Invatare Automata - Tema 2
Retele Neurale CodeHunt

Nume : Sava
Prenume: Dragos
Grupa: 342C4

Cuprins

I.Introducere

Pentru realizarea temei am folosit limbajul de programare Java.
In implementare am urmarit scheletul de cod scris in Python , oferit in arhiva cu enuntul temei.

II. Proiectarea retelei

A. Tip retea

Am decis sa folosesc o retea de tip Feed Forward.Pentru antrenarea retelei voi folosi metoda Backpropagation.

B. Numar de straturi

Reteaua va avea 3 straturi.Un strat de intrare,un strat “ascuns”,si un strat pentru iesire.

C. Numar de neuroni pe strat

Stratul de intrare are un numar de neuroni egal cu dimensiunea inputului.

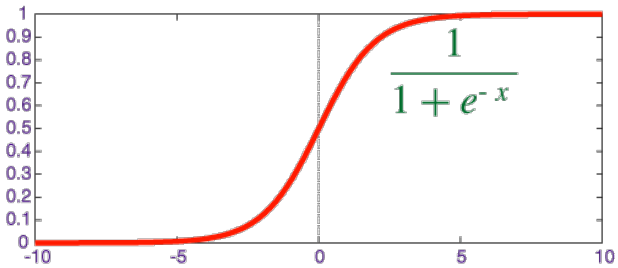
Stratul ascuns va avea un numar de neuroni egal cu : inputSize*C , unde C va fi o constanta(in urma testelor , am obtinut in medie cele mai bune rezultate pentru C=5).

Stratul de iesire are un numar de neuroni egal cu dimensiunea outputului.

D. Schema de legare

Fiecare neuron din stratul de intrare va primi input(va avea legatura) cu fiecare intrare.Fiecare neuron din urmatoarele straturi va fi legat la toti neuronii stratului anterior.

E. Functia de activare



Avand in vedere ca inputul consta din numere reale , am decis sa folosesc ca functie de activare functia Sigmoid .

III. Implementare

Function este o interfata ce va fi implementata de clasa FunctionImpl in care am implementat tot ce tine de functiile ce vor fi aproximate de retea.

Clasa NeuralNet implementeaza moulul de retea neurala.

- metoda feedForward calculeaza iesirile pentru neuronii fiecarui strat
- metoda backPropagation calculeaza erorile si actualizeaza ponderile

Clasa TestNeuralNet genereaza input-ul si antreneaza reteaua.

Pentru a antrena reteaua , la fiecare iteratie generez un nou input pe care il adaug intr-o lista si antrenez reteau cu toate input-urile din lista(Reteaua nu invata suficient de repede daca la fiecare iteratie o antrenam doar cu input-ul generat la pass-ul respectiv).

