ANTRENAREA SUPERVIZATĂ A REȚELELOR FEEDFORWARD MULTISTRAT ÎN PROBLEME DE CLASIFICARE

1. Considerații generale, motivație și obiectiv

Rețelele neuronale feed-forward multistrat cu noduri de intrare sigmoidale sunt utilizate în probleme de clasificare a datelor. Alegerea funcțiilor de activare are o influență majoră asupra performanțelor rețelei. Performanța rețelei MLP este calculată pe baza procentajului elementelor clasificate corect. Pașii urmați pentru crearea modelului conexiunilor folosite în rezolvarea problemei sunt : definirea unui set de trăsături folosite ca date de intrare precum și definirea structurii rețelei, a ratei de învățare și a metodelor de validare. Reperul de bază în utilizarea rețelelor neurale în recunoașterea de șabloane îl constituie setul de trăsături care trebuie să reprezinte fără ambiguitate toate șabloanele primite ca date de intrare pentru categoriile respective.

2. Cunoștințe prealabile necesare

- Capitolul ”Antrenarea supervizată a rețelelor neuronale”, secțiunea ”Rețele feed-forward multistrat” din cursul ”Aplicații ale rețelelor neurale în automatică”.
- Experiență de programare în MATLAB.

3. Breviar teoretic

Utilizarea rețelelor neuronale feed-forward multistrat în probleme de clasificare se bazează pe antrenarea rețelelor prin algoritmul Backpropagation. Denumirea algoritmului provine din faptul că informațiile sunt procesate în sens invers de la ieșirea rețelei către intrare.

3.1. Algoritmul de clasificare

Se consideră imaginea:

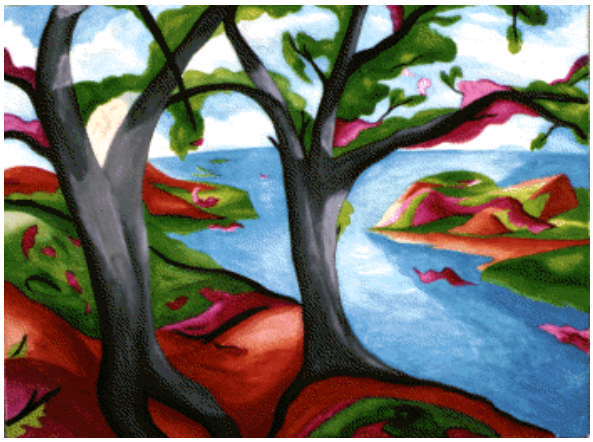


Figura 1. Imaginea inițială

Pentru clasificarea pixelilor imaginii considerate se ține cont de faptul că fiecare fixel **p** al imaginii color este caracterizat prin:

- un vector cu două componente, notat **g(p)** care conține coordonatele geometrice ale pixelului
- un vector cu trei componente, **c(p)** care exprimă coordonatele de culoare din cubul RGB.

Folosind coordonatele geometrice, să definim mulțimea pixelilor $\mathbf{G_k}$ care corespund regiunilor ocupate de culoarea $\mathbf{e_k}$, **k = 1, 2, 3**, **4**, unde:

- $\mathbf{e_1}$ semnifică culoarea neagră
- $\mathbf{e_2}$ semnifică culoarea roșie
- $\mathbf{e_3}$ semnifică culoarea verde și
- $\mathbf{e_4}$ semnifică culoarea albastră.

Evident mulțimile $\mathbf{G_k}$ sunt disjuncte. În cubul RGB scalat la [0, 1] x [0, 1] x [0, 1] definim mulțimea de culori asociată fiecărui element $\mathbf{e_k}$ cu **k = 1, 2, 3** sau **4**.

$$C_k = \left\{ \bigcup_p c(p) \middle| \, g(p) \in G_k \right\} \quad (1)$$

Ipoteză de lucru: Cel puțin din punct de vedere teoretic se poate considera că mulțimile C_1, C_2, C_3 și C_4 sunt distincte. Satisfacerea practică a acestei ipoteze depinde de calitatea imaginii color.

Algoritm

Pasul 1. Se selectează eșantioane de culoare reprezentative pentru cele trei tipuri de regiuni și se definesc mulțimile de eșantioane $\widetilde{\boldsymbol{C}}_1$, $\widetilde{\boldsymbol{C}}_2$, $\widetilde{\boldsymbol{C}}_3$ și $\widetilde{\boldsymbol{C}}_4$. Acestea constituie cea mai bună posibilitate de a obține informații parțiale despre mulțimile de culori $\mathbf{C_1}$, $\mathbf{C_2}$, $\mathbf{C_3}$ și $\mathbf{C_4}$ care sunt necunoscute ca valori concrete în cubul RGB. Se elimină vectorii identici din fiecare mulțime de culoare. În baza ipotezei de lucru considerate, $\widetilde{\boldsymbol{C}}_{\boldsymbol{k}} \subseteq \boldsymbol{C}_{\boldsymbol{k}}$, **k = 1, 2, 3, 4** și drept urmare mulțimile de eșantioane $\widetilde{\boldsymbol{C}}_1$, $\widetilde{\boldsymbol{C}}_2$, $\widetilde{\boldsymbol{C}}_3$ și $\widetilde{\boldsymbol{C}}_4$ sunt disjuncte.

