# Index

# 

[Index 1](#_Toc73138045)

[Preparation 15](#_Toc73138046)

[1. 15](#_Toc73138047)

[2. 16](#_Toc73138048)

[Part A 18](#_Toc73138049)

[1-4 18](#_Toc73138050)

[5 19](#_Toc73138051)

[6 19](#_Toc73138052)

[7 22](#_Toc73138053)

[8 23](#_Toc73138054)

[9 23](#_Toc73138055)

[10 24](#_Toc73138056)

[11 25](#_Toc73138057)

[12 26](#_Toc73138058)

[13 27](#_Toc73138059)

[14 28](#_Toc73138060)

[28](#_Toc73138061)

[15-19 29](#_Toc73138062)

[20 30](#_Toc73138063)

[21 35](#_Toc73138064)

[35](#_Toc73138065)

[Part C 41](#_Toc73138066)

[DATASET 1 41](#_Toc73138067)

[DATASET 2 47](#_Toc73138068)

[1 47](#_Toc73138069)

[2 49](#_Toc73138070)

[3 - Architecture of ANN including values of parameters (learning rate, activation function) 51](#_Toc73138071)

[4 - Results of 10-fold cross validation experiments (cost function value at each fold, averaged value) 52](#_Toc73138072) [54](#_Toc73138076)

[5 - Description of measures taken in order to increase ANN performance. Repeat 4th step for documenting improved model performance. 55](#_Toc73138077)

[Learning rate is increased %5 percentage: 0.0000875733 55](#_Toc73138078)

[Learning rate is decreased %5 percentage: 0.000079233 58](#_Toc73138079)

[Number of Hidden Layer Increased to 100 from 10 61](#_Toc73138080) [64](#_Toc73138084)

[Changing the Training Function to Bayesian regularization backpropagation 65](#_Toc73138085)

[Conclusion 69](#_Toc73138091)

# 

# 

# Preparation

## 1.

Illustration of a neuron with one input:

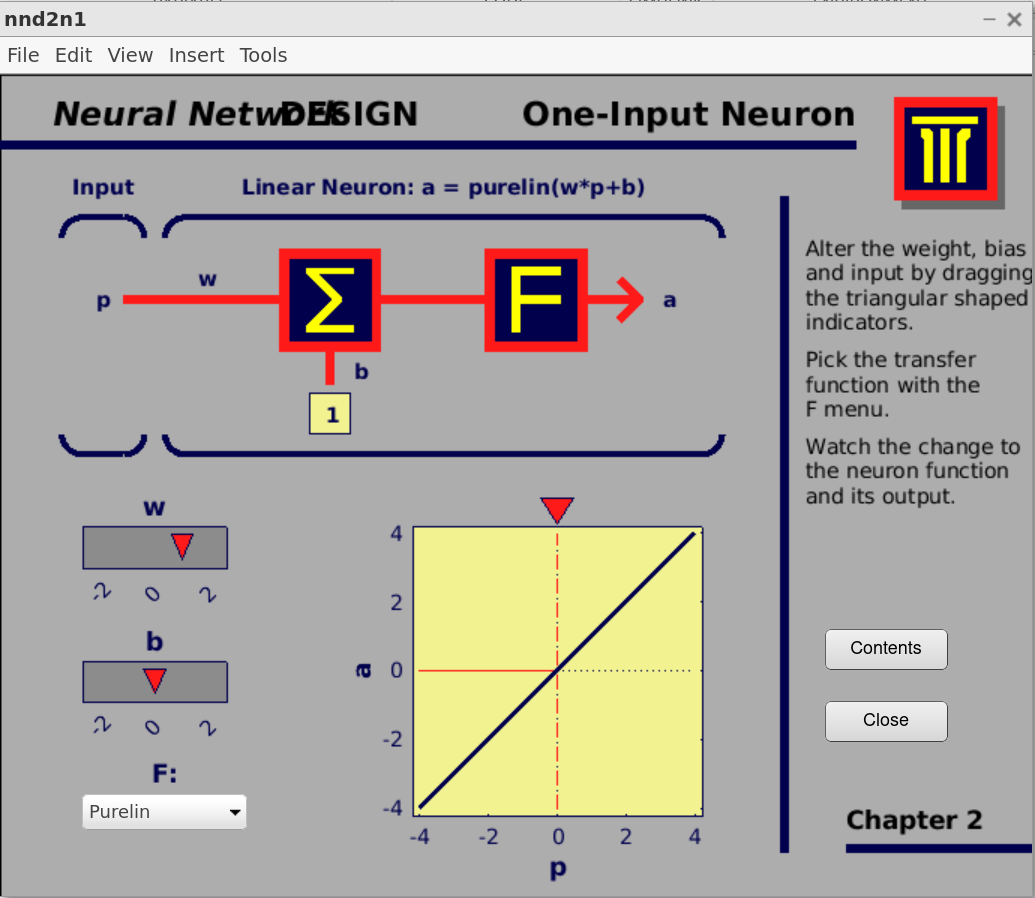
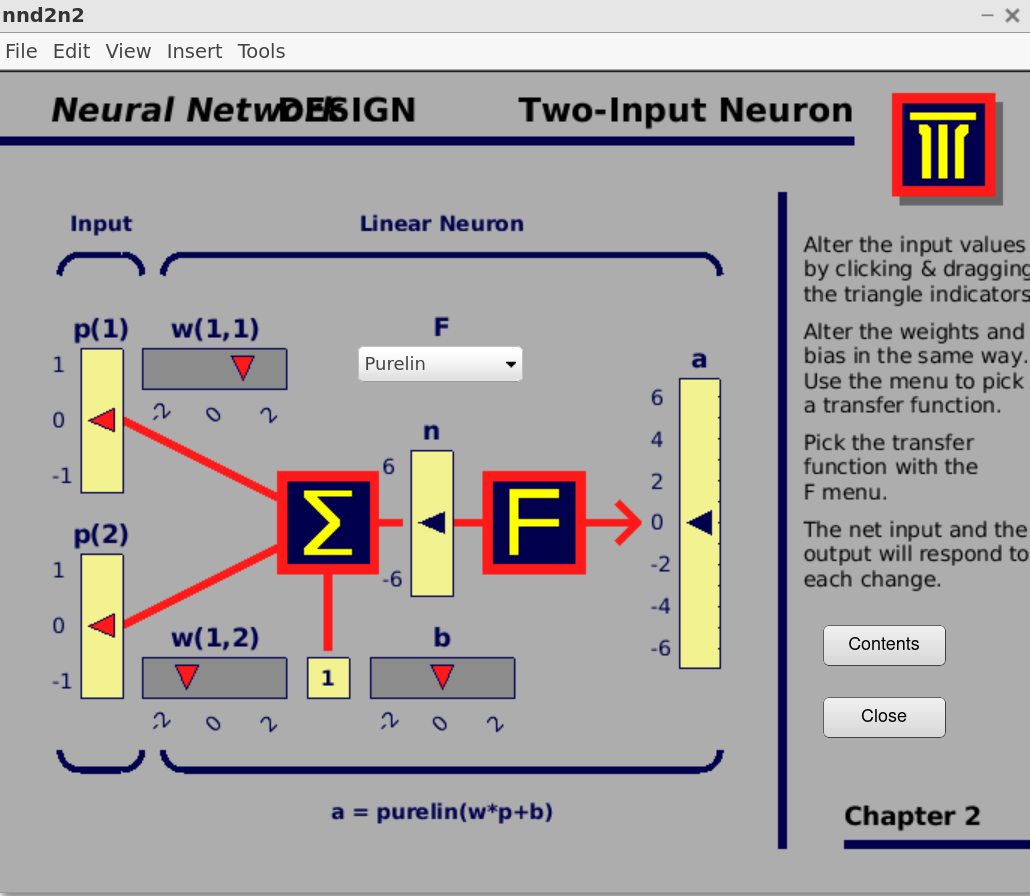


Illustration of a neuron with two inputs:



## 2.

1. **newlind** Design a linear layer.

newlind(X,T) takes an RxQ input matrix X, an SxQ target matrix T, and

returns a linearlayer designed to output T (with minimum sum square

error) given X.

newlind(X,T,Xi) can also solve for linear networks with input delays and

multiple inputs and layers by supplying input and target time series

data in cell array form:

X - NixTS cell array, each element X{i,ts} is an RixQ input matrix.

