ISTANBUL TECHNICAL UNIVERSITY

FACULTY of ELECTRICAL and ELECTRONICS ENGINEERING

ARTIFICIAL NEURAL NETWORKS AND SYSTEM IDENTIFICATION FOR CONTROL

Homework - II

Prof. Serhat Şeker

Eren Çakmak - 504181134

2 April 2019

Introduction

Aim of the first homework was to train a perceptron. Now, in this homework, a multilayered artificial neural network is wanted to be constructed as it is seen in the Figure 1. The objective of this network is to predict the incoming data with respect to input data. The data is derived from sine wave

$$y = \sin(2\pi f t),$$

Where frequency is denoted as f which equals to 50 Hz, and time is between 0 to 10 seconds which is denoted as t. Number of input data is selected as 10 and number of output data (predictions) is selected as 10. Thus, the network is auto-associated neural network. Number of the hidden layers is 4 and the number of node is 10, 12, 15, 12, 10 respectively.

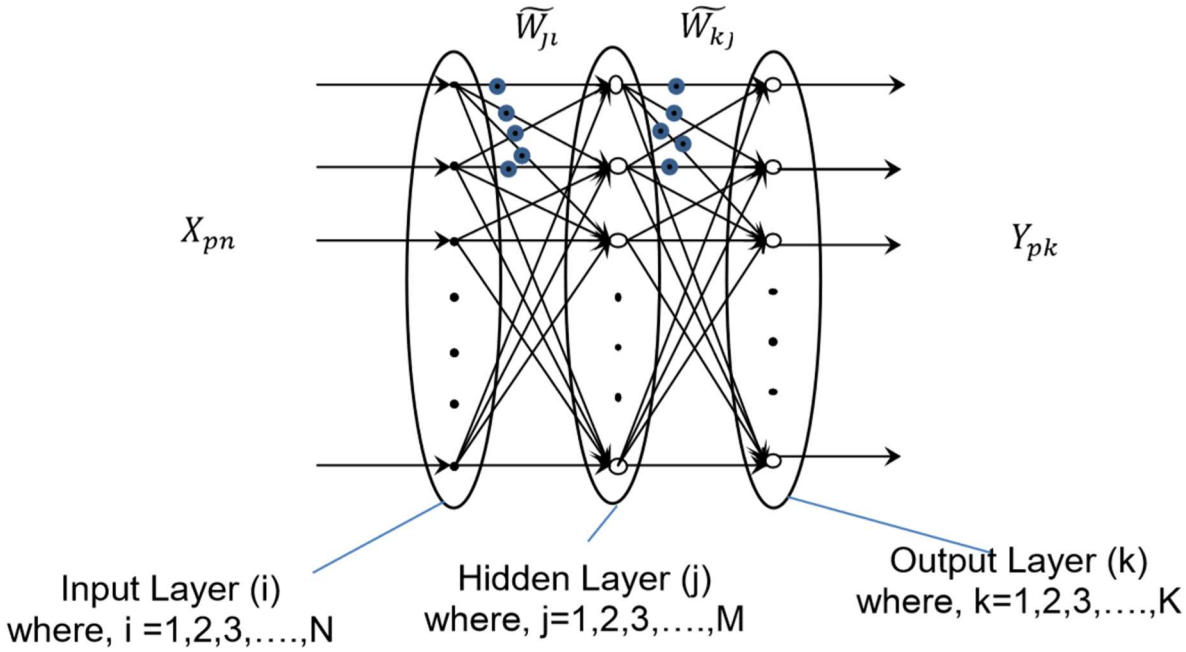

Figure 1: Multilayer Fully Connected Neural Network.