IV.Obesrvatii

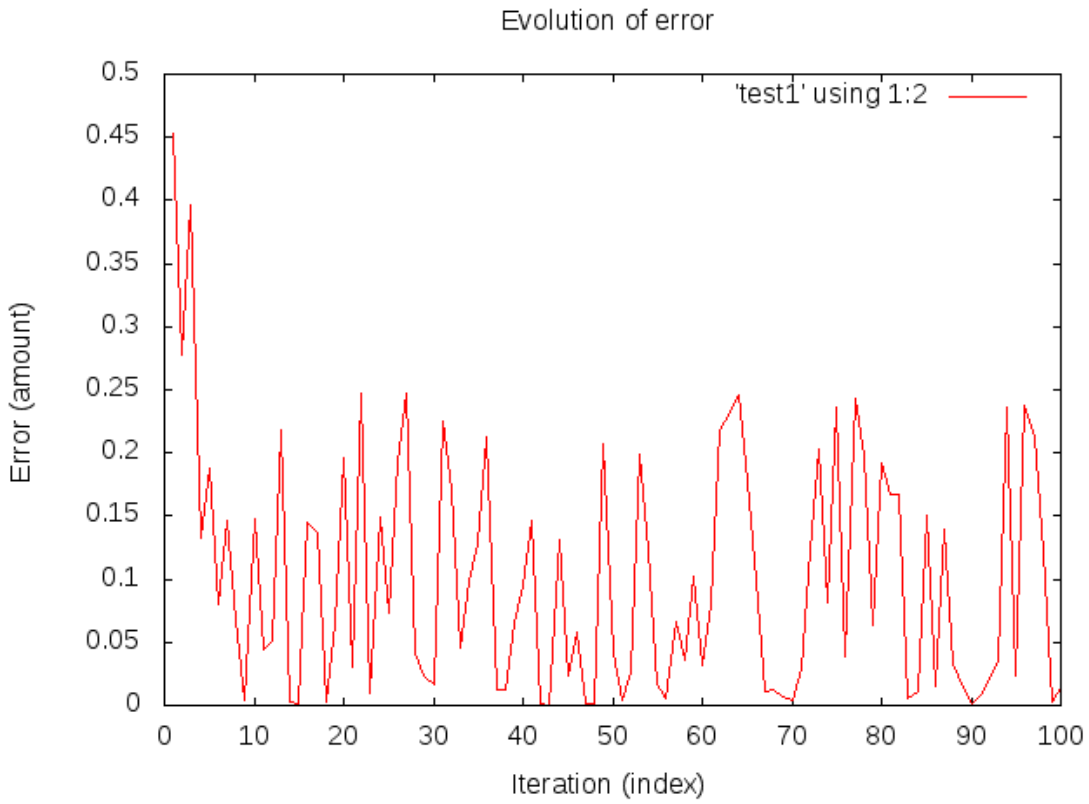
- Tema a fost implementata in linux.Pentru generarea scripturilor este necesar sa aveti instalat gnu-plot.
- Viteza de invatare = 0.85(avand in vedere ca numarul de iteratii este redus , doresc ca reteaua sa invete cand mai rapid)
- Am decis sa scalez input-ul folosind urmatoarea formula:

input_scalat = input/dimensiune_interval_domeniu(exemplu: pentru [-100;100] input_scalat = input/200)