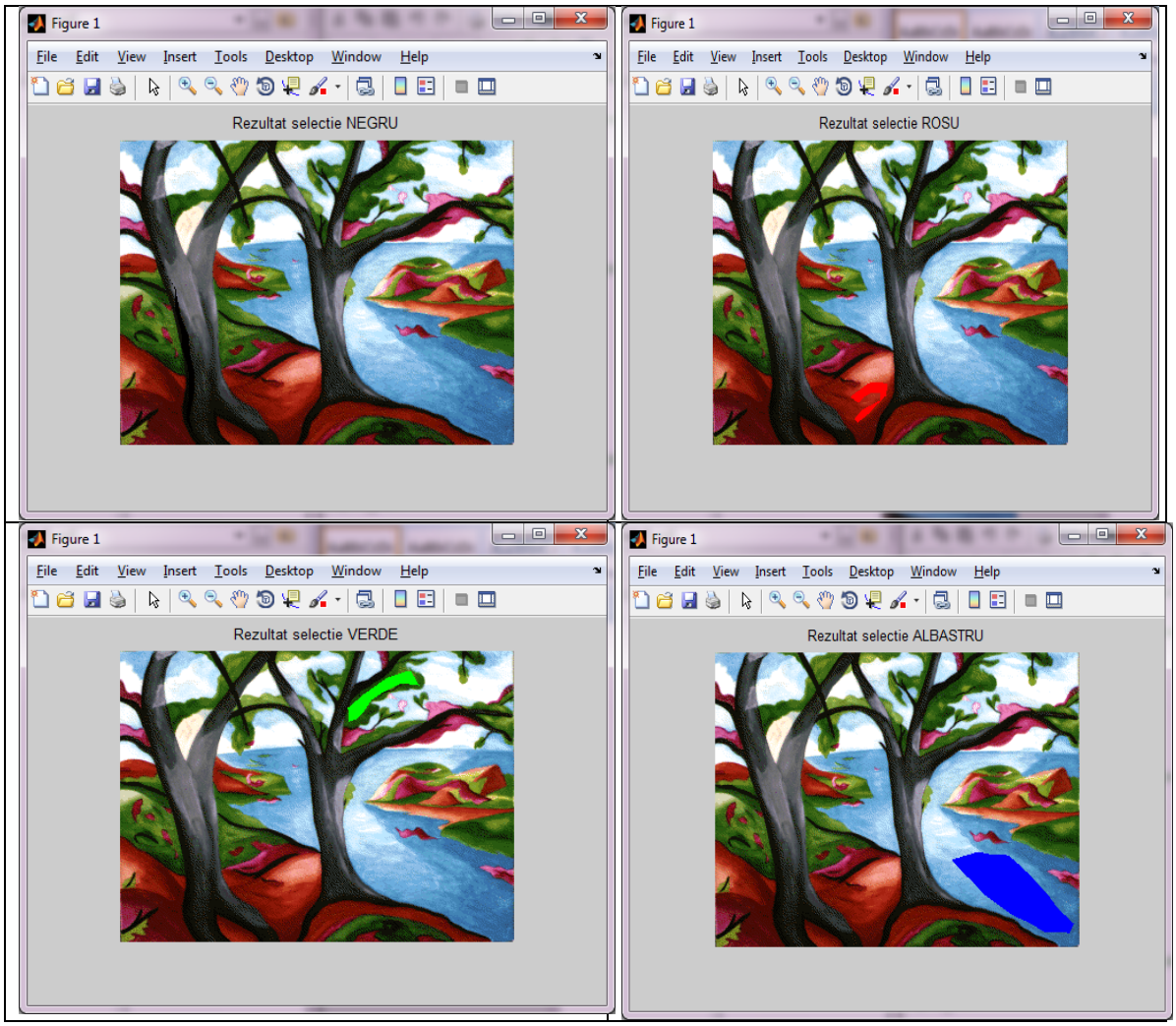


Figura 2. Alegerea eșantioanelor de culoare din regiuni reprezentative ale imaginii

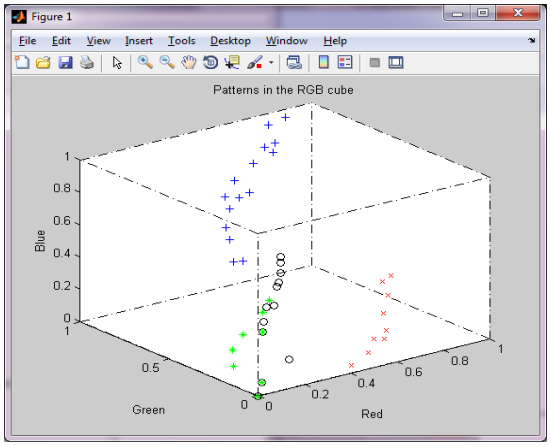


Figura 3. Plasarea culorilor din mulțimile de eșantioane $\widetilde{C}_1$, $\widetilde{C}_2$, $\widetilde{C}_3$ și $\widetilde{C}_4$ în cubul RGB scalat la [0, 1] x [0, 1] x [0, 1].

Pasul 2. Se antrenează o rețea neurală cu arhitectură adecvată astfel încât să clasifice în trei clase culorile din mulțimile de eșantioane $\widetilde{C}_1$, $\widetilde{C}_2$, $\widetilde{C}_3$ și $\widetilde{C}_4$.

Pasul 3. Se utilizează rețeaua antrenată pentru a clasifica toți pixelii din imagine în cele trei clase, corespunzătoare lui $\widetilde{C}_1$, $\widetilde{C}_2$, $\widetilde{C}_3$ și $\widetilde{C}_4$.

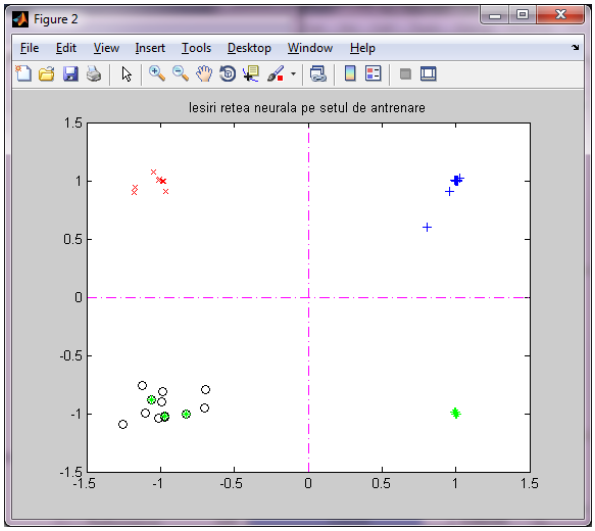


Figura 4. Ieșirile rețelei neurale pe setul de antrenare

Pasul 4. Se construiește o nouă imagine digitală în care se definesc următoarele patru regiuni care estimează regiunile ideale de interes Gk. Se utilizează trei culori distincte pentru a reprezenta regiunile estimate $\widetilde{G}_1$, $\widetilde{G}_2$, $\widetilde{G}_3$ și $\widetilde{G}_4$, pentru a asigura o mai bună vizualizare. Estimarea regiunilor ideale se va obține prin segmentare.

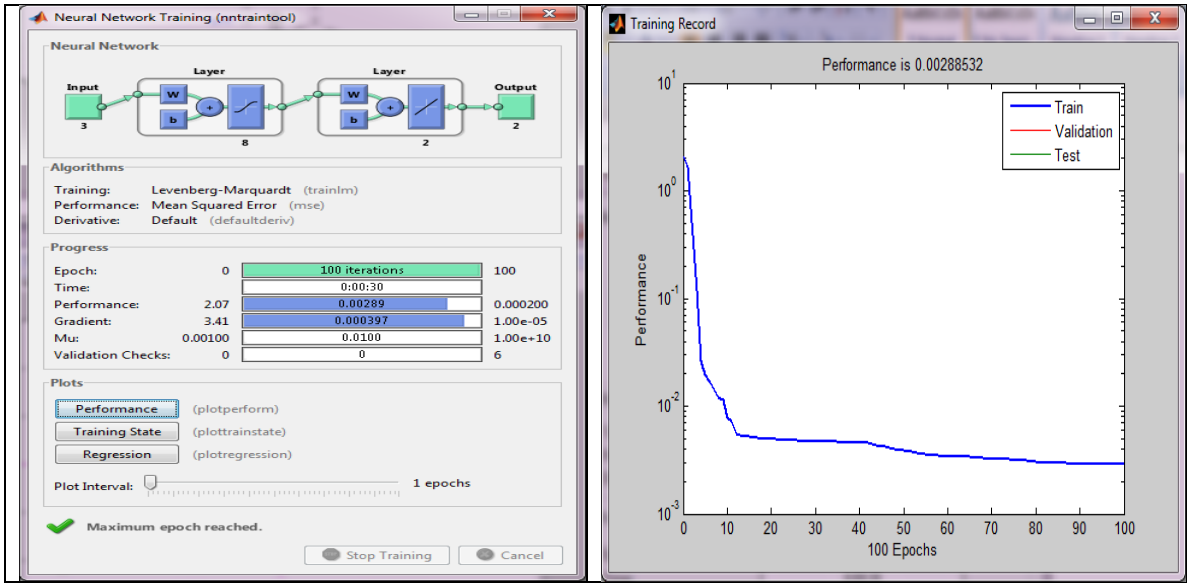


Figura 5. Performanța rețelei neurale antrenate prin algoritmul Levenberg-Marquardt

