții importante de utilizare a rețelelor Hopfield şi anume:

- ca memorii asociative;

- pentru rezolvarea problemelor de optimizare.

Stabilitatea reprezintă proprietatea rețelei de a se stabiliza (de a atinge o stare stabilă), indiferent de starea inițială. Pentru rețele neuronale sunt definite mai multe teoreme de stabilitate, dintre care se pot aminti: Cohen - Grossberg, Kosko, Abam.

Cohen şi Grossberg au demonstrat că rețelele neuronale recurente sunt stabile dacă şi numai dacă:

$$w_{ij} = w_{ji} \text{ şi } w_{ii} = 0.$$

Deci, orice rețea Hopfield este stabilă. Această teoremă a fost demonstrată cu ajutorul unei funcții Lyapunov, utilizată ca funcție de energie a rețelei.

Fiecărei stări a rețelei i se asociază o mărime E, denumită energie. E scade de fiecare dată când un neuron îşi schimbă starea. Fie ΔE schimbarea de energie datorată schimbării stării neuronului i.

Dacă O_i=0 şi se produce schimbarea O_i=1 rezultă că:

$$\sum_j w_{ij} O_j - T_i > 0, \Delta O_i > 0.$$

Dacă O_i=1 şi se produce schimbarea O_i=0 rezultă:

$$\sum_j w_{ij} O_j - T_i < 0, \Delta O_i < 0.$$

Deci:

$$\Delta O_i (\sum_j w_{ij} O_j - T_i) > 0.$$

Hopfield a definit schimbarea nivelului de energie a rețelei, ca urmare a schimbării stării neuronului i prin relația:

$$\Delta E = -\Delta O_i (\sum_j w_{ij} O_j - T_i).$$

Energia nodului i care conduce la această schimbare este

$$E_i = -O_i (\sum_j w_{ij} O_j - T_i) = -\sum_j w_{ij} O_j O_i + O_i T_i$$

Energia totală a rețelei, aşa cum a fost definită de Hopfield, este:

$$E = -1/2 \sum_i \sum_j w_{ij} O_j O_i + \sum_i O_i T_i.$$

În concluzie, tranzițiile de stare coboară nivelul de energie până când acest lucru nu mai este posibil, moment în care se produce stabilizarea rețelei. Stabilizarea rețelei poate fi locală (rețeaua s-a fixat pe un minim local al energiei) sau globală.

2. Rezolvarea problemelor de optimizare cu ajutorul rețelelor Hopfield

Tranzițiile de stare ale rețelelor neuronale Hopfield determină scăderea nivelului de energie a rețelei până când acest lucru nu mai este posibil (rețeaua s-a stabilizat). Prin funcționarea sa, rețeaua neuronală determină singură minimizarea funcției de energie. Deci, rețeaua neuronală poate fi utilizată pentru rezolvarea unor probleme de optimizare, putând determina singură nivelul funcției obiectiv.

Să presupunem, de exemplu, un sistem S, caracterizat cu ajutorul a N variabile de stare S1,...,Sn, fiecare putând lua valorile -1/0 sau 1. Putem afirma că starea sistemului S este dată de tuplul (S1,...,Sn).

Fie o funcție cost, E(S) definită pentru acest sistem, pătratică şi simetrică în raport de Si. Minimizarea lui E poate fi realizată cu ajutorul unei rețele Hopfield, având drept intensități ale conexiunilor coeficienții ecuației pătratice a lui E. Două dificultăți apar în rezolvarea cu ajutorul rețelelor Hopfield a problemelor de optimizare:

- problemele trebuie să accepte forma pătratică (cuadratică);
- minimul obținut de rerțeaua Hopfield poate fi local.

Exemplul 1. Rezolvarea problemei comis - voiajorului.

