EEVAL 2016 – student problem

Stanisław Jankowski

Problem: Design a multilayer perceptron for function approximation *y*=*f*(*x*) based on a given training set.

Network structure: 2 input neurons: x and 1 (bias), one hidden layer of m neurons, one output neuron.

Activation function of hidden neurons :

$$g(h) = \tanh(h) = \frac{\exp(h) - \exp(-h)}{\exp(h) + \exp(-h)}$$

Activation function of output neuron: linear.

Generation of data sets:

$$y(x) = \frac{\sin x}{x} + \varepsilon$$

where: $x \in (0,15)$ and ε – random number uniformly distributed in the interval (-0.1, 0.1)

Data sets - pairs {*x*,*y*}:

- `snn_a.txt`, - training set: 40 examples {*x*,*y*}uniformly distributed in the interval (0, 15)

`0.1·rand(40,1)`

- `snn_b.txt`. - test set: 400 examples {*x*,*y*} uniformly distributed in the interval (0, 15).

`0.1·rand(400,1)`

Learning method:

Goal: minimization of mean square error of function approximation.

Network structure: Select optimal number of hidden neurons based on the resulting training and test set mean square errors..

Weight initialization: random numbers in the interval [–0.15, 0.15]. Perform 20 network simulations for various initial conditions and select the best solution.

Learning algorithm: Levenberg-Marquardt algorithm 200 iterations (`trainlm`)

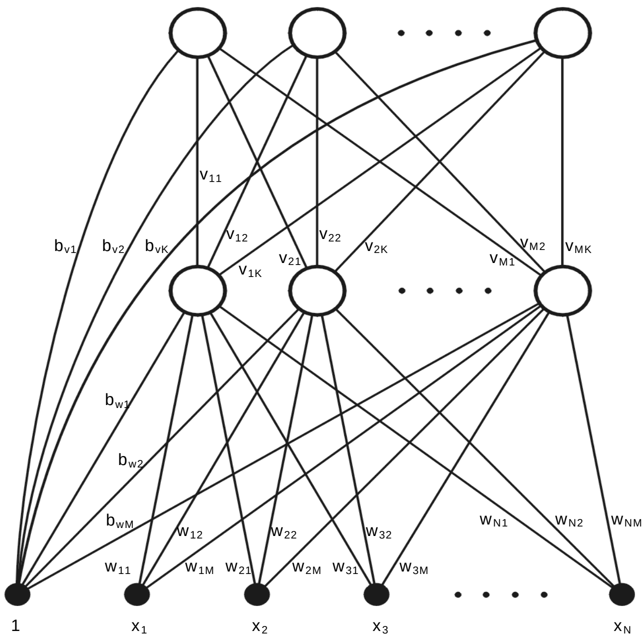

Fig.. 1. General scheme of multilayer perceptron.

Report:.

The report - pdf form, max. 4 pages A4 - should be sent to:
Stanisław Jankowski sjank@ise.pw.edu.pl

Content

1. Goal: multilayer perceptron for function approximation.
2. Data description and visualization
3. Neural network scheme – the best case
4. Result of approximation: mean square error (mse). Present the training set mse and test set mse as function of the number of hidden neurons table:

Number of hidden neurons	Training set mse	Test set mse
1 2 … m		

5. Plot of the best neural approximation.
6. Comments:
   a) information about the software tools;
   b) validation of the network structure (number of hidden neurons) based on simulation results, relation to the function complexity (shape).
7. Conlusions.
8. References (toolbox).

Neural network learning

```
function [net]= train_net(train_set,labels,hidden_neurons_count)
    %    %Parameters:
    %   train_set:
    %   labels - y
    %   hidden_neurons_count:
    %Return value:
    %   net – object representing a neural network

    %initialization
    %hidden neuron activation function- tanh,
    % output neuron activation - linear
    %
    net=newff(train_set',labels',hidden_neurons_count,...
              {'tansig', 'purelin'},'trainlm');

    rand('state',sum(100*clock));   %random numbers generator initialization
    net=init(net);                          %weights initialization
    net.trainParam.goal = 0.01;             %stop- mse criterion
    net.trainParam.epochs = 400;            %number of epochs iterations
    net=train(net,train_set',labels');      %network training
```