Lucrarea 4.a

Reteaua neurala Perceptron Multistrat (MLP)

1. Baza teoretica

1.1. Arhitectura perceptronului neliniar multinivel

Structura clasica de perceptron neliniar- "Non-linear Perceptron" (NP) este o retea fara reactie ("feedforward") care contine trei niveluri de neuroni, asa cum este reprezentat in Figura 1.1, de mai jos.

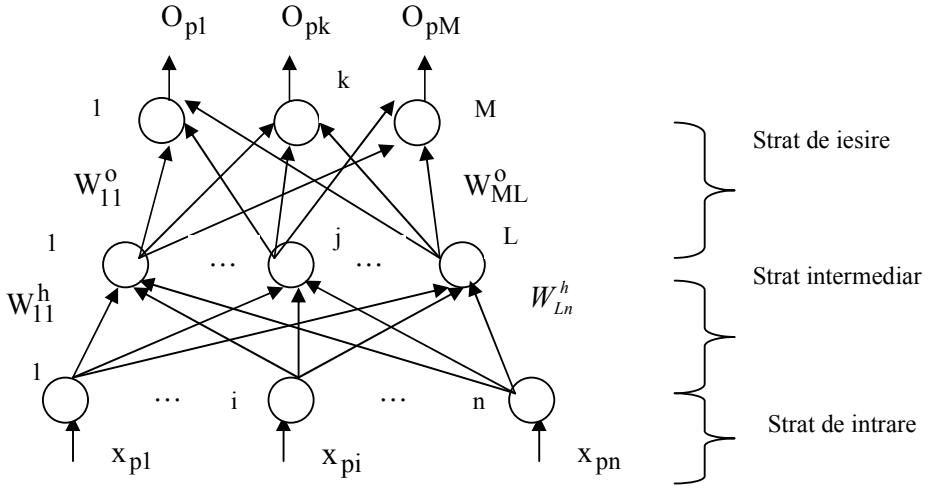


Figura 1.1. Retea perceptron neliniar cu trei straturi de neuroni (strutura clasica)

Primul strat al NP contine neuroni virtuali care nu realizeaza o prelucrare de semnal, ci doar o multiplexare, prelucrarea propriu- zisa avand loc doar in stratul intermediar si in cel de iesire.

A. **Ecuatiile unui neuron din stratul intermediar**

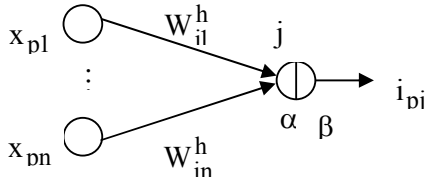


Figura 1.2. Conexiunile neuronului j (j=1,…,L) din stratul intermediar

- $X_p = (x_{p1},\ldots,x_{pn})^t$ are semnificatia de vector care se aplica la intrarea NP.
- $\{W_{ji}^h\}$, j=1,…,L, i=1,…,n reprezinta multimea ponderilor aferente stratului intermediar. L este numarul de neuroni ai stratului intermediar, iar n reprezinta numarul neuronilor de pe stratul de intrare. Indicele superior "h" care apare in notarea ponderilor acestui strat provine de la "hidden".

Exista doua etape de prelucrare: $\begin{cases} \alpha = \text{liniara} \\ \beta = \text{neliniara} \end{cases}$.

Etapa liniara (α)

In cadrul acestei etape vom defini activarea neuronului j din stratul intermediar**,** cand la intrare se aplica vectorul X_p**:**

$$(\alpha)\quad net_{pj}^h = \sum_{i=1}^{n} W_{ji}^h \cdot x_{pi},\ j=1,\ldots,L \tag{1.1}$$

unde net_{pj}^h este o variabila de stare interna (activarea neuronului j).

Daca $W_j^h = (W_{j1}^h,\ldots,W_{jn}^h)^t$, atunci:

$$net_{pj}^h = \left\langle W_j^h, X_p \right\rangle = \left[W_j^h\right]^t \cdot X_p,\ j=1,\ldots,L \tag{1.2}$$

Etapa neliniara β

Aceasta etapa presupune calcularea iesirii neuronului intermediar j, conform formulei:

$$(\beta)\quad i_{pj} = f_j^h\left(net_{pj}^h\right),\ j=1,\ldots,L \tag{1.3}$$

unde f_j^h este o functie neliniara de tipul $f:\Re\to\Re$, $f(x)=\dfrac{1}{1+e^{-\lambda x}}$ ($\lambda>0$ constant) (neliniaritate sigmoidala conform Figurii 1.3) sau $f(x)=\text{th } x$ (neliniaritate de tip tangenta hiperboloica, conform figurii Figurii 1.4).

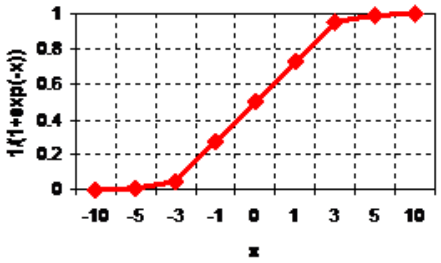


Figura 1.3. Functia sigmoida

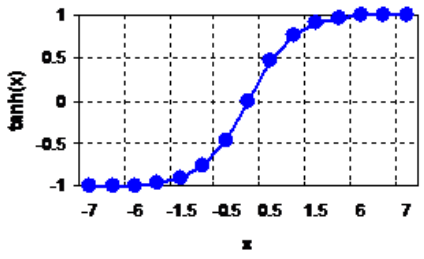


Figura 1.4. Functia tangenta hiperbolica

B. Ecuatiile unui neuron din stratul de iesire

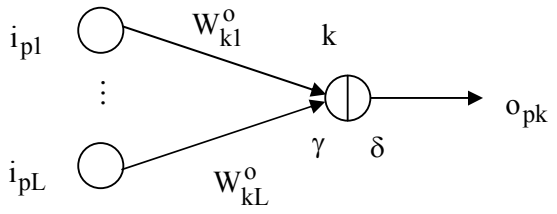


Figura 1.5. Conexiunile neuronului k (k=1,…,M) din stratul de iesire

$\{W^o_{kj}\}$, k=1,…,M, j=1,…,L reprezinta multimea ponderilor aferente stratului de iesire (indicele o=”output”). M este numarul de neuroni ai stratului de iesire.

