EENG 428 Laboratory --- Lab Session 5

Description:

In this session, the early basics of neural networks are to be introduced.

Prerequisites:

Attending students are expected to know:

- Basic Matlab commands.
- DH representation.
- Forward kinematics for serial manipulators.

Contents:

1- Biological neurons.

2- The brain.

3- The artificial neuron.

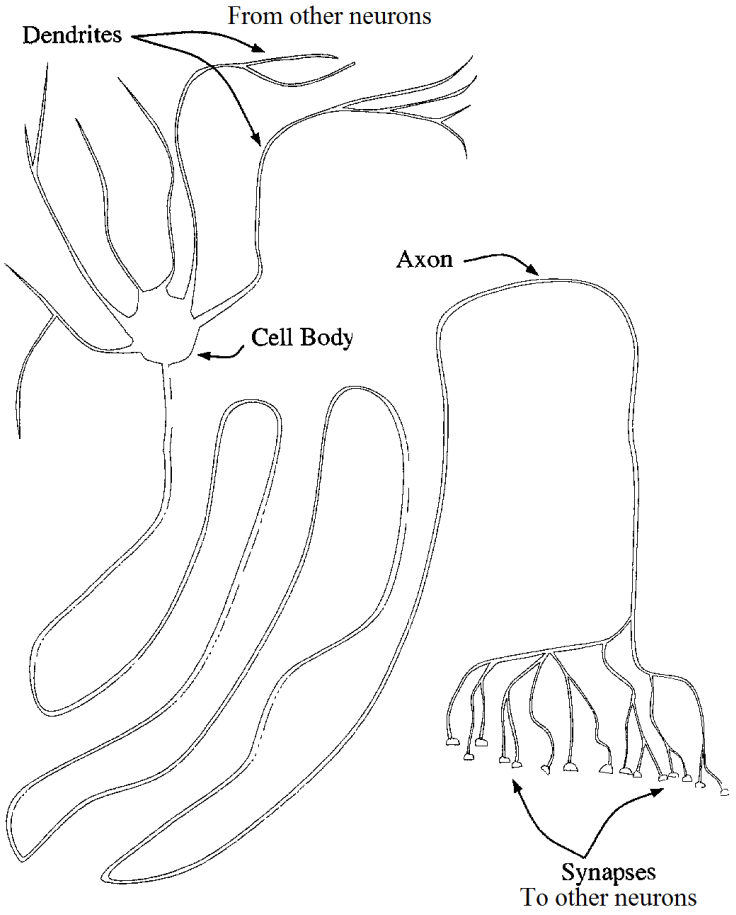
4- Types of artificial neural networks.

5- Multilayer feed forward artificial neural networks.

1- The Biological Neuron:

Basic facts:

- The neuron in the basic cell (unit) in any neural system.
- In a human brain, there are around 100 billion neurons.
- All the memories, the experiences, the skills and others… are stored in the brain, as a whole.
- Losing one neuron, does not effect a human, in fact, we lose about 190,000 neurons a day.
- Neurons do not renew (at all), which lead to the fact that we become more experienced day after day, but less intelligent.



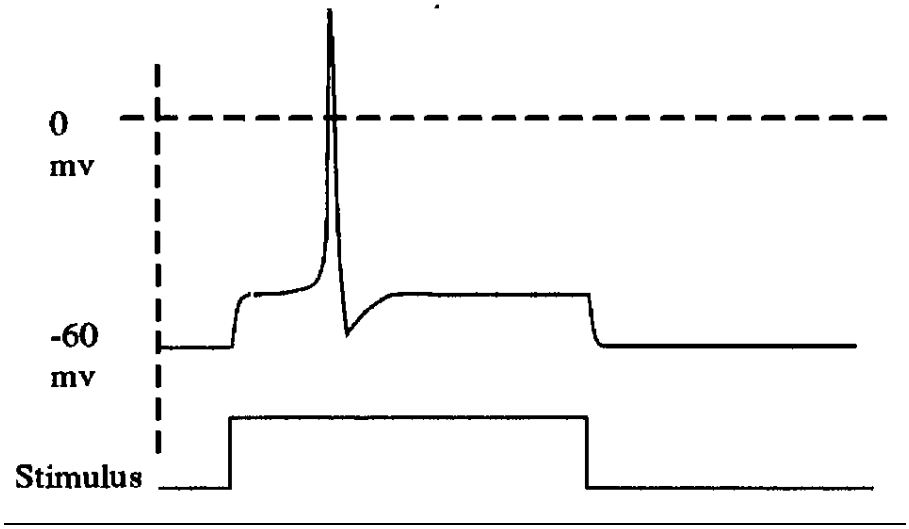
Neurons have different types, but they mostly consist of 3 parts. One part is the cell body, which is located in the brain (the head) (no cell bodies in the spinal cord). One part is the Dendrites, which are connections from other neurons to the

neuron body. And one part is the synapses which are connections from the neuron to other neurons or to muscles.

Fact:

Learning, is the process of strengthening the connections between different neurons in the brain. (Hebb's Rule or Hebbian Rule).