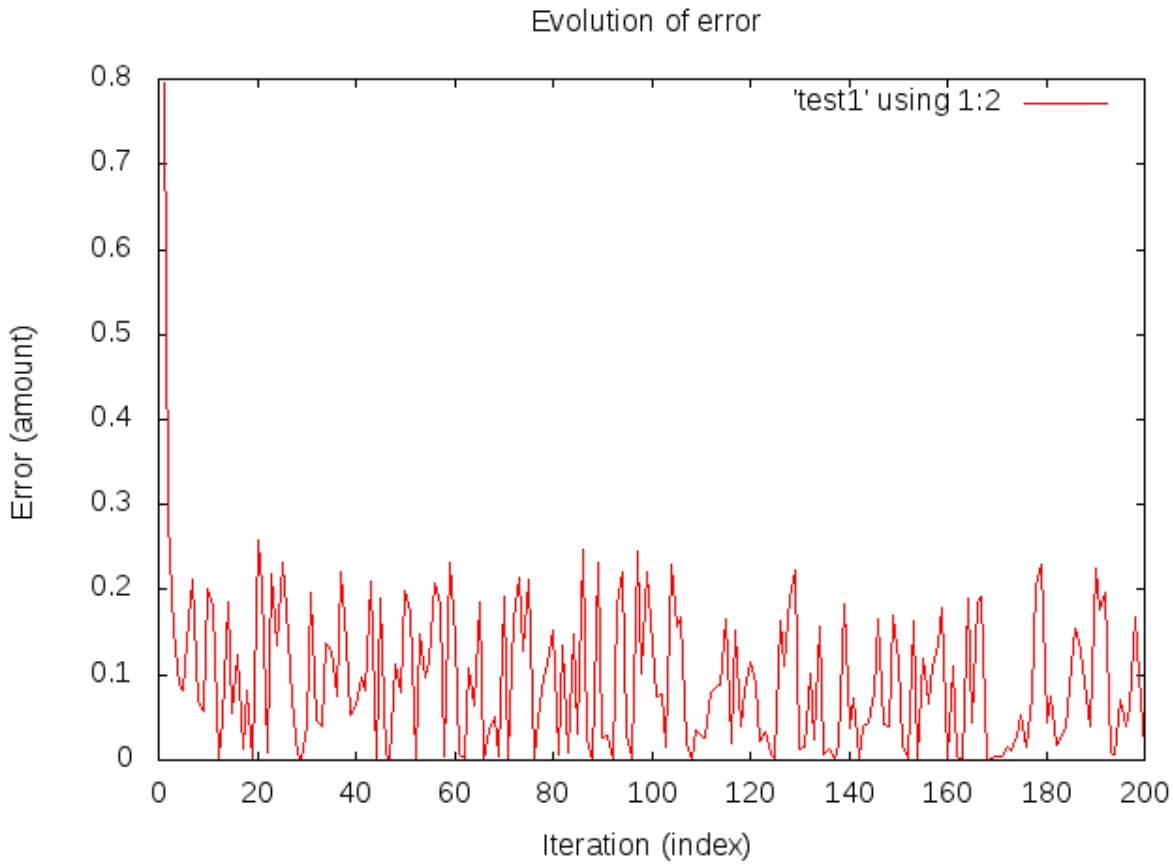
V. Grafice

➢ **Scenariul 1**

Primul grafic a fost realizat pentru 100 de iteratii.

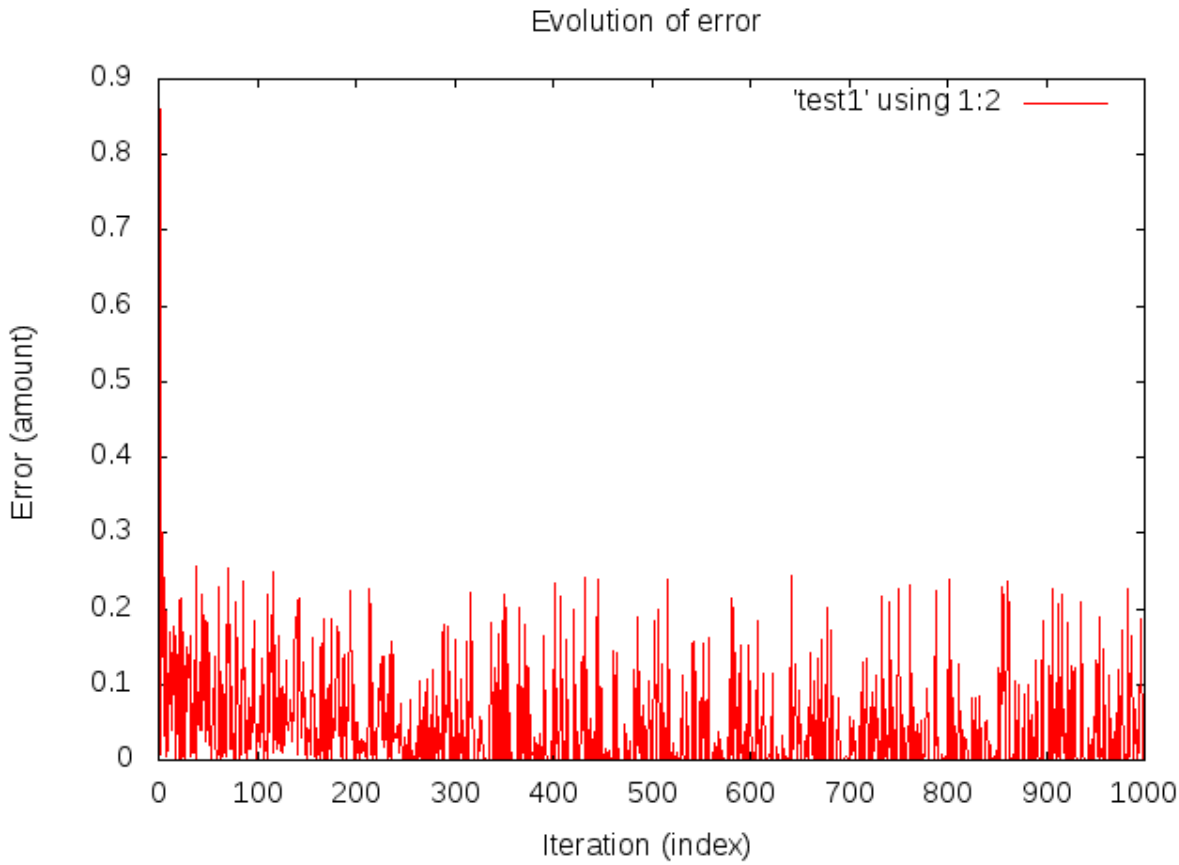


FinalError/InitialError ratio : 3.19% (cat la suta din eroare initiala reprezinta eroarea finala)



