Laborator 4 - Multilayer Perceptron

1 Obiective

Obiectivul acestei lucrări este de a înțelege și de a antrena o rețea neuronală de tipul Multilayer Perceptron. Se urmărește ca la finalul laboratorului studenții să:

- Știe să interpreteze un set de date, să știe care sunt trăsăturile, câți vectori de trăsături conține setul de date.
- Care sunt intrările în rețea și care sunt ieșirile. Cum se reprezintă ieșirile din rețea.
- Cum se face inițializarea ponderilor.
- Ce reprezintă ponderile unei rețele neuronale de tipul MLP și cum sunt reținute în Octave.
- Cum se calculează ieșirea din rețea pentru un set de date, cum se determină clasa de apartenență a unui vector de intrare.
- Ce reprezintă eroarea medie pătratică si eroarea de clasificare, care este diferența dintre cele două erori.
- Posibilitatea calculării ieșirii din rețea a unui vector de intrare având o funcție de activare diferită de cea prezentată în platformă.
- Împărțirea setului de date în set de antrenare și set de testare și semnificația acestor seturi de date.

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 2.1 este reprezentat Perceptronul simplu.

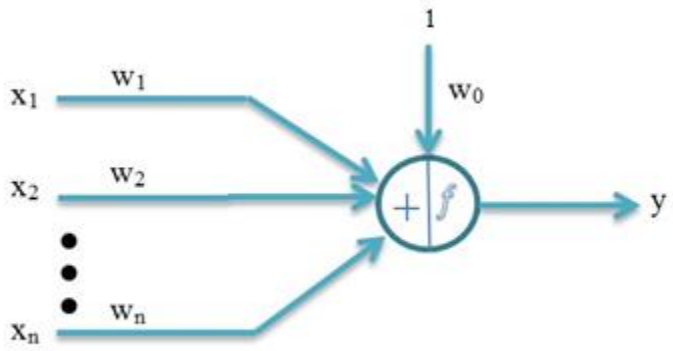

Fig 2.1 – Reprezentarea Perceptronului simplu

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:

$$f(net) = f(\sum_{i=1}^{n} x_i w_i + w_0) \tag{2.1}$$

Cea mai cunoscută funcție de activare a Perceptronului este funcția liniară, urmată de sunt funcțiile sigmoidale, printre care se regăsesc funcțiile logistică și tangențială.

a) Funcția logistică se definește astfel:

$$f_{log}(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

b) Funcția tangențială se definește astfel:

$$f_{tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

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

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

$$y = f(net) \tag{2.4}$$

2.2 Multilayer Perceptron

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. 2.2.