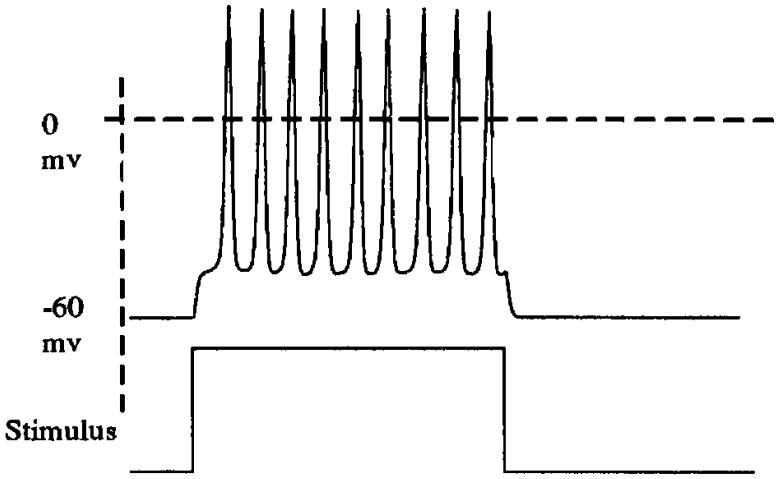
Stimulation of one neuron:



Firing

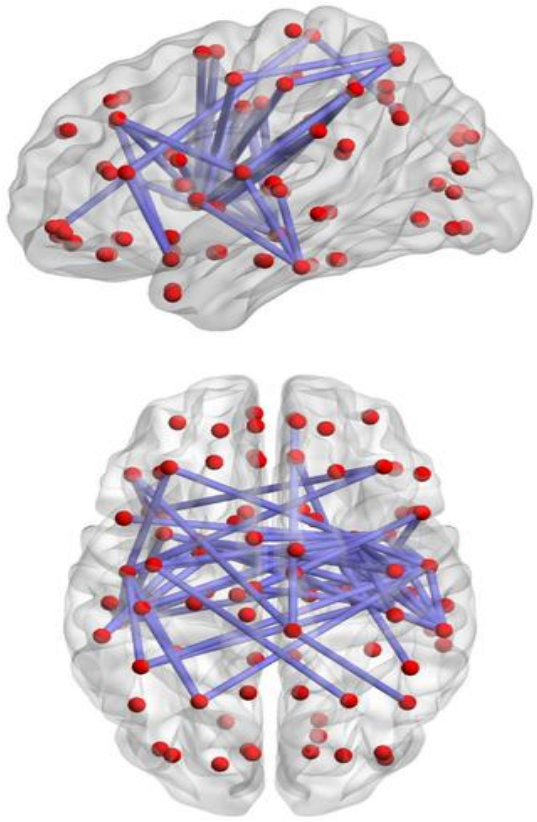
Facts:

- Increasing the stimulation, does not increase the feeling generated by one neuron (it might stimulate other neurons).
- Learning is the process of teaching a set of neurons to be stimulated simultaneously.
- The same neuron might have repeatable action of "firing", which is called bursting or spiking.



Spiking or Bursting

2- The brain:

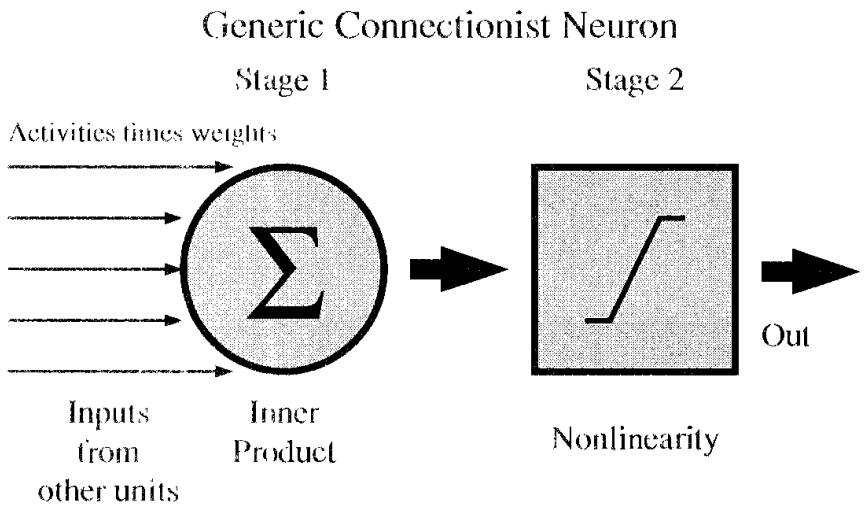


This photo simulates how different neurons in the brain are connected together

(which is the learning phenomena)

3- The artificial neuron:

Starting from the fact that a neuron can only fire or not. The artificial neuron takes input signals, and generate an output signal that is either zero or one.



The process of teaching (training) this neuron, is simply adjusting the weights for each input, that the result becomes as desired.

(see and, or, nor).

4- Artificial Neural Networks:

There are several types of neural networks, depending on the type of application.

For supervised learning, two types of networks are very familiar.

