SISTEME CU IA

RETELE NEURONALE ARTIFICIALE

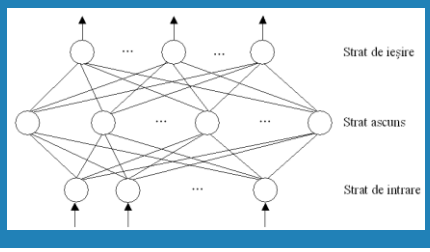
PERCEPTRONUL MULTISTRAT

Contextul general

- Pentru modelele neuronale ce foloseau neuronul formal nu existau algoritmi automati de invatare.
- Propunerea utilizarii unor neuroni ascunsi si a invatarii prin retropropagare a condus la PMS – Perceptronul Multistrat sau, in engleza, Multilayer Perceptron

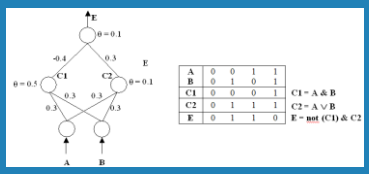
Contextul general

Arhitectura PMS



Contextul general

Invatatrea = formarea unor reprezentari interne asociate informatiei de intrare.



A	0	0	1	1	
B	0	1	0	1	
C1	0	0	0	1	C1 = A & B
C2	0	1	1	1	C2 = A ∨ B
E	0	1	1	0	E = not (C1) & C2

CUM ? Prin ajustarea ponderilor.

Regula delta - generalizata

Algoritmul de **retropropagare a erorii** a fost propus Rumelhart şi McClelland in anul 1986 si este denumit uneori şi **forma generalizată a regulii delta** (Δ).

Regula delta - generalizata

Setul de date de antrenare / invatare

Variabile			$f(x,y,z)$
x	y	z	
……	……	……	……
……	……	……	……
	……		
……	……	……	……

Fig. 4.3 – Tabelul de definiţie pentru setul de date de antrenare al perceptronului multistrat, în cazul a trei intrări – x, y şi z – şi a unei ieşiri – $f(x,y,z)$.

Initializarea ponderilor

Ponderile se iniţializează cu valori aleatorii, alese de obicei în intervalul (-1, 1).

Regula delta - generalizata

Ipoteze pentru aplicarea algoritmului

(i) se consideră cazul unei rețele de tip PMS care folosește neuroni ascunși;

(ii) funcțiile de activare ale neuronilor ascunși și ale celor de ieșire se consideră continue și derivabile;

(iii) dacă este cazul, mărimile de ieșire se scalează în intervale corespunzătoare funcției de activare folosite.

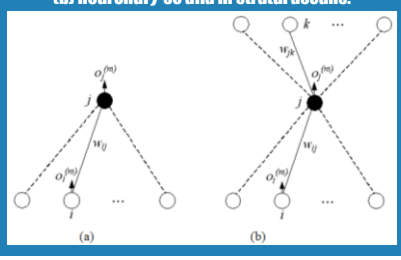
Regula delta - generalizata

2 etape principale

- propagarea inainte a modelului $x^{(m)}$ si calculul iesirii reale $o^{(m)}$.
- Retropropagarea erorii: se compara ieșirea reală o(m) cu valoarea dorită d (m) și termenul de eroare $e^{(m)} = o^{(m)} - d^{(m)}$ se propagă înapoi în rețea – de la stratul de ieșire, spre stratul de intrare – prin ajustarea ponderilor cu cantitatea $\Delta w^{(m)}$, conform principiului celor mai mici patrate.

Regula delta - generalizata

Explicativa: (a) neuronul *j* se află în stratul de ieșire sau (b) neuronul *j* se află în stratul ascuns.



Regula delta - generalizata

Propozitia 1

Pentru fiecare model de intrare – ieşire m din setul de antrenare, corecţia unei ponderi w_{ij} – notată $\Delta^{(m)}w_{ij}$ – pentru conexiunea dintre neuronul j şi neuronul i din stratul inferior este proporţională cu un termen de eroare $\delta_j^{(m)}$ asociat neuronului j:

$$\Delta^{(m)}w_{ij} = \eta \cdot \delta_j^{(m)} \cdot o_i^{(m)}$$

unde η este un factor de proporţionalitate, numit *rată de învăţare*.