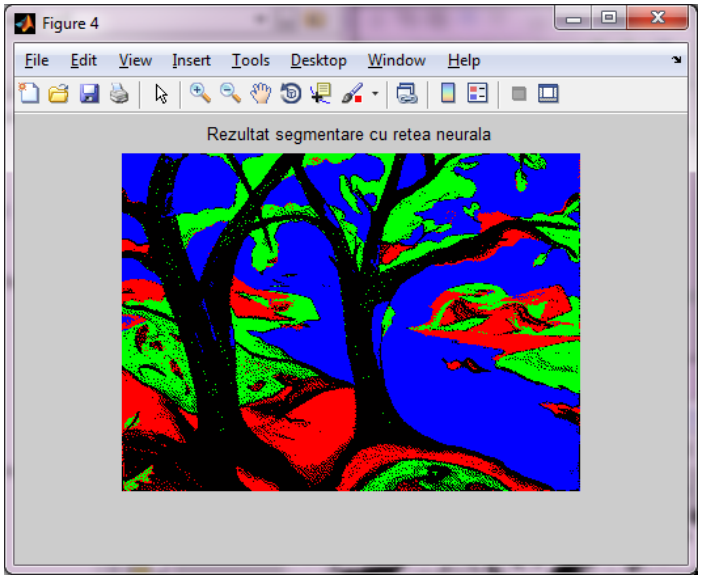


Figura 6. Separarea în clase – regiuni de interes realizată de o rețea neurală antrenată prin algoritmul Levenberg-Marquardt

Pasul 5. Se măsoară ariile regiunilor estimate $\widetilde{G}_1$, $\widetilde{G}_2$, $\widetilde{G}_3$ și $\widetilde{G}_4$ și se calculează procentele ocupate de fiecare.

3.2. Reducerea numărului de culori dintr-o imagine cu funcția imapprox

Funcția imapprox este folosită atunci când se dorește reducerea numărului de culori dintr-o imagine. Are la bază funcția rgb2ind și folosește aceleași metode de aproximare. De fapt, imapprox apelează ind2rgb pentru a converti imaginea în formatul RGB, după care apelează rgb2ind pentru a obține o imagine indexată cu mai puține culori decât imaginea inițială.

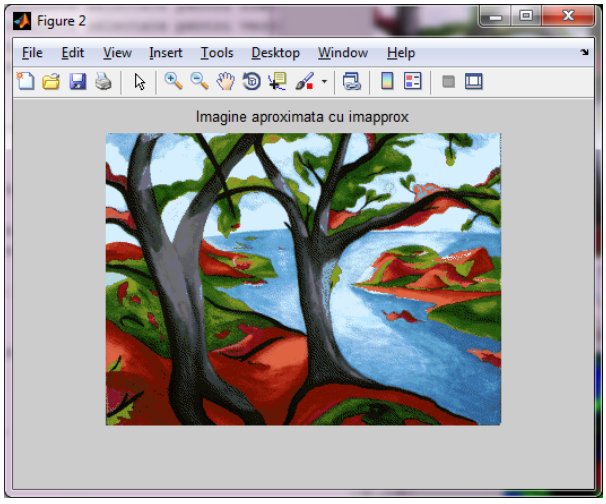


Figura 7. Aproximarea imaginii inițiale cu funcția imapprox

Calitatea imaginii rezultate depinde de metoda de aproximare utilizată, gama de culori a imaginii inițiale sau folosirea sau nu a oscilațiilor.

3.3. Identificarea regiunilor de interes cu funcția roicolor

Funcția **roicolor** poate fi folosită pentru a selecta regiuni de interes într-o imagine pe baza culorii și returnează o imagine binară.

```
BW = roicolor(A,low,high)
BW = roicolor(A,v)
```

Primul mod de apel returnează o regiune de interes ai cărei pixeli sunt în gama mapată de culori [low high]. BW este o imagine binară, semnificația unui pixel de 0 fiind că este în afara regiunii de interes, iar a unui pixel de 1 că aparține regiunii de interes.

```
BW = (A >= low) & (A <= high)
```

Cel de-al doilea tip de apel returnează o regiune de interes ai cărei pixeli corespund ca valoare elementelor din vectorul v. Semnificația elementelor de 1 ale imaginii binare BW este că elementele aferente din A corespund elementelor vectorului v. Imaginea A trebuie să fie numerică.

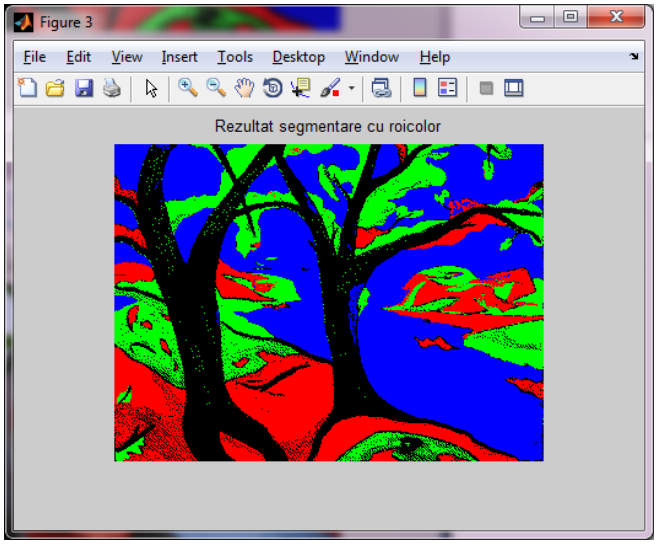


Figura 8. Aproximarea imaginii inițiale cu roicolor

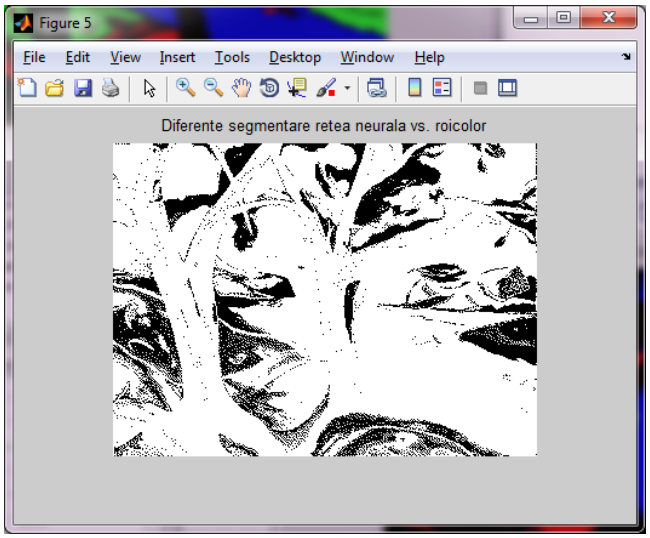


Figura 9. Diferențele între segmentarea utilizând o rețea neurală sau segmentarea cu funcția roicolor.

4. Problematica propusă pentru studiu

Problema 4.1.

Se consideră imaginea RGB din figura 1 reprezentând o un peisaj ce conține patru elemente diferite de culoare: negru, roșu, verde și albastru.
Să se precizeze componentele rețelei neurale:

- mulțimea vectorilor prototip;
- mulțimea vectorilor țintă;
- numărul de straturi și numărul de neuroni de pe fiecare strat;
- rutina de antrenare și rutina de învățare;
- numărul epocilor de antrenare;
- funcția de evaluare a performanțelor rețelei;
- denumirea și dimensiunile matricii care stochează imaginea segmentată.