- Feed forward multilayer neural networks. (very useful in system identification or replication)
- Hopfield neural networks. (very common in clustering, such as character recognition)

For unsupervised learning, the most famous type is knows as:

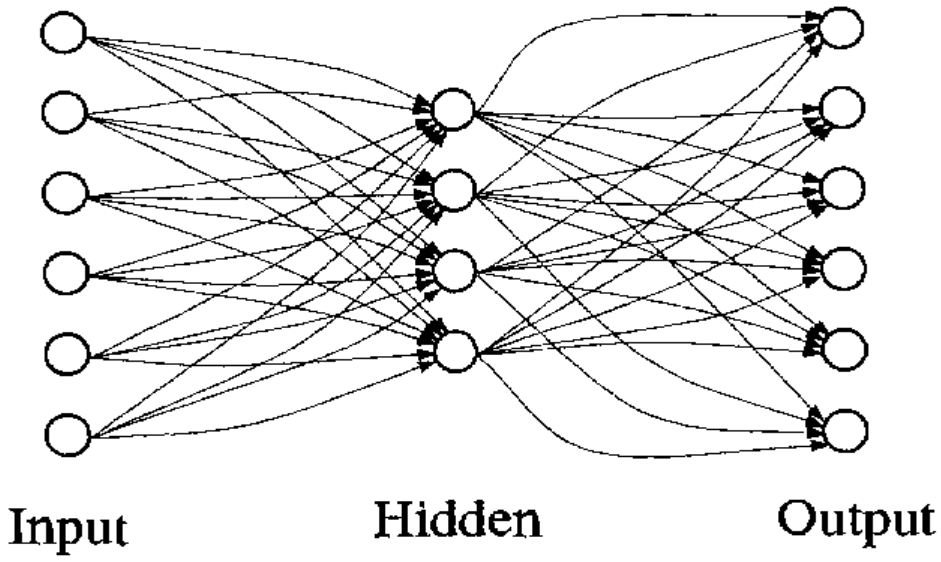
- Kohonen's neural network. (very dangerous applications such as chess engines or backgammon)

5- Multilayer feed forward artificial neural networks:

This network consist of 3 layers (at least). One layer is the input layer, one (or more) hidden layer(s), and one output layer.

The input layer is connected with the hidden layer, and the hidden layer is connected with the output layer.

The process of learning (training), is usually done by the "back-propagation algorithm", by providing a set of inputs and the desired set of outputs, and adjusting the weights of the network by minimizing the errors of the outputs using the "steepest decent optimization algorithm" .



For the mathematical background regarding training the network, see the handout for lab4 in the website.

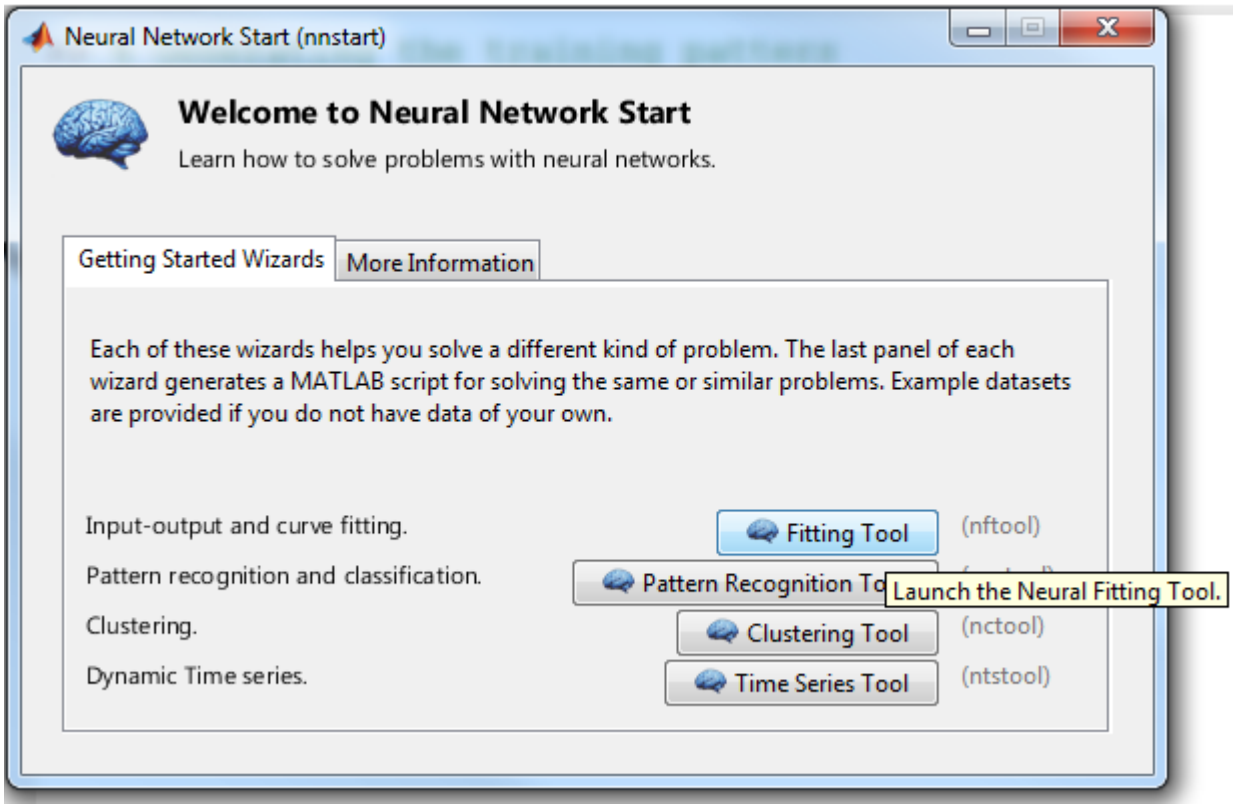
Applications on multilayer feed forward artificial neural networks:

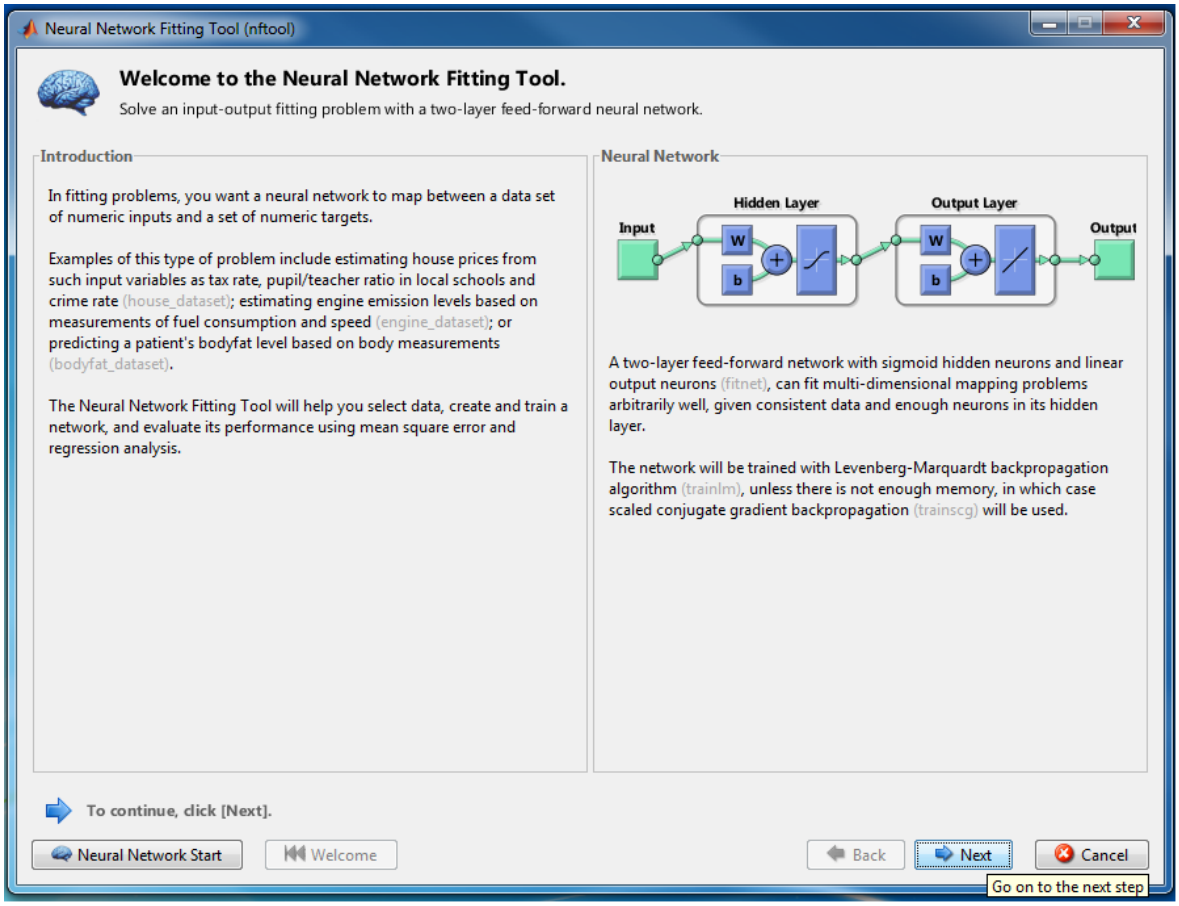
Example 1: (single input, single output)

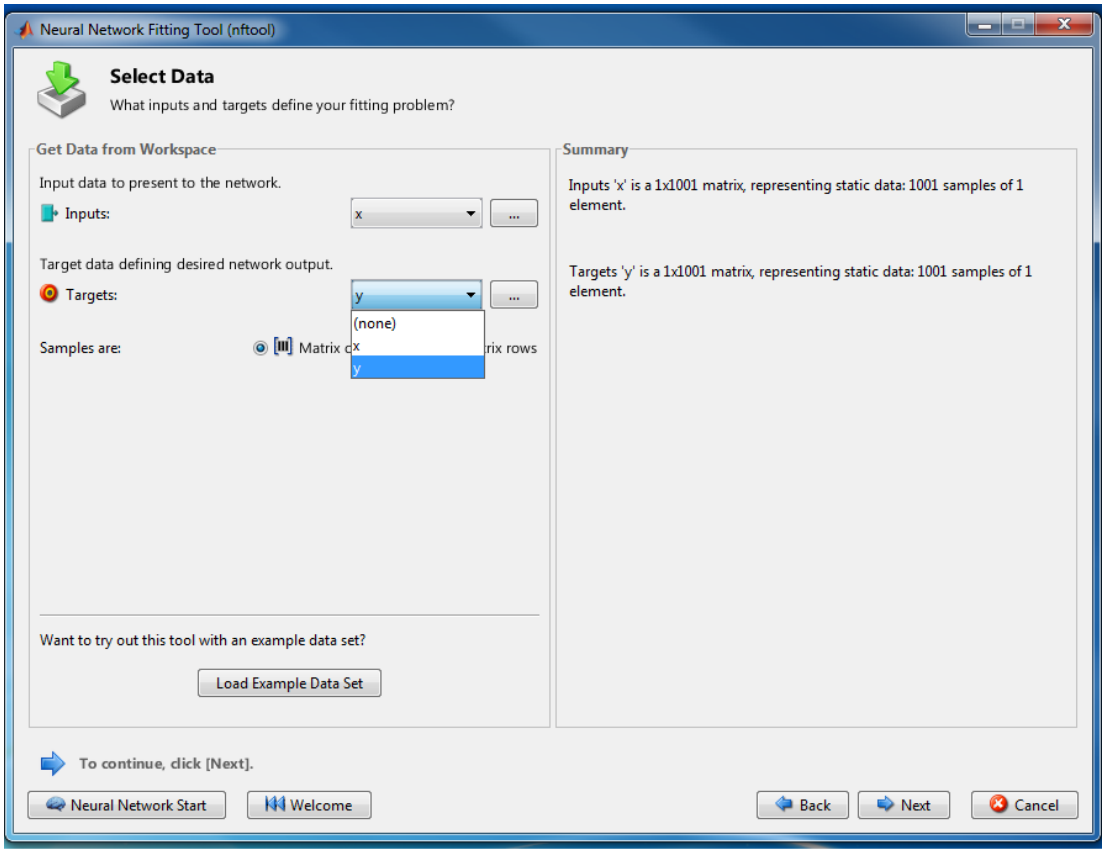
Train a network to implement the function Y= cos(2x)

