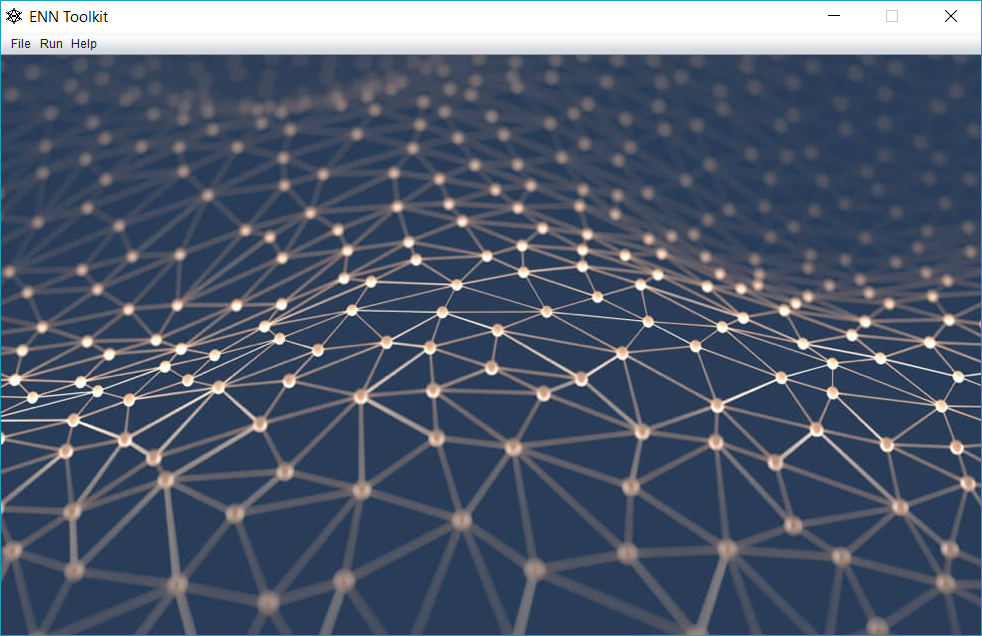
**Ghid de utilizare a “ENN Toolkit”**

Pentru a putea folosi acest produs program, asigurațivă ca aveți instalată Java pe PC dumneavoastră.

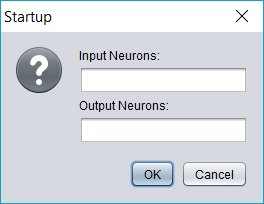


Apoi alegeți opțiunea de a deschide aplicația cu ”jar” după ce ați instalat java. “ENN Toolkit” este un produs program pentru antrenarea rețelelor neuronale extenice. Codul a fost scris în Java, deoarece este un limbaj puternic pentru interfață grafică și algoritmi. În această aplicație avem mai multe opțiuni:



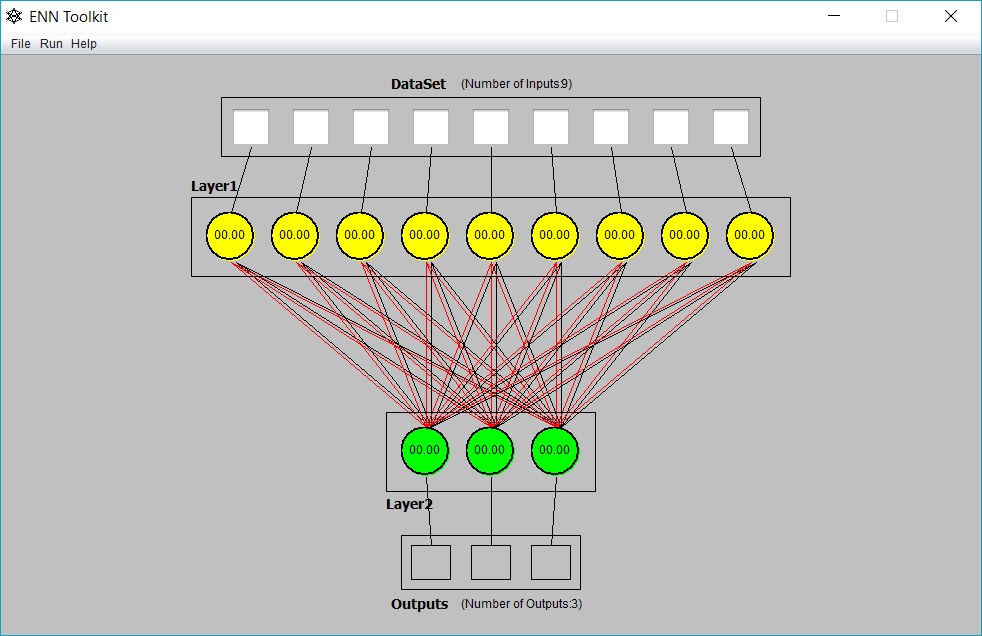
În meniul ”File” avem posibilitatea să creăm un proiect nou(New Project), să deschidem o rețea deja antrenată(Open) sau să salvăm rețeaua respectivă(Save As).