Să se determine procentele din aria totală a imaginii pe care le ocupă fiecare din cele patru elemente.
Se va studia și se va lansa în execuție programul **nn_segment** elaborat în MATLAB.

NEURAL NETWORK TOOLBOX – FUNCȚIILE INTERFEȚEI GRAFICE

5. Interfața grafică a NNToolbox

Interfața grafică a NNToolbox oferă următoarele facilități pentru a lucra cu rețele neurale:

>>nnstart

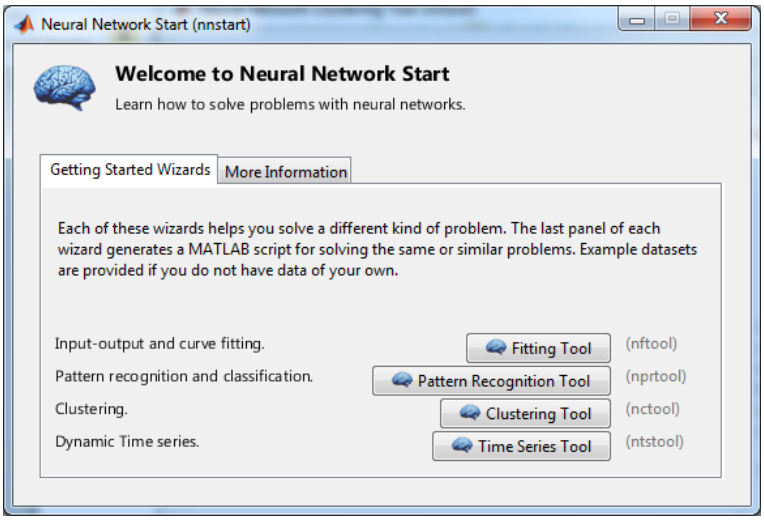
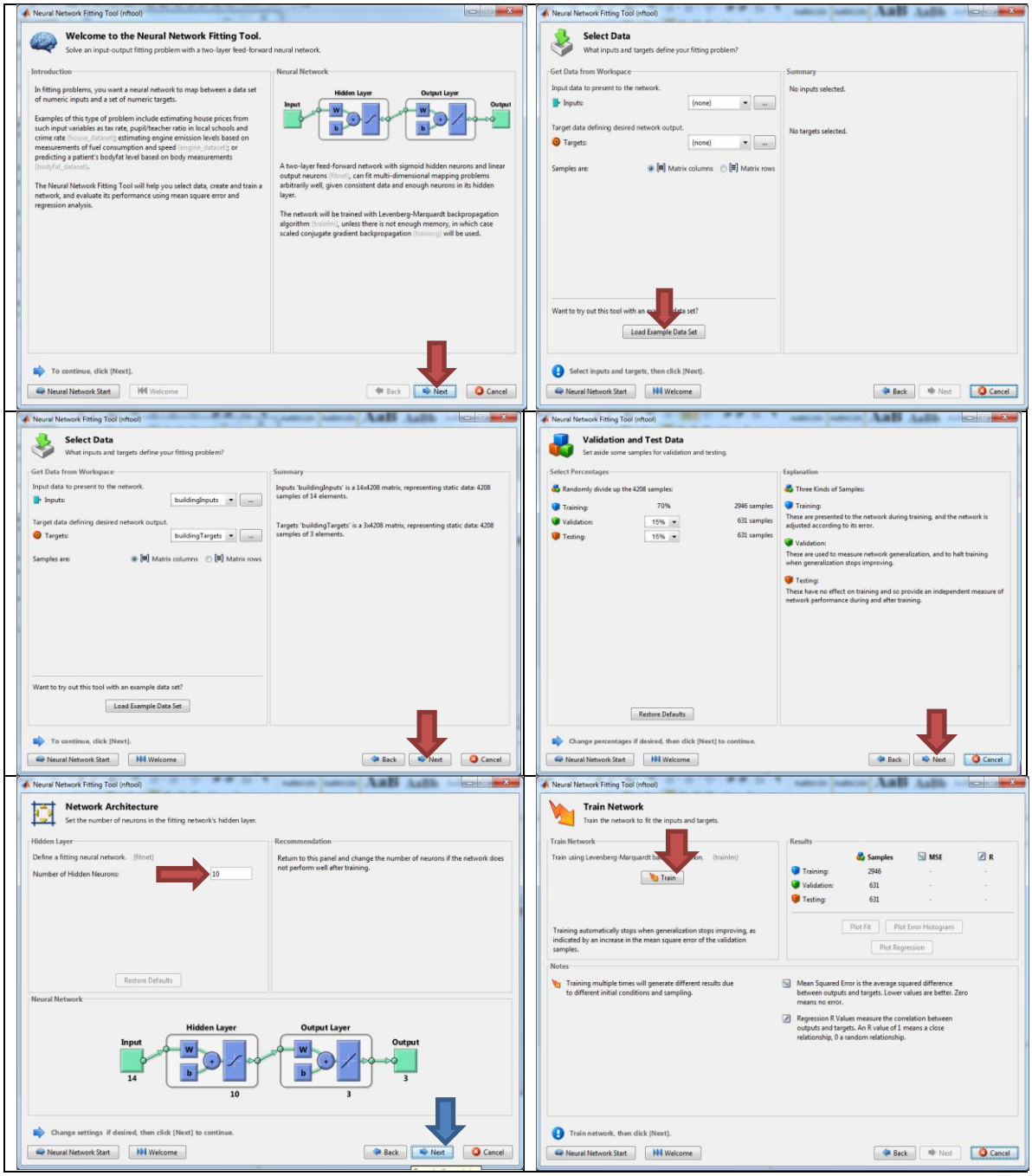
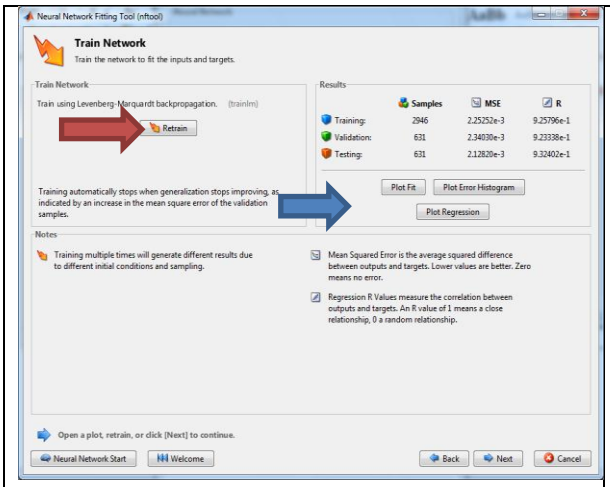