Exista doua etape de prelucrare: $\begin{cases} \gamma = \text{liniara} \\ \delta = \text{neliniara} \end{cases}$.

Etapa liniara (γ)

Activarea neuronului k din stratul de iesire, cand la intrarea NP se aplica vectorul X_p este data de formula:

$$(\gamma) \qquad net^o_{pk} = \sum_{j=1}^{L} W^o_{kj} \cdot i_{pj} \text{ , k=1,…,M} \tag{1.4}$$

unde net^o_{pk} este o variabila de stare interna (activarea neuronului k).

Etapa neliniara (δ)

Iesirea neuronului de iesire k este determinata in functie de activarea net^o_{pk} si este exprimata de formula:

$$(\underline{\delta}) \quad o_{pk} = f^o_k\left(net^o_{pk}\right), \text{ k=1,…,M} \tag{1.5}$$

f^o_k poate fi sigmoidala sau tangenta hiperbolica. k=1,…, K.

1.2. Algoritmul de instruire

Perceptronul multinivel are un algoritm de instruire supervizata, de tip “back-propagation”, in scopul minimizarii erorii pe lotul de instruire:

$$E = \frac{1}{K} \sum_{p=1}^{K} E_p \tag{1.10}$$

K fiind numarul de vectori din lotul de instruire, iar:

$$E_p = \frac{1}{2}\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)^2 \tag{1.11}$$

constituie eroarea determinata de vectorul p de instruire.

Pasul 1. Se initializeaza p = 1. Se aplica la intrarea retelei vectorul:

$$X_p = \left(x_{p1}, \cdots, x_{pn}\right)^T$$

pentru care se cunoaste vectorul de iesire ideal:

$$Y_p = \left(y_{p1,\ldots,} y_{pM}\right)^T$$

unde:

- n reprezinta dimensiunea vectorului de intrare,
- M numarul de clase (numarul neuronilor de pe stratul de iesire).

Apoi se genereaza aleator multimea ponderilor aferente stratului intermediar adica :

- $\left\{W_{ji}^h\right\}$, j=1, …, L, i = 1,…,n multimea ponderilor aferente stratului intermediar
- $\left\{W_{kj}^o\right\}$ k=1,…, M, j = 1,…,L, multimea ponderilor aferente stratului de iesire

unde L este numarul de neuroni de pe stratul intermediar.

Pasul 2. Se calculeaza activarile neuronilor din stratul intermediar folosind formula :

$$net_{pj}^h = \sum_{i=1}^{n} W_{ji}^h \cdot x_{pi}\text{ , } j=1,\ldots,L. \tag{1.1}$$

Pasul 3. Se calculeaza iesirile neuronilor din stratul intermediar folosind relatia :

$$i_{pj} = f_j^h\left(net_{pj}^h\right),\ j=1,..,L. \tag{1.3}$$

Pasul 4. Se calculeaza activarile neuronilor din stratul de iesire pe baza relatiei :

$$net_{pk}^o = \sum_{j=1}^{L} W_{kj}^o \cdot i_{pj}\text{ , } k=1,\ldots,M. \tag{1.4}$$

Pasul 5. Se calculeaza iesirile neuronilor de pe stratul de iesire conform relatiei :

$$o_{pk} = f_k^o\left(net_{pk}^o\right),\ k=1,..,M \tag{1.5}$$

Pasul 6. Se calculeaza termenii eroare pentru neuronii de iesire conform relatiei:

$$\delta_{pk}^o = \left(y_{pk} - o_{pk}\right)\cdot f_k^{'o}\left(net_{pk}^o\right), k = 1,\ldots,M \tag{1.12}$$

Pasul 7. Se calculeaza termenii eroare pentru neuronii din stratul intermediar:

$$\delta_{pj}^{h} = f_j^{'h}\left(net_{pj}^{h}\right)\cdot \sum_{k=1}^{M} \delta_{pk}^{o} \cdot W_{kj}^{o},\ j = 1,\ldots,L, \tag{1.13}$$

Pasul 8. Se rafineaza ponderile aferente stratului de iesire pe baza relatiei:

$$W_{kj}^{o}(t+1) = W_{kj}^{o}(t) - \eta \frac{\partial E_p}{\partial W_{kj}^{o}},\ k=1,\ldots M,\ j=1,\ldots,L \tag{1.14}$$

unde $\eta,\ 0 < \eta < 1$ este rata de invatare.

$$\frac{\partial E_p}{\partial W_{kj}^{o}} = -\left(y_{pk} - o_{pk}\right)\frac{\partial f_k^{o}}{\partial W_{kj}^{o}} = -\left(y_{pk} - o_{pk}\right)\cdot \frac{\partial f_k^{o}}{\partial\left(net_{pk}^{o}\right)}\cdot \frac{\partial\left(net_{pk}^{o}\right)}{\partial W_{kj}^{o}} \tag{1.15}$$

$$\frac{\partial\left(net_{pk}^{o}\right)}{\partial W_{kj}^{o}} = \frac{\partial}{\partial W_{kj}^{o}}\left(\sum_{j=1}^{L} W_{kj}^{o}\cdot i_{pj}\right) = i_{pj} \tag{1.16}$$

Inlocuind relatia (1.16) in relatia (1.15) rezulta:

$$\frac{\partial E_p}{\partial W_{kj}^{o}} = -\left(y_{pk} - o_{pk}\right)\cdot f_k^{'o}\left(net_{pk}^{o}\right)\cdot i_{pj} \overset{(2.12)}{=} -\delta_{pk}^{o}\cdot i_{pj} \tag{1.17}$$

Pe baza relatiei (1.17), formula (1.14) de rafinare a ponderilor aferente stratului de iesire este echivalenta cu formula urmatoare:

$$W_{kj}^{o}(t+1) = W_{kj}^{o}(t) + \eta\delta_{pk}^{o}\cdot i_{pj},\ k = 1,\ldots,M,\ j = 1,\ldots,L \tag{1.18}$$