FinalError/InitialError ratio : 1.65%

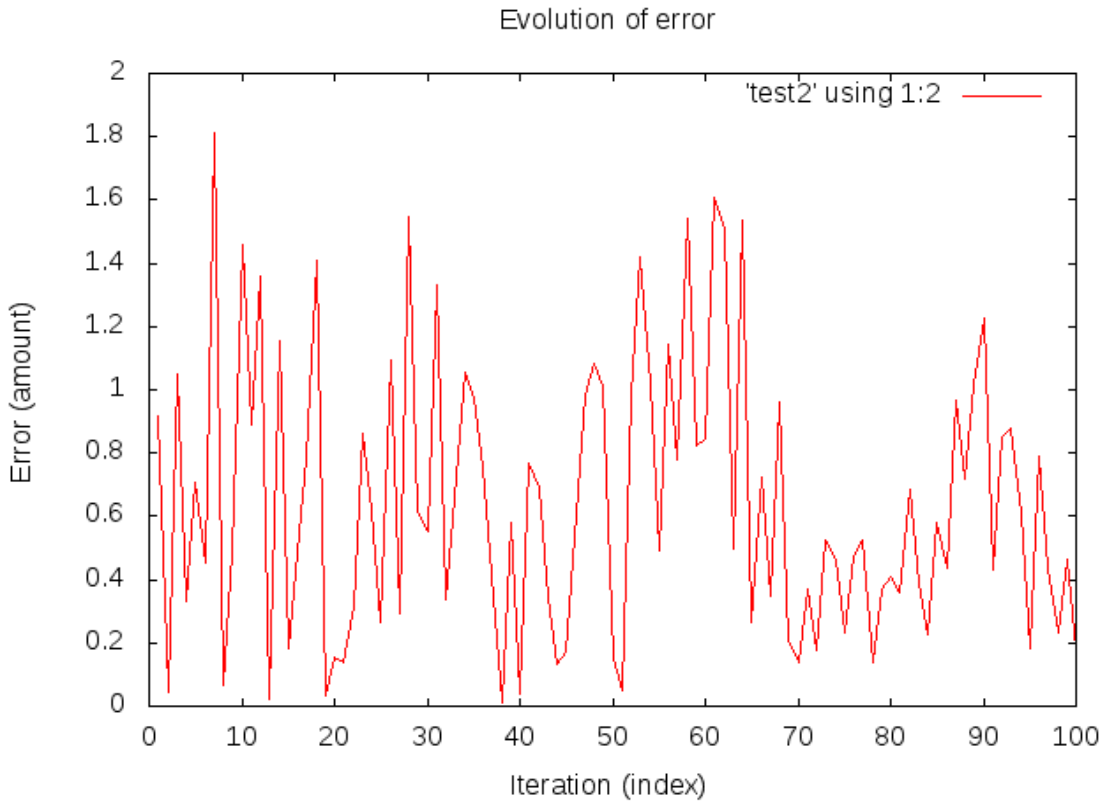
Dupa inca 100 de iteratii se poate observa imbunatatirea erorii , reteaua continuind sa invete.Totusi , aceasta imbunatatire nu este foarte mare.