Cînd apăsăm pe ”New Project”, se afișează o fereastră :



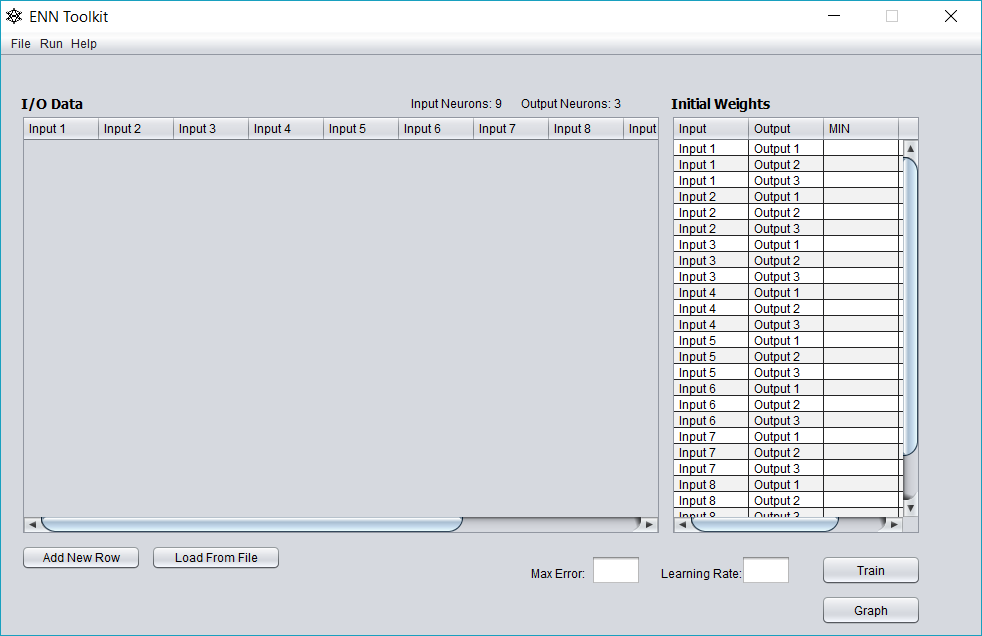
În prima căsuță indicăm numărul de neuroni de intrare, iar în a doua căsuță numărul de neuroni de ieșire.

Dacă setăm următorii parametri (9 și 3) obținem o rețea după cum urmează:



Se poate observa structura specifică rețelei neuronale extenice, cu 2 conexiuni între neuroni (min & max), am creat conexiunile cu culori diferite ca să fie o diferență grafică între ele. În această rețea lipsește stratul ”ascuns” ceea ce simplifică structura rețelei.

Următorul pas este de setare a data setului și matricii ponderilor. Pentru această facem click pe ”Train” din meniul ”Run”.

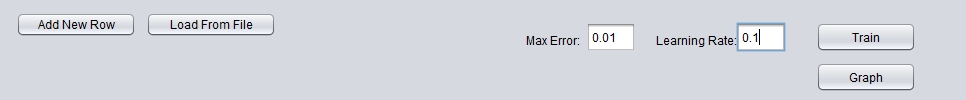


Aici avem posibilitatea să adăugăm manual datele, făcînd click pe ”Add New Row” sau să încărcăm datele dintr-un fișier .txt. De menționat este faptul, ca datele în fișierul txt trebuie să fie în format special :



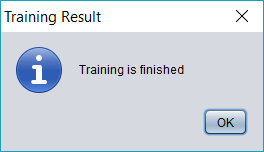
Aplicația singură decide ponderile pe baza min/max din setul de date, dar este posibilitaea de a seta ponderile manual. Numărul de ponderi se calculează automat în dependență de numărul neuronilor și conexiunilor acestora.