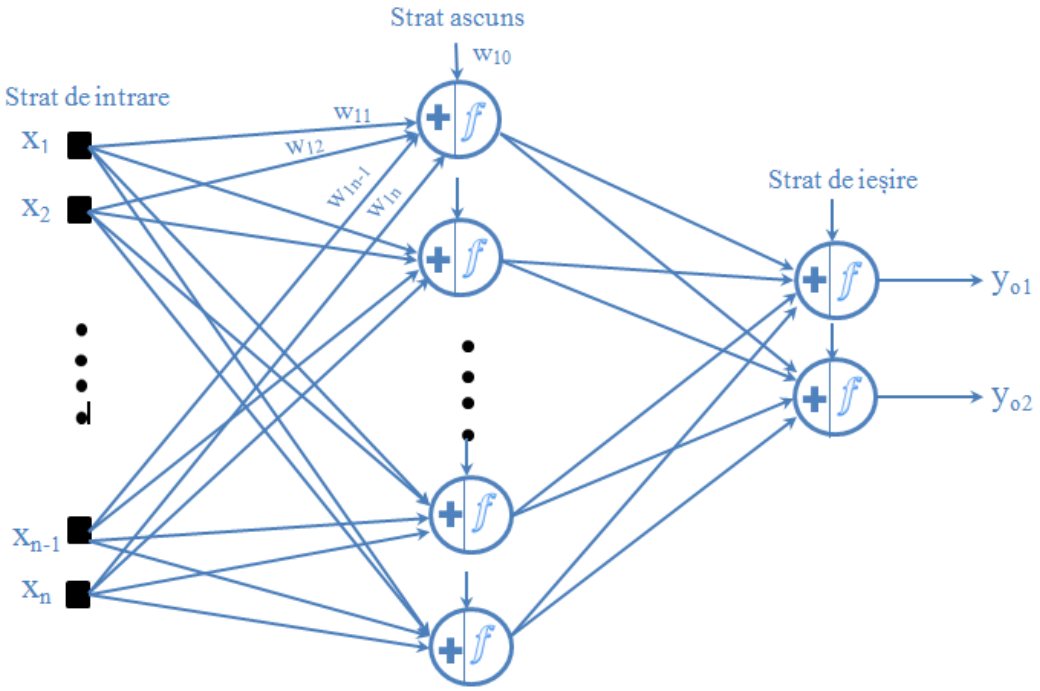

Fig 2.2– Topologia unei rețele neuronale de tip MLP cu un strat ascuns

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:

$$f(net_i) = f(\sum_{j=1}^{n} x_j w_{ij} + w_{i0}) \tag{2.5}$$

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 intrarea constantă 1. Acest lucru înseamnă că bias-ul poate fi considerat o pondere.

Sub formă vectorială, ecuația 2.5 devine:

$$\begin{bmatrix} net_1 \\ net_2 \\ \vdots \\ net_m \end{bmatrix} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1n} \\ w_{20} & w_{21} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m0} & w_{m1} & \cdots & w_{mn} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{2.6}$$

$$Net = W \cdot X \tag{2.7}$$

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. Astfel, ieșirea y_i este calculată după formula:

$$y_i = f(net_i) = \frac{1}{1 + e^{-net_i}} \tag{2.8}$$

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

$$\frac{\partial y_i}{\partial net_i} = f'(net_i) = y_i(1 - y_i) \tag{2.9}$$

Fiecare vector de intrare X_i, este alcătuit dintr-o serie de caracteristici și are o ieșire din rețea corespunzătoare țintei d_i. 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:

$$E = \frac{1}{2L}\sum_{j=1}^{L}\sum_{i=1}^{N}(d_i^j - y_i^j)^2 \tag{2.10}$$

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).

2.3 Algoritmul Myltilayer Perceptron Backpropagation

Pentru dezvoltarea algoritmului rețelei MLP sub formă secvențială (incrementală) se parcurg următorii 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

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

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

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

$$y_{li} = f_2(net_{li}^{o}) \tag{3.30}$$

%Sumarea erorii

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

END FOR

$$E = E/(2L) \tag{3.32}$$

$$p = p+1 \tag{3.33}$$

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

3 Desfășurarea lucrării

Exercitiu 3.1: Să se creeze functia ***plotData*** cu ajutorul căreia să poată fi reprezentate grafic trăsăturile din setul de date prezent în fișierul *lab04_data01.mat* din folderul *RNSF_Lab4*. Funcția va primi ca parametrii de intrare setul de date *X* și clasele de apartenență prezente în vectorul *D*.

Rezolvare:

I. Crearea funcției

Pasul 1: Se deschide o nouă funcție utilizând comanda *File →New Function.* Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea ***plotData***.
Pasul 2: Se va modifica fișierul deschis astfel: în locul numelui implicit Untitled, se va scrie numele funcției: ***plotData***. În locul argumentelor de intrare, input_args, se va scrie numele setului de date, *X* și a matricei ce conține ieșirile dorite, *D*. În locul argumentelor de ieșire nu se va scrie nimic, deoarece funcția nu va avea parametru de ieșire, ci doar va reprezenta graficul.

```
function plotData( x, d)
end
```