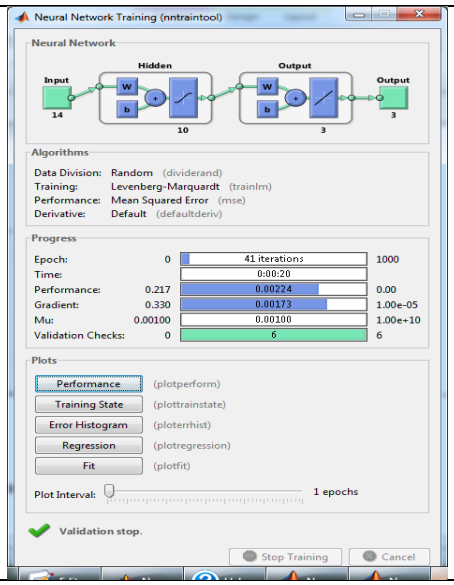
Figura xx. Neural Network Start

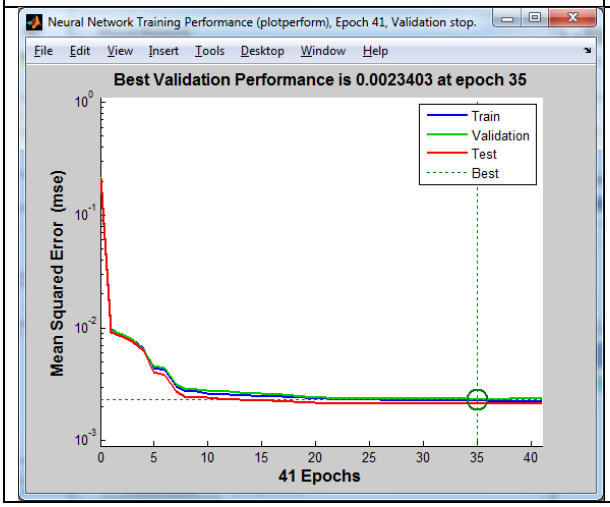
- **nctool** – toolbox pentru clasificare sau clusterizare folosind rețele neurale;
- **nftool** – toolbox pentru aproximare folosind rețele neurale;
- **nntool** – toolbox pentru gestiunea rețelelor și a datelor;
- **nntraintool** – toolbox pentru antrenarea unei rețele neurale;
- **nprtool** – toolbox pentru recunoașterea caracteristicilor;
- **ntstool** – toolbox pentru rețele neurale cu serii și
- **view** - pentru vizualizarea arhitecturii unei rețele neurale.

5.1. Nftool


Neural Network Fitting Tool (nftool)
Welcome to the Neural Network Fitting Tool.
Solve an input-output fitting problem with a two-layer feed-forward neural network.
Introduction
In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.
Examples of this type of problem include estimating house prices from such input variables as tax rate, pupil/teacher ratio in local schools and crime rate (housing_dataset); estimating engine emission levels based on measurements of fuel consumption and speed (engine_dataset); or predicting a patient's bodyfat level based on body measurements (bodyfat_dataset).
The Neural Network Fitting Tool will help you select data, create and train a network, and evaluate its performance using mean square error and regression analysis.
Neural Network
Input
Hidden Layer
Output Layer
Output
W
b
A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (fitnet), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.
The network will be trained with Levenberg-Marquardt backpropagation algorithm (trainlm), unless there is not enough memory, in which case scaled conjugate gradient backpropagation (trainscg) will be used.
To continue, click [Next].
Neural Network Start
Welcome
Back
Next
Cancel
Select Data
What inputs and targets define your fitting problem?
Get Data from Workspace
Input data to present to the network.
Inputs:
(none)
Target data defining desired network output.
Targets:
Samples are:
Matrix columns
Matrix rows
Summary
No inputs selected.
No targets selected.
Want to try out this tool with an example data set?
Load Example Data Set
Select inputs and targets, then click [Next].
buildingInputs
buildingTargets
Inputs 'buildingInputs' is a 14x4208 matrix, representing static data: 4208 samples of 14 elements.
Targets 'buildingTargets' is a 3x4208 matrix, representing static data: 4208 samples of 3 elements.
Validation and Test Data
Set aside some samples for validation and testing.
Select Percentages
Randomly divide up the 4208 samples:
Training:
70%
2946 samples
Validation:
15%
631 samples
Testing:
15%
631 samples
Explanation
Three Kinds of Samples:
Training:
These are presented to the network during training, and the network is adjusted according to its error.
Validation:
These are used to measure network generalization, and to halt training when generalization stops improving.
Testing:
These have no effect on training and so provide an independent measure of network performance during and after training.
Restore Defaults
Change percentages if desired, then click [Next] to continue.
Network Architecture
Set the number of neurons in the fitting network's hidden layer.
Hidden Layer
Define a fitting neural network. (fitnet)
Number of Hidden Neurons:
10
Recommendation
Return to this panel and change the number of neurons if the network does not perform well after training.
14
10
3
Change settings if desired, then click [Next] to continue.
Train Network
Train the network to fit the inputs and targets.
Train
Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.
Results
Samples
MSE
R
Training:
2946
Validation:
631
Testing:
631
Plot Fit
Plot Error Histogram
Plot Regression
Notes
Training multiple times will generate different results due to different initial conditions and sampling.
Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.
Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.
Train network, then click [Next].

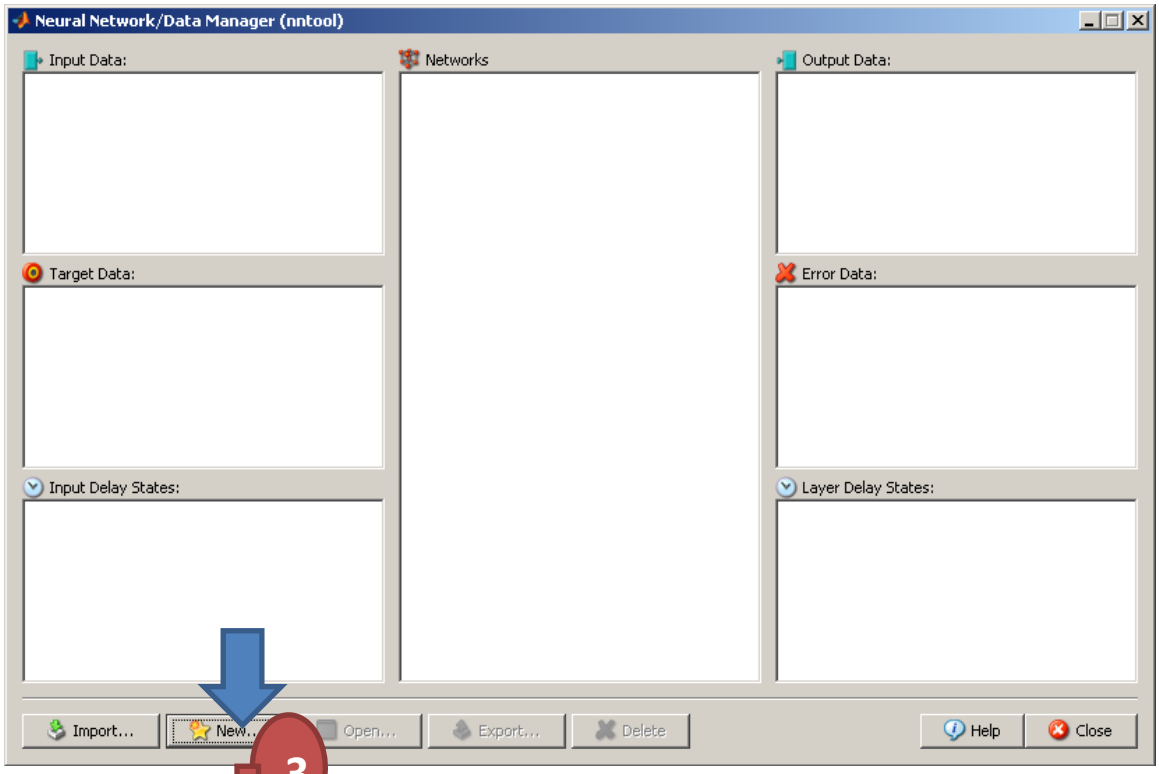

Neural Network Fitting Tool (nftool)
Train Network
Train the network to fit the inputs and targets.
Train Network
Train using Levenberg-Marquardt backpropagation. (trainlm)
Retrain
Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.
Results
Samples
MSE
R
Training: 2946 2.25252e-3 9.25796e-1
Validation: 631 2.34030e-3 9.23338e-1
Testing: 631 2.12820e-3 9.32402e-1
Plot Fit
Plot Error Histogram
Plot Regression
Notes
Training multiple times will generate different results due to different initial conditions and sampling.
Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.
Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.
Open a plot, retrain, or click [Next] to continue.
Neural Network Start
Welcome
Back
Next
Cancel

