Laboratory work 2

Work description

Artificial neural network with an elementary structure will be used during the work. The structure is a neuron with linear activation function (*purelin*(n)=*purelin*(*Wp+b*)=*Wp+b*). Neuron task is to forecast the *k*-th value of time series *a*(*k*) using *n* previous values *a*(*k-1*), *a*(*k-2*), ..., *a*(*k-n*). Model that is implemented using a premise that a dependency between a forecasted value and previous values can be described using a linear function is called autoregressive linear model of the n-th order.

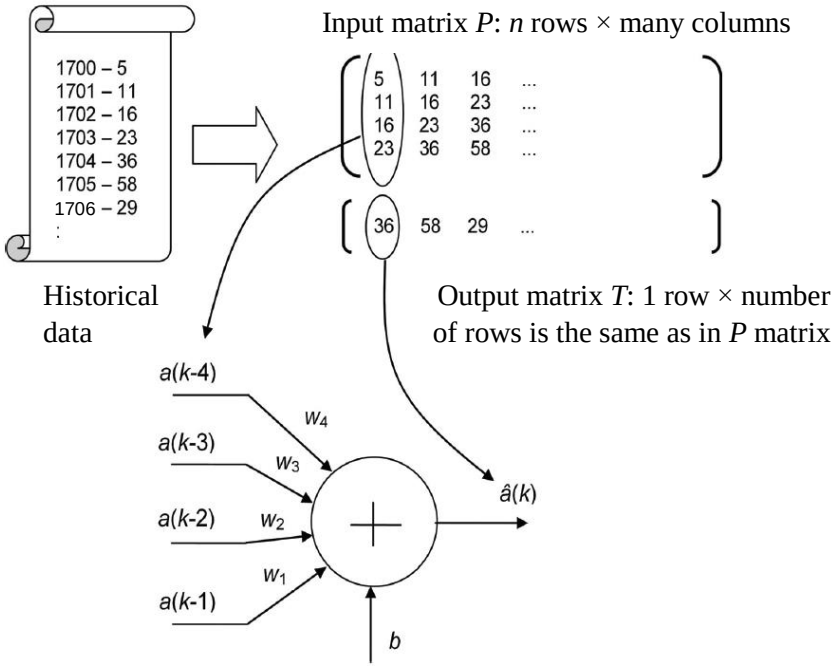

Figure 1.1. *Linear neuron schema that implements linear autoregressive model of the order n=4.*

Linear autoregressive model has the following form:

$$\hat{a}(k) = w_1 \cdot a(k-1) + w_2 \cdot a(k-2) + \cdots + w_n \cdot a(k-n) + b \qquad (1.1)$$

here $w_1, w_2, w_n \cdot$ and *b* are model parameters and $\hat{a}(k)$ means the next forecasted value of time series. Autoregressive model role in our work will be played by an artificial neuron. Previous time series values will be given as inputs to the neuron. Model parameters will be substituted by neuron weights. Forecasted value will be at the neuron output (see figure 1.1). Expression 1.1 can be rewritten in the following way:

$$a(k) = w_1 \cdot a(k-1) + w_2 \cdot a(k-2) + \cdots + w_n \cdot a(k-n) + b + e(k) \qquad (1.2)$$

here $e(k) = a(k) - \hat{a}(k)$ is a forecast error at the *k*-th time series.

Having a set of historical data we will search for optimal values of autoregressive model parameters. That means that we will seek to set such $\hat{a}(k)$ forecast value to be as much close to the real $a(k)$ value, i.e. its mean squared error of forecasted values $e(k)$ be as small as possible for a whole data set. Then the developed model will be possible to use in a practice.

Plum activity on the sun will be forecasted in this work. The activity will be expressed by a number of plums during a certain calendar year (see figure 1.2). This activity is cyclic and cycle is 11 years.

Historical data of sun plum activity will be used to describe input data that are given to neuron inputs. The same data will be used as neuron outputs, i.e. as "supervisor's answers" – see figure 1.1. Additionally, the data set will be split out onto training and test sets.

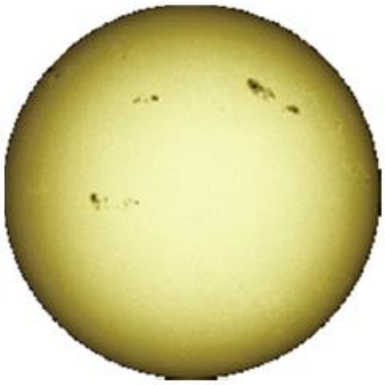

Figure 1.2. Sun plums.

Next, we will perform a procedure of selection optimal values of model weight coefficients. In a case of a linear neuron that can be done in two ways: either using a set of equations or using an iterative technique, i.e. using a supervised neuron learning. Test experiments will be composed after calculation of model weight coefficients in order to check model forecast quality.

Preparation. Read the following function descriptions to be used in the work. Type `help` and a function name in *Matlab* command prompt:

```
newlind  – ;
newlin   – .
train    – .
sim      – .
maxlinlr – .
```

Work instructions

1. Download `sunspot.txt` and save it in *Matlab* working directory. File contains historical data on sun plum activity during 1700 - 2014 years.

