Laborator 3 - Multilayer Perceptron

1. Obiective

Obiectivul acestei lucrări este de a înțelege și de a dezvolta rețeaua neuronală Multilayer Perceptron utilizată pentru a clasifica cifrele de la 0 la 9 analizând imagini ce conțin cifrele scrise.

2. Bază teoretică

2.1. Perceptron simplu

Perceptronul este cea mai simplă formă a rețelelor neuronale utilizat pentru clasificarea pattern-urilor liniar separabile. Acesta conține un singur neuron cu sinapse ce pot fi ajustate prin modificarea ponderilor și a bias-ului. Prin acest algoritm, Rosenblatt a demonstrat că dacă două clase sunt liniar separabile, atunci perceptronul converge către o suprafață de decizie de forma unui hiperplan între cele două clase. Deoarece perceptronul are un singur neuron, acesta este limitat la clasificarea vectorilor în doar două clase. În Fig 3.1 este reprezentat Perceptronul simplu.

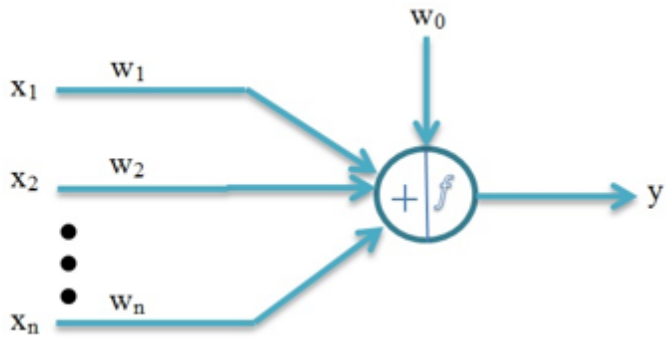

Fig 3.1 – Reprezentarea Perceptronului simplu

Unde x sunt mărimile de intrare în neuron, n este dimensiunea vectorului de intrare (numărul de caracteristici), w_0 bias-ul, iar y este ieșirea neuronului.

Potențialul de activare al Percepronului poate fi calculat după formula:

$$net = \sum_{i=1}^{n} x_i w_i + w_0 \qquad (3.1)$$

Cele mai cunoscute funcții de activare ale Perceptronului sunt funcțiile sigmoidale, printre care se regăsesc funcțiile logistică și tangențială.

a) Funcția logistică se definește astfel:

$$f_{log}(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

b) Funcția tangențială se definește astfel:

$$f_{tanh}(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.3}$$

Spre deosebire de funcţia logistică, funcţia tangentă hiperbolică poate lua şi valori negative: $f_{tanh}(x)\ \epsilon\ [-1, 1]$.

Ieșirea neuronului este calculată aplicând potențialului de activare funcția de activare corespunzătoare.

$$y = f(a) \tag{3.4}$$

2.2. Multilayer Perceptron Feed-Forward Backpropagation

Multilayer Perceptron este o rețea neuronală de tip Feed-Forword alcătuită dintr-un număr de unități (neuroni) conectate prin ponderi.

Rețelele neuronale de tip Feed-Forward permit semnalului să meargă într-o singură direcție, de la intrare către ieșire.

Rețelele neuronale multistrat sunt alcătuite din unități, corespunzătoare neuronilor, ordonate în mai multe straturi. Primul strat este numit strat de intrare, ultimul este numit strat de ieșire, iar straturile intermediare poartă denumirea de straturi ascunse.

Stratul de intrare primește un vector de activare extern, pe care îl transmite mai departe următorului strat de neuroni cu ajutorul ponderilor, ce reprezintă conexiunile dintre straturi. Aceștia procesează informația primită și o transmit mai departe următorului strat. Mai exact spus, vectorul de intrare se propagă de-a lungul rețelei neuronale, determinând obținerea unui răspuns la ieșire, toată funcționarea rețelei bazându-se pe conexiunile dintre neuroni date de ponderi. Topologia unei astfel de rețele cu un singur strat ascuns este prezentată în Fig. 3.2.