Regula delta - generalizata

Propozitia 2

Dacă neuronul j se află în stratul de ieşire, termenul de eroare $\delta_j^{(m)}$ se calculează în funcţie de abaterea între valoarea reală $o_j^{(m)}$ şi cea dorită $d_j^{(m)}$ şi derivata funcţiei de activare f a neuronului j în raport cu intrarea netă corespunzătoare modelului m, notată $net_j^{(m)}$:

$$\delta_j^{(m)} = \left(d_j^{(m)} - o_j^{(m)}\right) \cdot f'\left(net_j^{(m)}\right)$$

Regula delta - generalizata

Propozitia 2 - continuare

Dacă neuronul j se află în stratul ascuns, fiind legat prin conexiuni sinaptice cu neuronii k din stratul de ieşire, termenul de eroare $\delta_j^{(m)}$ este proporţional cu suma tuturor termenilor de eroare asociaţi neuronilor de ieşire k, modificaţi de ponderile conexiunilor respective w_{jk} şi cu derivata funcţiei de activare în raport cu intrarea netă $net_j^{(m)}$:

$$\delta_j^{(m)} = \left(\sum_k \delta_k^{(m)} \cdot w_{jk}\right) \cdot f'\left(net_j^{(m)}\right)$$

Regula delta - generalizata

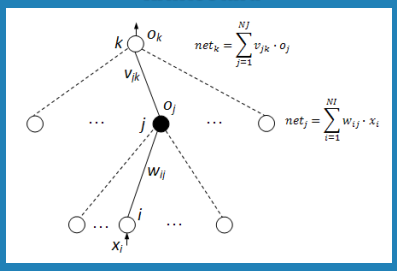
Propozitia 3

Regula delta – generalizata are la baza principiul minimizarii erorii patratice, care descrie abaterea patratica intre valorile reala si dorita la iesirea retelei:

$$E^{(m)} = \frac{1}{2}\sum_{j=1}^{J}\left(d_j^{(m)} - o_j^{(m)}\right)^2$$

Regula delta - generalizata

Arhitectura



Regula delta - generalizata

Principiu

Retropropagarea erorii dupa regula delta – generalizata corespunde de fapt unei minimizari a erorii *E* dupa metoda gradientului :

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \eta \cdot \nabla E\ (\boldsymbol{w}^{t}) = \boldsymbol{w}^{t} - \eta \cdot \Delta \boldsymbol{w}^{t}$$

$$w_{ij}^{t+1} = w_{ij}^{t} - \eta \cdot \frac{\partial E}{\partial w_{ij}}\bigg|_{w_{ij}} = w_{ij}^{t} - \Delta w_{ij}^{t}$$

Regula delta - generalizata

Principiu - continuare

Daca se renunta la indicele *m* care indica numarul de ordine al modelului din setul de date de antrenare si se considera cazul general al unei retele cu *NK* neuroni pe stratul de iesire,, expresia erorii pentru unul din modelele de antrenare este:

$$E = \frac{1}{2}\sum_{k=1}^{NK}(o_k - d_k)^2$$

Regula delta - generalizata

Actualizarea ponderii v_{jk}

$$\frac{\partial E}{\partial v_{jk}} = (o_k - d_k)\cdot\frac{\partial o_k}{\partial v_{jk}} = (o_k - d_k)\cdot f'(net_k)\cdot\frac{\partial net_k}{\partial v_{jk}} =$$

$$= \underbrace{(o_k - d_k)\cdot f'(net_k)}_{\text{termenul de eroare } \delta_k}\cdot o_j$$

$$\Rightarrow \Delta v_{jk} = \eta\cdot\delta_k\cdot o_j$$

Regula delta - generalizata

Actualizarea ponderii w_{ij}

$$\frac{\partial E}{\partial w_{ij}} = \sum_{k=1}^{NK}(o_k - d_k)\cdot\frac{\partial o_k}{\partial w_{ij}} = \sum_{k=1}^{NK}(o_k - d_k)\cdot f'(net_k)\cdot\frac{\partial net_k}{\partial w_{ij}} =$$