2. Load a contents of the file into *Matlab* working memory: `load sunspot.txt`

3. Check whether a corresponding matrix had been loaded– the first column corresponds to years, the second – sun plums. Delete `sunspot` variable – `clear sunspot .`

4. The first task that should implement our program is to draw a diagram of sun plum activity during 1700-2014. Figure has to be fully described (axis and figure titles)
   In scenario editor window (*File/New/M-File*) write commands

```
clear
close all
```
Read about a figure plotting instrument (`help plot`). For example, `plot` function call can be the following:
```
figure(1)
plot(sunspot(:,1),sunspot(:,2),'r-*')
```
Add axis and figure titles (`xlabel, ylabel, title`).

5. Let us set the order of autoregressive model will be 2 (*n*=2). That is, we premise that next year's plum forecast is possible based only on two previous years. Then, a neuron will have only two inputs. Supplement the scenario by describing matrices `P` and `T`, that contain input (training) data and output data correspondingly (see figure 1.1).
```
L = length(sunspot);        % data length
P = [sunspot(1:L-2,2)' ;    % input data
     sunspot(2:L-1,2)'];    % matrix
T = sunspot(3:L,2)';        % output data vector
```
Check content and size of matrices `P` and `T` (`size`).

6. Read about the parameters of functions: `plot3`, `grid`, `zlabel`. Plot a diagram in a new graphical window (`figure 2`) with the following content – inputs and outputs (`P` and `T` correspondingly). Supplement the scenario. E.g. to call plotting function `plot3` you may use `plot3(P(1,:),P(2,:),T,'bo')` . Run the scenario, analyze the figure. Add axis and figure titles. What is a graphical interpretation of optimal values of neuron weight coefficients `w1`, `w2`, `b` ? *Note*: to rotate a figure you may use *Rotate 3D* icon.

7. Let us select from inputs `P` and outputs `T` data set fragments of the first 200 members – so called training data set. Using that set we will calculate optimal values of neuron weight coefficients (parameters of autoregressive model). The rest data will be used for model testing. Then, using existing matrices `P` and `T` , let us define two new ones – `Pu` and `Tu`. For example: definition of a `Pu` matrix may be as follows `Pu = P(:,1:200);`
Run the scenario, check the fact of creation of corresponding matrices, and check their contents and size.

8. Create an artificial neuron of the structure that had been discussed before. Calculate its weight coefficients using matrix calculus (function `newlind`). To do so, use matrices of training data `Pu` and `Tu`. Name a variable that describes a network by `net`. Add a corresponding command to the scenario.

9. Display corresponding neuron weight coefficient values:
```
disp('neuron weight coefficient values:' )
disp( net.IW{1} )
disp( net.b{1} )
```

Assign corresponding weight coefficient values to auxiliary variables
```
w1 = net.IW{1}(1)
w2 = net.IW{1}(2)
b = net.b{1}
```

10. During next step perform a testing of the developed model – i.e. check its forecast quality using network simulation. Initially we will do so using training data that was used for calculation of weight coefficients.

    For example, we need to forecast sun plum activity for 1702–1901. For that purpose we supply into neuron inputs data sets corresponding the following years: 1700 and 1701, 1701 and 1702, …, 1899 and 1900. This can be done in a computerized way using input data from matrix `Pu`. Following this, we will obtain resulting vector `Tsu`, i.e. forecasted sun plum activity values for 1702–1901 :

    ```
    Tsu = sim(net,Pu)
    ```

    Since we have known values during the analyzed period (`Tu`), we can compare them with forecasted ones (`Tsu`). To do so we can use several figures on the same graphical window (`hold on`). The resulting window should contain different colors for each diagram and a legend (`legend`).

    Add corresponding commands to the scenario.

11. Do the same simulation for the rest of the data. Draw a comparison diagram depicting forecasted `Ts` (neuron outputs) and true values `T`. *Note*: you should not recalculate weight coefficients.

12. Create forecast error vector `e` (see expression 1.2). In a new window draw error diagram (`plot`). Describe axis and chart titles.

13. Draw forecast error histogram (`hist`).

14. Using (1.3) calculate *Mean-Square-Error, MSE and MAD (see explanation in presentation file)*:

$$MSE = \frac{1}{N}\sum_{k=1}^{N}\left(a(k) - \hat{a}(k)\right)^2 = \frac{1}{N}\sum_{k=1}^{N} e(k)^2 \qquad (1.3)$$

    MAD = $median(|a(k) - \hat{a}(k)|)$

    The following assignment items contain a description of how to modify scenario in order to calculate weight coefficients using iterative method- using neuron training.

15. Make a copy of the previous scenario. Start editing a fresh copy under a different file name.

16. Using function `newlin` (see application illustration: `help newlin`) create *a direct neuron*. Define function parameters (input delay – 0 and learning rate (`lr`) – between 0 and 1).

17. Define needed learning error goal and number of epochs, e.g.:
    ```
    net.trainParam.goal = 100;
    net.trainParam.epochs = 1000;
    ```

18. Use function `train` to train a network, e.g.:
    `net = train(net,Pu,Tu);` Calculate network forecast errors MSE and MAD.

19. Save and run the scenario. Answer the questions:
    - What are the new weight values of neural network ?
    - What is the value of mean squared error ?

20. Repeat the procedure with new parameters set in #17. Investigate their impact to learning process and forecasting quality. Which is the maximum value of learning rate `lr` that enables process convergence?

21. Change number of network inputs to *n*=6, *n*=11. Correspondingly redefine definitions of matrices `P` and `T`. Investigate an impact of model structure change on forecast quality (graphically and in a written form). Thus, you need to perform all previous experiments with `newlin` function using different input sets sizes– 6 and 11. **This task is important and mandatory.**

Work report should contain scenarios with comments, created figures, answers to questions and comments on the accomplished investigation, and conclusions.