Standard normal distribution is added to the training data as

$$x = \sin(100\pi t) + 0.01 * \mathcal{N}(0,1), \qquad 0 \le t \le 10.$$

Because the input data have to be is discrete, input matrix (denoted as X) and the desired output matrix (denoted as T) is derived as follows,

$$X_{10xk} = \begin{pmatrix} x[n] & x[n+1] & \cdots & x[n+k] \\ x[n+1] & x[n+2] & \cdots & x[n+k+1] \\ \vdots & \vdots & \cdots & \vdots \\ x[n+9] & x[n+10] & \cdots & x[n+k+9] \end{pmatrix},$$

$$T_{10xk} = \begin{pmatrix} x[n+10] & x[n+11] & \cdots & x[n+k+10] \\ x[n+11] & x[n+12] & \cdots & x[n+k+11] \\ \vdots & \vdots & \cdots & \vdots \\ x[n+19] & x[n+20] & \cdots & x[n+k+19] \end{pmatrix},$$

Where n is the starting time, which is zero for our study.

The adaptive learning algorithm is chosen as the Backpropagation algorithm. The noiseless sine wave is used as validation data to observe the validation error.

The code is written in python 2.7. The code is can be accessed from my github repository: https://github.com/erenleicter/ANN-Sys-Iden-Cont.

An Update for the Backpropagation Algorithm

In the first homework the backpropagation algorithm is derived. But it is just for one perceptron and needs to be updated. Fortunately, all the steps are almost same for every perceptron.

Remember that the derivative of the total error with respect to weights of the perceptron $\frac{\partial E}{\partial w_i}$ was

$$\frac{\partial E_p}{\partial w_i} = \frac{\partial E_p}{\partial e_p}\frac{\partial e_p}{\partial y_p}\frac{\partial y_p}{\partial V_p}\frac{\partial V_p}{\partial w_i}.$$

E_p: E for p^{th} vector in a batch,

e_p: error for p^{th} vector in a batch,

y_p: output of the network for p^{th} vector in a batch,

V_p: net input of nonlinear (activation) function for p^{th} vector in a batch,

E_p: E for p^{th} vector in a batch,

w_i: i^{th} weight,

x_i: i^{th} value of p^{th} vector in a batch.

If the derivative of total error is calculated with respect to weights of the weight of the last hidden layer, $\frac{\partial E}{\partial w_{kj}}$ is found as,

$$\frac{\partial E^p}{\partial w_{kj}} = \frac{\partial E^p}{\partial e^p}\frac{\partial e^p}{\partial y_k^p}\frac{\partial y_k^p}{\partial V_k^p}\frac{\partial V_k^p}{\partial w_{kj}},$$

If the output of the hidden layers is denoted as O, the equation becomes,

$$\frac{\partial E^p}{\partial w_{kj}} = -\delta_k O_j^p.$$

$$\delta_k = -\frac{\partial E^p}{\partial e^p}\frac{\partial e^p}{\partial y_k^p}\frac{\partial y_k^p}{\partial V_k^p} = e_p f'(V_k^p)$$

To find $\frac{\partial E}{\partial w_{kj-1}}$, we use δ_k as follows,

$$\frac{\partial E^p}{\partial w_{kj-1}} = \frac{\partial E^p}{\partial V_k^p}\frac{\partial V_k^p}{\partial O_j^p}\frac{\partial O_j^p}{\partial V_j^p}\frac{\partial V_j^p}{\partial w_{kj-1}},$$

$$\frac{\partial E^p}{\partial w_{kj-1}} = -\delta_j O^p_{j-1},$$

$$\delta_j = \sum_k \left(\delta_k w_{kj}\right) f'\left(V^p_j\right).$$

Finally, $\frac{\partial E^p}{\partial w_{ji}}$ becomes as follows,

$$\frac{\partial E^p}{\partial w_{j-n,i}} = \frac{\partial E^p}{\partial V^p_{j-(n-1)}} \frac{\partial V^p_{j-(n-1)}}{\partial O^p_{j-n}} \frac{\partial O^p_{j-n}}{\partial V^p_{j-n}} \frac{\partial V^p_{j-n}}{\partial w_{j-n,i}},$$

$$\frac{\partial E^p}{\partial w_{j-n,i}} = -\delta_{j-n} O^p_{j-n},$$

$$\delta_{j-n} = \sum_j \left(\delta_{j-(n+1)} w_{j-(n+1),j-n}\right) f'\left(V^p_{j-(n+1)}\right)$$