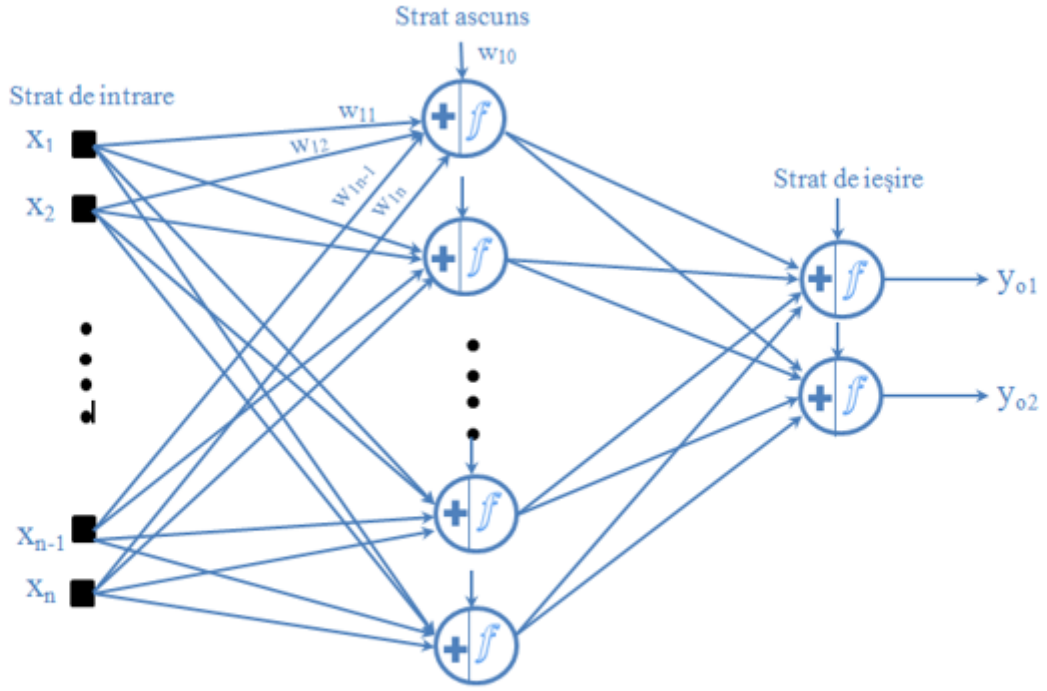

Fig 3.2– Topologia unei rețele neuronale de tip MLP cu un strat ascuns

Fiecare neuron *i* din rețea reprezintă o unitate de procesare simplă, care calculează funcția de activare a vectorului de intrare x_j, cu ajutorul ponderilor astfel:

$$net_i = \sum_{j=1}^{n} x_j w_{ij} + w_{i0} \tag{3.5}$$

Unde *n* reprezintă numărul de neuroni din stratul anterior neuronului *i*, w_{ij} reprezintă ponderile ce fac legătura dintre unitatea *j* și unitatea *i*, iar w_{i0} reprezintă bias-ul neuronului *i*. Pentru o reprezentare omogenă, w_{i0} este înlocuit cu o pondere care are ieșirea constantă 1. Acest lucru înseamnă că bias-ul poate fi considerat o pondere.

Sub formă vectorială, ecuația 3.5 devine:

$$\begin{bmatrix} net_1 \\ net_2 \\ \vdots \\ net_m \end{bmatrix} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1n} \\ w_{20} & w_{21} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m0} & w_{m1} & \cdots & w_{mn} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{3.6}$$

$$Net = W \cdot X \tag{3.7}$$

Unde m reprezintă numărul de neuroni din stratul ascuns, iar n numărul de caracteristici ale datelor de intrare.

Activarea unității *i* se realizează prin trecerea valorii obținute în urma calculării *net* printr-o funcție de activare neliniară, asemănător Perceptronului. Astfel, ieșirea y_i este calculată după formula:

$$y_i = f(net_i) = \frac{1}{1 + e^{-net_i}} \tag{3.8}$$

O proprietate importantă a acestei funcții este că poate fi derivată ușor, forma derivativă fiind:

$$\frac{\partial y_i}{\partial net_i} = f'(net_i) = y_i(1 - y_i) \tag{3.9}$$

Fiecare vector de intrare $\boldsymbol{X}$, este alcătuit dintr-o serie de caracteristici și are o ieșire din rețea corespunzătoare țintei D. După antrenarea rețelei, când o intrare x_i este prezentată, rezultatul y_i al ieșirii din rețea ar trebui să fie egal cu vectorul dorit d_i. Distanța dintre țintă și vectorul de ieșire reprezintă eroarea rețelei, denumită și funcția cost, și se calculează astfel:

$$E = \frac{1}{2L}\sum_{j=1}^{L}\sum_{i=1}^{N}(d_i^j - y_i^j)^2 \tag{3.10}$$

Unde N este numărul de unități din stratul de ieșire și L numărul de vectori aplicați la intrare.

Împlinirea scopului presupune găsirea unui minim global E.

Algoritmul de propagare inversă a erorii, presupune modificarea ponderilor începând de la ieșire către intrare cu scopul de a minimiza eroarea rețelei, cu ajutorul algoritmului de gradient negativ (regula Widrow-Hoff).