$$= \sum_{k=1}^{NK}\underbrace{(o_k - d_k)\cdot f'(net_k)}_{\text{termenul de eroare } \delta_k}\cdot v_{jk}\cdot\frac{\partial o_j}{\partial w_{ij}} =$$

$$= \left(\sum_{k=1}^{NK}\delta_k\cdot v_{jk}\right)\cdot\frac{\partial o_j}{\partial w_{ij}} = \left(\sum_{k=1}^{NK}\delta_k\cdot v_{jk}\right)\cdot f'(net_j)\frac{\partial net_j}{\partial w_{ij}} =$$

$$= \underbrace{\left(\sum_{k=1}^{NK}\delta_k\cdot v_{jk}\right)\cdot f'(net_j)}_{\text{termenul de eroare } \delta_j}\cdot x_i \quad\Rightarrow\quad \Delta w_{ij} = \eta\cdot\delta_j\cdot x_i$$

Regula delta - generalizata

Functia sigmoid logistic

Functia de activare:

$$f(x) = \frac{1}{1+e^{(-x+b)}}$$

... si derivata sa:

$$f'(x) = \frac{-(-1)\cdot e^{(-x+b)}}{\left[1+e^{(-x+b)}\right]^2} = \frac{e^{(-x+b)}}{\left[1+e^{(-x+b)}\right]^2} = \frac{1+e^{(-x+b)}-1}{\left[1+e(-x+b)\right]^2} =$$

$$= \frac{1}{1+e^{(-x+b)}} - \frac{1}{\left[1+e^{(-x+b)}\right]^2} = f(x) - \left[f(x)\right]^2 = f(x)\cdot\left[1-f(x)\right]$$

Regula delta - generalizata

Functia sigmoid logistic - continuare

Ipoteza: se considera ca PMS utilizeaza numai functii de activare de tip sigmoid logistic.

$$\frac{\partial E}{\partial v_{jk}} = (o_k - d_k)\cdot o_k\cdot(1-o_k)\cdot o_j \qquad v_{jk}^{t+1} = v_{jk}^{t} - \eta\cdot(o_k - d_k)\cdot o_k\cdot(1-o_k)\cdot o_j$$

$$\frac{\partial E}{\partial w_{ij}} = \left[\sum_{k=1}^{NK}(o_k - d_k)\cdot o_k\cdot(1-o_k)\cdot v_{jk}\right]\cdot o_j\cdot(1-o_j)\cdot x_i$$

$$w_{ij}^{t+1} = w_{ij}^{t} - \eta\cdot\left[\sum_{k=1}^{NK}(o_k - d_k)\cdot o_k\cdot(1-o_k)\cdot v_{jk}\right]\cdot o_j\cdot(1-o_j)\cdot x_i$$

Algoritmul de retropropagare

– Forma elementara –

1. Definirea arhitecturii rețelei PMS: numărul de neuroni de pe fiecare strat (I, J, K) şi setul de date de antrenare: $\{x^{(m)}, d^{(m)}\}$ $m = 1,\ldots,M$. Definirea numărului de cicluri de antrenare: C_{max}.
2. Definirea parametrilor rețelei: ratele de învățare pentru ponderile v şi w, notate η_1, respectiv η_2.
3. Inițializarea ponderilor rețelei cu valori aleatorii în intervalul (-1, 1):

$$v_{jk} = 2\cdot random(\,) - 1;$$
$$w_{ij} = 2\cdot random(\,) - 1;$$

($i = 1,\ldots,I$; $j = 1,\ldots, J$; $k = 1,\ldots,K$).

Algoritmul de retropropagare
– Forma elementara –

4. Ajustarea poderilor:

for $c = 1$ to C_{max} do.
for $m = 1$ to M do.
// Propagare înainte în primul strat
for $j = 1$ to J do
$y_j = 0$;
for $i = 1$ to I do $y_j = y_j + w_{ji} \cdot x_i$
// Propagarea înainte în al doilea strat
for $k = 1$ to K do
$o_k = 0$;
for $j = 1$ to J do $o_k = o_k + v_{kj} \cdot y_j$