Următorul pas, setăm ”Eroarea maximă admisibilă” și rata de învățare:

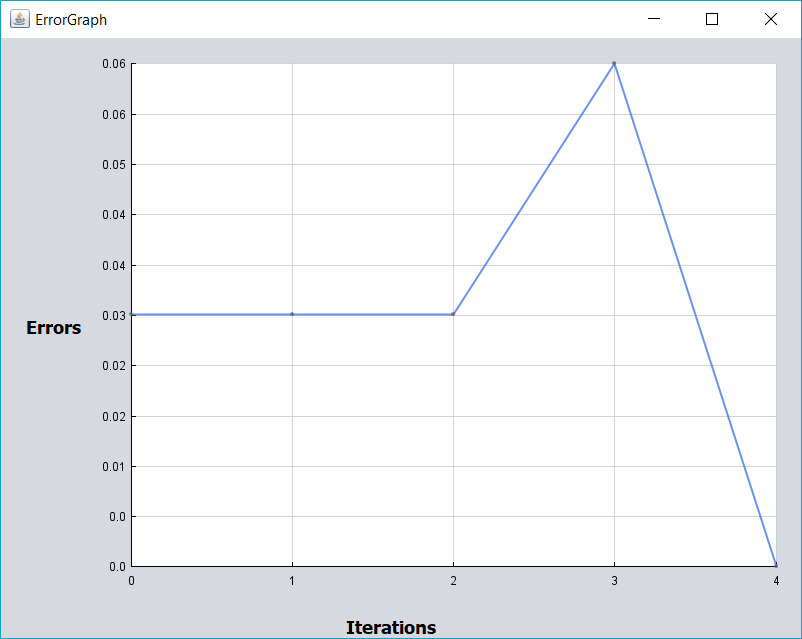


Rata de învățare dictează cât de repede mergem spre minim. Dacă e prea mare poate vom țopăi în jurul minimului, iar dacă e prea mică durează mult să ajungem la el. Mai rău de atât, în absența unei rate de învățare adecvate am putea rămâne blocați într-un minim local.

Cînd rețeaua este antrenată, o fereastră ne afișează asta:



Avem posibilitatea să vizionăm graficul rețelei, care ne afișează numărul de epoci de antrenare și erorile înregistrate.



În cazul respectiv, putem observa că au fost 5 iterații și erorile respective:

1 -- 0.033333335

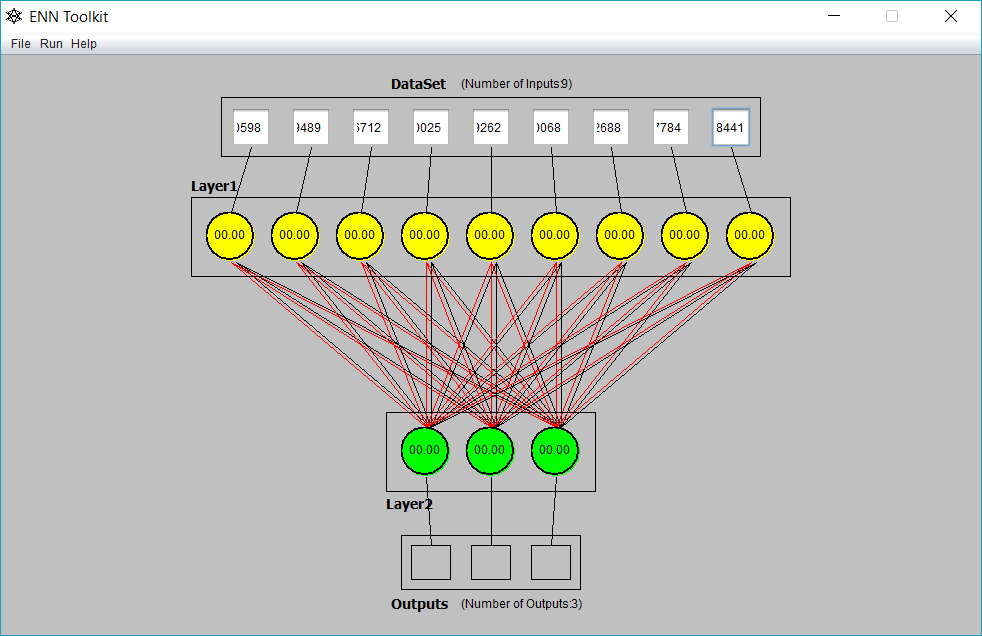
2 -- 0.033333335

3 -- 0.033333335

4 -- 0.06666667

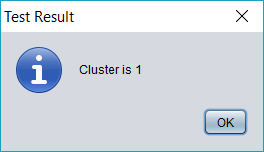
5 -- 0.0

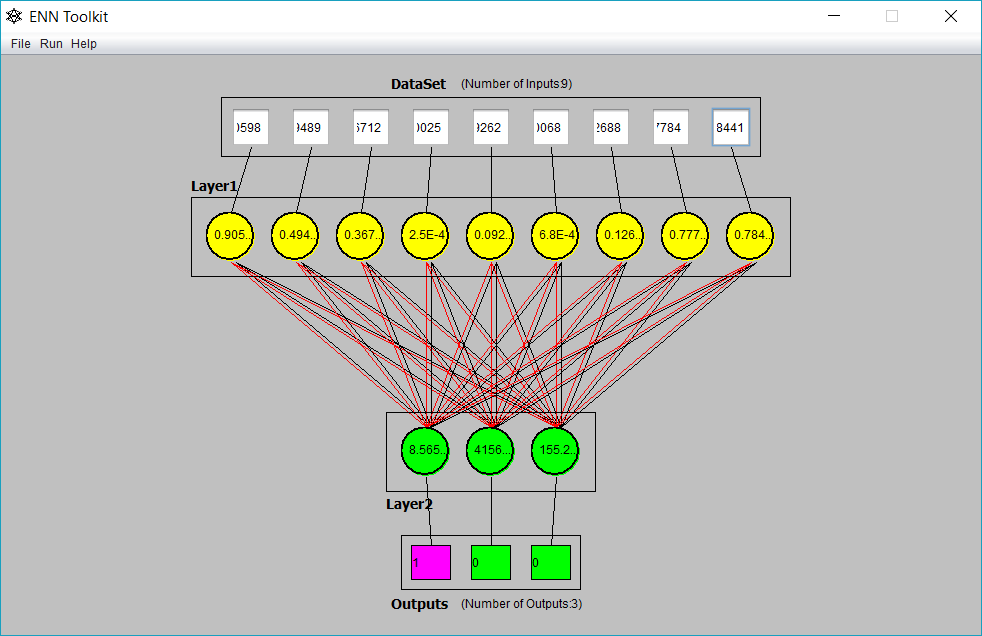
După ce a fost antrenată rețeaua, ea poate fi testată, pentru aceasta facem click pentru opțiunea ”Test” din meniul Run.



Revenim la starea inițială, aici în căsuțele neuronilor de intrare setăm datele pentru testare. Apoi facem click pe opțiunea ”Run” din meniul ”Run”.

O fereastră pop up ne afișează clusterul cu cea mai mică distanță, și respectiv clasificarea pe care o așteptă. Ex. eu am introdus date, ce reprezintă prima clasificare din 3. Respectiv distanța cea mai mică e pentru primul cluster:



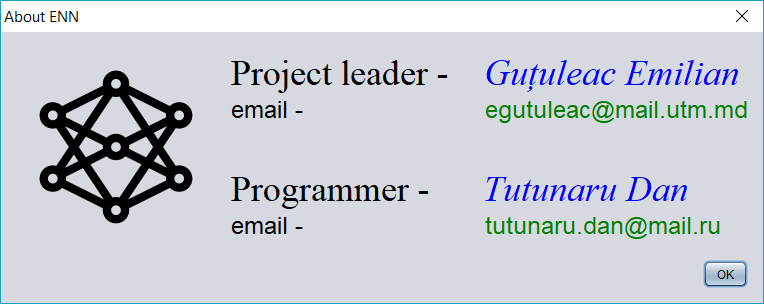


Putem vedea eficiența acestei rețele, care pentru un numărul mic de iterații ne afișeaza clasificarea corectă. Deoarece am setat data ce aparțin primei clasificări din cele 3 primim rezultatul care conține cea mai mică distanță pentru prima clasificare. Rețeaua este antrenată atunci cînd atingem eroare mică admisibilă și distanța cea mai mică este egală cu clasificarea ajustată. De exemplu: dacă avem prima clasificare I. Avem 3 distanțe ED1, ED2, ED3. Rețeaua este antrenată atunci cînd ED1=1.

Acum avem posibilitea să salvăm rețeaua antrenată, pentru aceasta facem click pe opțiunea ”Save As” din meniul ”File”, aceasta va salva reațeaua într-o extenise deosebită .enn, ca mai apoi să o putem utiliza.



În meniul Help, avem date despre autorii aplicației și un link către fișierul documentației.



Mulțumesc.