Regula Widrow-Hoff reprezintă un algoritm de gradient negativ care are rolul de a minimiza eroarea medie pătratică prin ajustarea ponderilor. Este utilizat în general on-line, ponderile fiind modificate la fiecare iterație. Astfel, se presupune intrarea alcătuită dintr-un singur vector $\boldsymbol{X}$, de n caracteristici, intrare având ieșirea corectă notată cu D (D fiind un scalar), iar ponderile curente notate cu $\boldsymbol{W}$ ($\boldsymbol{W}$ fiind un vector de n valori). Atunci, eroarea este dată de formula:

$$E(\boldsymbol{W}) = (D - \boldsymbol{W}^T\boldsymbol{X})^2, \tag{3.11}$$

Această ecuație mai poate fi scrisă și sub formă de suma astfel:

$$E_l(W) = \frac{1}{2}\sum_{i=1}^{N}\left(d_i^l - y_i^l\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\left(d_i^l - f_i(\sum_{k=0}^{M} w_{ik}x_k^l)\right)^2 \tag{3.12}$$

Iar gradientul este dat de ecuația 3.13 și respectiv 3.14:

$$\nabla E = \langle\frac{\partial E}{\partial W_0} \dots \frac{\partial E}{\partial W_n}\rangle \tag{3.13}$$

$$\frac{\partial E}{\partial W_j} = 2(D - \boldsymbol{W}^T\boldsymbol{X})\boldsymbol{X}_j \tag{3.14}$$

Unde $j = \overline{1,n}$, n fiind numărul de intrări în rețea.

Deoarece trebuie specificat pasul, η, pentru gradientul negativ, valoarea 2 din formula 3.14 poate fi introdus în rata de învățare η. Ținând cont de faptul că gradientul negativ presupune avansarea în direcția -∇E, forma finală a algoritmului devine:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \eta(D - \boldsymbol{W}^T\boldsymbol{X})\boldsymbol{X} \tag{3.15}$$

Unde *W(t+1)* reprezintă ajustarea ponderii pentru următoarea iterație.

Această procedură de ajustare a ponderilor se mai numește și regula delta deoarece, în general, eroarea $D - W^T X$ se notează cu δ.

Ecuația 3.15 presupune adaptarea ponderilor la momentul *t+1* ținând cont de semnul diferenței dintre ieșirea corectă și ieșirea oferită de rețea. Astfel, dacă această diferență este negativă, adică dacă ieșirea dorită este mai mică decât ieșirea dată de rețea, atunci ponderea următoare se va obține prin scăderea din ponderea curentă a diferenței dintre cele două ieșiri înmulțită cu rata de învățare, ajustând în acest mod ponderea în sensul minimizării erorii.

În cazul rețelei neuronale multistrat, eroarea pătratică pentru un lot de învățare de dimensiunea L, este:

$$E(W) = \frac{1}{2}\frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{N_o}\left(d_{li} - f_2\left(\sum_{k=0}^{N_h} w_{ik}^{o} f_1\left(\sum_{j=0}^{N_i} w_{kj}^{h} x_{lj}\right)\right)\right) \tag{3.16}$$

În acest caz, δ_h și δ_o sunt:

$$\delta_{li}^{o} = (d_{li} - y_{li})y_{li}(1 - y_{li}) \tag{3.17}$$

$$\delta_{li}^{o} = y_{li}(1 - y_{li})\sum_{i=1}^{N_o} w_{ik}^{o}\delta_{li}^{o} \tag{3.18}$$

Una dintre problemele acestui algoritm presupune alegerea ratei de învățare corespunzătoare. O alegere bună depinde de forma funcției eroare. O rată de învățare mică va determina un timp mare de convergență, în timp ce o rată mare de învățare va duce la oscilații ale erorii, împiedicând scăderea acesteia sub o anumită valoare. De asemenea, se poate demonstra că, deși în anumite circumstanțe rețeaua converge către un minim local, nu se poate garanta că algoritmul va găsi minimul global al funcției eroare. În Fig 3.3 este evidențiată descreșterea erorii medii pătratice odată cu modificarea ponderilor, în sensul gradientului negativ.