Încercăm să definim problema sub forma unui sistem caracterizat printr-un set de variabile de stare binare şi o funcție de energie pătratică, astfel încât minimul funcției de energie să fie

soluție la problemă (cel mai scurt drum). Hopfield a caracterizat sistemul cu ajutorul unei tabele, în care fiecare linie corespunde unui oraș, iar fiecare coloană unui pas în circuit. O intrare în tabelă, (x,i) are valoarea 1, dacă orașul x este vizitat la pasul i și 0, altfel. Multe tabele posibile de valori descriu stări ilegale. O stare legală presupune un singur 1 pe fiecare linie și pe fiecare coloană. Aceste tabele sunt reprezentate în rețeaua Hopfield prin N^2 neuroni, fiecare neuron reprezentând o intrare a tabelei. Funcția de energie propusă de Hopfield este de forma:

$$E=AC_1+BC_2+CC_3+DM,$$

unde termenii au următoarea formă:

a) $C_1 = \sum_x \sum_i \sum_{i \neq j} O_{xi} \cdot O_{xj}$

Se defavorizează astfel stările cu mai mult de un 1 pe o linie x.

b) $C_2 = \sum_i \sum_x \sum_{x \neq y} O_{xi} \cdot O_{yi}$

Se defavorizează stările cu mai mult de un 1 pe coloana i.

c) $C_3 = (\sum_x \sum_i O_{xi} - N)^2$

Se încearcă să se forțeze sistemul să se stabilizeze pe stările cu N poziții setate pe 1.

d) $M = \sum_x \sum_{y \neq x} \sum_i d_{xy} \cdot O_{xi} (O_{y,i+1} + O_{y,i-1})$,

unde d_{XY} este distanța între x și y. Acesta reprezintă criteriul de lungime minimă. A,B,C,D sunt constante.

Comparând această funcție cu funcția de energie a rețelei Hopfied se obține:

$$w_{xi,yj} = -A\rho_{xy}(1-\rho_{ij}) - B\rho_{ij}(1-\rho_{xy}) - -C - Dd_{xy}(\rho_{j,i+1} + \rho_{j,i-1})$$

unde $\rho_{ij} = 1$, dacă i=j sau $\rho = 0$, altfel.

Odată ce intensitățile conexiunilor sunt determinate și ceilalți parametri aleși, rețeaua rezolvă problema prin stabilizarea sa pe soluție.

Exemplul 2. Rezolvarea problemelor de programare liniară

Considerăm următoarea problemă de programare liniară:

$\min f(x) = \sum_j c_j x_j$ cu restricțiile:

$$r_i(x): \ \sum_j a_{ij} x_j - b_j \geq 0, \quad i = \overline{1,m}.$$

Pentru această problemă trebuie să definim o funcție de energie E(x), al cărui minim X* să fie soluție la problema de programare liniară. O modalitate este de a transforma problema cu restricții într-o problemă de optimizare fără restricții, prin adăugarea unui termen de penalizare la funcția obiectiv: E = fct.cost+ restricții. Se încearcă minimizarea funcției cost și simultan maximizarea numărul de restricții satisfăcute. Pentru aceasta poate fi folosită funcția de energie:

$$E(x,k) = \sum_{j=1}^{n} c_j x_j + k \sum_{i=1}^{m} ([r_i(x)]\_)^2,$$

unde k>0 și $[r_i(x)]\_ = \min\{0, r_i(x)\}$.

Se pune problema cât de bine reușește problema de minimizare fără restricții să aproximeze problema de programare liniara originală. S-a aratat ca parametrul de penalizare k determină acuratețea acestei aproximări. Cu cât parametrul k este crescut mai mult aproximarea devine mai corectă, astfel încât soluția problemei fără restricții converge către soluția problemei restricționate.

O alta metodă de a rezolva problema este de a construi funcția de energie:

$$E(x,k) = \sum_{j=1}^{n} c_j x_j - k \sum_{i=1}^{m} \min\{0, r_i(x)\}, \ k>0.$$

Spre deosebire de tehnica anterioară, parametrul de penalizare nu trebuie să fie foarte mare sau să tindă la infinit pentru a asigura soluția exactă. Problema minimizării funcției de energie duce la următorul sistem de ecuații diferențiale:

$$\frac{dx_j}{dt} = -\mu_j (c_j + \sum_{i=1}^{m} S_i a_{ij}),$$

$x_j(0)=0$, k>0, $\mu_j>0$, (j=1,2,...,n).

Funcția S_i ia valoarea 0, dacă restricția este îndeplinită sau -k, altfel.

