UNIVERSITÀ DEGLI STUDI DI MILANO

Introduction to Neural Networks using Matlab

Enrique Muñoz Ballester

Dipartimento di Informatica
via Bramante 65, 26013 Crema (CR), Italy
enrique.munoz@unimi.it

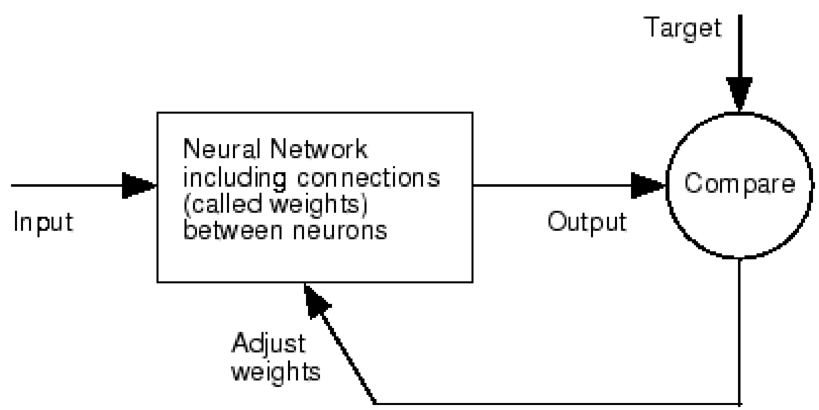
Material

- Download slides data and scripts:

https://homes.di.unimi.it/munoz/teaching.html

Neural networks

- Inspired by biological nervous systems
- Collection of connected artificial neurons
- Interesting properties:
  - Adaptive
  - Non-linear
  - General approximators



Neural networks applications

- Aerospace
  - Autopilot, flight path simulation, control systems, fault detection
- Automotive
  - Automatic guidance
- Banking
  - Check reading, credit evaluation, credit card activity checking
- Defense
  - Target tracking, object discrimination, new sensors
- Electronics
  - Process control, chip failure analysis

Neural network applications

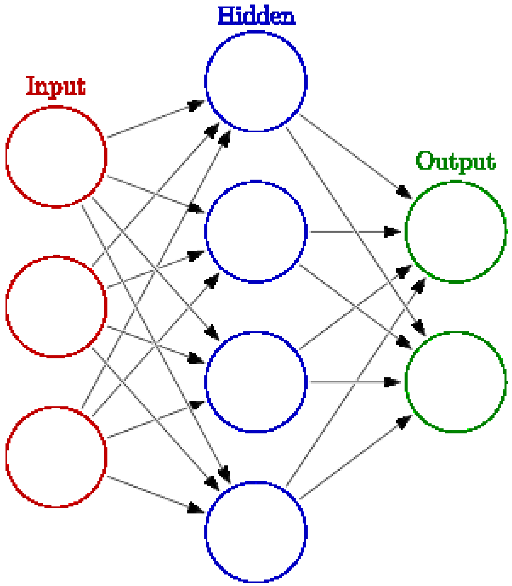
- Entertainment
  - Animations, special effects
- Financial
  - Portfolio trading program, currency price prediction
- Industrial
  - Manufacturing control, machine diagnosis, visual quality inspection systems
- Medical
  - Cancer analysis, prosthesis design, ECG analysis
- Oil and gas
  - Exploration

Neural networks applications

- Robots
  - Trajectory control, manipulator controlers, vision systems
- Speech
  - Speech recognition, text-to-speech synthesis
- Telecommunications
  - Image and data compression, automated information services
- Transportation
  - Truck brake diagnosis, vehicle scheduling

Neural networks in Matlab

- What can we do with NN Toolbox?
  - Fit Data with a Neural Network
  - Classify Patterns with a Neural Network
  - Cluster Data
  - Time Series Prediction and Modeling