T - NtxTS cell array, each element T{i,ts} is an VixQ matrix.

Xi - NixID cell array, each element Xi{i,k} is an RixQ matrix, default = [].

returns a linear network with ID input delays, Ni network inputs, Nl layers,

and designed to output T (with minimum sum square error) given input P.

1. **newlin** Create a linear layer.

Syntax

net = newlin(P,S,ID,LR)

net = newlin(P,T,ID,LR)

Description

Linear layers are often used as adaptive filters

for signal processing and prediction.

newlin(P,S,ID,LR) takes these arguments,

P - RxQ matrix of Q representative input vectors.

S - Number of elements in the output vector.

ID - Input delay vector, default = [0].

LR - Learning rate, default = 0.01;

and returns a new linear layer.

newlin(P,T,ID,LR) takes the same arguments except for

T - SxQ2 matrix of Q2 representative S-element output vectors.

NET = newlin(PR,S,0,P) takes an alternate argument,

P - Matrix of input vectors.

and returns a linear layer with the maximum stable

learning rate for learning with inputs P.

1. **train** Train a neural network.

train Train a neural network.

[NET,TR] = train(NET,X,T) takes a network NET, input data X

and target data T and returns the network after training it, and a

a training record TR.

[NET,TR] = train(NET,X) takes only input data, in cases where

the network's training function is unsupervised (i.e. does not require

target data).

[NET,TR] = train(NET,X,T,Xi,Ai,EW) takes additional optional

arguments suitable for training dynamic networks and training with

error weights. Xi and Ai are the initial input and layer delays states

respectively and EW defines error weights used to indicate

the relative importance of each target value.

1. **sim** Simulate a Simulink model

SimOut = sim('MODEL', PARAMETERS) simulates your Simulink model, where

'PARAMETERS' represents a list of parameter name-value pairs, a structure

containing parameter settings, or a configuration set. The SimOut

returned by the sim command is an object that contains all of the logged

simulation results. Optional PARAMETERS can be used to override existing

block diagram configuration parameters for the duration of the simulation.

This syntax is referred to as the 'Single-Output Format'.

SINGLE-OUTPUT FORMAT

--------------------

SimOut = sim('MODEL','PARAMETER\_NAME1',VALUE1,'PARAMETER\_NAME2',VALUE2, ...)

SimOut = sim('MODEL', PARAM\_NAME\_VAL\_STRUCT)

SimOut = sim('MODEL', CONFIGSET)

1. **maxlinlr** Maximum learning rate for a linear layer.

maxlinlr is used to calculate learning rates for NEWLIN.

maxlinlr(X) takes an RxQ matrix of Q R-element input vectors and

returns the maximum learning rate for a linear layer without a bias

for stable learning on inputs X.

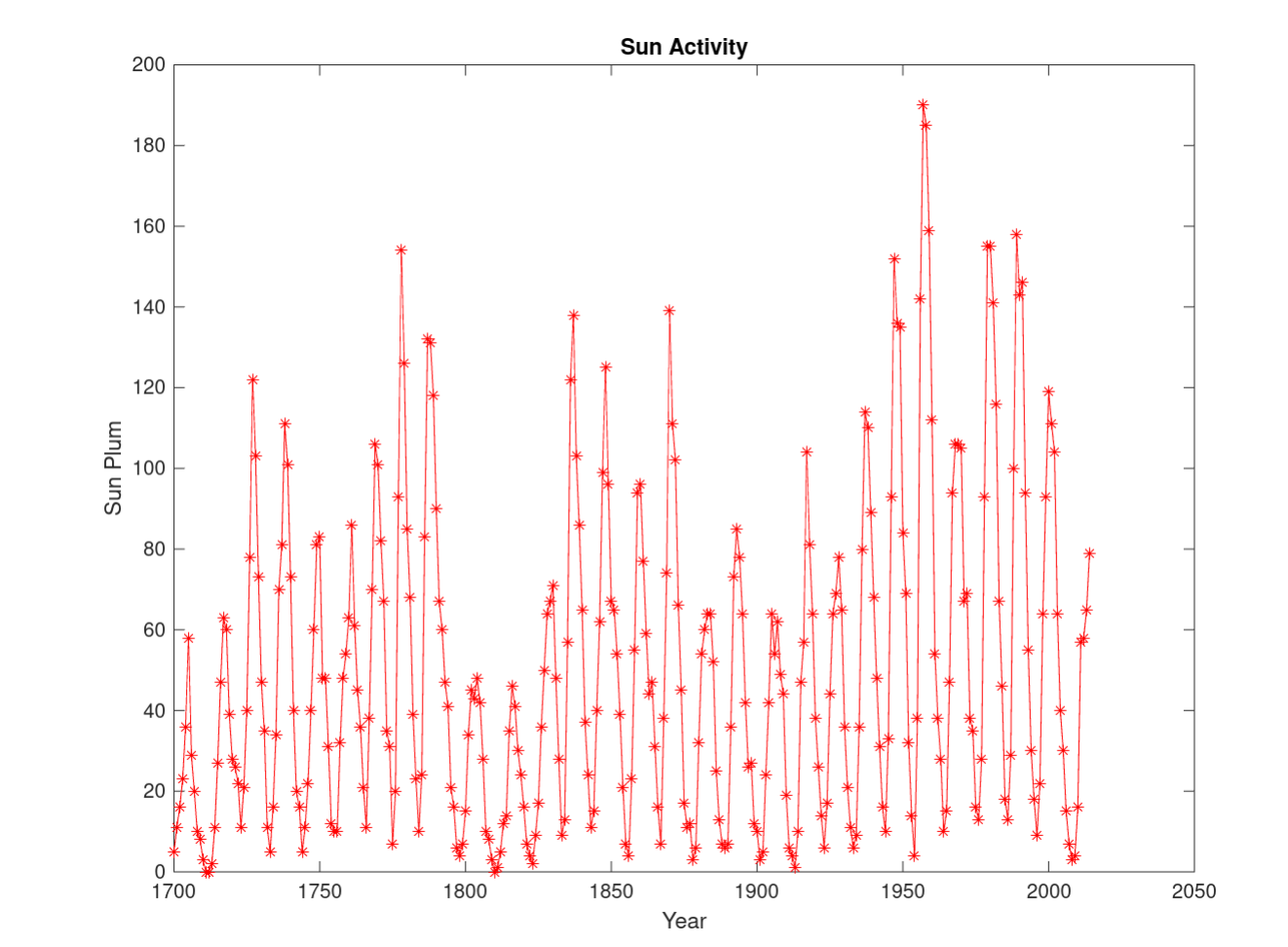
maxlinlr(X,'bias') returns the maximum learning rate for

a linear layer with a bias.