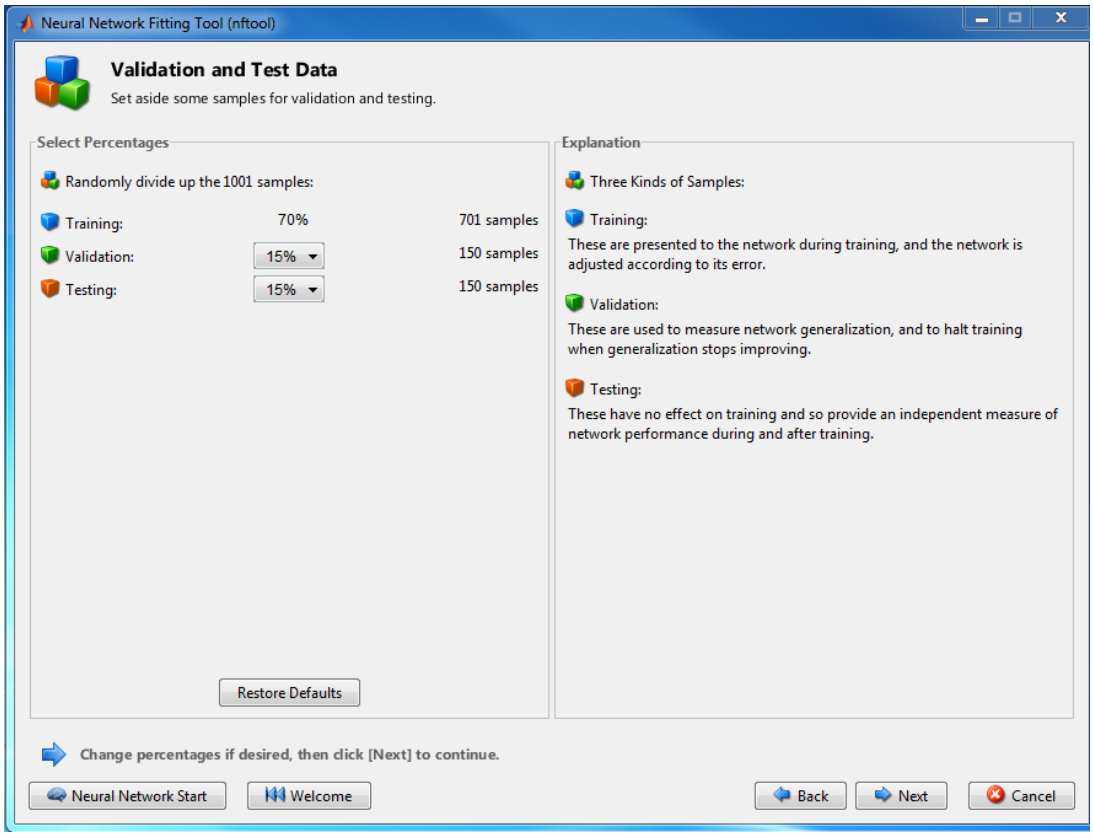
```
% Generating the training patters
x=-5:0.01:5;
y=cos(2*x);
```