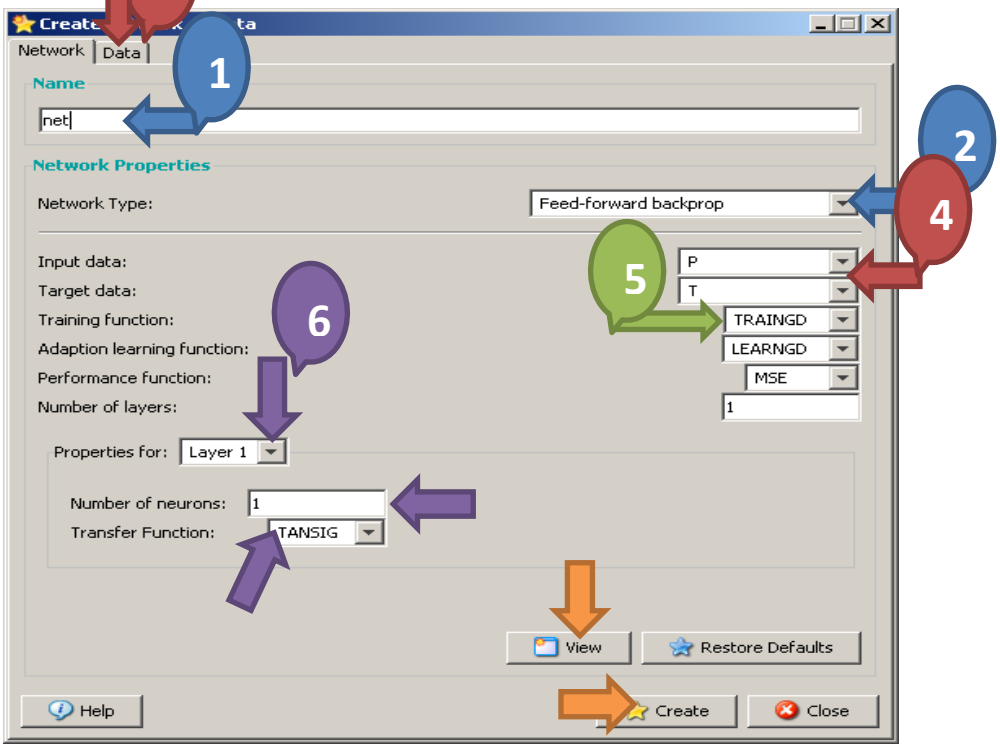

Neural Network Training (nntraintool)
Neural Network
Input
Hidden
Output
Output
14
10
3
3
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)
Progress
Epoch: 0 41 iterations 1000
Time: 0:00:20
Performance: 0.217 0.00224 0.00
Gradient: 0.330 0.00173 1.00e-05
Mu: 0.00100 0.00100 1.00e+10
Validation Checks: 0 6 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)
Plot Interval: 1 epochs
Validation stop.
Stop Training
Cancel

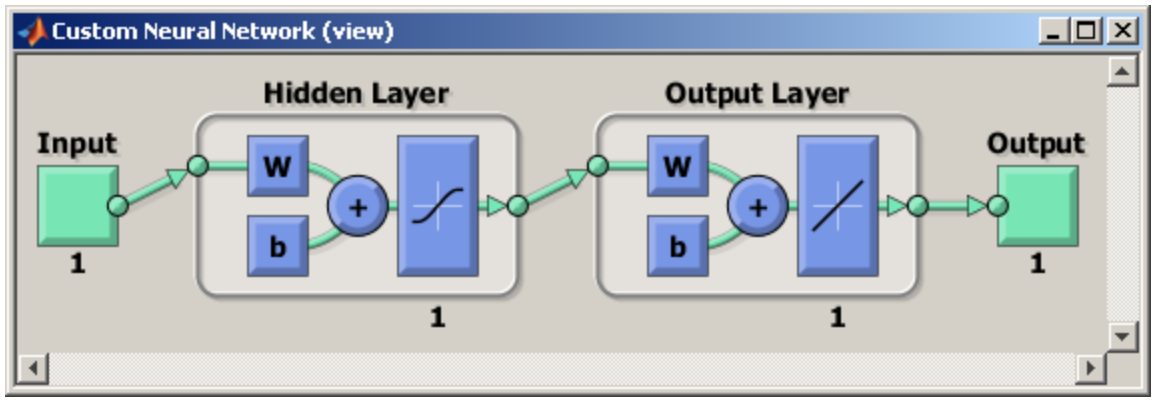

Neural Network Training Performance (plotperform), Epoch 41, Validation stop.
File Edit View Insert Tools Desktop Window Help
Best Validation Performance is 0.0023403 at epoch 35
Train
Validation
Test
Best
Mean Squared Error (mse)
41 Epochs