Pasul 3: În interiorul corpului funcției se vor reprezenta grafic trăsăturile setului de date *X*, trăsătura de pe coloana 2 în funcție de trăsătura de pe coloana 1 cu marker de tip „x” și culorile: roșie pentru clasa 1 și albastră pentru clasa 2. Pentru facilitarea găsirii vectorilor din cele două clase, se poate utiliza funcția ***find***. Această funcție a fost explicată detaliat în platforma laboratorului 2, RNSF_Lab2, Secțiunea 3, Exercițiul 3.5.

```
function plotData( x, d )
figure, plot(x(find(d==1),1),x(find(d==1),2),'xr')
hold on, plot(x(find(d==2),1),x(find(d==2),2),'xb')
hold off
end
```

II. Apelarea funcției

Pasul 1: Se deschide un script nou utilizând comanda *File →New.* Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea Exercitiu03_1.
Pasul 2: Se va încărca setul de date, *lab04_data01.mat* utilizând funcția ***load***.

```
clear all, close all, clc

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

Pasul 3: Se va apela funcția ***plotData***

```
plotData( X, D )
```

Exercitiu 3.2: Să se antreneze o rețea neuronală de tipul Multilayer Perceptron cu ajutorul funcțiilor ***mlp*** și ***mlptrain*** prezente în folder-ul RNSF_Lab4 →netToolbox, care să clasifice setul de date ***lab04_data01.mat***. Setul de date conține setul de antrenare *X*, care prezintă pe fiecare linie un vector de intrare, iar pe fiecare coloană este stocată o trăsătură și vectorul *D*, care conține clasele de apartenență pentru fiecare vector de intrare din *X*. Se vor folosi 25 de neuroni în stratul ascuns.

Rezolvare:
Pasul 1: Se deschide un script nou utilizând comanda *File →New*. Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea *„Exercitiu03_2.m”*.
Pasul 2: Se va încărca setul de date, *lab04_data01.mat* utilizând funcția ***load***.

```
clear all, close all, clc

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

Pasul 2: Se vor stabili parametrii cunoscuți, necesari rezolvării problemei: numărul vectorilor de intrare, *nvect*, și numărul neuronilor de intrare, *nin*, din rețea pot fi determinate din setul de date *X* utilizând funcția ***size***, iar numărul de neuroni din stratul ascuns, *nhidden*, va fi setat la 25, numărul neuronilor din stratul de ieșire, *nout*, poate fi extras din vectorul *D*, utilizând funcțiile ***length*** și ***unique*** (funcția ***unique*** returnează toate valorile distincte care se găsesc într-un vector sau matrice), iar funcția de activare a neuronilor din stratul de ieșire, *outfunc*, va fi setaată ca funcție logistică.

```
[nvect, nin] = size(X);
nhidden = 25;
nout = length(unique(D));
outfunc = 'logistic';
```

Pasul 3: Se va transforma vectorul *D* într-o matrice, *Out*, de dimensiune *[nvect x nout]*, astfel încât să se cunoască care dintre neuronii de ieșire vor trebui să se activeze pentru a oferi răspunsul corect. Acest lucru poate fi realizat astfel: se presupune că primul neuron din stratul de ieșire răspunde pentru clasa cu eticheta 1, al doilea neuron din stratul de ieșire răspunde pentru clasa cu eticheta 2, etc. Pentru ca un neuron să se activeze, acesta trebuie să fie setat pe 1, iar restul neuronilor să fie setați pe 0. De exemplu, *D(1)* are eticheta 1, astfel că matricea *Out*, pe prima linie și prima coloană va avea valoarea 1, iar pe prima linie și a doua coloană va avea valoarea 0 (cunoscând că setul de date conține doar două clase). Însă, pe linia 863 a vectorului *D*, *D(863)*, avem eticheta 2. Motiv pentru care în matricea *Out*, pe linia 863, *Out(863,1)*, și prima coloană vom avea valoarea 0, iar pe cea de-a doua coloană, *Out(863,2)*, vom seta neuronul la 1.

```
Out = zeros(nvect,nout);

for i=1:nvect
    Out(i,D(i)) = 1;
end
```

Pasul 4: Se va introduce folderul netToolbox la path-urile de căutare ale Octavului.

```
addpath('path\netToolbox')
```

'path' reprezintă calea către netToolbox.