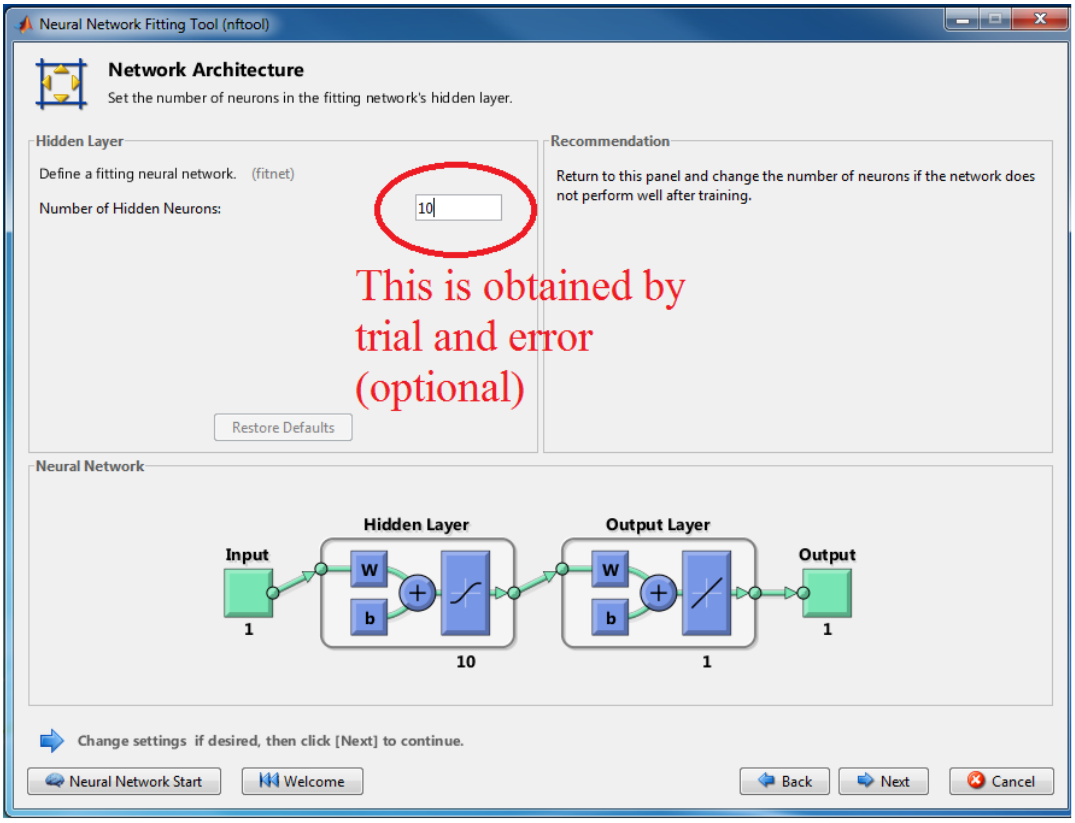
```
nnstart % starting the ANN toolbox (GUI)
```

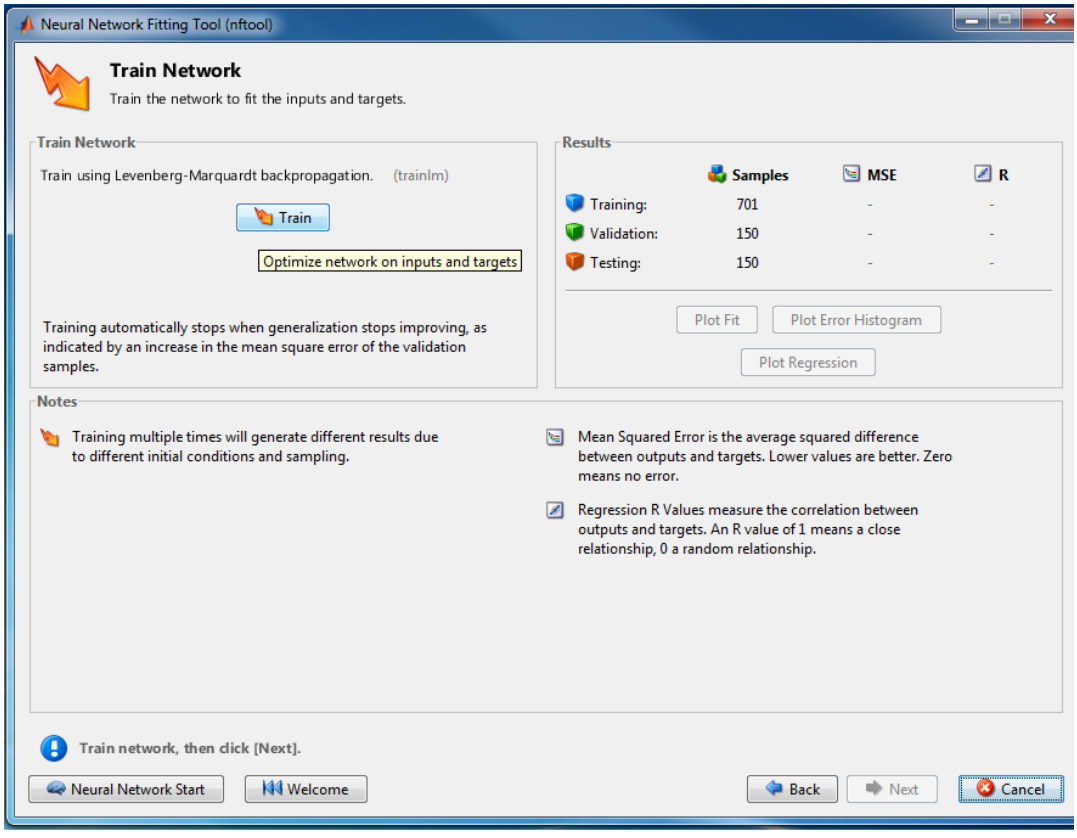

Neural Network Start (nnstart)
Welcome to Neural Network Start
Learn how to solve problems with neural networks.
Getting Started Wizards
More Information
Each of these wizards helps you solve a different kind of problem. The last panel of each wizard generates a MATLAB script for solving the same or similar problems. Example datasets are provided if you do not have data of your own.
Input-output and curve fitting.
Fitting Tool
(nftool)
Pattern recognition and classification.
Pattern Recognition To
Launch the Neural Fitting Tool.
Clustering.
Clustering Tool
(nctool)
Dynamic Time series.
Time Series Tool
(ntstool)