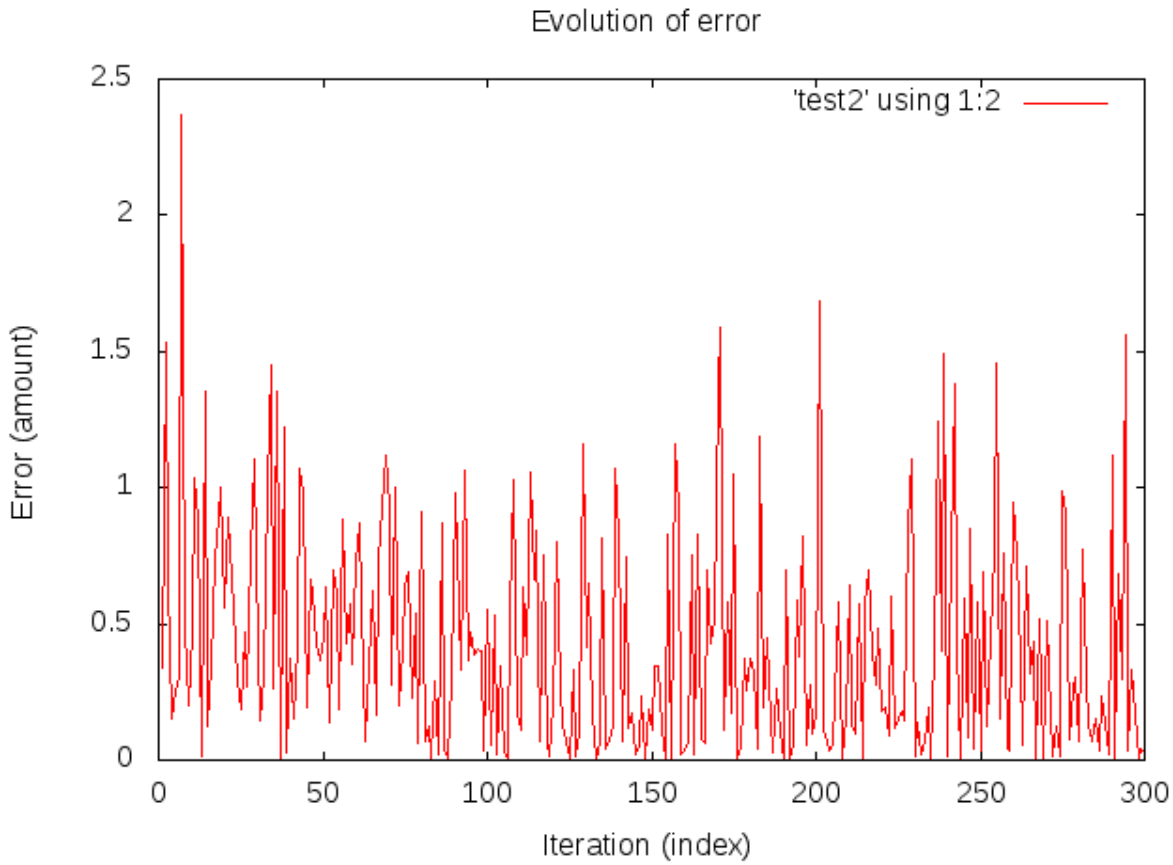
FinalError/InitialError ratio : 1.60%

Pentru 1000 de iteratii nu se mai observa fluctuatii foarte mari , reteaua incepand sa se stabilizeze ; cresterea numarului d eiteratii nu pare a mai avea un impact major asupra rezultatului.