În ideea de a realiza o rețea neuronală discretă se poate transforma sistemul

de ecuații diferențiale într-un sistem de ecuații cu diferențe, folosind regula Euler: $x_j^{(k+1)} = x_j^{(k)} - \mu_j[c_j + \sum_{i=1}^{m} S_i^{(k)} a_{ij}]$,

unde: i=1,2,...,m; k=0,1,2,...; j=1,2,...,n.
Algoritmul de calcul este următorul:
Pasul 1. Citește A, b, c; Setează k si μ; Alege soluția inițiala $xj^{(0)}$,(j=1,2,...,n).
Pasul 2. Calculeaza $r_i(x)$, (i=1,2,...,m).
Pasul 3. Calculeaza $S_{i(k)}$, (i=1,2,...,m).
Pasul 4. Calculeaza $x_{j(k+1)}$, (j=1,2,...,n).
Pasul 5. Daca $x_{j(k+1)} > x_{j(k)} > x_{j(k-1)}$ sau $x_{j(k+1)} < x_{j(k)} < x_{j(k-1)}$ pentru (j=1,2,...,n) Atunci Stop, altfel $x_{j(k)} = x_{j(k-1)}$ si $x_{j(k+1)} = x_{j(k)}$ pentru (j=1,2,...,n) si reia de la pasul 2.

Procedura de funcționare a rețelei Hopfield pentru rezolvarea problemelor de optimizare

1) Se determină o funcție de energie bazată pe restricțiile problemei.
2) Se compară funcția de energie de la etapa 1 cu funcția de energie a unei rețele Hopfield, în scopul stabilirii parametrilor rețelei.
3) La momentul t=0: Oj(t) primește valori inițiale (valori aleatoare mici)
4)Repetă (t>0): $O_j(t+1) = F[\sum_i w_{ji} O_i(t)]$

până la echilibru.
Patternul activărilor la echilibru reprezintă soluția optimă.

3. Simulator - rețele Hopfield

Simulatorul urmărește rezolvarea problemelor de programare liniară cu ajutorul rețelelor neuronale Hopfield.

3.1. Scurte precizări

- Se folosește o funcție de energie de forma:

$$E(x,k) = \sum_{j=1}^{n} c_j x_j - k \sum_{i=1}^{m} \min\{0, r_i(x)\}, \; k>0;$$

- Rețeaua se consideră optimizată atunci când toate elementele s-au stabilizat; un element se consideră stabilizat atunci când valorile sale nu mai sunt în creștere sau descreștere;
- Rata de învățare și constanta de penalizare se dau de către utilizator.

3.2. Tipuri de fișiere folosite

*.CFG - fișier de configurare; conține informațiile referitoare la rețea: numele rețelei, rata de învățare, constanta de penalizare, numărul de variabile (neuroni), numărul de restricții, matricea sistemului, vectorul coeficienților funcției obiectiv, vectorul termenilor liberi;
*.OPT - conține valori temporare ale neuronilor (outputurilor) salvate pe parcursul procesului de optimizare, precum și numele rețelei de optimizat.

3.3. Opțiunile simulatorului

După cum se observă din figura 2, meniul principal are opțiunile: Configurare, Optimizare, Informații.

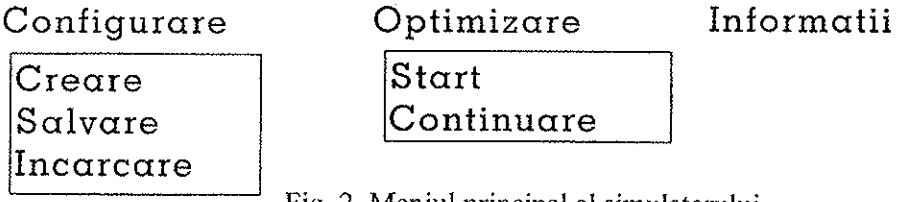

Fig. 2. Meniul principal al simulatorului

A. Configurare

Acestei opțiuni îi este ataşat un meniu popup, care are următoarele opțiuni:
A.1. *Creare*. Opțiunea serveşte pentru crearea unei noi configurații de rețea. Se introduce numele rețelei şi în continuare se cere introducerea următoarelor informații:
- parametrul de penalizare - se introduce o valoare între 1 si 999;
- numărul de variabile - se introduce o valoare între 2 şi 20;
- numărul de restricții - se introduce o valoare între 2 şi 20;
- coeficienții funcției obiectiv - se introduc valori cuprinse între 0 şi 10000;
- matricea sistemului - se introduc valori cuprinse între -10000 şi 10000;