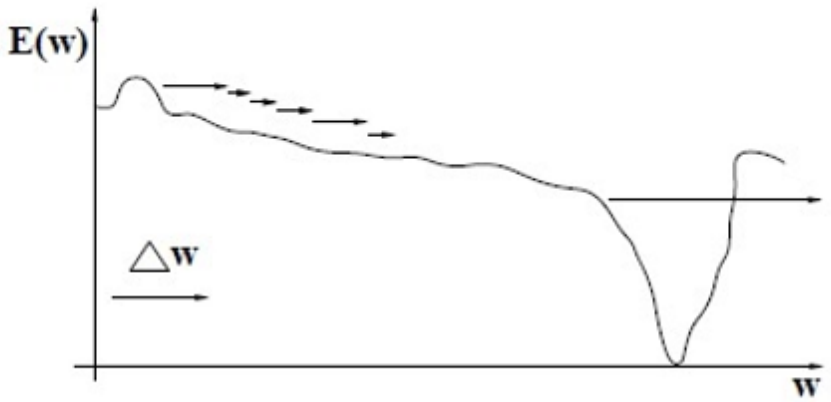

Fig 3.3 – Scăderea erorii medii pătratice în sensul gradientului negativ

2.3. Algoritmul Myltilayer Perceptron Backpropagation

În acest laborator, se va dezvolta algoritmul rețelei MLP sub formă secvențială (incrementală) prin parcurgerea următorilor pași:

1. Inițializarea aleatoare a ponderilor
2. Repetă

FOR l = 1,L DO

% Etapa FORWARD (propagarea înainte a vectorului de intrare)

% Calcularea potențialului de activare a neuronilor din stratul ascuns

$$net_{lk}^{h} = \sum_{j=0}^{N_i} w_{kj}^{h} x_{lj} \tag{3.19}$$

% Calcularea funcției de activare a neuronilor din stratul ascuns

$$y_{lk}^{h} = f_1\left(net_{lk}^{h}\right) \tag{3.20}$$

% Calcularea potențialului de activare a neuronilor din stratul de ieșire

$$net_{li}^{o} = \sum_{k=0}^{N_h} w_{ik}^{o} y_{lk}^{h} \tag{3.21}$$

% Calcularea funcției de activare a neuronilor din stratul de ieșire

$$y_{li} = f_2(net_{li}^{o}) \tag{3.22}$$

% Etapa BACKWARD (propagarea erorii)

% Calcularea erorii rețelei față de ieșirea dorită

$$\delta_{li}^{o} = f_2'(net_{li}^{o})(d_{li} - y_{li}) \tag{3.23}$$

%Calcularea erorii propagată către neuronii din stratul ascuns

$$\delta_{lk}^{h} = f_1'\left(net_{lk}^{h}\right) \sum_{i=1}^{N_o} w_{ik}^{o} \delta_{li}^{o} \tag{3.24}$$

% Etapa de ajustare a ponderilor

$$w_{kj}^{h} = w_{kj}^{h} + \eta \delta_{lk}^{h} x_{lj} \tag{3.25}$$

$$w_{ik}^{o} = w_{ik}^{o} + \eta \delta_{li}^{o} y_{lk}^{h} \tag{3.26}$$

END FOR

% Calcularea erorii

E = 0

FOR l=1,L DO

%Etapa FORWARD

$$net_{lk}^{h} = \sum_{j=0}^{N_i} w_{kj}^{h} x_{lj} \quad (3.27)$$

$$y_{lk}^{h} = f_1\left(net_{lk}^{h}\right) \quad (3.28)$$

$$net_{li}^{o} = \sum_{k=0}^{N_h} w_{ik}^{o} y_{lk}^{h} \quad (3.29)$$

$$y_{li} = f_2\left(net_{li}^{o}\right) \quad (3.30)$$

%Sumarea erorii

$$E = E + \sum_{i=1}^{L} (d_{li} - y_{li})^2 \quad (3.31)$$

END FOR

$$E = E/(2L) \quad (3.32)$$

$$p = p+1 \quad (3.33)$$

CAT TIMP $p > p_{max}$ SAU $E < E^*$

3. Desfășurarea lucrării

În această lucrare se urmărește determinarea ieșirii unei rețele neuronale de tip Multilayer Perceptron pentru fiecare vector dintr-un set de date ce conține imagini cu cifre scrise de la 0 la 9, utilizând ponderile rezultate în urma antrenării rețelei. De asemenea, se dorește determinarea performanței rețelei neuronale prin calcularea erorii medii pătratice pe setul de antrenare și prin calcularea erorii de clasificare a acesteia pe setul de testare.