Pasul 9. Se rafineaza ponderile aferente stratului intermediar:

$$W_{ji}^{h}(t+1) = W_{ji}^{h}(t) - \eta \frac{\partial E_p}{\partial W_{ji}^{h}},\ j=1,\ldots,L,\ i=1,\ldots,n \tag{1.19}$$

$$E_p = \frac{1}{2}\sum_{k=1}^{M}\left[y_{pk} - f_k^{'o}\left(net_{pk}^{o}\right)\right]^2 = \frac{1}{2}\sum_{k=1}^{M}\left[y_{pk} - f_k^{'o}\left(\sum_{j=1}^{L} W_{kj}^{o}\cdot i_{pj}\right)\right]^2 \tag{1.20}$$

$$\frac{\partial E_p}{\partial W_{ji}^{h}} = \frac{1}{2}\sum_{k=1}^{M}\frac{\partial}{\partial W_{ji}^{h}}\left(y_{pk} - o_{pk}\right)^2 = -\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)\cdot \frac{\partial o_{pk}}{\partial\left(net_{pk}^{o}\right)}\cdot \frac{\partial\left(net_{pk}^{o}\right)}{\partial i_{pj}}\cdot \frac{\partial i_{pj}}{\partial\left(net_{pj}^{h}\right)}\cdot \frac{\partial\left(net_{pj}^{h}\right)}{\partial W_{ji}^{h}}\ (2.21) \tag{1.21}$$

$$\frac{\partial E_p}{\partial W_{ji}^{h}} = -\sum_{k=1}^{M}\left(y_{pk} - o_{pk}\right)\cdot f_k^{'o}\left(net_{pk}^{o}\right)\cdot W_{kj}^{o}\cdot f_j^{'h}\left(net_{pj}^{h}\right)\cdot x_{pi} \tag{1.22}$$

$$\frac{\partial E_p}{\partial W_{ji}^h} = -f_j'^h\left(net_{pj}^h\right)\cdot\sum_{k=1}^{M}\left(y_{pk}-o_{pk}\right)\cdot f_k'^o\left(net_{pk}^o\right)\cdot W_{kj}^o\cdot x_{pi} \overset{(1.12)}{=} -f_j'^h\left(net_{pj}^h\right)\cdot[\sum_{k=1}^{M}\delta_{pk}^o\cdot W_{kj}^o]\cdot x_{pi} \tag{1.23}$$

$$\frac{\partial E_p}{\partial W_{ji}^h} \overset{(1.13)}{=} -\delta_{pj}^h\cdot x_{pi} \tag{1.24}$$

Tinand cont de relatia (1.24), formula (1.19) de rafinare a ponderilor aferente stratului intermediar devine formula (1.25)

$$W_{ji}^h(t+1) = W_{ji}^h(t) + \eta\cdot\delta_{pj}^h\cdot x_{pi},\ j=1,\ldots,L,\ i=1,\ldots,n \tag{1.25}$$

Pasul 10. Se calculeaza eroarea datorata vectorului p de instruire data de formula:

$$E_p = \frac{1}{2}\sum_{k=1}^{M}\left(y_{pk}-o_{pk}\right)^2 \tag{1.11}$$

Pasul 11. Daca p < N se face p = p + 1, adica se introduce un nou vector la intrarea retelei. Altfel se calculeaza eroarea corespunzatoarea epocii respective (prin epoca se intelege parcurgerea intregului lot de vectori) folosind formula:

$$E = \frac{1}{N}\sum_{p=1}^{N}E_p \tag{1.10}$$

si se incepe o noua epoca de instruire.

Algoritmul de instruire se termina dupa un anumit numar (fixat) de epoci.

1.3. Analiza complexitatii algoritmului de instruire

Daca consideram drept functie de activare:

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

se stie ca:

$$f'(x) = f(x)[1-f(x)]$$

Rezulta:

$$f_k'^o\left(net_{pk}^o\right) = o_{pk}\left(1-o_{pk}\right),\ k=1,\ldots,M \tag{1.26}$$

$$f_j'^h\left(net_{ph}^o\right) = i_{pj}\left(1-i_{pj}\right),\ j=1,..,L \tag{1.27}$$

Relatia (1.1) care apare la *Pasul 2* al algoritmului de instruire pentru a calcula activarile neuronilor de pe stratul intermediar necesita $L\cdot n$ inmultiri.

La *Pasul 4* al algoritmului de invatare sunt necesare $M \cdot L$ inmultiri pentru a calcula cu formula (1.4) activarile neuronilor de pe stratul de iesire.

Relatia (1.12) utila la determinarea termenilor eroare pentru neuronii de iesire se poate rescrie in relatia echivalenta (1.28), pe baza presupunerii din relatia (1.26).

$$\delta_{pk}^{o} = \left(y_{pk} - o_{pk}\right) \cdot o_{pk} \cdot \left(1 - o_{pk}\right), \; k=1,\ldots,M \tag{1.28}$$

In acest caz se efectueaza 2M inmultiri la *Pasul 6* al algoritmului de instruire.

Formula (1.13) de calcul a termenilor eroare pentru neuronii din stratul intermediar devine:

$$\delta_{pj}^{h} = i_{pj} \cdot \left(1 - i_{pj}\right) \cdot \sum_{k=1}^{M} \delta_{pk}^{o} \cdot W_{kj}^{o}, \; j=1,\ldots,L \tag{1.29}$$

Pentru a minimiza numarul de inmultiri care ar trebui efectuate la *Pasul 7* in vederea determinarii termenilor eroare pentru neuronii din stratul intermediar, folosind formula (1.29) se face o operatie de atribuire $z_j = i_{pj} \cdot \left(1 - i_{pj}\right)$ (1.30). Formula (1.29) va fi inlocuita cu (1.31), de mai jos:

$$\delta_{pj}^{h} = z_j \cdot \sum_{k=1}^{M} \delta_{pk}^{o} \cdot W_{kj}^{o} \tag{1.31}$$

care necesita $M \cdot L + L$ inmultiri.

Pentru a rafina la *Pasul 8* ponderile aferente stratului de iesire folosind formula (1.18) avem de efectuat $M \cdot L$ inmultiri.

Din formula (1.19) de rafinare a ponderilor aferente stratului intermediar, prezenta la *Pasul 9* se observa ca se realizeaza $n \cdot L$ inmultiri la nivelul acestui pas al algoritmului de instruire.