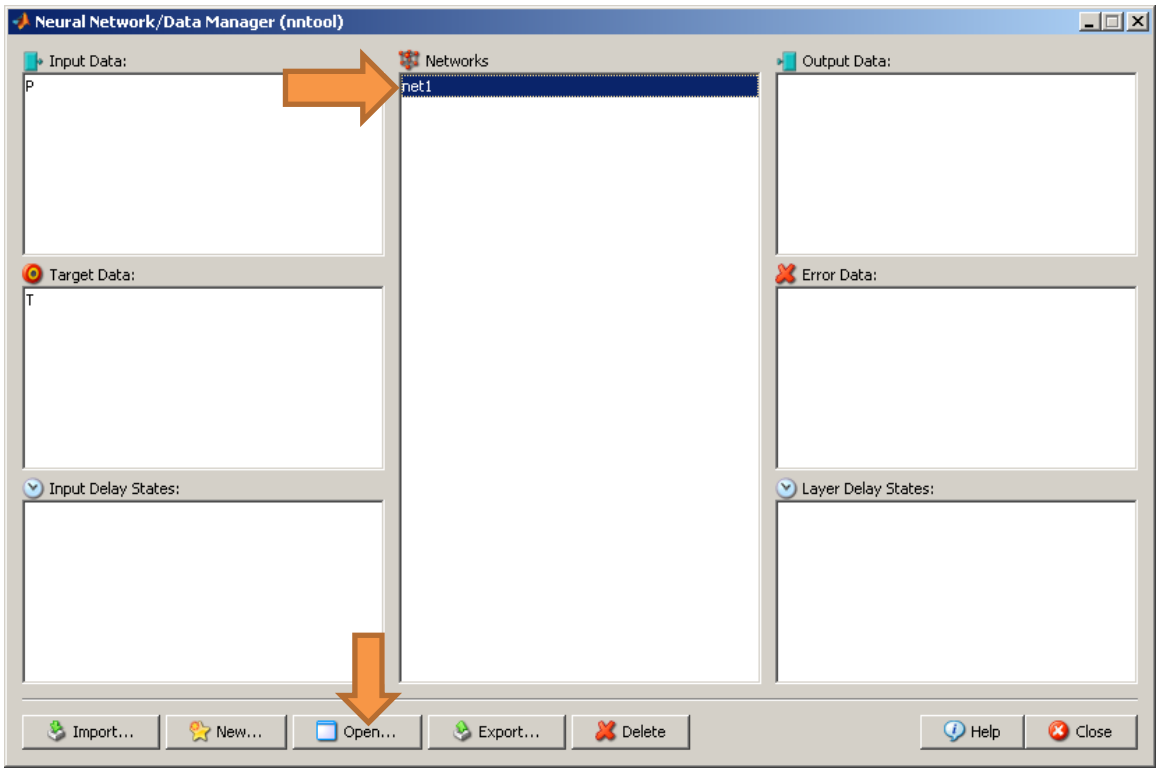
5.2. Nntool

>>nntool

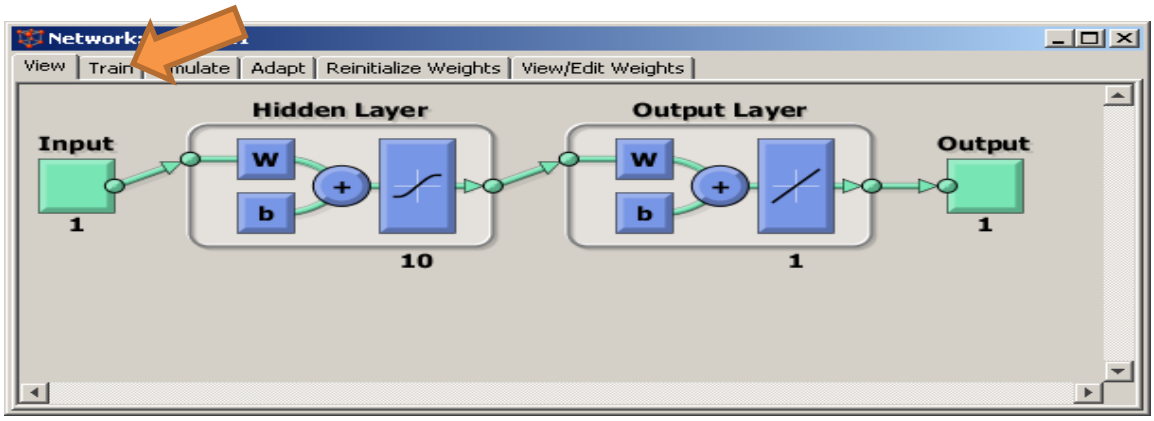


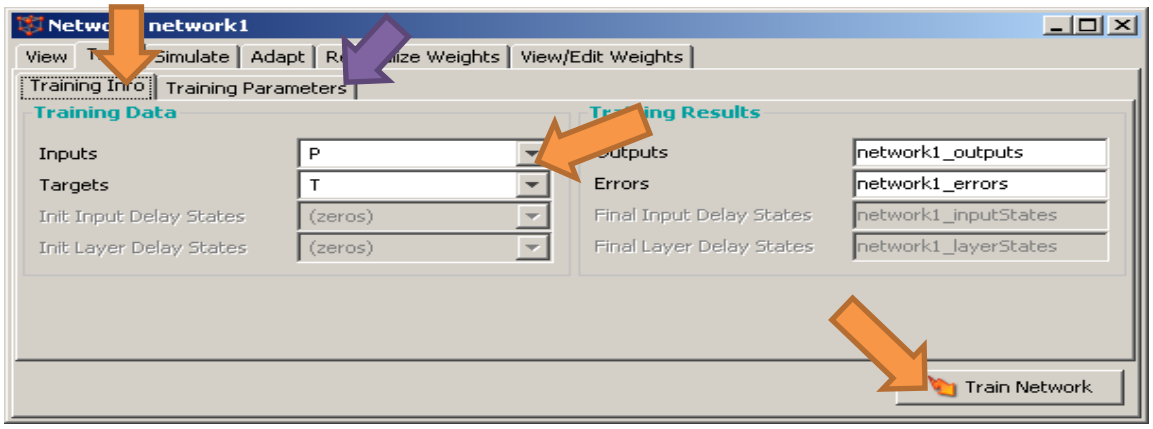


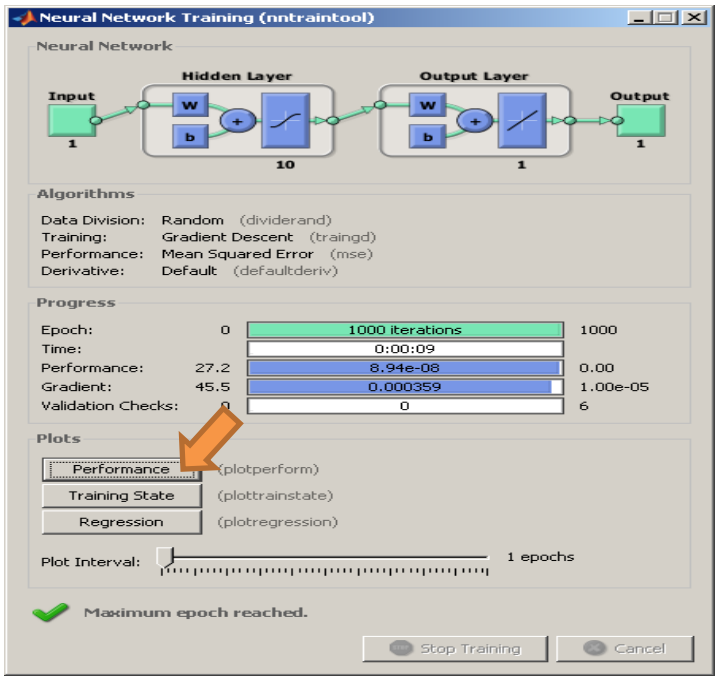

Custom Neural Network (view)
Hidden Layer
Output Layer
Input
Output
W
b
+
1
1
1
1

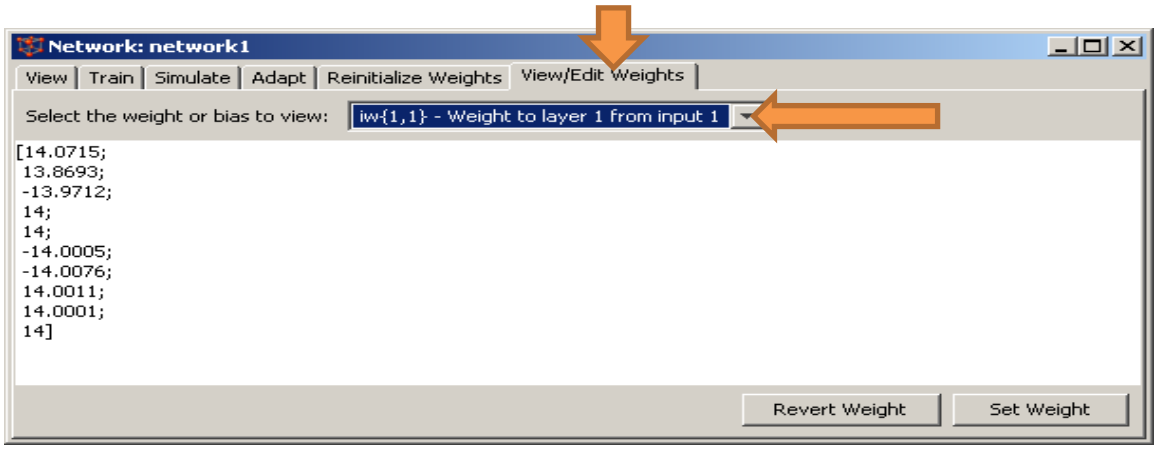

Neural Network/Data Manager (nntool)
Input Data:
P
Networks
net1
Output Data:
Target Data:
T
Error Data:
Input Delay States:
Layer Delay States:
Import...
New...
Open...
Export...
Delete
Help
Close