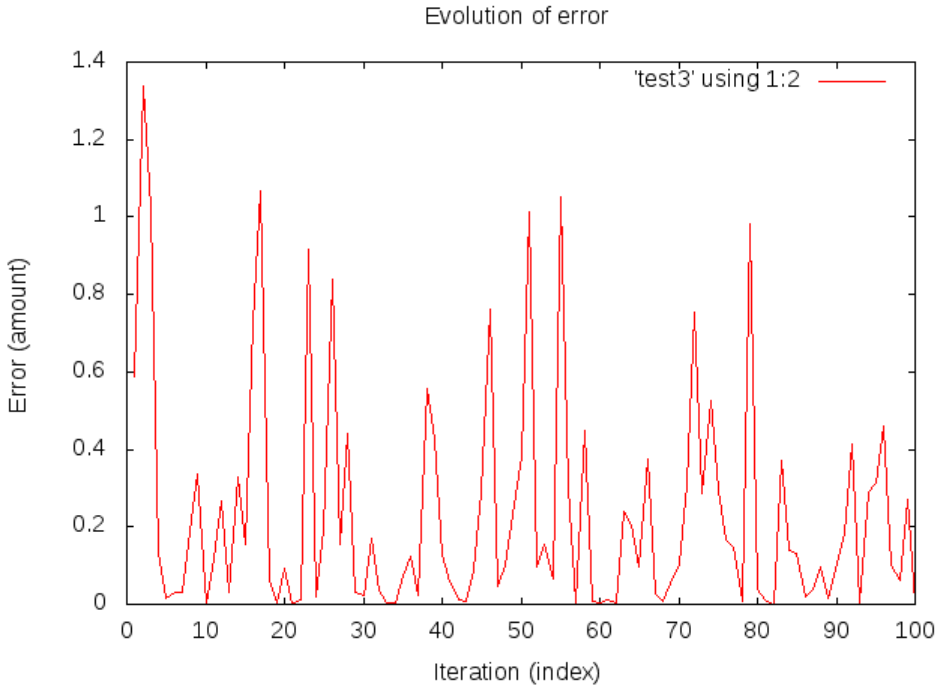
- **Scenariul 2**



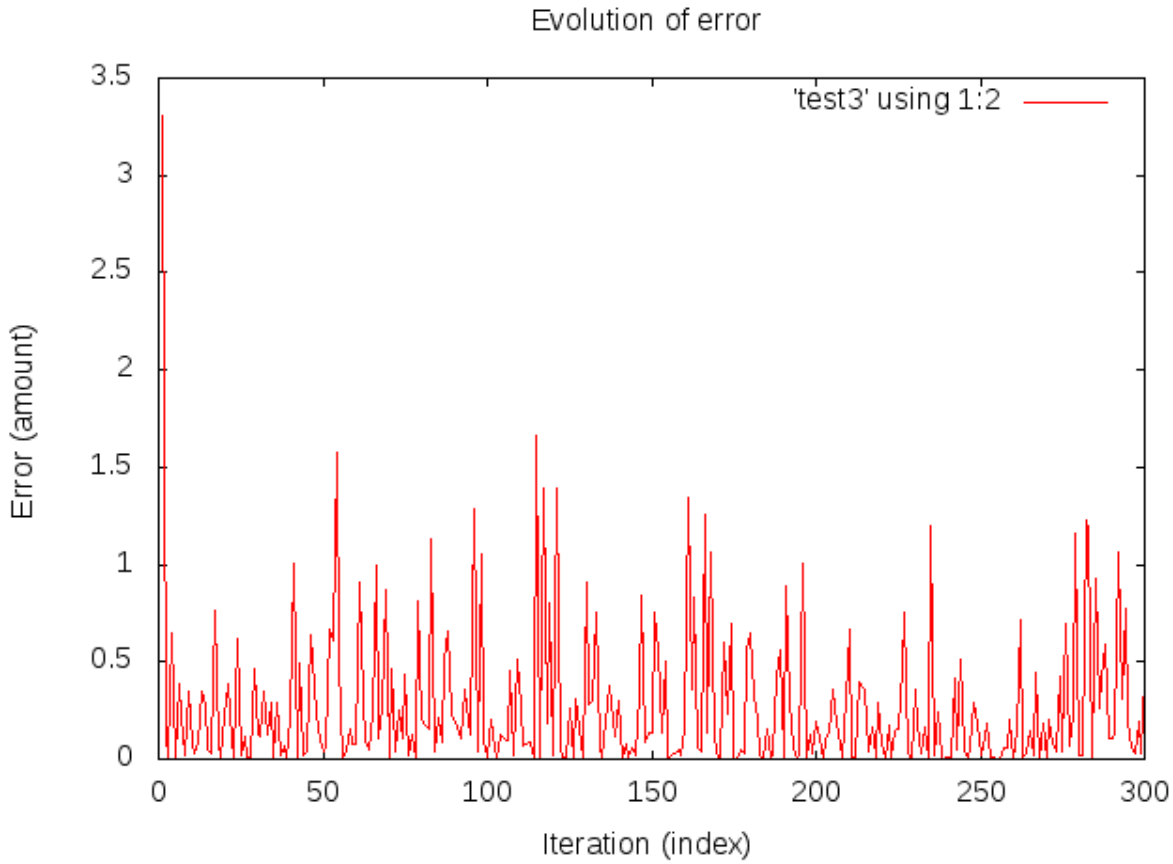
FinalError/InitialError ratio : 19.42%


Evolution of error
'test2' using 1:2
Error (amount)
Iteration (index)
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250
300