3.1. Încărcarea setului de date și stabilirea parametrilor utili

Setul de date utilizat pentru antrenarea rețelei neuronale este alcătuit din imagini ce conțin cifre scrise de la 0 la 9. Imaginile din setul de antrenare au o dimensiune de 20x20 pixeli și sunt reprezentate de imagini binare (conțin doar valori de 0 și 1). Dacă se urmăresc imagini ce aparțin aceleiași clase, se poate observa că aceeași cifră poate scrisă în mod diferit, în funcție de tipul de scriere al fiecărui subiect. Din acest motiv este necesară antrenarea unei rețele capabilă să învețe diferitele tipuri de scriere pentru fiecare cifră și să poată clasifica aceste cifre independent de subiect.

Toate imaginile din setul de date utilizat pentru antrenare au fost liniarizate și salvate în matricea **X**, unde fiecare linie din matrice reprezentă câte un vector de intrare, iar pe fiecare coloană a matricei este prezentă o caracteristică a vectorului respectiv.

Pentru fiecare vector de intrare, $\mathbf{X}_j$, există un corespondent d_j care reprezintă clasa de apartenență a vectorului respectiv.

Utilizând acest set de date, a fost antrenată o rețea neuronală de tip MLP cu un singur strat ascuns, având 25 de neuroni în stratul ascuns. După antrenare, au fost salvate ponderile din stratul ascuns în variabila **Wh** și ponderile din stratul de ieșire în variabila **Wo**.

Pentru încărcarea acestor date de intrare, se va folosi funcția load, sub forma:

```
load('lab04_data01.mat');
```

Opțional:

Să se reprezinte o imagine din setul de date, **X**, cu ajutorul uneia dintre funcțile: ***imshow*** sau ***imagesc***.

Observație: Ne amintim că fiecare imagine din setul de date a fost liniarizată, motiv pentru care trebuie realizată transformarea inversă, în formă bidimensională, a imaginii. Pentru a realiza acest lucru, se poate utiliza funcția ***reshape***, sau se poate implementa această transformare.

Obiectiv1: Stabilirea parametrilor utilizați

1.1. Inițializarea parametrilor cunoscuți
- nr. neuronilor din stratul ascuns: 25
- nr. straturilor ascunse: 1

1.2. Se dorește determinarea următorilor parametrii:
- numărul neuronilor din stratul de intrare
- numărul neuronilor din stratul de ieșire
- numărul vectorilor de intrare

Obiectiv2: Stabilirea ieșirii din rețea pentru fiecare vector de intrare.

2.1. Pentru fiecare vector $\mathbf{X}_j$ din setul de date se dorește stabilirea ieșirii din stratul ascuns, y_{jh}:

- se calculează net_{jh} pentru vectorul respectiv conform ecuației:

$$net_{jh} = w_{0h} + \sum_{i=1}^{N_i} w_{ih} x_{ji} \tag{3.1}$$

- se determină ieșirea din fiecare neuron din stratul ascuns, aplicând funcția de activare:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

$$y_{jh} = f(net_{jh}) \tag{3.3}$$

2.2. Pentru vectorul $\mathbf{X}_j$, se determină valoarea obținută în urma introducerii lui y_{jh} în stratul de ieșire.

- se calculează netjo pentru vetorul de intrare conform ecuației:

$$net_{jo} = w_{0o} + \sum_{h=1}^{N_h} w_{ho} y_{jh} \tag{3.4}$$

- se determină ieșirea din fiecare neuron din stratul de ieșire, aplicând funcția de activare:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

$$y_{jo} = f(net_{jo}) \tag{3.6}$$

Obiectiv3: Calcularea erorii medii pătratice pe setul de date de antrenare.

Se dorește determinarea performanței rețelei neuronale calculând eroarea medie pătratică pe setul de date, utilizând următoarea ecuație:

$$E = \frac{1}{2L} \sum_{j=1}^{L} \sum_{o=1}^{No} \left(d_{jo} - y_{jo}\right)^2 \tag{3.7}$$

Obiectiv4: Testarea rețelei neuronale utilizând setul de date de test, ce conține imagini noi pentru rețeaua neuronală.

4.1. Încărcarea setului de date de test

```
load('lab04_data02_01.mat');
```

4.2. Determinarea ieșirii din rețeaua neuronală a fiecărui vector de test.

4.3. Stabilirea apartenenței la clasă pentru fiecare vector de test.

Pentru a determina clasa de apartenență, se alege din stratul de ieșire neuronul care prezintă cel mai mare răspuns pentru vectorul respectiv.

4.4. Calcularea erorii de clasificare

Se dorește determinarea numărului de vectori din setul de test clasificați incorect, în raport cu numărul de neuroni de test. Această eroare poate fi calculată utilizând următoarea ecuație:

$$E_{cl} = \frac{Nr.\,vectori\ clasificați\ greșit}{Nr.\,total\ vectori} \tag{3.8}$$