Numarul total de multiplicari care se efectueaza intr-o epoca este:

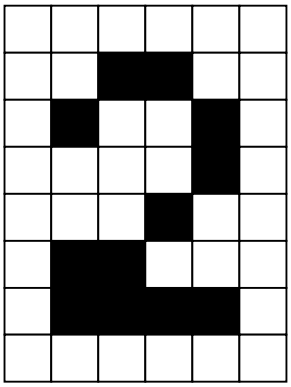
$$K*(2*L*n+3*M*L+2*M+L) \tag{1.32}$$

iar numarul total de multiplicari necesari tuturor epocilor este formula de mai sus inmultita cu numarul total de epoci de instruire.

2. Problema

Fie o retea MLP de recunoastere a celor 10 cifre. Cifrele sunt reprezentate prin imagini binare (valorile pixelilor pot fi 0 sau 1) de dimensiune 8 x 6 pixeli. Lotul de antrenare contine cate 5 imagini pentru fiecare cifra. Se cere:

- numarul neuronilor din stratul de intrare
- numarul neuronilor din stratul de iesire
- numarul total de ponderi daca pe stratul intermediar sunt 5 neuroni
- numarul de iteratii al unei epoci
- daca toate ponderile retelei sunt 0.5, sa se afle iesirile neuronilor din stratul intermediar daca la intrarea retelei se introduce imaginea de mai jos:



3. Aplicatii de laborator

3.1 Clasificarea vinurilor folosind reteaua MLP

Obiectiv: clasificarea vinurilor (din baza de date Wine) folosind reteaua MLP.

Baza de date *Wine* contine 3 tipuri de vinuri, fiecare avand cate 13 parametri:

1. Alcohol 2. Malic acid 3. Ash 4. Alcalinity of ash 5. Magnesium 6. Total phenols

7. Flavanoids 8. Nonflavanoid phenols 9. Proanthocyanins 10. Color intensity

11. Hue 12. OD280/OD315 of diluted wines 13. Proline

Baza de date contine:

• in *Clasa1:* 59 de vectori; in *Clasa2:* 71 de vectori; in *Clasa3*: 48 de vectori.

Toti vectorii sunt 13-dimensionali.

Pentru a se vizualiza baza de date, deschideti fisierul *wine.data* salvat in folderul curent.

Vectorii din baza de date au fost impartiti in 2 loturi:

- *lot_antrenare*, continand 50% din vectorii fiecarei clase
- *lot_testare*, continand restul de 50% din vectorii fiecarei clase

Ambele loturi sunt etichetate, stiindu-se fiecare vector din ce clasa provine. Etichetarile sunt salvate in *target_antrenare* pentru *lot_antrenare* si *target_testare* pentru *lot_testare*.

Pentru clasificarea utilizand retele neuronale se va porni *Toolboxul* pentru *Retele Neuronale*. Din programul Matlab, din coltul stanga jos se apasa *Start/Toolboxes/Neural Network/NNtool*, ca in captura de mai jos.

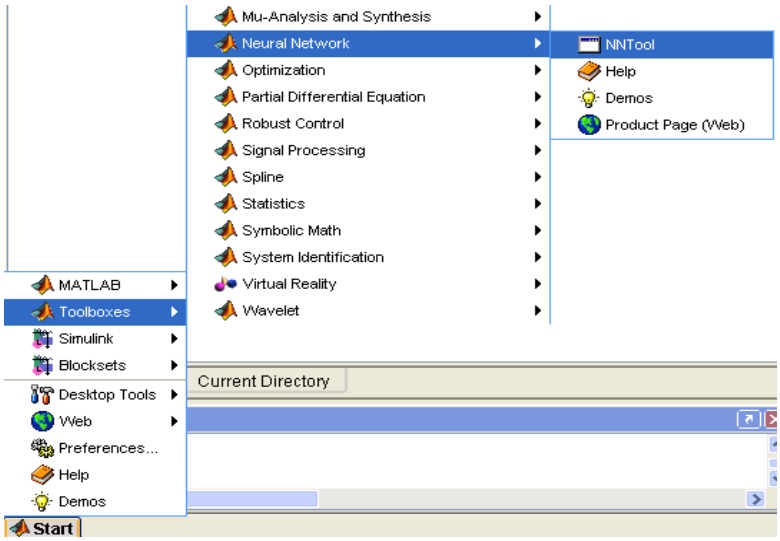


Figura 3.1. Toolboxul din Matlab

Se va deschide o fereastra precum cea din captura de mai jos:

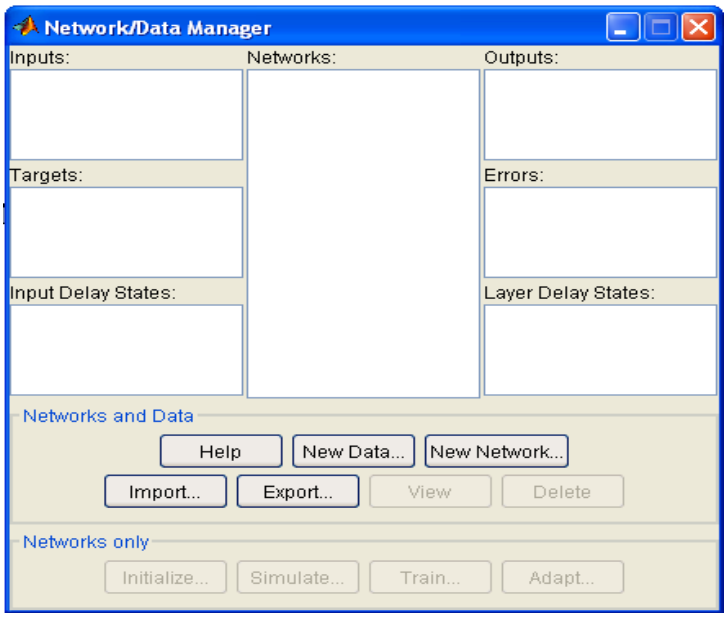


Figura 3.2. Toolboxul Neural Network din Matlab

Pentru implementarea retelei Perceptron Neliniar s-a folosit *Toolboxul* de *Neural Network* din *Matlab*. S-a folosit o retea de tipul *Feed forward backpropagation* cu 3 straturi.