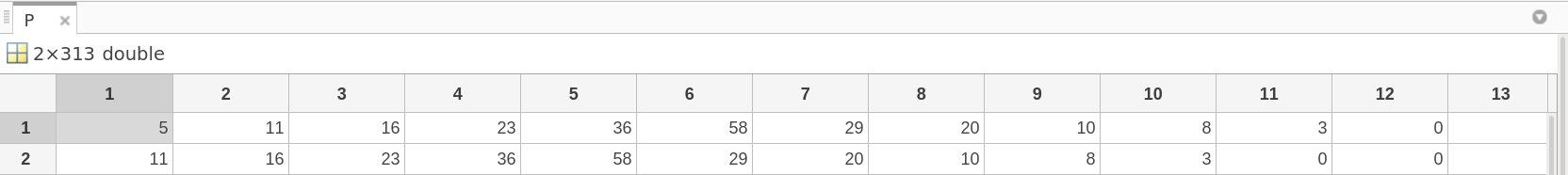
# Part A

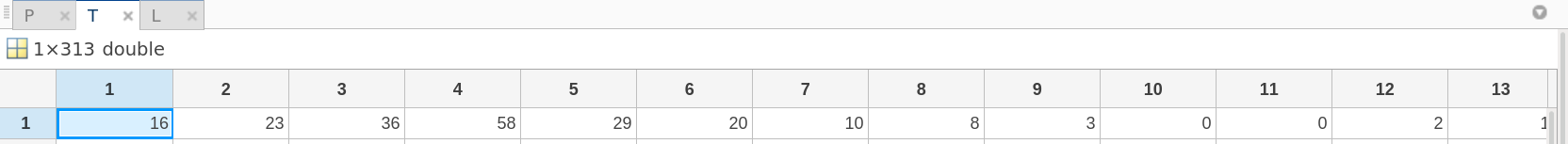
### 1-4

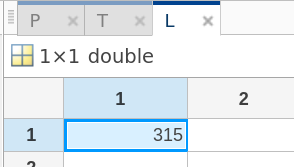
**plot(sunspot(:,1),sunspot(:,2),'r-\*')**



### 5



****

****

### 6

**plot3**

Plot lines and points in 3-D space.

plot3() is a three-dimensional analogue of PLOT().

plot3(x,y,z), where x, y and z are three vectors of the same length,

plots a line in 3-space through the points whose coordinates are the

elements of x, y and z.

plot3(X,Y,Z), where X, Y and Z are three matrices of the same size,

plots several lines obtained from the columns of X, Y and Z.

**grid Grid lines.**

grid ON adds major grid lines to the current axes.

grid OFF removes major and minor grid lines from the current axes.

grid MINOR toggles the minor grid lines of the current axes.

grid, by itself, toggles the major grid lines of the current axes.

grid(AX,...) uses axes AX instead of the current axes.

grid sets the XGrid, YGrid, and ZGrid properties of

the current axes. If the axes is a polar axes then grid sets

the ThetaGrid and RGrid properties. If the axes is a geoaxes, then grid

sets the Grid property.

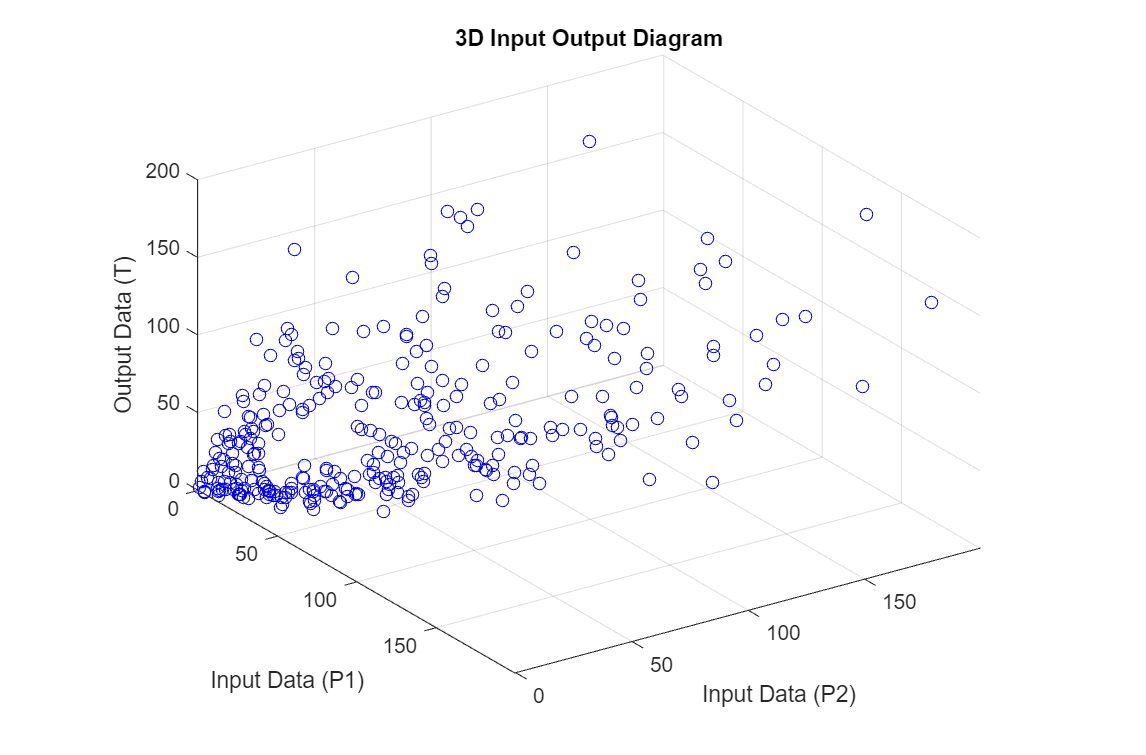
**zlabel Z-axis label.**

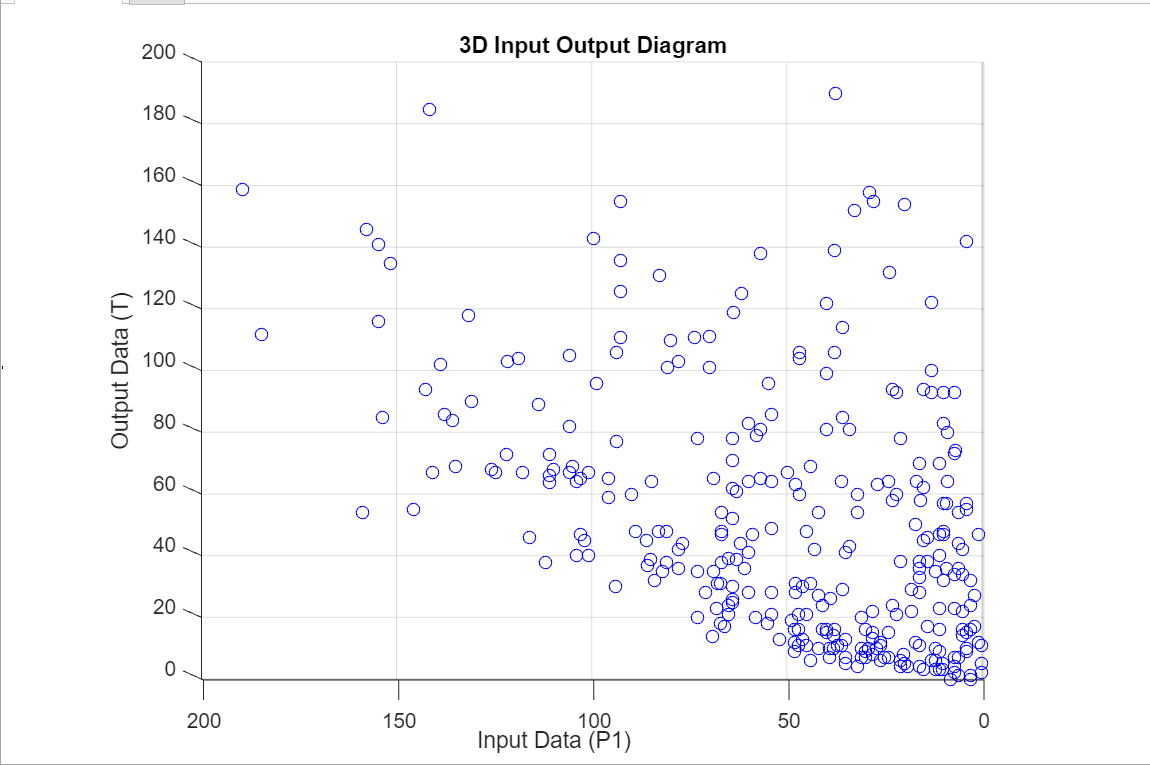
zlabel('text') adds text beside the Z-axis on the current axes.