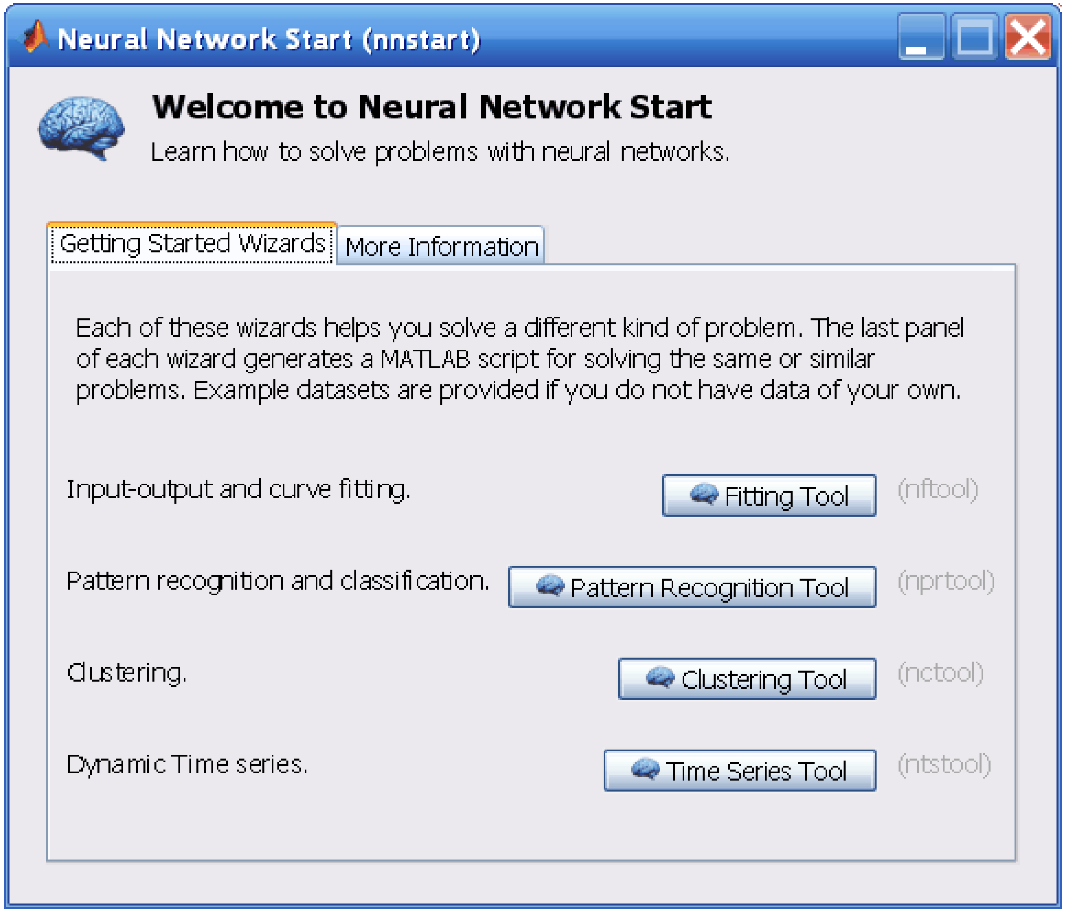


Neural networks in Matlab

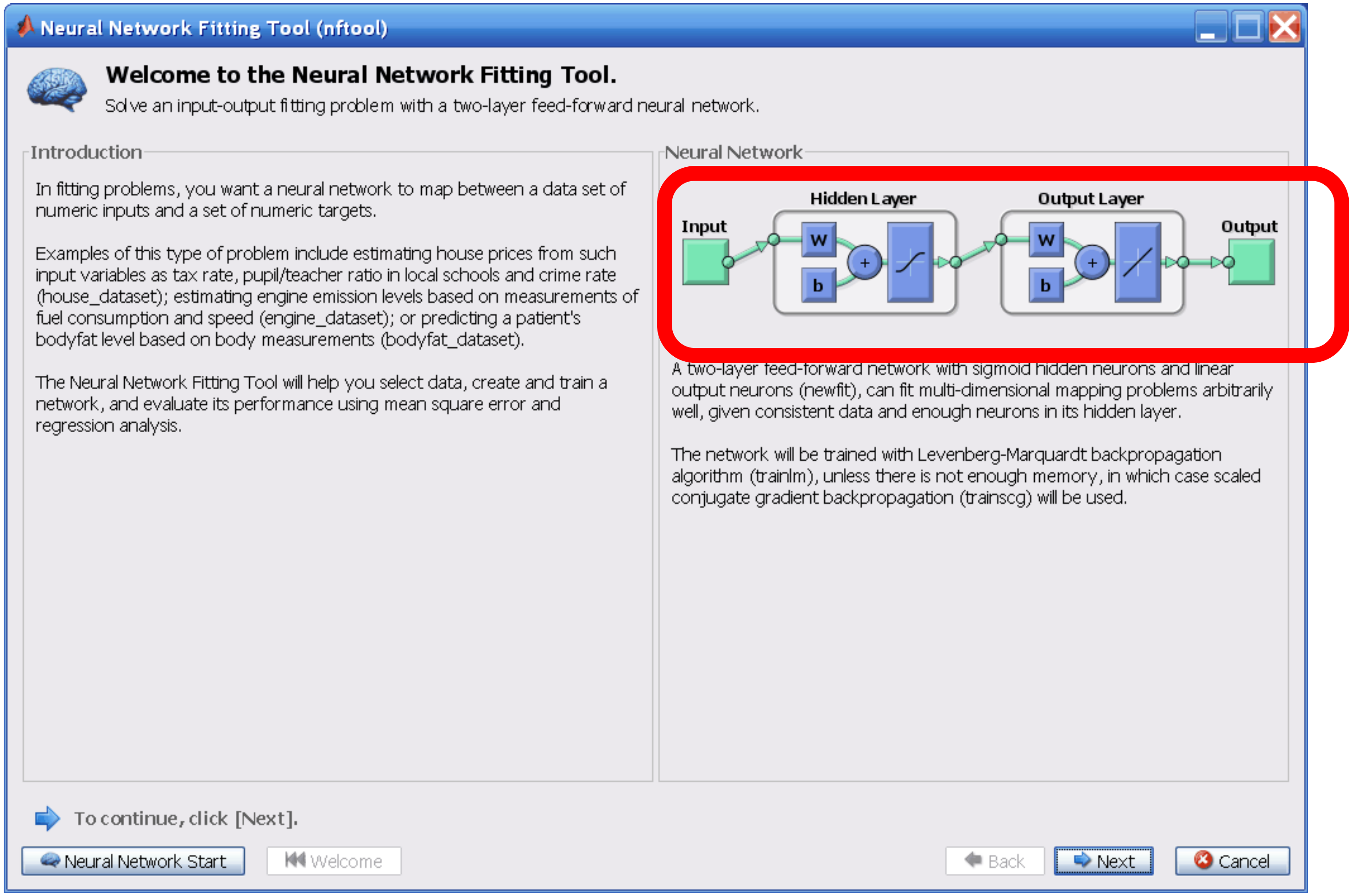
1. Loading data source
2. Selecting attributes required
3. Decide training, validation, and testing data
4. Data manipulations and Target generation
5. Neural Network creation (selection of network architecture) and initialisation
6. Network Training and Testing
7. Performance evaluation