Stratul de intrare are 13 neuroni (dimensiunea vectorilor = 13) iar stratul de iesire are 3 neuroni (sunt 3 clase de vinuri). Neuronii din stratul de iesire sunt de forma:

[1 0 0] ➔ vectorul apartine clasei 1

[0 1 0] ➔ vectorul apartine clasei 2

[0 0 1] ➔ vectorul apartine clasei 3

Parametrii ce vor fi variati in timpul lucrarii sunt:

- numarul neuronilor din stratul intermediar, functia de invatare, numarul de epoci

In continuare se va proceda astfel:

1. se va importa fisierul cu lotul de vectori pentru antrenare *lot_antrenare* apasand butonul *Import* si apoi:

• la sectiunea *Source* se va selecta *Import/Load from disk file/Browse/L4/Wine_Data*

• la sectiunea *Select a variable* se va selecta *lot_antrenare*

• la sectiunea *Destination* se va selecta *Inputs*

2. se va importa fisierul cu etichetari *target_antrenare* apasand butonul *Import* si apoi:

• la sectiunea *Select a variable* se va selecta *target_antrenare*

• la sectiunea *Destination* se va selecta *Targets*

3. se va crea apoi o retea folosind butonul *New Network* si apoi:

• *Network Name*: sa va alege un nume al retelei, de exemplu: *perceptron*

• *Network Type*: tipul retelei. Se va alege *Feed-forward backprop*

• *Input ranges:* se va selecta *Get from input: lot_antrenare*

• *Number of layers:* se vor selecta 2 straturi (aceste 2 straturi sunt cel intermediar si cel de iesire. Stratul formal de intrare il adauga el automat).

• *Properties for Layer1*: Proprietatile stratului intermediar

Number of neurons: **10**

Transfer Function: TANSIG

• *Properties for Layer2*: Proprietatile stratului de iesire

Number of neurons: **3**

Transfer Function: TANSIG

Se va apasa *Create* si in acest moment reteaua este configurata.

4. Pentru a vedea cum arata reteaua, selectati mai intai numele retelei si apoi dati *View*.

O sa va apara urmatoarea fereastra:

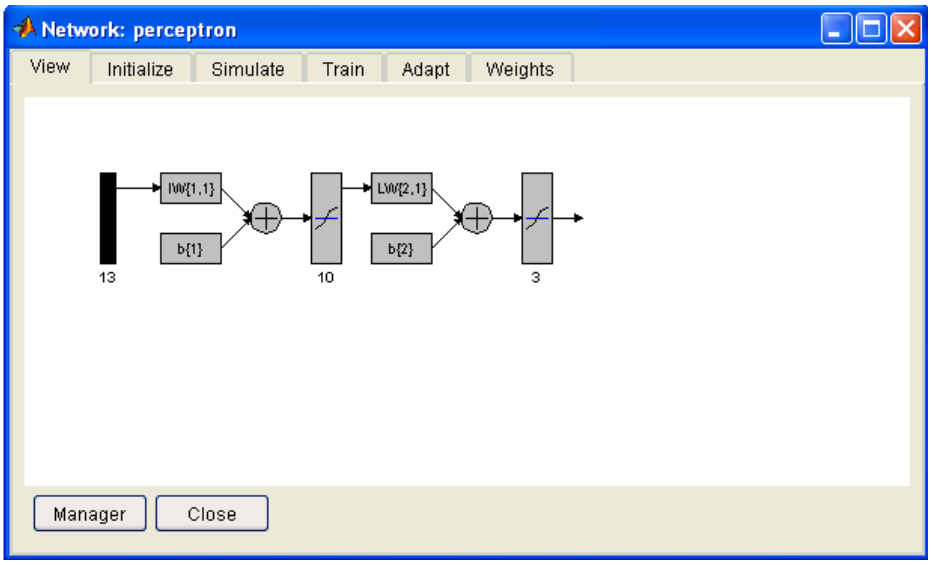


Figura 3.3. Reteaua MLP

5. Pentru a antrena reteaua apasati *Train* si apoi:

• *Training Info/ Training Data:*

- *Inputs*: selectati *lot_antrenare*
- *Targets*: selectati *target_antrenare*

• *Training Parameters:*

- *epoch:* selectati numarul de epoci egal cu **40**

Puteti antrena acum reteaua apasand *Train Network*

Se va afisa un grafic al evolutiei *Mean Square Error (MSE)* in functie de epoca.

Va veti nota performanta finala.

6. Pentru a testa reteaua, trebuie mai intai sa importati lotul de test si cel cu iesirile asociate lui si anume *lot_testare si target_testare*. Veti proceda la fel ca si cu lotul de vectori pentru antrenare, apasand butonul *Import* si apoi:

- la sectiunea *Select a variable* se va selecta *lot_testare*
- la sectiunea *Destination* se va selecta *Inputs*
- la sectiunea *Select a variable* se va selecta *target_testare*
- la sectiunea *Destination* se va selecta *Targets*

7. Apasati *Simulate* si apoi:

• *Simulation Data/Inputs:* selectati *lot_testare*

• *Simulation results/Outputs*: denumiti fisierul in care vor fi salvate iesirile. De exemplu: *perceptron_outputs_sim*

8. Exportati apoi fisierul cu iesirile calculate si cel cu iesirile dorite (*perceptron_outputs_sim si target_testare*).

Pentru a vedea scorul de recunoastere scrieti in *Matlab* in *Command Window* linia de cod:

```
[val ,pos] = max(perceptron_outputs_sim,[],1);
classnumber = sum(target_testare.*repmat([1;2;3],1,length(target_testare)), 1 );
scor = length(find(pos==classnumber))/length(classnumber)*100
```

Reglare parametri

• Se va reconfigura reteaua si se vor relua toti pasii descrisi anterior pentru numarul de neuroni pe stratul intermediar egal cu 20, 30 si apoi 40, ceilalti parametri ramanand neschimbati. Se va nota scorul de recunoastere pentru fiecare caz in parte.

• Pentru 30 neuroni pe stratul intermediar se va relua experimentul modificand de aceasta data numarul de epoci la 20, 30, 50 si 100

• Pentru 20 neuroni pe stratul intermediar si 75 de epoci se vor alege alte 2 functii de invatare si se va nota scorul de recunoastere

Se va reprezenta grafic evolutia scorului de clasificare corecta in functie de numarul de neuroni de pe stratul intermediar.