Pasul 5: Stabilirea arhitecturii și parametrii necesari rețelei. Se va apela funcția ***mlp***, cu parametrii de intrare: *nin*, *nlayer*, *nhidden*, *nout*, *outfunc*, *alpha*. Unde *nin* reprezintă numărul de neuroni din stratul de intrare, *nlayer* reprezintă numărul de straturi ascunse, *nhidden* reprezintă numărul de neuroni din stratul ascuns, *nout* reprezintă numărul de neuroni din stratul de ieșire, iar *outfunc* reprezintă funcția de activare a neuronilor din stratul de ieșire. Parametrul de ieșire a funcției, denumit *net*, va fi reprezentat de o structură ce conține câmpurile:

- type = 'mlp'
- nin = numărul neuronilor de intrare
- nlayer = numărul de straturi ascunse
- nhidden = numărul neuronilor din stratul ascuns
- nout = numărul neuronilor din stratul de ieșire
- nwts = numărul total de ponderi și bias-uri
- outfn = șir de caractere ce descrie funcția de activare a neuronilor din stratul de ieșire
- w_h = ponderile din stratul ascuns, celulă de dimensiune {nlayer, 1}, unde:
  - w_h{1,1} are dimesiune *[nhidden(1) x nin]*
  - w_h{2,1} are dimesiune *[nhidden(2) x nhidden(1)]*
  - ...
  - w_h{nlayer, 1} are dimensiune *[nhidden(nlayer) x nhidden(nlayer-1)]*
- b_h = bias-urile neuronilor din stratul ascuns, celulă de dimensiune {nlayer, 1} unde:
  - b_h{1,1} are dimensiune *[1 x nhiddden(1)]*
  - b_h{2,1} are dimensiune *[1 x nhiddden(2)]*
  - ...
  - b_h{nlayer,1} are dimensiune *[1 x nhiddden(nlayer)]*
- w_o = ponderile neuronilor din stratul de ieșire, dimensiune *[nout x nhidden]*
- b_o = biasu-urile neuronilor din stratul de ieșire, dimensiune *[1 x nout]*

Câmpurile se pot apela utilizând operatorul ***punct „.”***, exemplu: *net.type*, *net.nout*.

De asemenea, este important de menționat faptul că funcția ***mlp*** inițializează aleator ponderile.

```
net = mlp(nin, nhidden, nout, outfunc);
```

Pasul 6: Pentru a putea verifica rezultatul obținut, nu vom începe antrenarea rețelei cu ponderile aleatoare obținute la Pasul 7, ci vom încărca o altă structură net (ce conține ponderile semi-antrenate), astfel încat după un numar *I* de iterații, să putem avea toți aproximativ aceleași valori de ieșire, aceleași valori ale erorii, etc.

De asemenea, vom reprezenta grafic și setul de date împreună cu suprafața de separație pentru a observa cum se modifică suprafața după antrenare.

```
load('net_init01.mat');
plotData(X, D )
hold on,
plot_boundary(X, net), hold off
```

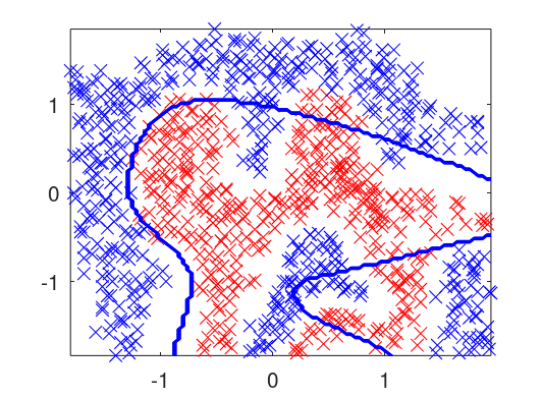

Fig. 3.1 Suprafața de separație înainte de antrenare

Pasul 7: Antrenarea rețelei. Se va antrena rețeaua neuronala MLP utilizând funcția ***mlptrain***. Parametrii de intrare ai funcției sunt: *net*, structura încărcată anterior la Pasul 5, *X*, setul de date de intrare, *Out*, matricea valorilor corespunzătoare ieșirii dorite din rețea și numărul maxim de interații ale rețelei, care va fi setat în acest caz 100. Parametrii de ieșire din funcția ***mlptrain*** sunt *net*, structura încărcată la Pasul 5 având ponderile obținute în urma antrenării și nu ponderile inițializate aleator și vectorul *cost* care reține valorile costului obținute la fiecare iterație a rețelei.

```
[net, cost] = mlptrain(net, X, Out, 300);
```