Neural Network Fitting Tool: GUI

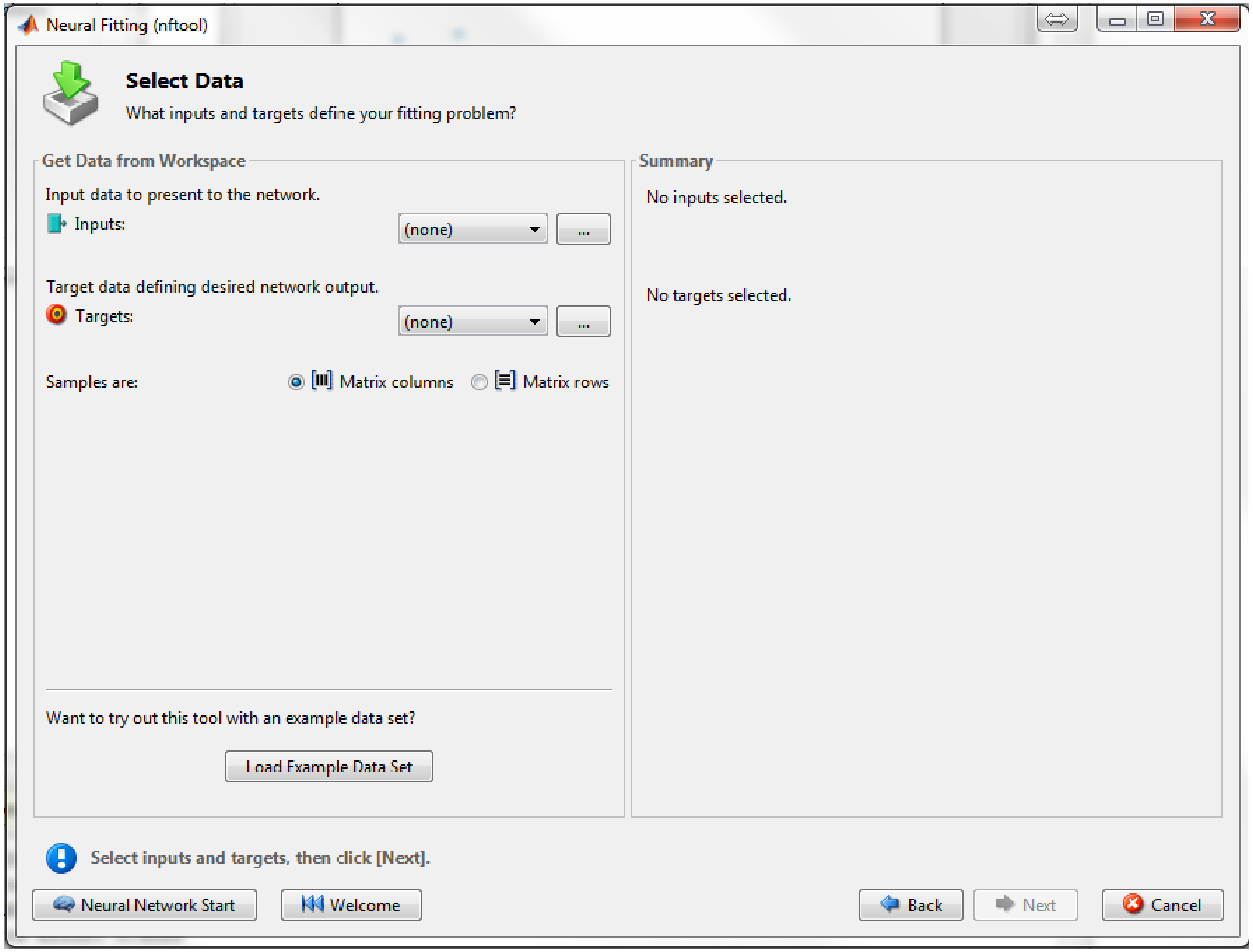
- nnstart



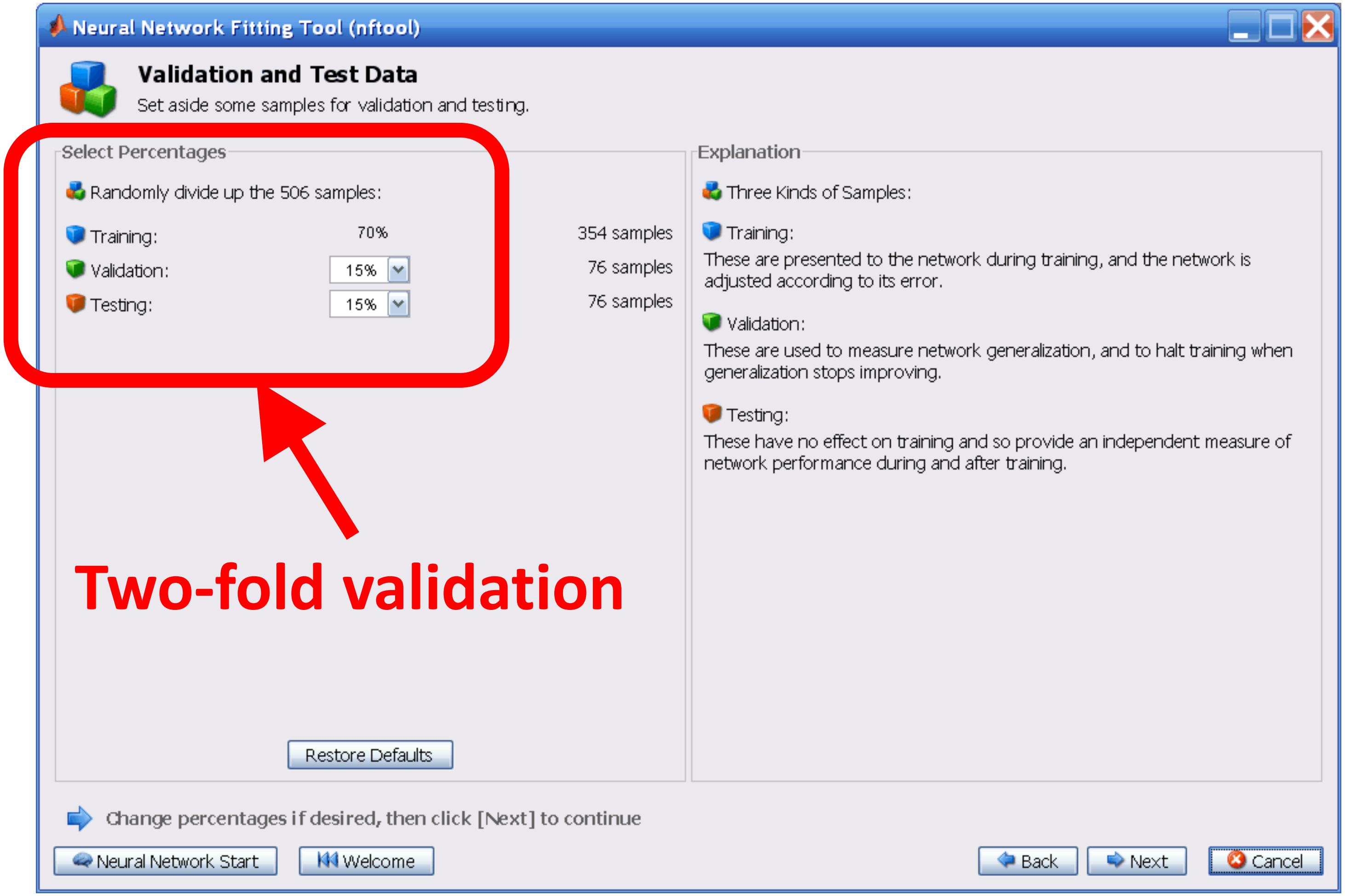
Neural Network Fitting Tool: GUI



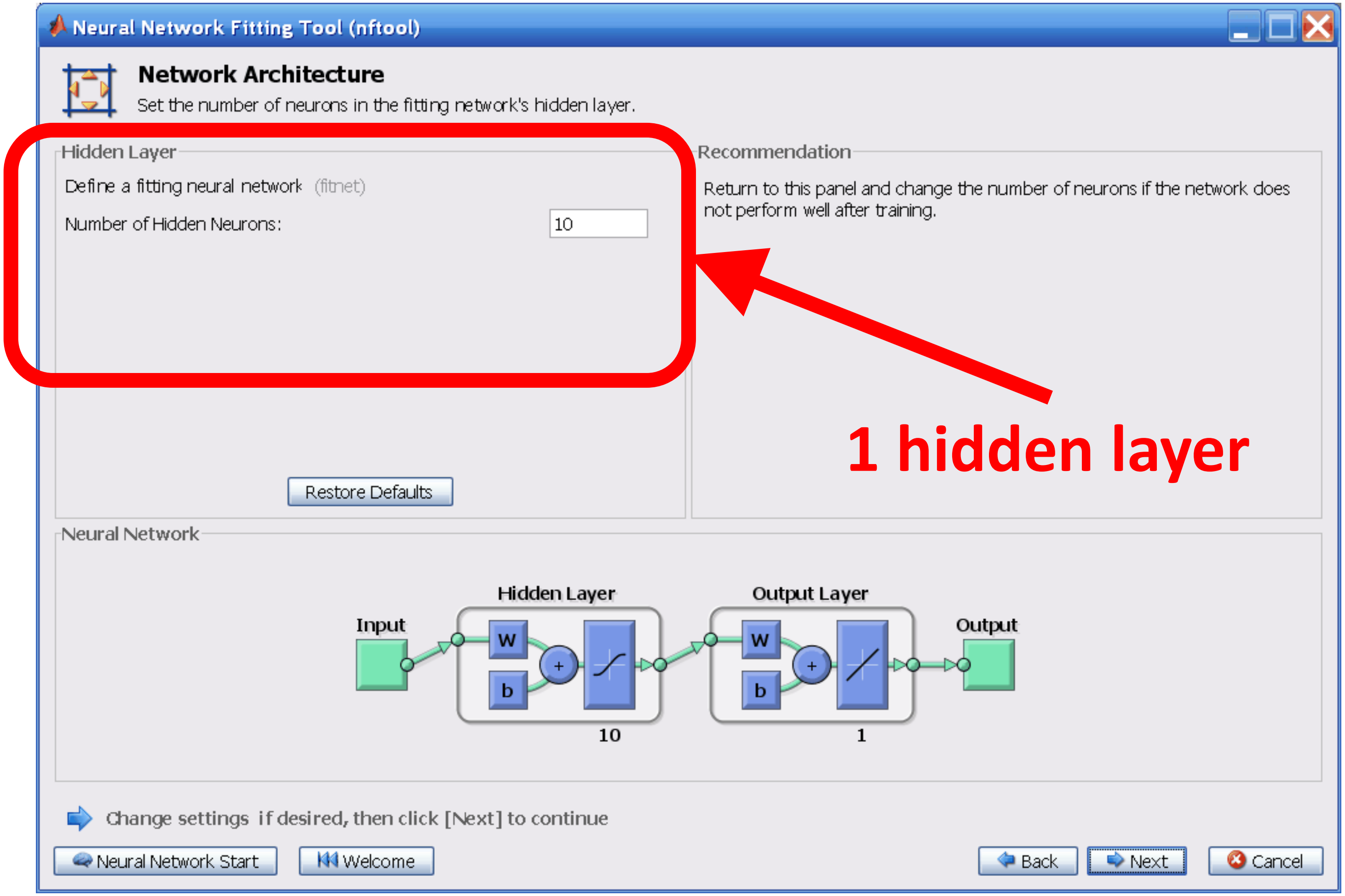
Neural Network Fitting Tool: GUI



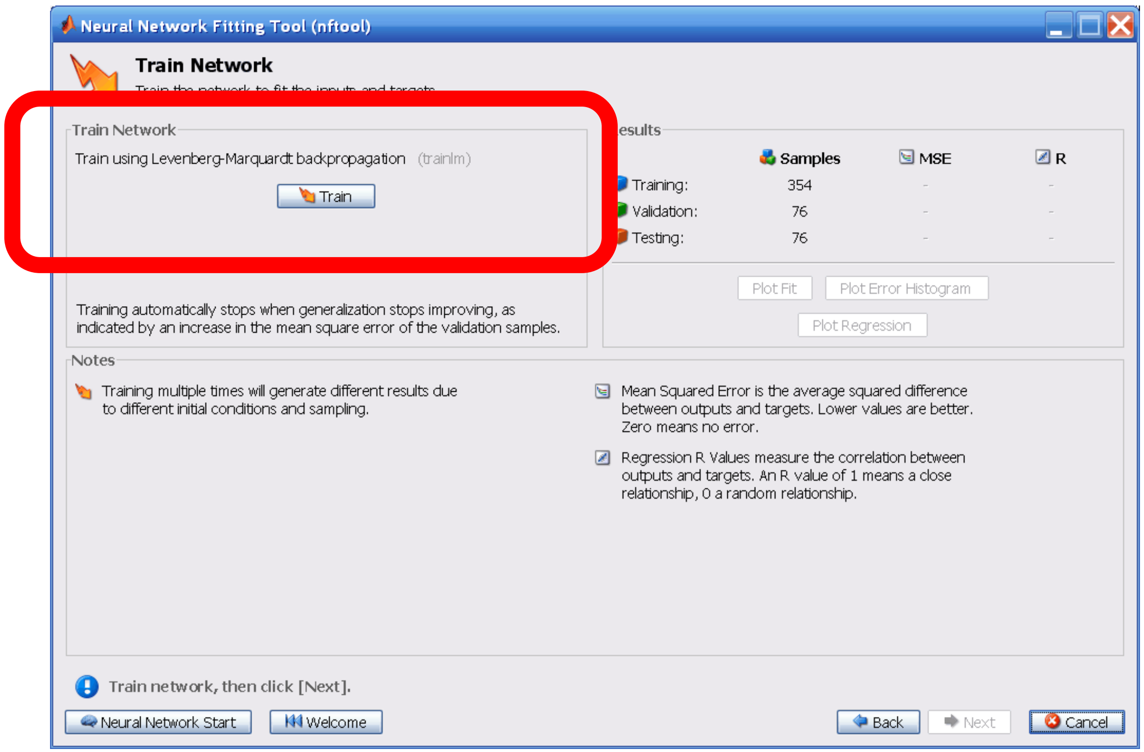
Neural Network Fitting Tool: GUI



Neural Network Fitting Tool: GUI

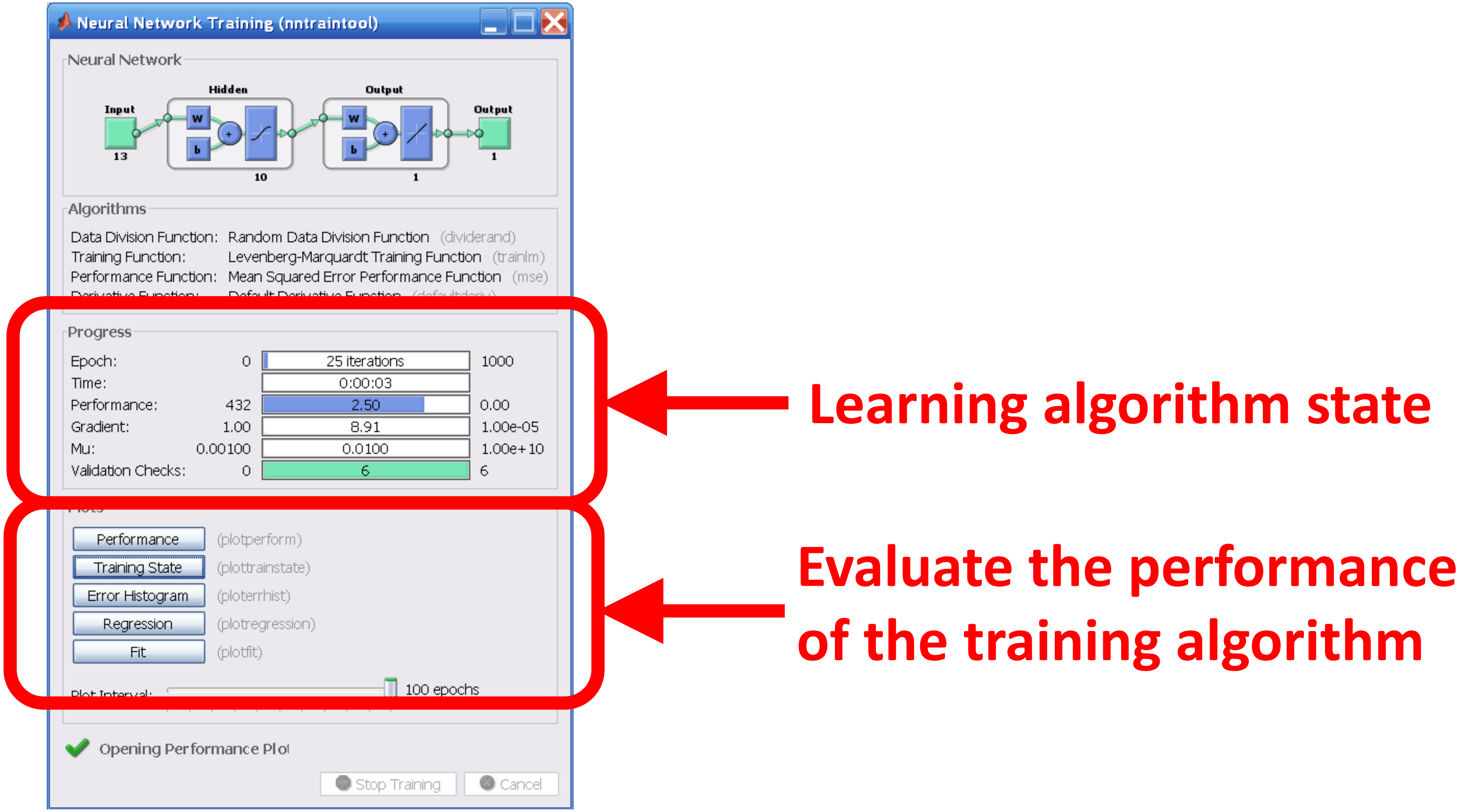


Neural Network Fitting Tool: GUI

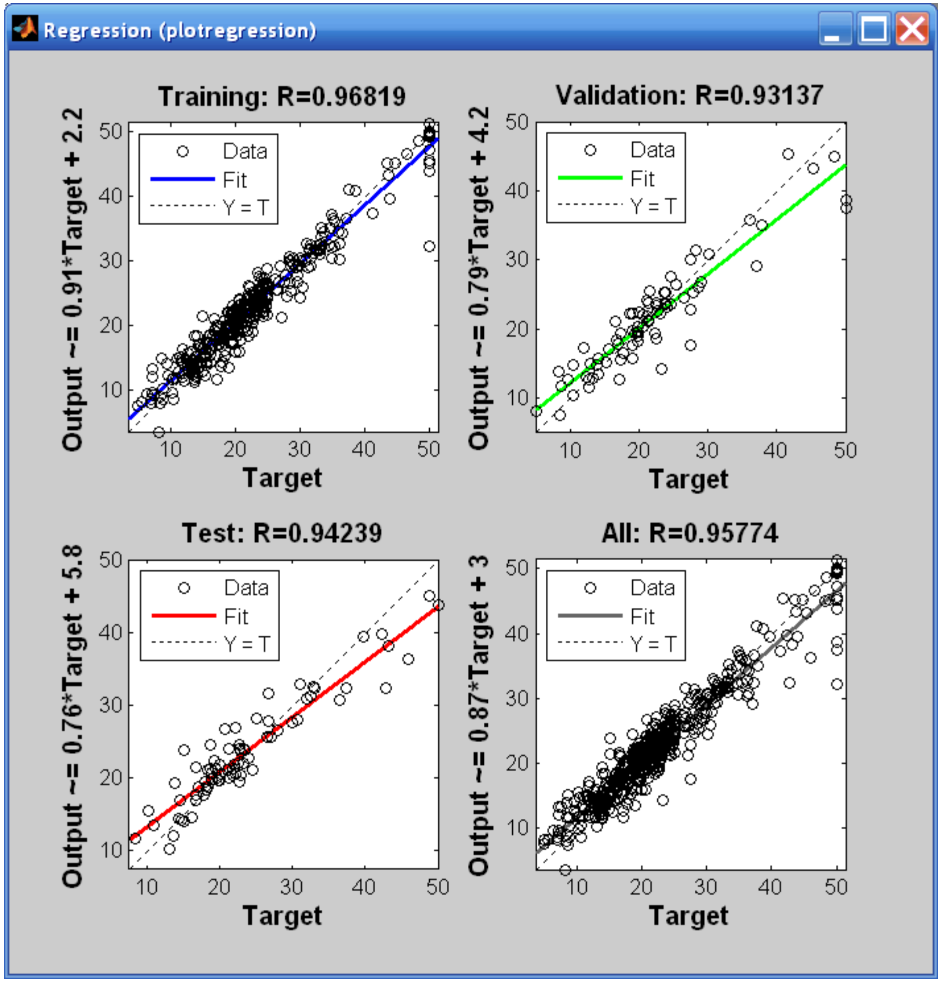


The **Levenberg–Marquardt** algorithm (LMA),also known as the damped least-squares (DLS) method, provides a numerical