3.2. Predictia afluxului folosind reteaua MLP

Obiectiv: optimizarea parametrilor reţelelor de tip Perceptron, utilizate în lucrarea „*Data Fusion and Neural Networks for Disaster Forecasting: Flood Prediction Case*" în sensul minimizării erorii medii pătratice şi a maximizării corelaţiei.

Se vor varia numărul neuronilor din stratul intermediar (*L)* între 3 şi 6 şi numărul epocilor de antrenare (*E)* între 3 şi 15. Se vor face câte două grafice şi tabelele aferente pentru fiecare *L*:

- eroarea medie pătratică în funcţie de numărul epocilor
- corelaţia în funcţie de numărul epocilor.

Experimentul se poate realiza cu ajutorul funcţiei *prezicator* din Matlab care are următoarea sintaxă:

[yid,y,rms,corel] = prezicator;

Unde:

- *yid* reprezintă ieşirile ideale
- *y* sunt reşirile reale ale sistemului
- *rms* este eroarea medie pătratică
- *corel* – corelaţia.

Valorile coeficienţilor care împart intervalul în 3 sunt $\alpha = 0.76$ si $\beta = 1.55$.

Bibliografie

[1] V.Neagoe, O. Stănăşilă – Recunoasterea formelor si retele neurale – algoritmi fundamentali, Ed. Matrix Rom, Bucureşti, 1998.

[2] V. Neagoe, C. Tudoran, G. Strugaru, *Data Fusion and Neural Networks for Disaster Forecasting: Flood Prediction Case*

Data Fusion and Neural Networks for Disaster Forecasting: Flood Prediction Case

Prof. Dr. Victor-Emil Neagoe, Mr. Cristian Tudoran and Mr. Gabriel Strugaru
Faculty of Electronics, Telecommunications, and Information Technology
'Politehnica' University of Bucharest, P.O. Box 16-37, RO-062510, Bucharest
Romania

victoremil@gmail.com

ABSTRACT

A model of Adaptive Data Fusion Reservoir Inflow Forecasting using Concurrent Neural Networks (ADAFIFCON) is presented. It uses a fusion of previous rainfall and reservoir inflow data. The system consists of three backpropagation neural networks. Each neural module is trained to estimate a specific class of data dynamics: low, medium and high gradients. The decision fusion module uses a concurrent strategy. The model is applied to forecast the reservoir inflow for St-Jean Lake, Quebec, Canada. The method may be applied for disaster prediction and management for NATO (Science for Peace) Projects.

1.0 INTRODUCTION

Multisensor data fusion is an emerging technology drawn from artificial intelligence, pattern recognition, statistical estimation, and other areas. Fusion multisensor data has significant advantages over simple source data, obtaining a more accurate estimate of a physical phenomenon. Data fusion provides new modelling opportunities in other areas of the physical and social sciences, which includes geographical and environmental research.

In hydrological research, a significant effort has been concentrated to river flow prediction task. Flash floods are dangerous phenomena, which have produced in the past important economic losses and in some cases, life losses. A flood warning systems is a technical way to reduce such risks. If the hydrological system includes a dam equipped with control gates, improved criteria for gates operation during the flood can be assessed. There have been many recent papers and contributions regarding the applications of backpropagation neural networks (BPNN) for river discharge (or reservoir inflow) forecasting.

We further present a model of **A**daptive **Da**ta **F**usion Reservoir **I**nflow **F**orecasting using **Co**ncurrent **N**eural **N**etworks (ADAFIFCON). It uses a fusion of rainfall and inflow data (previous samples of rainfall and reservoir inflow data). This multi-system consists of a set of three concurrent backpropagation neural networks, corresponding to the three classes of rainfall sample gradients: low, medium and high. The model is applied for the reservoir of St-Jean Lake, Quebec, Canada.

Neagoe, V.-E.; Tudoran, C.; Strugaru, G. (2006) Data Fusion and Neural Networks for Disaster Forecasting: Flood Prediction Case. In *Emerging and Future Technologies for Space Based Operations Support to NATO Military Operations* (pp. P6-1 – P6-6). Meeting Proceedings RTO-MP-RTB-SPSM-001, Poster 6. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int/abstracts.asp.

2.0 ADAPTIVE DATA FUSION RESERVOIR INFLOW FORECASTING WITH CONCURRENT NEURAL NETWORKS (ADAFIFCON)

We propose the data fusion system for reservoir inflow forecasting (ADAFIFCON) shown in Figure 1. It consists of a set of three concurrent backpropagation neural networks. Each neural module is designed to estimate a specific class of data dynamics: low, medium and respectively high gradients.

The input data which are amalgamated by the data fusion system corresponds to:

- the previous samples of the reservoir inflow:

  y[n-1], y[n-2], y[n-3], … y[n-q]

- the previous rainfall samples:

  x [n-1], x [n-2], … x [n-p]

The training set (consisting of rainfall data and corresponding inflow data) is divided before training into three time domains : D_L, D_M and D_H, The classification of a given sample pair { x[n] , y[n] } is given by the following decision rule based on the gradient of the rainfall adjacent samples :

$$0 < | x[n] - x[n-1] | \leq \alpha \quad \Rightarrow \quad \{x[n], y[n]\} \in D_L$$

$$\alpha < | x[n] - x[n-1] | \leq \beta \quad \Rightarrow \quad \{x[n], y[n]\} \in D_M$$

$$\beta < | x[n] - x[n-1] | \quad \Rightarrow \quad \{x[n], y[n]\} \in D_H \qquad (\text{with } 0 < \alpha < \beta),$$

where D_L, D_M and D_H are the time domains corresponding to the labels ***low***, ***medium*** and ***high***. Each neural network (L, M, or H) is trained using the samples of the corresponding domain (subset), characterized by its rainfall dynamics (D_L, D_M or D_H).

After training, the three neural modules (Figure 1) estimate in parallel the output task (reservoir inflow). The decision fusion module uses a concurrent strategy by choosing at each step the best fitting neural module. Namely, at each forecasting step one chooses the neural network which obtained the best estimation accuracy at the previous step.