În continuare, vom afișa suprafața de separare și după antrenare pentru a vedea îmbunătățirile aduse clasificării.

```
plotData(X, D )
hold on,
plot_boundary (X, net), hold off
```

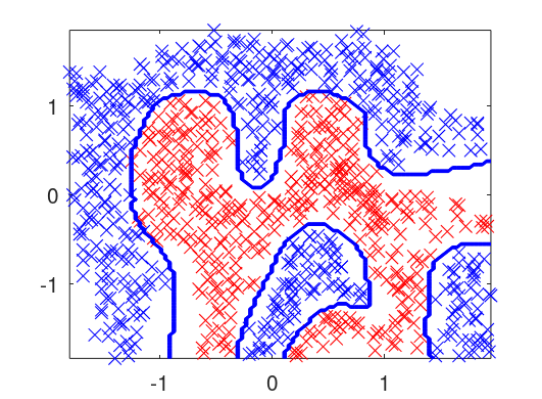

Fig. 3.2 Suprafața de separație după antrenare

Pasul 8: Plotarea erorii.

```
figure, plot(cost)
xlabel('Nr. iteratie')
ylabel('Cost')
```

În imaginea **Error! Reference source not found.** este reprezentată eroarea pentru fiecare iterație.

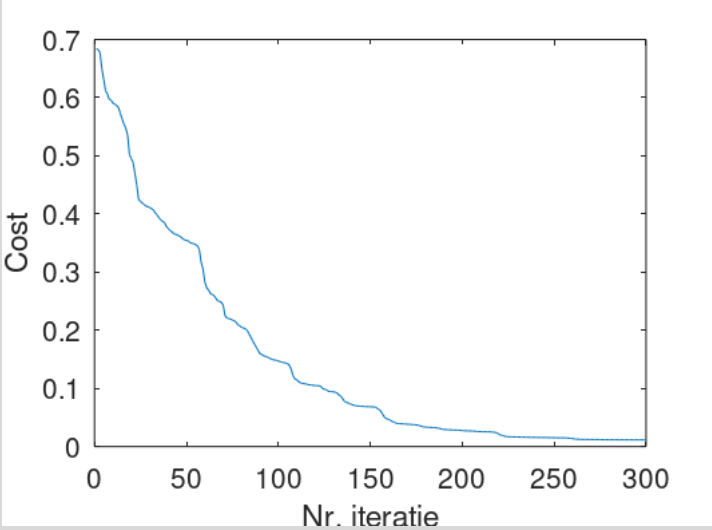

Fig. 3.3 – Costul rețelei la fiecare iterație

La iterația 300, costul va trebui să fie aproximativ: Cost: 1.144425e-02

Pasul 8: Salvarea structurii net.

```
save('net.mat','net')
```

Se va salva structura *net* pentru utilizarea acesteia în următoarele exerciții.

(5p) Exercițiu Punctat! *Exercitiu 3.3:* Să se definească funcția cu denumirea ***feedForwardLogistic*** care primește ca parametrii de intrare un set de date **X** și o structura **net** inițializată și oferă ca parametru de ieșire ieșirea din rețea pentru toți vectorii de intrare din setul de date **X**. Funcția de activare a neuronilor va fi funcția logistică* pentru toți neuronii din stratul ascuns și din stratul de ieșire.