- vectorul termenilor liberi - se introduc valori între -10000 şi 10000;

A.2. *Salvare*. Este opţiunea care serveşte la salvarea configuraţiei curente într-un fişier pe disc. Se cere introducerea numelui fişierului, căruia i se adaugă extensia .CFG. Dacă fişierul este deschis cu succes se salvează configuraţia şi apoi se continuă lucrul pe aceeaşi configuraţie.

A.3. *Încărcare*. Opţiunea serveşte la încărcarea unei configuraţii dintr-un fişier de pe disc. Se cere introducerea numelui fişierului, căruia i se adaugă extensia .CFG. Dacă fişierul este găsit se încarcă noua configuraţie cu care va continua lucrul, dacă nu, lucrul va continua cu vechea configuraţie.

B. Optimizare

Această opţiune are ataşat un meniu popup cu următoarele opţiuni:

B.1. *Start*. Foloseşte pentru startarea procesului de optimizare, generându-se o soluţie iniţială. Soluţia iniţială se generează setându-se toate valorile neuronilor pe 0.

B.2. *Continuare*. Foloseşte pentru continuarea procesului de optimizare, pornind cu rezultate intermediare preluate dintr-un fişier. Se cere numele fişierului fără extensie, căruia i se adaugă automat extensia .OPT. Dacă fişierul este găsit se reia procesul de optimizare plecând de la rezultatele citite din fişier.

La începutul procesului de optimizare, pornind fie de la o soluţie iniţială, fie de la o soluţie intermediară, se cere numărul de iteraţii care se doreşte a fi parcurs şi rata de învăţare. Dacă, după parcurgerea acestor iteraţii, se doreşte salvarea rezultatelor intermediare într-un fişier, se cere numele fişierului fără extensie, căruia i se adaugă automat extensia .OPT şi se salvează rezultatele intermediare. Dacă în timpul acestui proces se ajunge la soluţia optimă, procesul de optimizare se opreşte şi se afişează soluţia.

C. Informaţii

Această opţiune se foloseşte pentru afişarea unor informaţii despre configuraţia existentă şi anume: numele reţelei, constanta de penalizare, numărul variabilelor (neuronilor), numărul restricţiilor, coeficienţii funcţiei obiectiv, matricea sistemului, vectorul termenilor liberi.

3.4. Exemplu de utilizare

Fie următoarea problemă de programare liniară:

$$\min f(x) = 3x_1 + x_2 + 2x_3,$$

cu restricţiile:

$$\begin{cases} 3x_1 - 2x_2 + 4x_3 - 8 \geq 0 \\ -x_1 - 2x_2 - x_3 + 9 \geq 0 \\ -2x_1 - x_2 + 6 \geq 0 \end{cases}$$

Pentru această problemă s-a creat configuraţia respectivă, alegându-se un parametru de penalizare k=1.

În procesul de optimizare s-a ales o rată de învăţare μ=0.00002.

După parcurgerea a 35480 de iteraţii reţeaua s-a stabilizat, oferind următoarea soluţie: x1=0, x2=0, x3=2.000054. Valoarea funcţiei obiectiv a rezultat 4.000028.

3.5. Concluzii

Cu cât rata de învăţare este mai mare, cu atât reţeaua tinde să se stabilizeze într-un număr mai mic de iteraţii. Alegerea unei rate de învăţare mai mici are însă avantajul că soluţiile oferite de reţea tind să se apropie mai mult de cele reale.

4. Bibliografie

1. Cichocki A., Unbehauen R. - Neural Networks for Optimization and Signal Processing, John Wiley & Sons, Ltd & B.G. Teubner, Stuttgart, 1993.

2. Davalo E., Naim P. - Neural Networks, MacMillan Education, Ltd, 1991.

3. Dumitrescu D., Costin H. - Reţele neuronale. Teorie şi aplicaţii, Editura Teora, Bucureşti, 1996.