zlabel('text','Property1',PropertyValue1,'Property2',PropertyValue2,...)

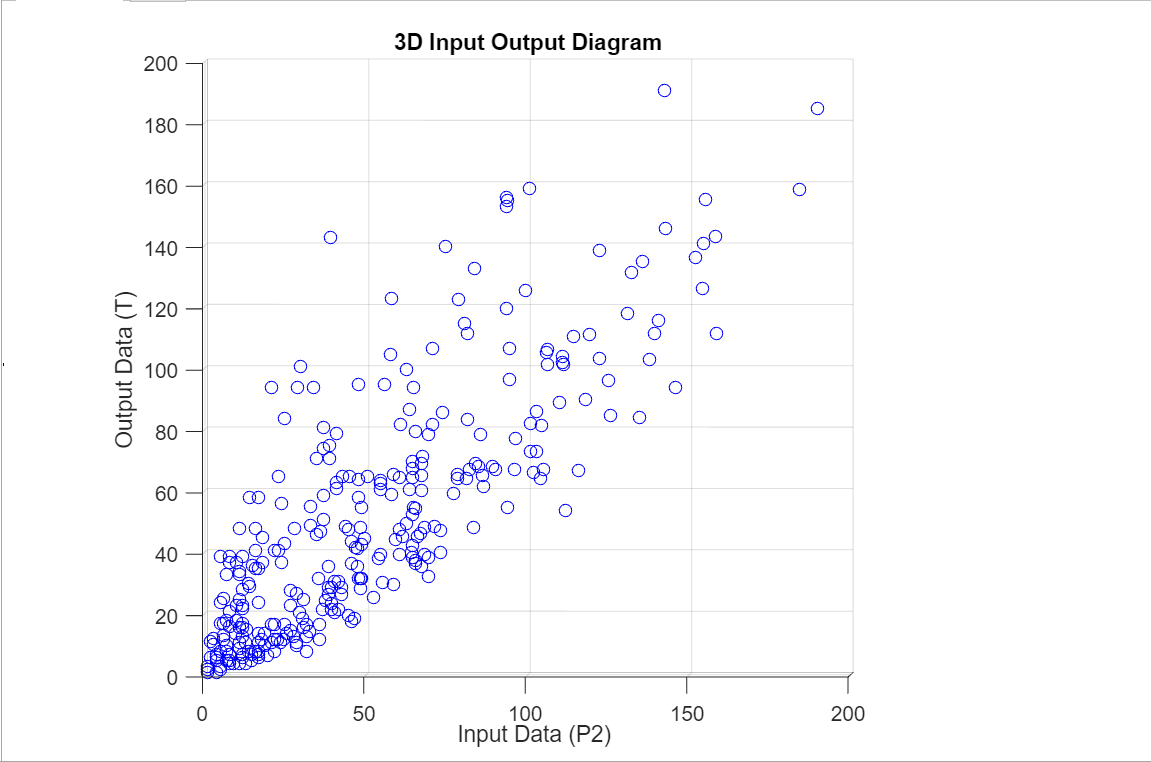
sets the values of the specified properties of the zlabel.

zlabel(AX,...) adds the zlabel to the specified axes.





There is **weak** positive correlation between Output Data (T) and Input Data (P1)



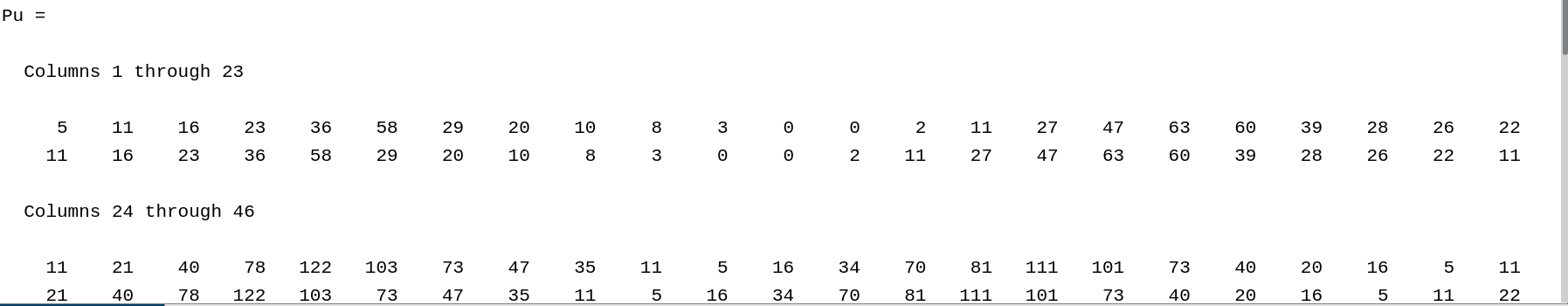
There is **strong** positive correlation between Output Data (T) and Input Data (P2)

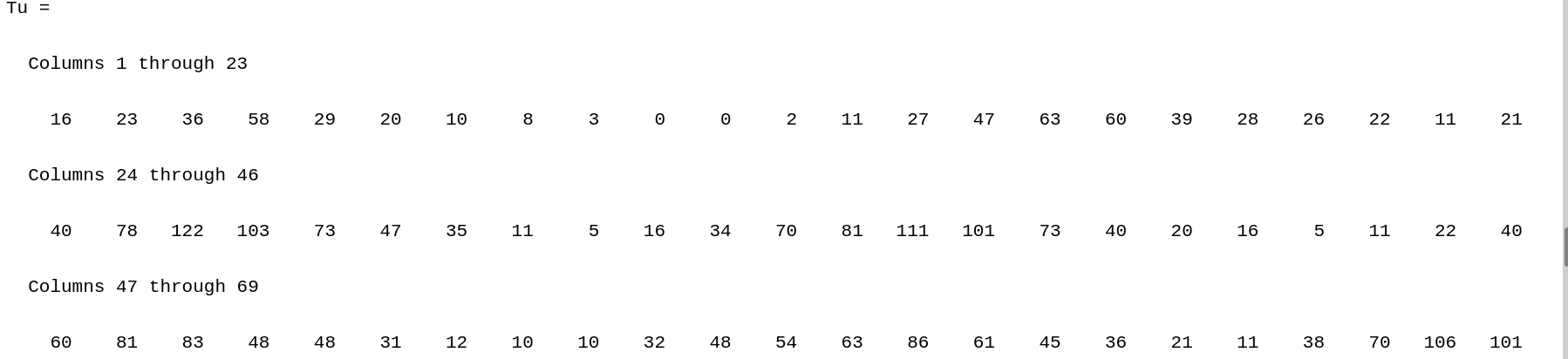


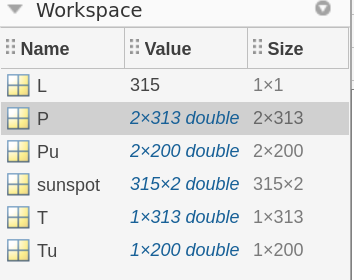
There is **strong** positive correlation between Input Data (P1) and Input Data (P2)