Să se încarce funcția **(doar funcția definită în cerință)** pe Moodle la exercițiul cu numele **GR[Nr.Grupa]_RNSF_LAB04_ex1.** Exercițiul restricționează încărcarea altui fișier decât cel din cerință: un fișier cu extensia **.m** și denumirea ***feedForwardLogistic*** care conține declarația funcției:

```
function y = feedForwardLogistic(X, net)
% Corp Functie de completat
end
```

*funcția logistică:
$$f(u) = \frac{1}{1 + e^{-u}}$$

OBS: i) Acest fișier se poate găsi și în secțiunea Edit pe Moodle.
ii) Setul de date cu care va fi testat este setul "lab04_data01.mat" și structura net salvată anterior la *Exercițiul 3.2*.

Instrucțiuni opționale de rezolvare:
Pasul 1: Se deschide un script nou utilizând comanda *File →New*. Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea Exercitiu03_3.

Pasul 2: Se încarcă setul de date *lab04_data01.mat* și structura *net.mat* salvată anterior.

Pasul 3: Se calculează ieșirea din rețea. Pentru a obține ieșirea din rețea:

1. Inițial se calculează ieșirea din stratul ascuns pentru fiecare neuron (ecuația este caracteristică unui singur vector de intrare):
$$y_i^h = f\left(\sum_{j=1}^{nin} x_j w_{ij}^h + w_{i0}^h\right)$$
Unde, i reprezintă neuronul din stratul ascuns: i = $\overline{1, nhidden}$ (nhidden corespunde numărului de neuroni din straul ascuns), j reprezintă trăsătura: j = $\overline{1, nin}$ (nin reprezintă numărul de intrări în rețea, echivalent cu numărul de trăsături), x_j corespunde valorii j din vectorul de trăsături introdus în rețea, w_{ij}^h corespunde ponderii j pentru neuronul i din stratul ascuns, w_{i0}^h corespunde bias-ului neuronului i din stratul ascuns, iar f reprezintă funcția de activare a neuronilor din stratul ascuns. Ecuația funcției de activare specifică problemei pentru stratul ascuns este funcția logistică:
$$f(u) = \frac{1}{1 + e^{-u}}$$
În cazul în care rețeaua are mai multe straturi ascunse, ieșirile din stratul anterior vor fi intrare pentru stratul curent, ieșirea calculându-se asemănător.

2. Calcularea ieșirii pentru neuronii din stratul de ieșire (ecuația este caracteristică unui singur vector de intrare):
$$y_i^o = f\left(\sum_{j=1}^{nhidden} y_j^h w_{ij}^o + w_{i0}^o\right)$$
Unde i reprezintă neuronul din stratul de ieșire: i = $\overline{1, nout}$ (nout corespunde numărului de neuroni din straul de ieșire), j reprezintă intrarea în neuronul de ieșire: j = $\overline{1, nhidden}$ (nhidden reprezintă numărul de neuroni din stratul ascuns), y_j^h corespunde valorii j din vectorul obținut în urma caluclării ieșirilor din stratul ascuns, w_{ij}^o corespunde ponderii j pentru neuronul i din stratul de ieșire, w_{i0}^o corespunde bias-ului neuronului i din stratul

de ieșire, iar f reprezintă funcția de activare a neuronilor din stratul ascuns. Ecuația funcției de activare specifică problemei pentru stratul ascuns este funcția logistică:

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

Primii 5 vectori de ieșire vor fi de forma (sunt valori aproximative):

```
  8.8335e-01   1.1465e-01
  9.9295e-01   6.5610e-03
  1.7126e-06   1.0000e+00
  1.0000e+00   1.2184e-06
  1.5735e-05   9.9998e-01
```

(3p) Exercițiu Punctat! *Exercitiu 3.4* Să se definească funcția cu denumirea ***mse*** care calculează eroarea medie pătratică. Funcția primește ca parametrii de intrare ieșirile reale y din rețea, împreună cu ieșirile dorite D și returnează eroarea medie pătratică, E.