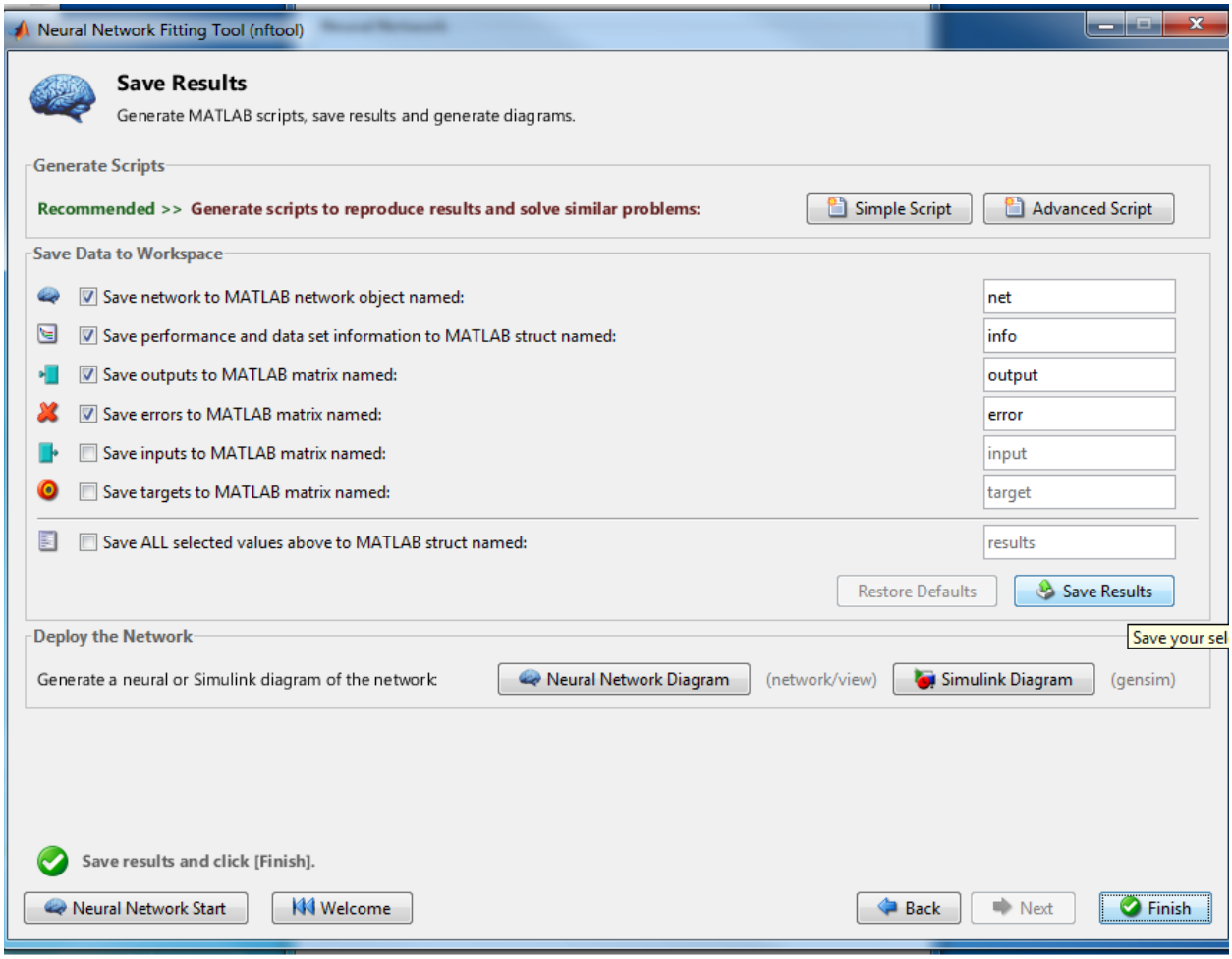

Neural Network Fitting Tool (nftool)
Welcome to the Neural Network Fitting Tool.
Solve an input-output fitting problem with a two-layer feed-forward neural network.
Introduction
In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.
Examples 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 (house_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
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
Go on to the next step


Neural Network Fitting Tool (nftool)
Select Data
What inputs and targets define your fitting problem?
Get Data from Workspace
Input data to present to the network.
Inputs:
x
Target data defining desired network output.
Targets:
y
(none)
x
y
Samples are:
Matrix c
rix rows
Want to try out this tool with an example data set?
Load Example Data Set
Summary
Inputs 'x' is a 1x1001 matrix, representing static data: 1001 samples of 1 element.
Targets 'y' is a 1x1001 matrix, representing static data: 1001 samples of 1 element.
To continue, click [Next].
Neural Network Start
Welcome
Back
Next
Cancel


Neural Network Fitting Tool (nftool)
Validation and Test Data
Set aside some samples for validation and testing.
Select Percentages
Randomly divide up the 1001 samples:
Training:
70%
701 samples
Validation:
15%
150 samples
Testing:
15%
150 samples
Restore Defaults
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.
Change percentages if desired, then click [Next] to continue.
Neural Network Start
Welcome
Back
Next
Cancel


Neural Network Fitting Tool (nftool)
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.
This is obtained by trial and error (optional)
Restore Defaults
Neural Network
Input
1
Hidden Layer
W
b
10
Output Layer
W
b
1
Output
1
Change settings if desired, then click [Next] to continue.
Neural Network Start
Welcome
Back
Next
Cancel


Neural Network Fitting Tool (nftool)
Train Network
Train the network to fit the inputs and targets.
Train Network
Train using Levenberg-Marquardt backpropagation. (trainlm)
Train
Optimize network on inputs and targets
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: 701 - -
Validation: 150 - -
Testing: 150 - -
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 Start
Welcome
Back
Next
Cancel




```
>> net(pi)
ans =
   1.0000
>>
```