Algoritmul de retropropagare
– Forma elementara –

for $j = 1$ to J do
// Adaptarea ponderilor pentru al II-lea strat
for $k = 1$ to K do

$$v_{jk} = v_{jk} + \eta_1 \cdot \left(d_k^{(m)} - o_k\right) \cdot o_k \cdot (1 - o_k) \cdot y_j$$

// Adaptarea ponderilor pentru primul strat}
for $i = 1$ to I do

$$w_{ij} = w_{ij} + \eta_2 \cdot \sum_{k=1}^{K} \left[\left(d_k^{(m)} - o_k\right) \cdot o_k \cdot (1 - o_k) \cdot v_{jk}\right] \cdot o_k \cdot (1 - o_k) \cdot x_i^{(m)}$$

5. Rețeaua a fost antrenată pe cele M modele, în C_{max} cicluri, iar caracteristicile sale se găsesc în ponderile v_{jk} şi w_{ij}.

Procedee de accelerare a convergentei

(i) optimizarea procesului de inițializare a ponderilor din rețea,
(ii) stabilizarea procesului de ajustare a ponderilor,
(iii) accelerarea propriu-zisă a convergenței prin aplicarea unor tehnici de optimizare mai eficiente şi
(iv) selectarea unei arhitecturi a rețelei care să asigure performanțele cele mai bune.

Accelerare a convergentei – Initializarea ponderilor

(i) Procedura standard: initializarea cu valori aleatoare, mici, in intervalul (-1, 1) sau (-0.5 , 0.5);

(ii)Regula Russo:

$$-\frac{2.4}{I} \le w_{ij} \le \frac{2.4}{I}$$

unde **I** – numarul de conexiuni de intrare ale neuronului.

Accelerare a convergentei – Initializarea ponderilor

(iii) Procedura Nguyen – Widrow:

se defineste parametrul:

$$\beta = 0.7 \cdot J^{\frac{1}{I}}$$

iar ponderile se initializeaza cu relatia:

$$w_{ij} = \frac{\beta}{\|w^*\|} \cdot w^*_{ij}$$

unde: $[w^*] = \{w^*_{11}, \ldots, w^*_{IJ}\}$

Accelerare a convergentei – Termen de moment

Scop: reducerea oscilatiilor traiectoriei pe suprafata erorii.

Solutie: introducerea in termenul de corectie a ponderilor a unui termen de "*moment*" proportional cu viteza de deplasare (valoarea corectiei din iteratia anterioara).

$$\mathbf{z}^{t+1} = \mathbf{z}^t - \eta \cdot \nabla E(\mathbf{z}^t) + \mu \cdot (\mathbf{z}^t - \mathbf{z}^{t-1})$$

sau:

$$z_{pq}^{t+1} = z_{pq}^t - \eta \cdot \left.\frac{\partial E}{\partial z_{pq}}\right|_{z_{pq}^t} + \mu \cdot (z_{pq}^t - z_{pq}^{t-1})$$

Accelerare a convergentei – Termen de moment

Efecte :

- la începutul procesului de antrenare, când corecțiile ponderilor sunt relativ însemnate, se asigură deplasarea în direcția generală a micşorării erorii, evitându-se „prinderea” în minime locale;
- termenul de moment contribuie la atenuarea oscilațiilor şi netezirea traiectoriei aproximațiilor succesive pe suprafața erorii.

Accelerare a convergentei – Rata de invatare

Reducerea progresiva a ratei de invatare:

- In faza initiala se foloseste o rata de invatare mare: deplasarea pe suprafața erorii se face cu paşi mari, ceea ce permite depăşirea minimelor locale.
- Dupa apropierea de minimul cautat: reducerea valorii ratei de învățare permite stabilizarea procesului de aproximații succesive în zona acestui minim, reducând riscul depăşirii sale.

Accelerare a convergentei – Rata de invatare

Principii:

- Dacă în două iterații succesive derivata $\partial E/\partial w$ păstrează semnul (eroarea E continuă să scadă), rata de învățare este mărită pentru accelerarea apropierii de minim;
- Dacă, între două iterații succesive derivata $\partial E/\partial w$ schimba semnul (eroarea E începe să crească), rata de învățare este micşorată pentru revenirea pe partea de pantă descrescătoare.