### 7

**Training Dataset:**

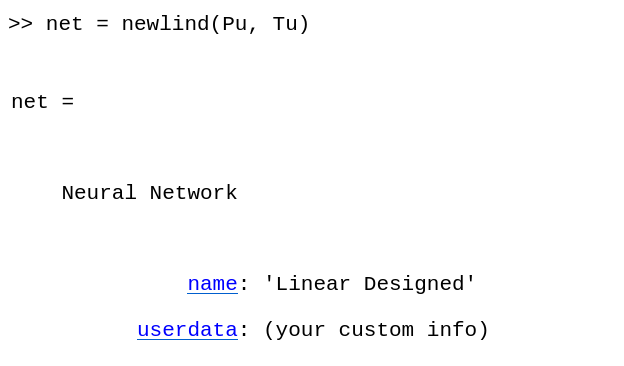






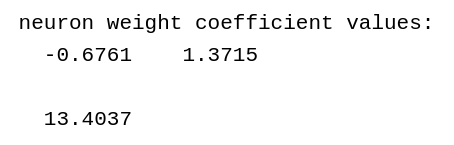
### 8

Creating net:



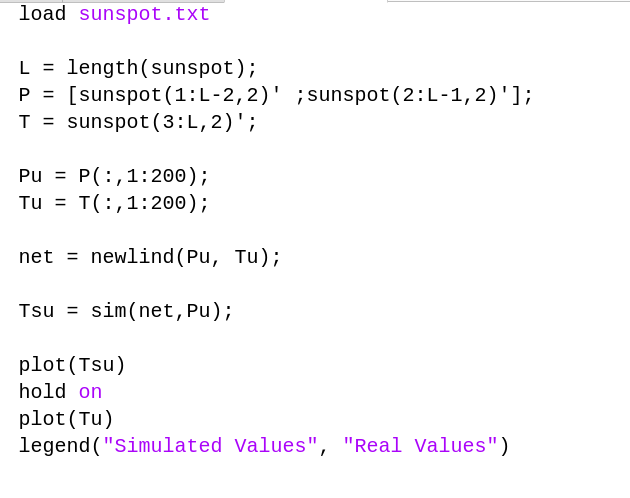
### 9

Neuron weight coefficient values:

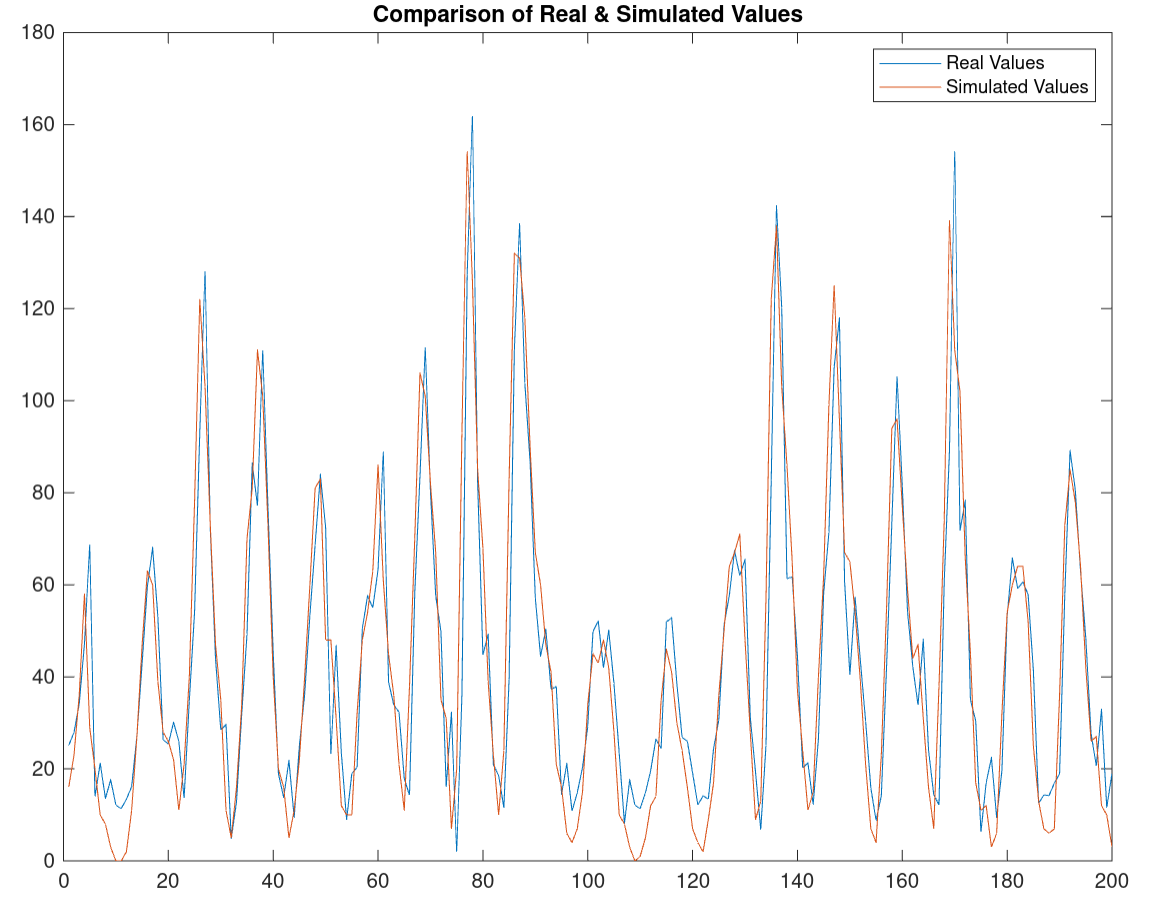


### 

### 10

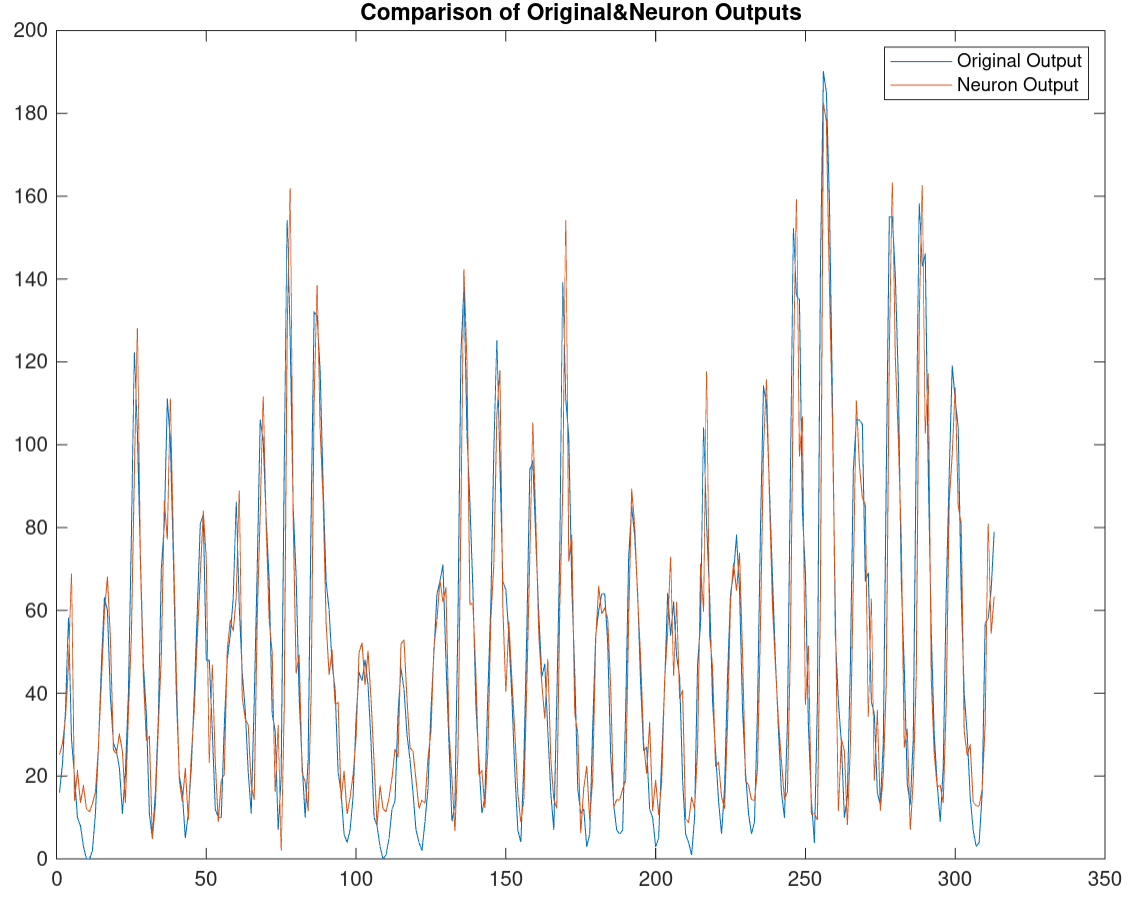


Comparison of Tsu (Simulated 200) and Tu (Real 200):



### 11

Comparison of Ts and T:



### 12

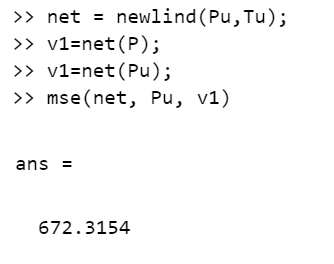
### 13

### 14

### 

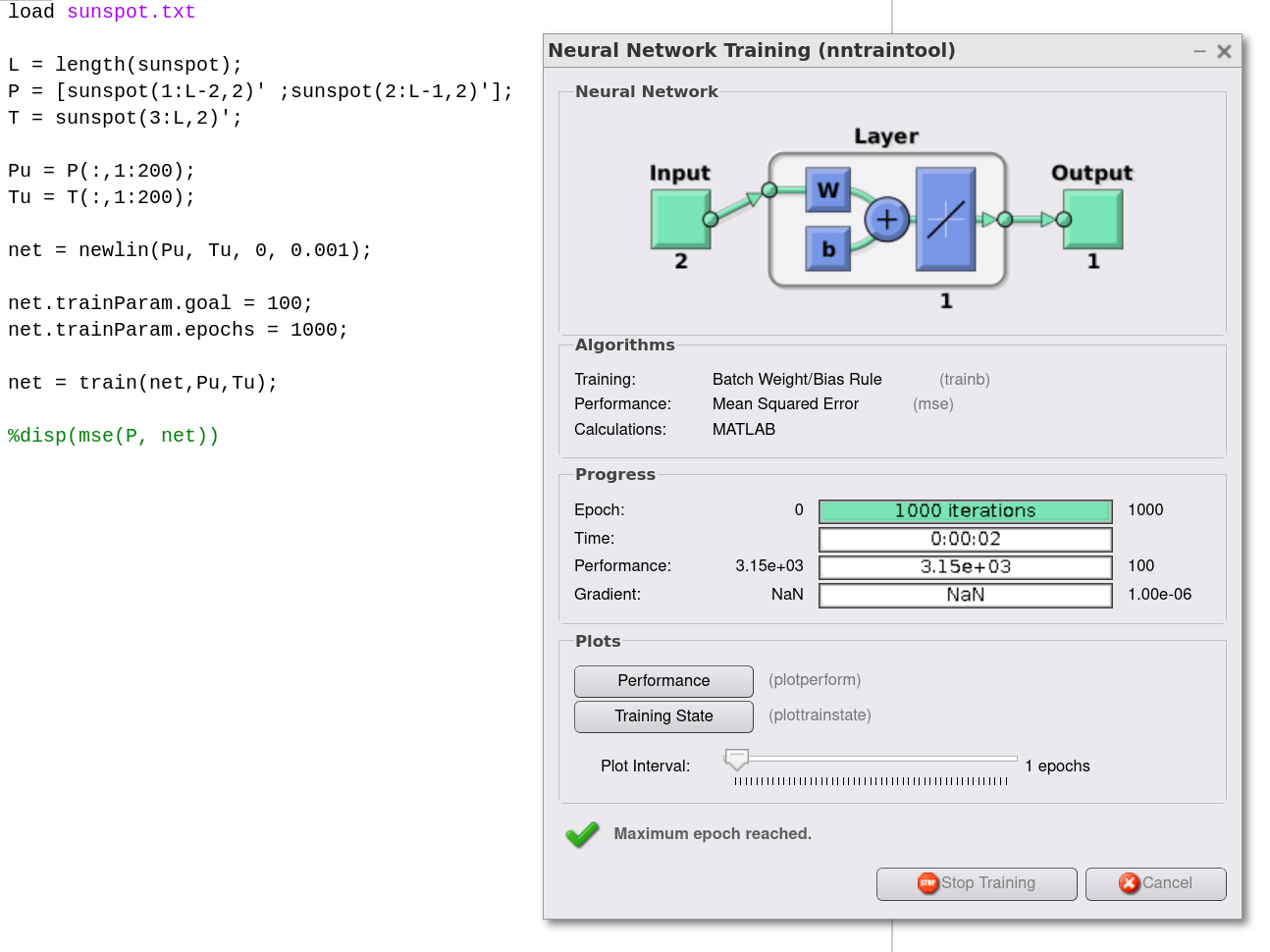
I tried to create the formula in MATLAB.