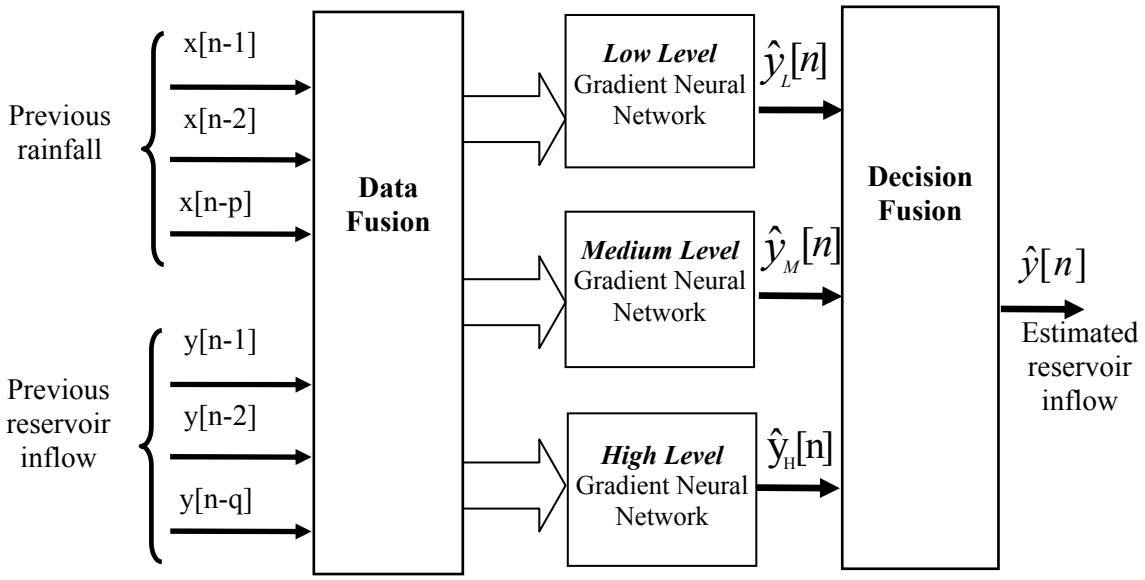


Figure 1: Adaptive data fusion reservoir inflow forecasting with concurrent neural networks (ADAFIFCON)

3.0 EXPERIMENTAL RESULTS

3.1 Hydrological and meteorological datasets

We considered the quarter monthly inflows data as well as corresponding rainfall data for the St-Jean Lake reservoir, Quebec, Canada. The data covered the period of the years 1953-1982.

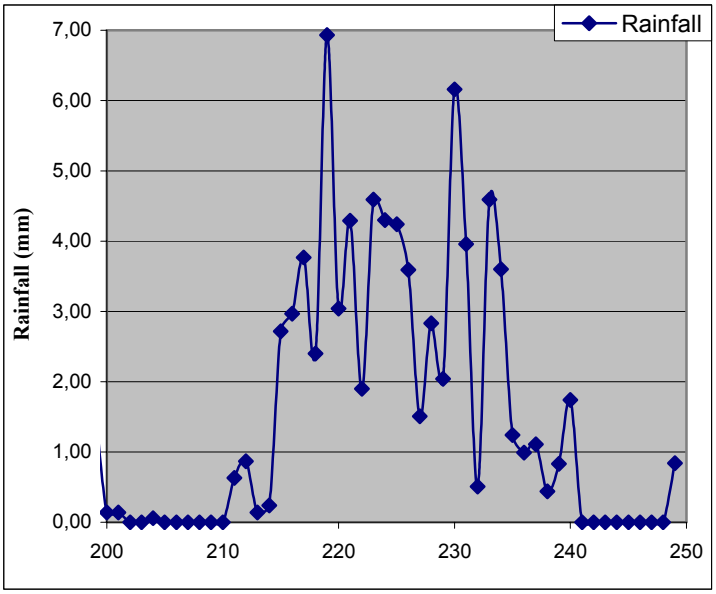


Figure 2: Observed rainfall

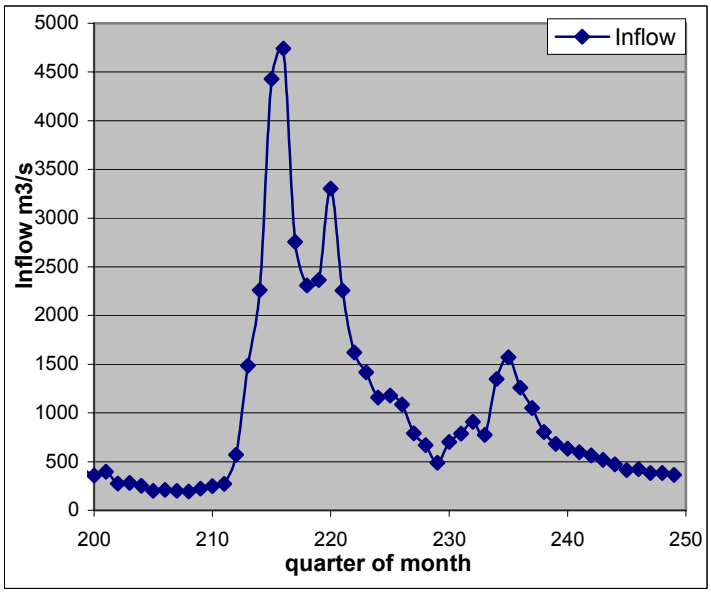


Figure 3: Observed reservoir inflow

In Figures 2 and 3 one can see an example of the observed rain-fall and the corresponding St-Jean Lake reservoir inflow. The sampling period is equal to a quarter of month.

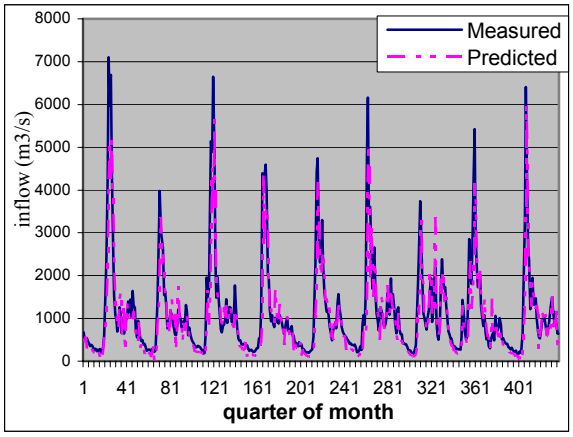


Figure 4: Measured vs. Predicted Inflows (m^3/s)