Accelerare a convergentei – Rata de invatare

Adaptarea ratei de invatare:

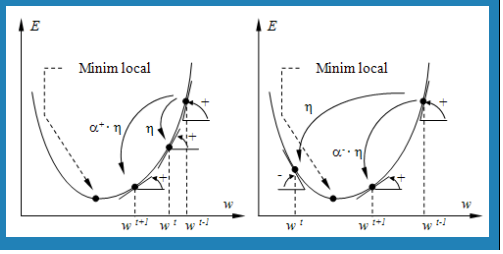
$$\eta^{t+1} = \alpha^{+} \cdot \eta^{t} \quad daca \quad \left(\frac{\partial E}{\partial w}\right)^{t-1} \cdot \left(\frac{\partial E}{\partial w}\right)^{t} > 0$$

$$\eta^{t+1} = \alpha^{-} \cdot \eta^{t} \quad daca \quad \left(\frac{\partial E}{\partial w}\right)^{t-1} \cdot \left(\frac{\partial E}{\partial w}\right)^{t} < 0$$

$$\eta^{t+1} = \eta^{t} \quad daca \quad \left(\frac{\partial E}{\partial w}\right)^{t-1} \cdot \left(\frac{\partial E}{\partial w}\right)^{t} = 0$$

Accelerare a convergentei – Rata de invatare

Adaptarea ratei de invatare:



Accelerare a convergentei – Functia RProp

RProp – Resilient Propagation (Propagare Elastica)

Principiu:

Algoritmul *RProp* nu foloseşte mărimile derivatelor $\partial E/\partial z_{ps}$ ci numai semnele acestora. Se foloseşte câte un coeficient de adaptare δ_{ps} pentru fiecare pondere z_{ps} care îşi modifică valoarea în funcţie de evoluţia semnelor derivatelor funcţiei eroare.

Accelerare a convergentei – Functia RProp

Principiu:

$$\delta_{ps}^{(t+1)} = \begin{cases} \eta^+ \cdot \delta_{ps}^{(t)} & daca \quad \left(\frac{\partial E}{\partial z_{ps}}\right)^{t+1} \cdot \left(\frac{\partial E}{\partial z_{ps}}\right)^{t} > 0 \\ \eta^- \cdot \delta_{ps}^{(t)} & daca \quad \left(\frac{\partial E}{\partial z_{ps}}\right)^{t+1} \cdot \left(\frac{\partial E}{\partial z_{ps}}\right)^{t} < 0 \\ \delta_{ps}^{(t)} & daca \quad \left(\frac{\partial E}{\partial z_{ps}}\right)^{t+1} \cdot \left(\frac{\partial E}{\partial z_{ps}}\right)^{t} = 0 \end{cases}$$

Accelerare a convergentei – Functia RProp

Adaptarea ponderilor:

$$\Delta z_{ps}^{(t+1)} = \begin{cases} -sign\left[\left(\frac{\partial E}{\partial z_{ps}}\right)^{t+1}\right] \cdot \delta_{ps}^{(t+1)} & daca \quad \left(\frac{\partial E}{\partial z_{ps}}\right)^{t+1} \cdot \left(\frac{\partial E}{\partial z_{ps}}\right)^{t} \geq 0 \\ -sign\left[\Delta z_{ps}^{(t)}\right] \cdot \delta_{ps}^{(t+1)} & daca \quad \left(\frac{\partial E}{\partial z_{ps}}\right)^{t+1} \cdot \left(\frac{\partial E}{\partial z_{ps}}\right)^{t} < 0 \end{cases}$$

$$z_{ps}^{(t+1)} = z_{ps}^{(t)} + \Delta z_{ps}^{(t+1)}$$

Criterii de oprire

Criteriul numarului maxim de cicluri de antrenare

- T_{max} prea mic: oprirea in minime locale;
- T_{max} prea mare: specializarea retelei pe setul de date de antrenare (supra-antrenare sau supra-invatare).
- Consecinta: T_{max} modest si teste off-line.

Criterii de oprire

Criteriul setului de date test

Setul initial de date se imparte in:

- Setul de natrenare
- Setul test

Antrenarea se face pe setul de antrenare si procesul se intrerupe cand, dupa un numar dat de cicluri succesive eroarea pe setul test incepe sa creasca.