To summarize, updates of the weights become as

$$\Delta_p w_{kj} = \eta \delta_k x^p_i,$$

$$\Delta_p w_{j+1,j} = \eta \delta_{j+1} O^p_{j+1},$$

$$\Delta_p w_{ji} = \eta \delta_j x^p_i,$$

and the δ of nodes are given as

$$\delta_k = e_p f'\left(V^p_k\right),$$

$$\delta_j = \begin{cases} \sum_j \left(\delta_{j+1} w_{j+1,j}\right) f'\left(V^p_j\right), & \textit{if } j \textit{ is not the last hidden layer} \\ \sum_k \left(\delta_k w_{kj}\right) f'\left(V^p_j\right), & \textit{if } j \textit{ is the last hidden layer} \end{cases}$$

Results

In the first homework, sigmoid function is selected as the non-linear function. However, because the sine wave changes between the interval -1 and 1, hyperbolic tangent function ($\tanh(x)$) is selected as the non-linear function. As it is depicted in Figure 2, tanh function takes value between the interval -1 and 1.

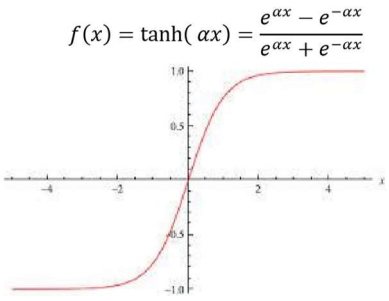

Figure 2: Hyperbolic tangent function.

Learning rate is selected as 0.01 and the network trained for 50,000 iterations. Results given in the Figure 3. At the end of the training, total error found as 0.0939 and validation error found as 3.4023. At the testing stage, 10 sample from a noisy sine wave is given to the network. As it is understood from the figure, predicted values (output data) is more likely to ideal sine wave.

These results shows that even if the data is noisy, the neural network is successful to predict the incoming data. Moreover, the network improves the data. Thus, this network can be used as a filter as well as a predictor.

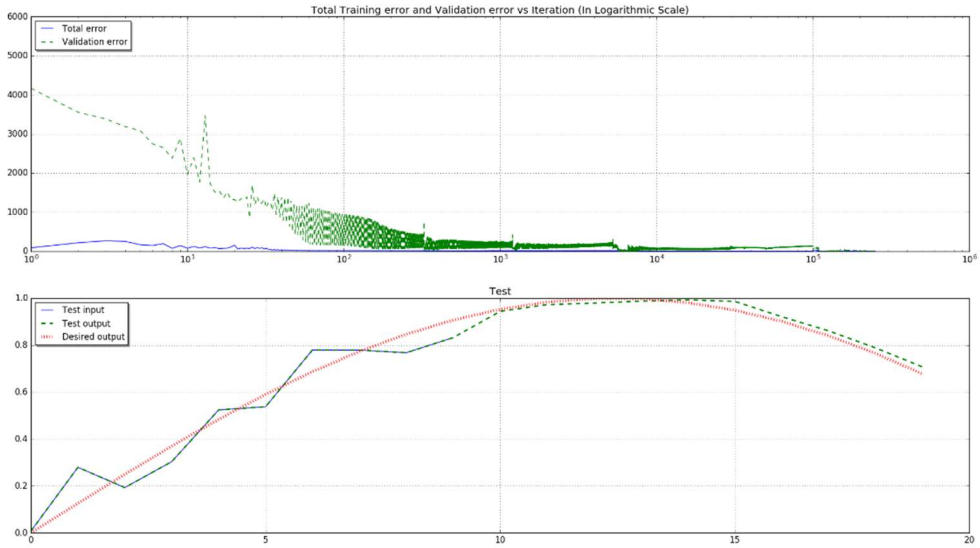

Figure 3: Results of the training and the test stages.