<u>Programming task 1</u>

Approximation and Prediction
With Feedforward Neural Networks

Multilayer Feedforward Backpropagation

The programming tasks should be accounted for as short reports. You are allowed to work in groups of a maximum of two persons. If you have any problems feel free to ask me. Also discussions with other groups are appreciated and welcome.

a) Our aim is to solve a set of linear equations by the use of a single layer feedforward Neural Network.

Consider the following set of equations

$$Ax = b$$

A is an (m-by-k) matrix and b is an (m-by-1) vector.

This set of equations have either infinite number of solutions or one unique solution or no solutions. We want to find the pseudo-inverse solution to this set of equations.

- Define the appropriate training sequence and train a single layer linear neural network to find the pseudo-inverse of the matrix A by choosing a random matrix of size (30-by-10) and a random vector b of size (30-by-1). Use Gaussian distribution of the random elements.
- During training plot the norm of the error (which is a measure of how well we have approximated the solution)

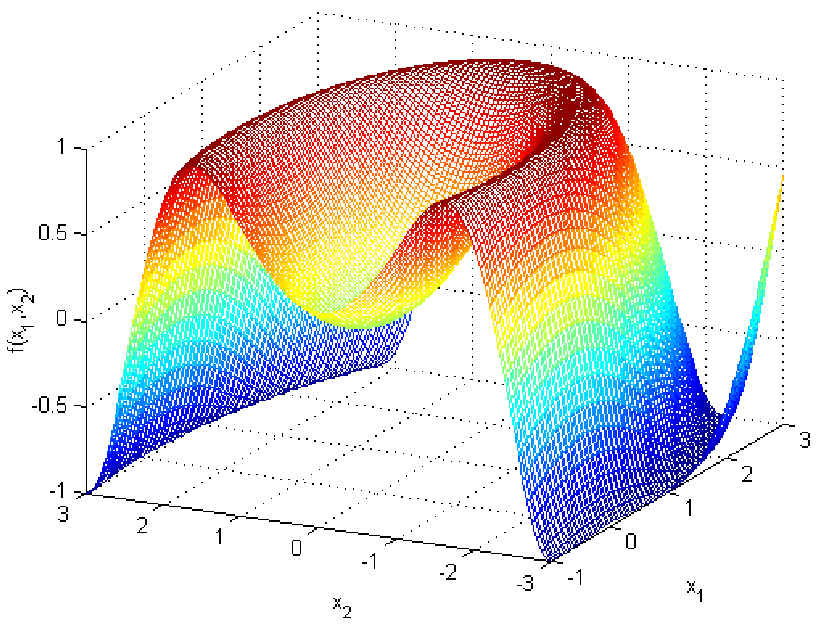
  $$\|Ax - b\| \qquad \text{where } x = A^{+}b$$

  Use an appropriate stopping criteria.
- Calculate the error norm by using Matlab's pseudo-inverse (pinv).
- Calculate the norm of the difference between the final weights W^T and the pseudo-inverse given by Matlab.
- Use a decreasing stepsize alfa during batches.

b) In this task we want to approximate a given nonlinear function, (which is the same as non-linear system identification), with a 3 layer feedforward neural network. The network should have a configuration of 2-5-4-1, which means 2 input sites and 5 neurons in first hidden layer, 4 neurons in the second hidden layer and one output neuron. All neurons should have the bipolar sigmoid functions.
The function to be approximated is

$$f(x_1, x_2) = \sin\left(\tfrac{x_1^2}{4} + \tfrac{x_2^2}{2}\right) \qquad , x_1 \in [-1,3], x_2 \in [-3,3]$$

The function has a "sofa" like shape and the plot below shows the function.



The input consists of the two parameters x_1 and x_2 which should lie in the specified input region.

- Create training sequences by sampling the region where the inputs are specified and use the specified function to create the target sequence.
- Train the network using back-propagation with this training sequence and plot the error during training. Some averaging of the error can be appropriate for the plot. Use an appropriate stopping criteria. Use bias terms in the weights.
- Make a plot similar to the one above using the neural network's output. This means that the weights should be kept fixed when the input is chosen from the specified input region.
- Choose different stepsizes and make a comment on the sensitivity of the choices.
- Use a momentum term (choose any appropriate value) when updating the weights and plot the error during training. Some averaging of the error can be appropriate for the plot. Is the convergence speed higher with momentum?

c) In this task we wish to make a prediction of a non-linear system. Consider the following system

$$y(n) = \sqrt{0.1x(n) + 0.5x(n-1)} + 0.3x(n-2)^2 + \sin(x(n-3)).$$

The system is artificially created and a I do not know anything about its behaviour. Our aim is to try to predict one step ahead given past values of the input and past values of the output.

- Create a noise sequence as input. Create the corresponding output from the non-linear system.
- Use a 3 layer neural network with 10 bipolar sigmoid units in the first hidden layer, 5 bipolar sigmoid units in the second hidden layer and a single linear neuron in the output. Use a TDL (Tapped Delay Line) of size 4 to create the input vector from the sequence x(n).
- Train the network to predict the output sequence y(n+1) given the input sequence x(n), x(n-1), x(n-2), x(n-3) and the output y(n), y(n-1), y(n-2), y(n-3), …., during the sequence presentation, that is, on-line through the sequence. Use bias terms in the weights.
- Make a averaged plot of the error
- Save the weights on a disk after training, plot the weights from the output layer. I will use this information to see if there are many local minimas in the error surface.