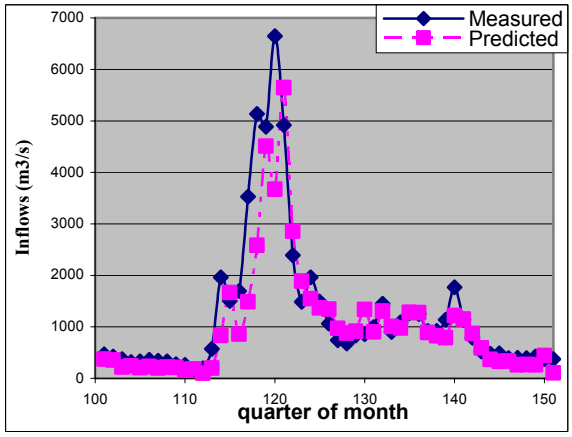


Figure 5: Measured vs. Predicted Inflows (m^3/s)

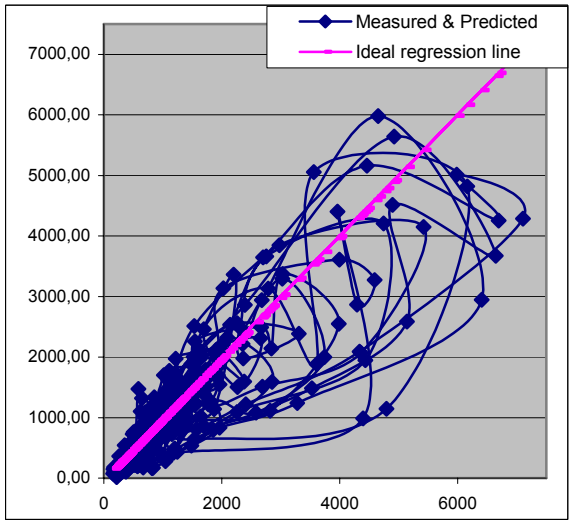


Figure 6: Measured and Predicted Inflows (in m^3/s)

3.2 Model evaluation

Each backpropagation neural module has a feedforward architecture with the following dimensions :

- number of input neurons equals the number of input data
- number of hidden neurons = 4
- one output neuron

We used a number of 100 training epochs. For performance evaluation, we consider the following measures:

- the root mean square error (Rmse) between the ideal and the forecasted reservoir inflow data
- the correlation coefficients of the ideal inflow sequence and the estimated one.

The experimental results are given in Table 1 and Figures 4-6.

Table 1: Performance evaluation of the ADAFIFCON model

	Inputs		Rmse (m^3/s)	Correlation
	Reservoir Inflow	Rainfall	Test.	Test.
ADAFIFCON (Multi-system) (3-nets)	y[n-1], y[n-2], y[n-3]	x[n], x[n-1], x[n-2]	0.596	0.8552
	y[n-1], y[n-2], y[n-3]	none	0.627	0.8532
	y[n-1], y[n-2], y[n-3]	x[n-1], x[n-2]	0.588	0.8585
	y[n-1], y[n-2]	x[n], x[n-1]	0.591	0.8558
	y[n-1], y[n-2]	x[n-1], x[n-2]	0.580	0.8629
	y[n-1], y[n-2]	none	0.616	0.8536
	y[n-1], y[n-2]	x[n-1]	0.592	0.8479
Mono-system (single net)	y[n-1], y[n-2], y[n-3]	x[n], x[n-1], x[n-2]	0.669	0.8506
	y[n-1], y[n-2], y[n-3]	none	0.626	0.8539
	y[n-1], y[n-2], y[n-3]	x[n-1], x[n-2]	0.661	0.8560
	y[n-1], y[n-2]	x[n], x[n-1]	0.679	0.8556
	y[n-1], y[n-2]	x[n-1], x[n-2]	0.648	0.8576
	y[n-1], y[n-2]	x[n-1]	0.653	0.8581
	y[n-1], y[n-2]	none	0.627	0.8560
Naive	y[n-1]	none	0.628	0.8570

The best result corresponds to the case of using 4 inputs: 2 previous rainfall samples {x[n-1], x [n-2]} and 2 previous inflow samples {y [n-1], y [n-2]}. The advantage of using the proposed multiple network system over a single network system is obvious. The case of naïve prediction is also considered for comparison.

The method may be applied for disaster prediction and management in NATO Science for Peace Projects.

4.0 REFERENCES

[1] Coulibaly, P., Anctil, F., Bobée, B., *Daily Reservoir Inflow Forecasting Using Artificial Neural Networks with Stopped Training Approach*, Journal of Hydrology, 230(3-4), 2000, pp. 244-257.

[2] Garcia-Bartual, R., *Short Term River Flood Forecasting with Neural Networks*, Proceedings of the First Biennial Meeting of the International Environmental Modelling and Software Society, June 2002, vol. 2, pp. 160--165.

[3] Hall, D. Linnas, J., *An Introduction to Multisensor Data Fusion*, Proceedings IEEE, Vol. 85, No.1, Jan. 1997, pp. 6-23.

[4] Maier, H., Dandy, C., *Neural Networks for Prediction and Forecasting of Water Resources Variables: A Review of Modelling Issues and Applications*, Environmental Modelling&Software, vol. 15, 2000, pp. 101-124.

[5] Neagoe, V., Ropot, A., *Concurrent Self-Organizing Maps for Pattern Classification*, Proc. First IEEE International Conference on Cognitive Informatics, ICCI 2002, 19-20 August 2002, Calgary, Alberta, Canada, pp. 304-312.

[6] Neagoe, V., Iatan, R., Iatan, I., *A Nonlinear Neuro-Fuzzy Model for Prediction of Daily Exchange Rates*, Proceedings of World Automation Congress, WAC'04, Seville, 3, (2004), IEEE Catalog 04EX832C.

[7] Neagoe, V., Tudoran, C., Strugaru, G., *A Neural Data Fusion Model for Hydrological Forecasting*, Proceedings of World Automation Congress (WAC'06), Budapest, Hungary, July 24-26, 2006, TSI Press, San Antonio, Texas, ISBN: 1-889335-26-6, IEEE Catalog number 06EX1486.

[8] See, L., Abrahart, R., *Multi-Model Data Fusion for Hydrological Forecasting*, Computers and Geosciences, Vol. 27, 2001, pp.987-994.