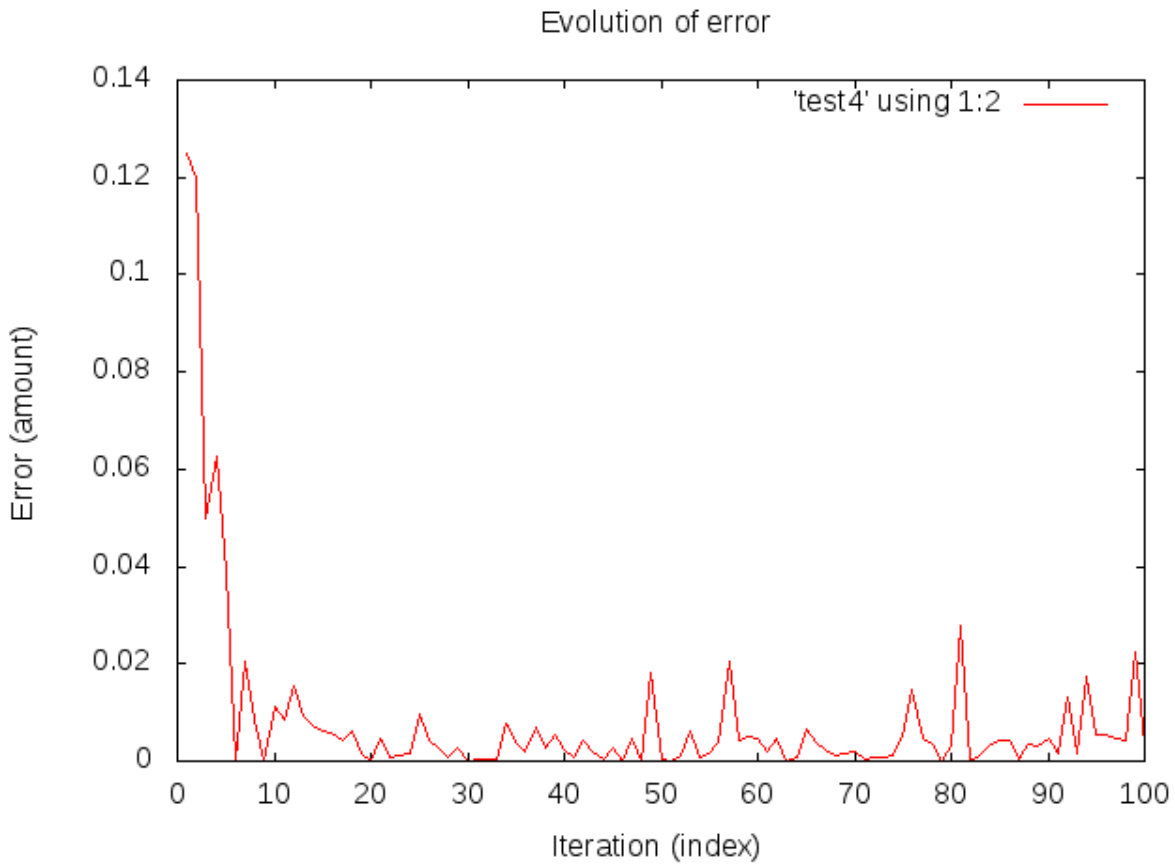
➢ **Scenariul 3**



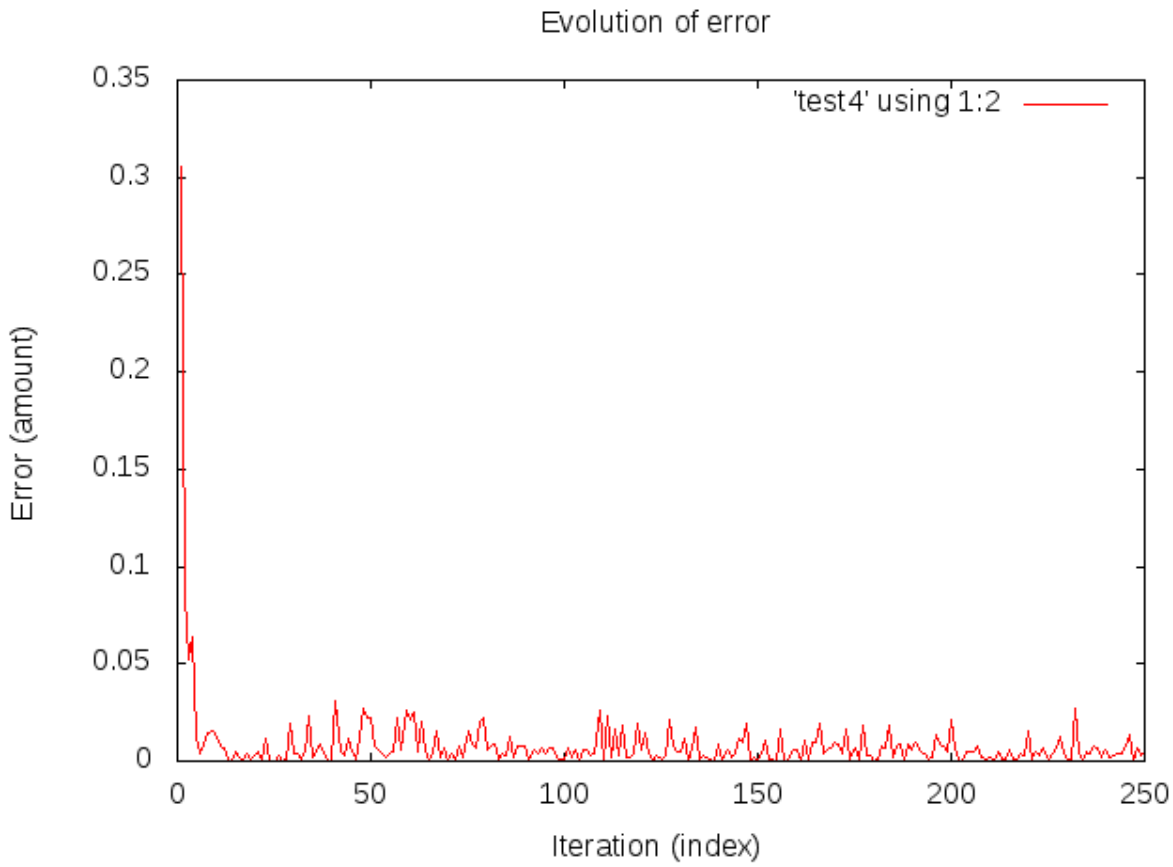
FinalError/InitialError ratio : 0.13%



➢ **Scenariul 4**


Evolution of error
'test4' using 1:2
Error (amount)
Iteration (index)
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
10
20
30
40
50
60
70
80
90
100

FinalError/InitialError ratio : 2.39%



FinalError/InitialError ratio : 1.71%

Si in acest caz, imbunatatirea este destul de mica , dupa cresterea numarului cu inca 150 de iteratii.Reteaua invata destul de repede in primele 100 de iteratii , astfel incat este normal ca apoi diferentele sa fie din ce in ce mai pici , ponderile tinzand sa convearga spre valorile lor optime.

VI.Mod utilizare

Pentru compilarea surselor folositi make.
Pentru a rula folositi comanda:
java TestNeuralNet tip_functie(poate fi 1,2,3 sau 4).

VII.Arhiva

- Makefile
- Readme
- Script folosit pentru generarea graficelor
- Fisierele sursa:

- ❏ Function.java
- ❏ FunctionImpl.java
- ❏ NeuralNetwork.java
- ❏ TestNeuralNetwork.java