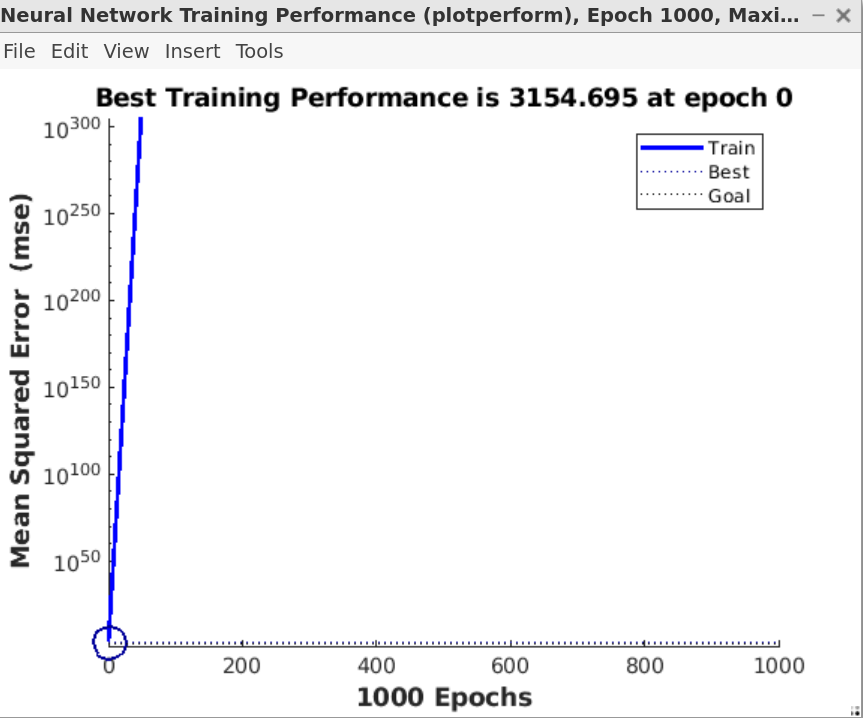
MSE = -0.0584



### 15-19

**Lr = 0.001 - Goal = 100 - Epochs = 1000;**



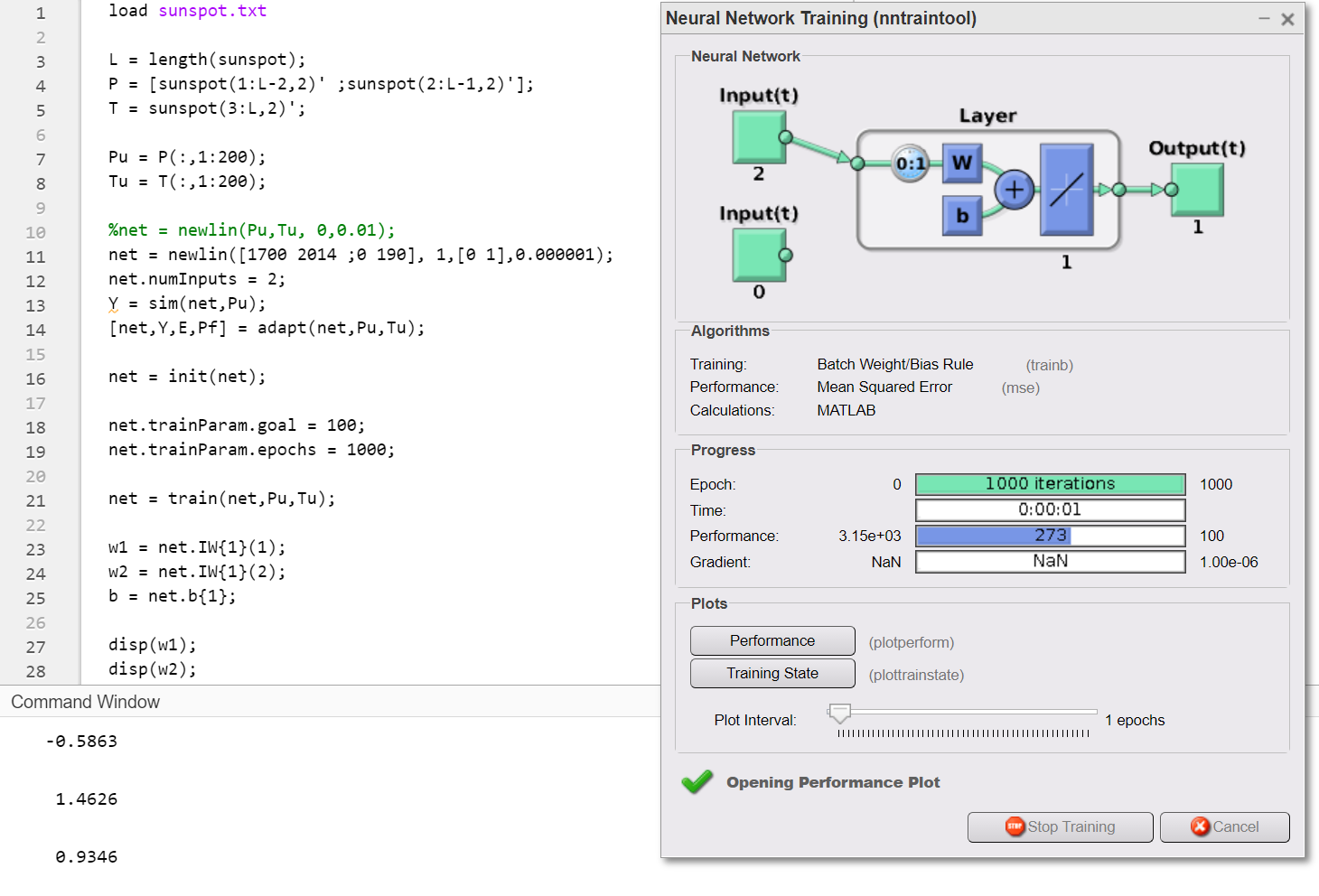


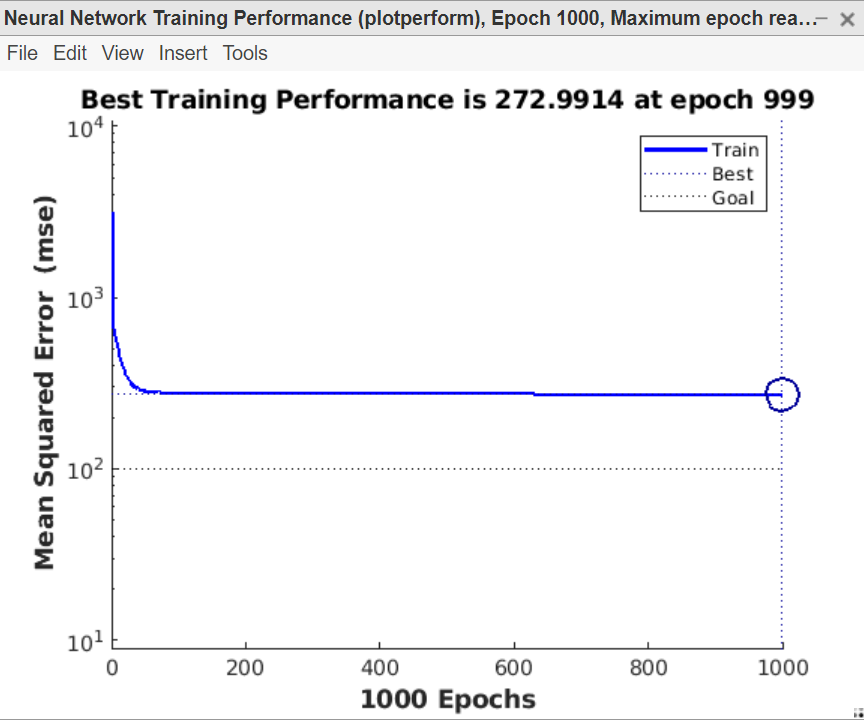
**mse = 3.1551e+03**

This result is not exactly what we want. The reason for those bad results is learning rate. Learning rate is high, we need to decrease it.