solution to the problem of minimizing a function, generally nonlinear, over a space of parameters of the function. These minimization problems arise especially in least squares curve fitting and nonlinear programming.

Neural Network Fitting Tool: GUI



Neural Network Fitting Tool: GUI



- Why four results?
- We need to evaluate:
  - learning capability
  - generalization capability

Example 1

- Consider humps(x) function in Matlab, is given by:

```
y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
```

- Build a neural network to fit the data generated by humps-function between [0,2]

Exercises

1. Obtain the neural network approximation of the signal

   x = (0: 0.1: 5);

   y=erf(x);

2. Obtain the neural network approximation of a sinusoidal signal

   x=(0:0.0001:0.05)

   y=sin(100*pi*x - 2*pi*0.75)

Neural Networks in real applications

- The GUI has been used only for discussing basic concepts

- In real applications, it is better to use command-line functions

Example 2

- One-dimensional fitting with command-line functions
  - 1Ddata.mat contains data describing the connections between the enzymes X and Y.
  - The vectors describing the enzymes are X_train and Y_train.
  - Feedforward neural networks should be used to learn the connections between the enzymes X and Y.
  - The generalization capability of trained neural networks should be evaluated on X_test and Y_test.

Note: download data from https://homes.di.unimi.it/munoz/teaching.html

Exercises

3. Re-run the script example1 and analyze the results
4. Try different numbers of epochs and plot the results in terms of validation MAE
5. Try different transfer functions and plot the results in terms of validation MAE
6. Try different numbers of neurons and plot the results and plot the results in terms of validation MAE
7. Try different numbers of hidden layers and plot the results in terms of validation MAE

Exercises

8. Two-dimensional fitting
   - Load 2Ddata.mat, it contains data describing the connections of the enzymes X1 and X2 with Y.
   - Use feedforward neural networks to learn the connections between the enzymes X and Y.
   - Try to optimize the parameters (epochs, transfer functions, number of neurons, number of layers) to obtain a low validation error
   - Plot the results using function scatter3 (training and test)