Să se încarce funcția **(doar funcția definită în cerință)** pe Moodle la exercițiul cu numele **GR[Nr.Grupa]_RNSF_LAB04_ex2.** Exercițiul restricționează încărcarea altui fișier decât cel din cerință: un fișier cu extansia **.m** și denumirea **mse** care conține declarația funcției:

```
function E = mse(y, D)
% Corp Functie de completat
end
```

Eroarea medie pătratică se va calcula după ecuația:

$$E = \frac{1}{2L} \sum_{j=1}^{L} \sum_{i=1}^{nout} (d_{ji} - y_{ji})^2$$

Unde d_{ji} este ieșirea dorită pentru vectorul de intrare j și neuronul de ieșire i, y_{ji} reprezintă ieșirea reală (ieșirea obținută după trecerea vectorului prin rețea) pentru vectorul de intrare j și neuronul de ieșire i, L reprezintă numărul de vectori de intrare în rețea, iar nout reprezintă numărul de neuroni din stratul de ieșire.

Eroarea obținută (estimativă) este de forma: E = 0.000558

Obs: i) Eroarea s-a obținut pe setul de date “lab04_data01.mat” utilizând ponderile antrenate la Exercițiul 3.2 (după parcurgerea celor 300 de iterații).
ii) Funcția poate fi testată pe Moodle cu orice alt set de date și ponderi ducând implicit la alte valori ale ieșirii reale și ale ieșirii dorite.

(2p) Exercițiu Punctat! *Exercitiu 3.5* Se consideră un set de test, T. Să se definească funcția ***eroareClasificare***, funcție ce primește ca parametrii de intrare ieșirea reală a vectorilor de test din rețea, y_t, împreună cu ieșirea dorită D_t și returnează eroarea de clasificare.

Să se încarce funcția **(doar funcția definită în cerință)** pe Moodle la exercițiul cu numele **GR[Nr.Grupa]_RNSF_LAB04_ex3.** Exercițiul restricționează încărcarea altui fișier decât cel din cerință: un fișier cu extansia **.m** și denumirea ***eroareClasificare*** care conține declarația funcției:

```
function Ecl = eroareClasificare(y_t, D_t)
% Corp Functie de completat
end
```

Eroarea de clasificare este definită astfel:

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

Obs: i) Exemplu de vectori de test:

```
T = [-1.5 1.7; 0.1 0.15; 1.85 -1.85; -1 -1.5; -0.5 -0.5];.

D_t = [2; 1; 2; 1; 2];
```

Verificare: Eroarea de clasificare va fi Ecl = 0.4000

Exercitiu 3.6: Să se antreneze o rețea neuronală de tipul Multilayer Perceptron cu ajutorul funcției ***mlptrain*** prezente în folder-ul *RNSF_Lab4 →netToolbox*, care să clasifice setul de date *lab04_data02.mat*. Setul de date conține setul de antrenare *X*, care prezintă pe fiecare linie un vector de intrare, iar pe fiecare coloană este stocată o trăsătură și vectorul *D*, care conține clasele de apartenență pentru fiecare vector de intrare din *X*. Se va utiliza structura *net* salvată în *net_init02.mat* pentru a putea verifica rezultatele. În stratul ascuns au fost introduși 30 de neuroni.

Explicarea setului de date:

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 care au fost liniarizate, obținându-se un vector de dimensiune 1x400. 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, X_j, există un corespondent D_j care reprezintă clasa de apartenență a vectorului respectiv.

Pentru vizualizarea unei imagini se poate utiliza următorul cod Octave:

```
I = reshape(X(5,:),20,20);

figure, imagesc(I)
```

Se poate observa că în imagine este reprezentată cifra 5 și este conform ieșirii dorite, *D(5)*.

Verificare:

Valoarea costului dupa 100 de iterații (aproximativ): Cost: 1.882003e-01

Exercitiu 3.7: Să se calculeze eroarea medie pătratică pentru setul de date *lab04_data02.mat*, utilizând ponderile obținute la *Exercitiu 3.5*.

Verificare: Eroarea medie patratică va fi de ordinul 10^{-2} → E = 0.020792

Obs: Este posibil ca programul să ruleze mai mult timp. Pentru a putea verifica rezultatul într-un timp mai scurt, puteți calcula eroarea medie pătratică doar pentru primii 100 de vectori din setul de date. În acest caz, ne așteptăm ca eroarea să fie aproximativ: E = 0.027426.