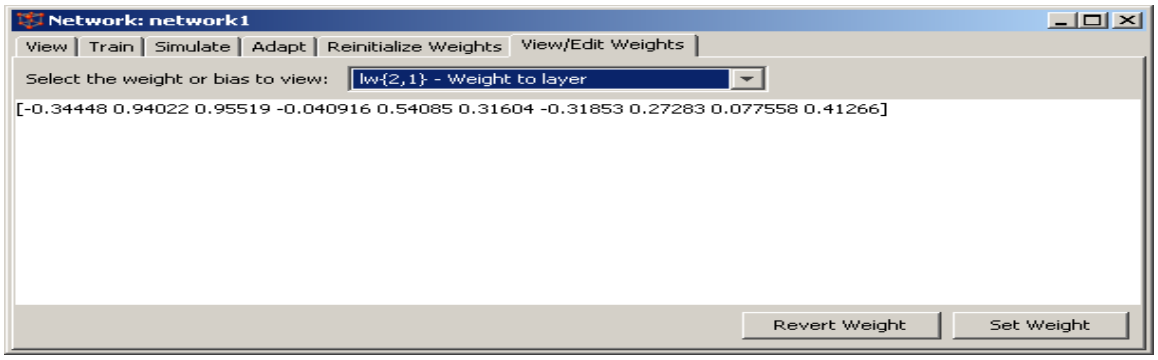
5.3. Nntraintool

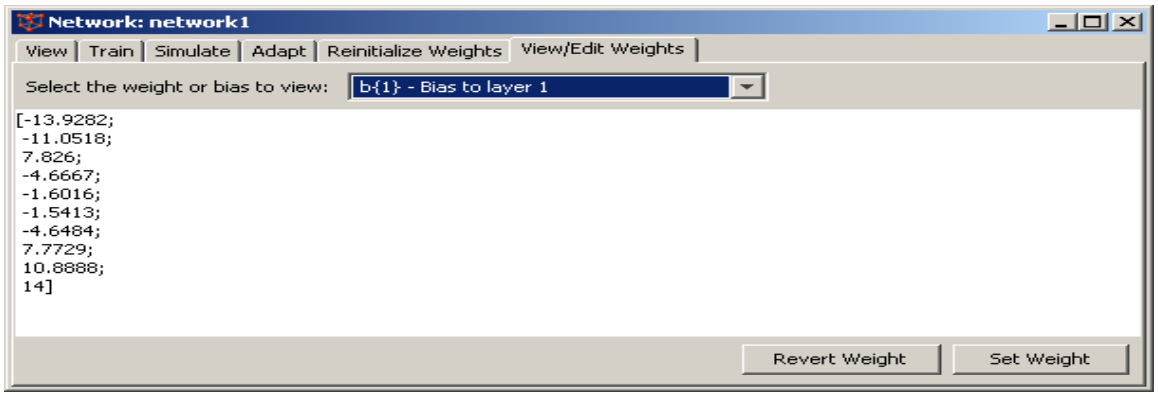


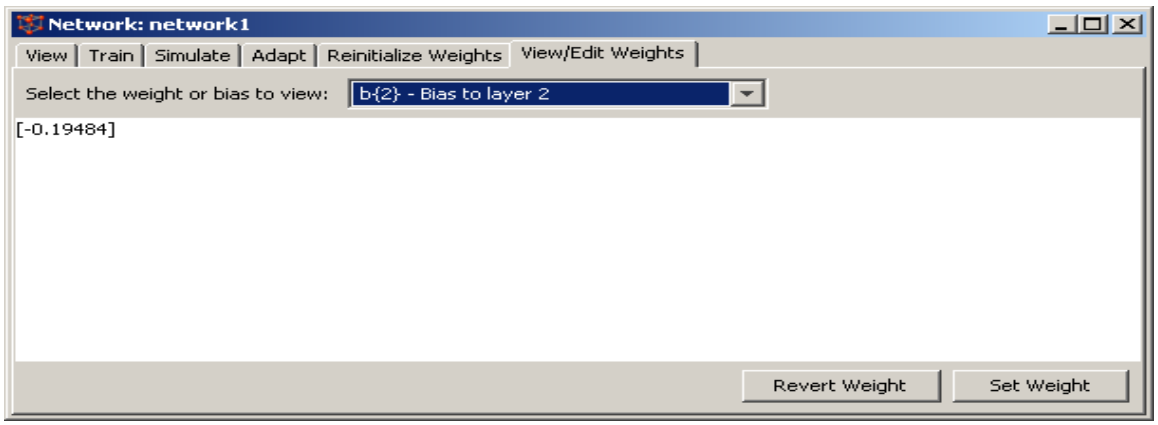





Network: network1
View
Train
Simulate
Adapt
Reinitialize Weights
View/Edit Weights
Select the weight or bias to view:
iw{1,1} - Weight to layer 1 from input 1
[14.0715;
13.8693;
-13.9712;
14;
14;
-14.0005;
-14.0076;
14.0011;
14.0001;
14]
Revert Weight
Set Weight


Network: network1
View
Train
Simulate
Adapt
Reinitialize Weights
View/Edit Weights
Select the weight or bias to view:
iw{2,1} - Weight to layer
[-0.34448 0.94022 0.95519 -0.040916 0.54085 0.31604 -0.31853 0.27283 0.077558 0.41266]
Revert Weight
Set Weight


Network: network1
View
Train
Simulate
Adapt
Reinitialize Weights
View/Edit Weights
Select the weight or bias to view:
b{1} - Bias to layer 1
[-13.9282;
-11.0518;
7.826;
-4.6667;
-1.6016;
-1.5413;
-4.6484;
7.7729;
10.8888;
14]
Revert Weight
Set Weight


Network: network1
View
Train
Simulate
Adapt
Reinitialize Weights
View/Edit Weights
Select the weight or bias to view:
b{2} - Bias to layer 2
[-0.19484]
Revert Weight
Set Weight

6. Problematica propusă pentru studiu

Problema 6.1.

Să se construiască o rețea neurală cu două straturi, cu s = 2, 4 sau 8 neuroni, cu funcție de activare sigmoid bipolar pe stratul ascuns și 1 neuron cu funcție de activare liniară pe stratul de ieșire. Se consideră vectorul prototip P calculat după formula:

$$P = \frac{i-1}{50} \; cu \; i = 1, \dots, 51$$

și vectorul țintă calculat după formula:

$$T = 1 + \exp\left(-P(i)\right) \cdot \sin\left(2 \cdot \pi \cdot P(i) - \frac{\pi}{2}\right) cu \; i = 1, \dots, 51.$$

Pentru fiecare din cele trei cazuri să se deseneze

- arhitectura rețelei;
- graficul evoluției erorii medii pătratice pe parcursul antrenării;
- graficul funcției (P,T) și al aproximării cu rețeaua MLP pe aceeași figură.

și să se noteze:

- numărul de iterații cât a durat antrenarea;
- ponderile și deplasările aferente neuronilor de pe stratul ascuns;
- ponderile pentru conexiunile dintre stratul de ieșire și stratul ascuns.