Suggestion: many parts of the script are similar to example2

Note: download data from https://homes.di.unimi.it/munoz/teaching.html

Divide into training and test set

- To validate a neural network model it is necessary to test the results on a dataset different from the training dataset
- In the previous exercises it was pre-divided
- Usually it is necessary to do divide the original dataset
  - Use of dividerand function

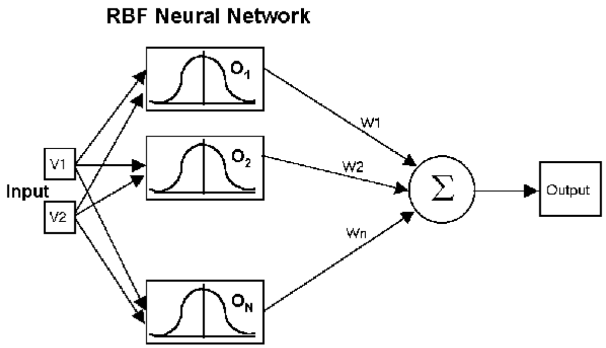
```
[trainInd, valInd, testInd] =
dividerand(sizeDB, trainProp, valProp, testProp);
P_train=P(:,trainInd);
T_train=T(:,trainInd);
P_test=P(:,testInd);
T_test=T(:,testInd);
```

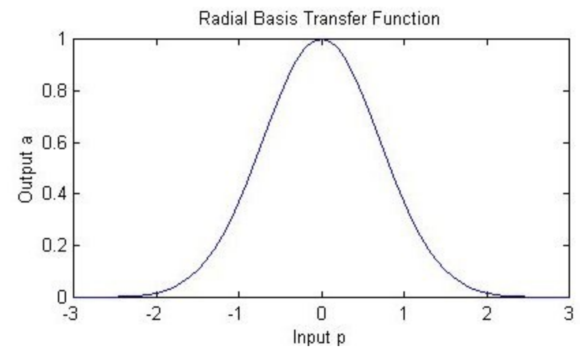
Exercises

9. N-dimensional fitting
   - Load bodyfat dataset (load('bodyfat_dataset.mat');)
   - Train a neural network to estimate the bodyfat of someone from various measurements. Bodyfat_dataset contains two variables
     - bodyfatInputs - a 13x252 matrix defining thirteen attributes for 252 people:
       - Age (years), Weight (lbs), Height (inches), Neck circumference (cm), Chest circumference (cm), Abdomen circumference (cm), Hip circumference (cm), Thigh circumference (cm), Knee circumference (cm), Ankle circumference (cm), Biceps (extended) circumference (cm), Forearm circumference (cm), Wrist circumference (cm)
     - bodyfatTargets - a 1x252 matrix of associated body fat percentages, to be estimated from the inputs.
   - Divide dataset into training and testing sets (use dividerand function)
   - Evaluate performance in terms of MAE
   - Plot original target values and predicted target values (for training and test sets)
   - Try to optimize the parameters (epochs, transfer functions, number of neurons, number of layers)

Radial Basis Function networks (RBF)

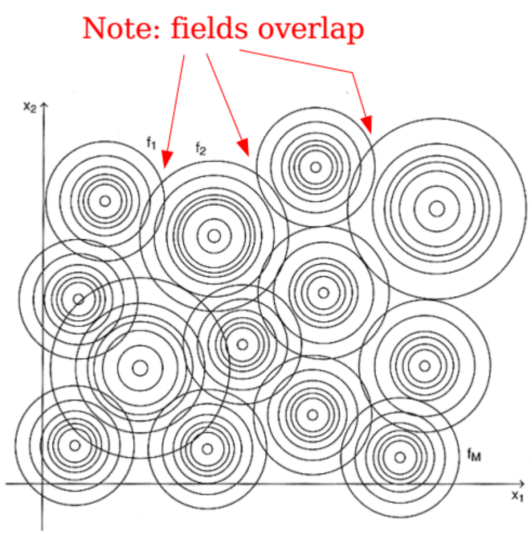
- A radial basis network is a network with two layers
- It consists of a hidden layer of radial basis neurons and an output layer of linear neurons
- The weights and biases of each neuron in the hidden layer define the position and width of a radial basis function
- Each linear output neuron forms a weighted sum of these radial basis functions
- With the correct weight and bias values for each layer, and enough hidden neurons, a radial basis network can fit any function with any desired accuracy





Radial Basis Function networks (RBF)

- Receptive fields overlap a bit, so there is usually more than one unit active.
- But for a given input, the total number of active units will be small.



Example 3

- Consider humps(x) function in Matlab, given by:

  ```
  y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
  ```

- Build a RBF network to fit the data generated by humps-function between [0,2]

Exercises

10. Surface reconstruction
    - Load doll.mat, it contains data describing the 3D reconstruction of a doll's face (P describes axes X and Y, T axes Z)
    - Divide dataset into training and testing sets (use dividerand function)
    - Build a RBF network to learn the connections between the P and T
    - Try to optimize the parameters (goal and spread) to obtain a low validation error
    - Plot the results using function scatter3 (training and test)

Note: download data from https://homes.di.unimi.it/munoz/teaching.html