It is obvious that we did not train for the value (pi). But the result is accurate (cos(2pi)=1)

Example 2: (several inputs, several outputs)

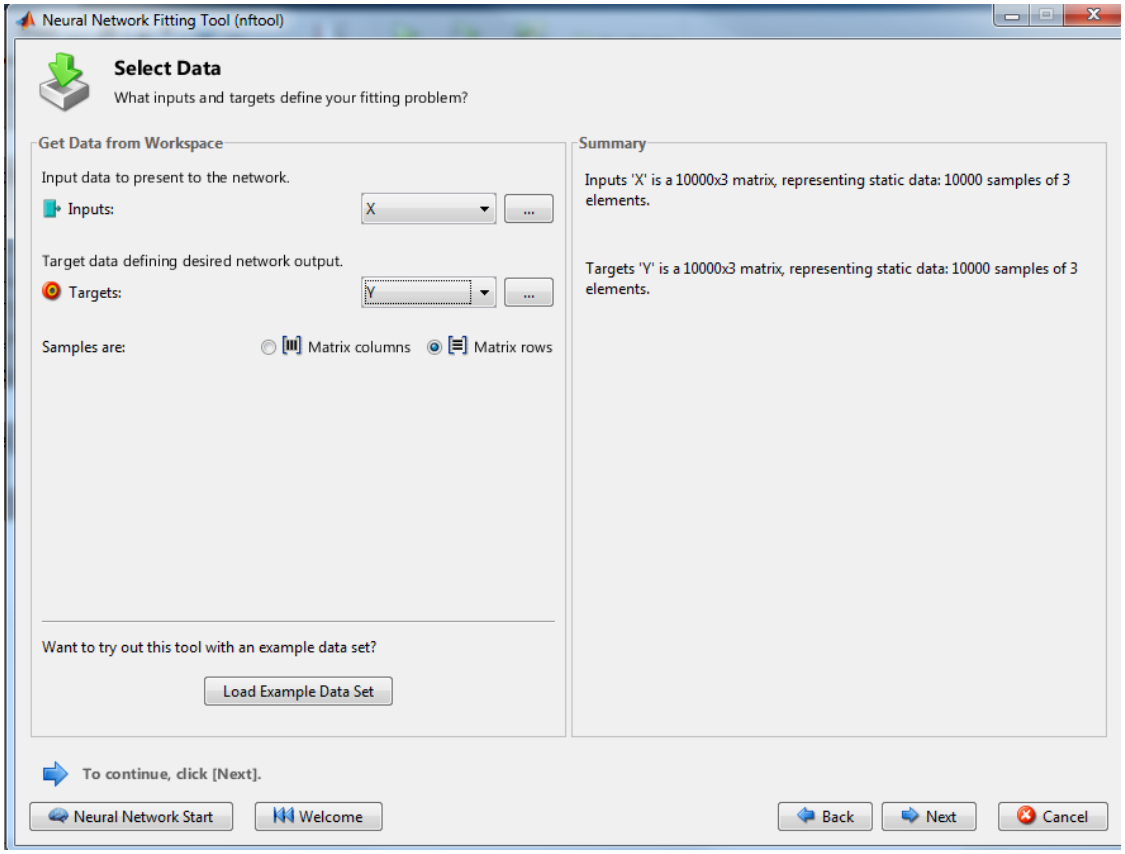
Train a multilayer neural network that can implement the following functions:

f1(x,y,z)= x^3 + 4y -6z

f2(x,y,z)= 5*cos(x) + 44y -16z

f3(x,y,z)= x^2 + y^2 –z

```
% Generating the training patters
x=10*randn(1,10000);
y=10*randn(1,10000);
z=10*randn(1,10000);
f1=x.^3+4*y-6*z;
f2=5*cos(x)+44*y-16*z;
f3=x.^2+y.^2-z;
%Obtaining the input/output vectors
X=[ x' y' z'];
Y=[f1' f2' f3'];
nnstart
```




```
F=net([2 2 2]');
g1=F(1)
g2=F(2)
g3=F(3)

f1=2^3+4*2-6*2
f2=5*cos(2)+44*2-16*2
f3=2^2+2^2-2
```

```
g1 =
   3.9688
g2 =
  56.1519
g3 =
   6.0216
f1 =
     4
f2 =
  53.9193
f3 =
     6
```

Homework:

Train a multilayer ANN that finds the eigenvalues of a 2*2 matrix.

(hint: Read the notes uploaded on the website)

Bonus: (200 points added to your total grade)

It is known that the forward kinematics problem is very simple in serial manipulators, while inverse kinematic problem is hard.

For a 3DOF manipulator of your choice:

1- Find the forward kinematic formula using DH representation. (T=A1*A2*A3)
2- Generate random joint variables values in acceptable range (joint limits).
3- Generate the corresponding position vector according for these joint variables. (ignore the orientation part)
4- Use the vectors in part 2 and 3 to train a multilayer feed forward artificial neural network to find the inverse kinematics (given the position, find the joint variables).
5- Verify your work by testing some output of the network (Partial solutions will not be accepted)

   Hint: see the research article uploaded on the website, and prepare similar work.

YOU HAVE EXACTLY 6 DAYS TO SUBMIT. SUBMISSION SHOULD BE

THE EMAIL: lab.eeng428@gmail.com