### 20

**Lr = 0.000001 - Goal = 100 - Epochs = 1000;**





**w1 = -0.5863 w2 = 1.4626 b = 0.9346**

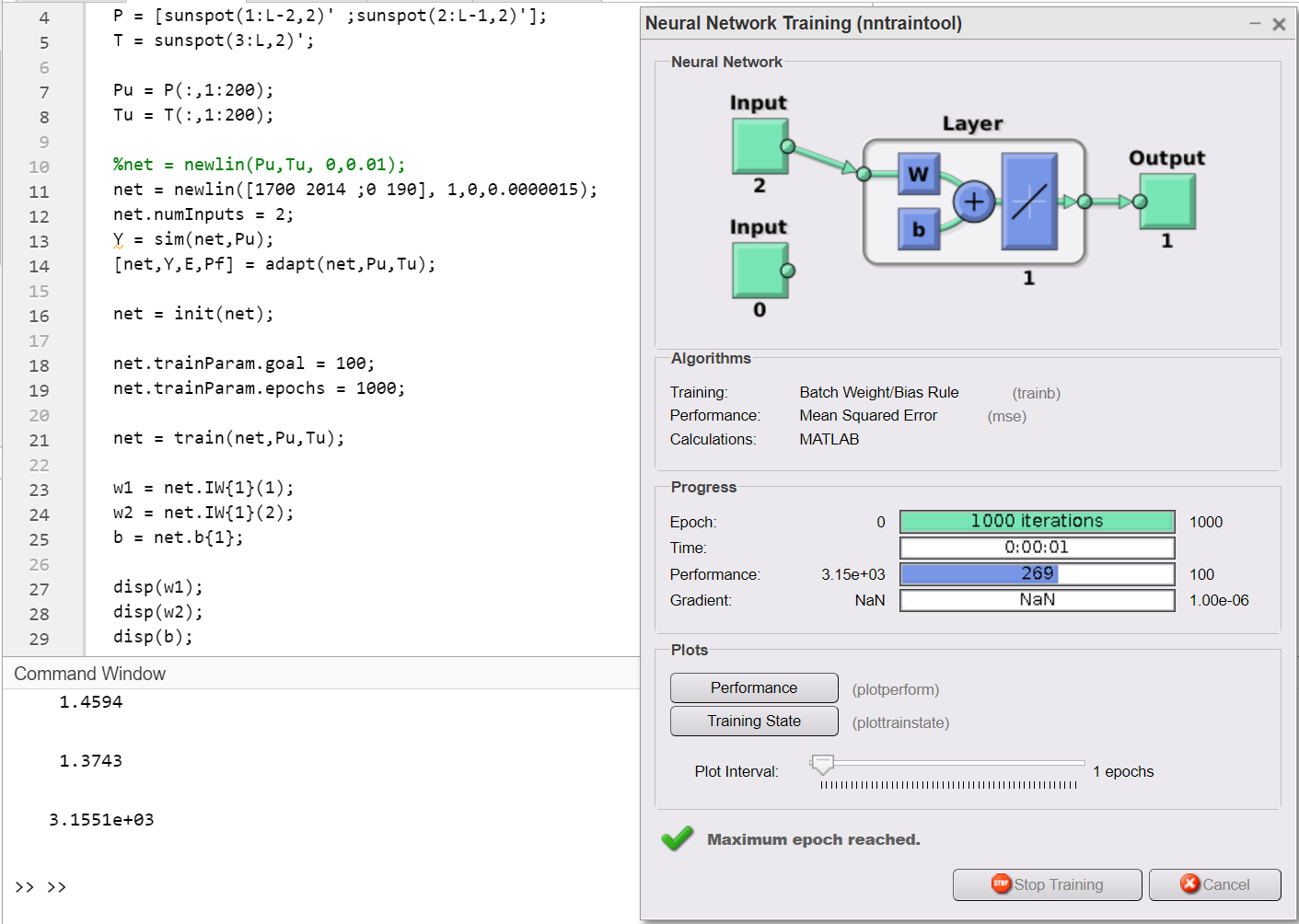
As we decreased the learning rate to 0.000001, we got more useful results.

**Lr = 0.0000005 - Goal = 100 - Epochs = 1000;**



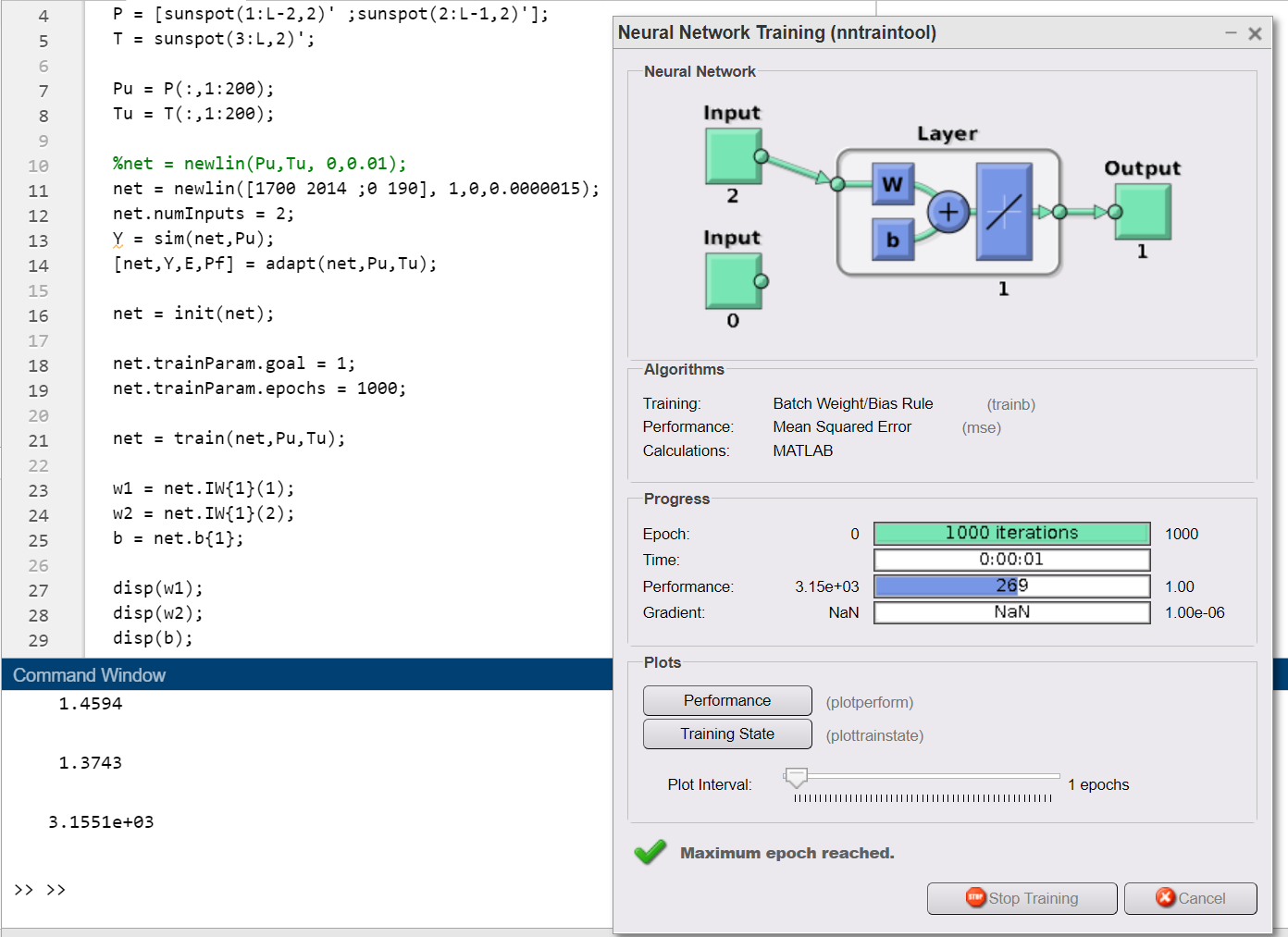
I have decreased LR from 0.000001 to 0.0000005 (half of before). But the result, worse than before. And it is getting worse if we decrease the LR.

**Lr = 0.0000015 - Goal = 100 - Epochs = 1000;**



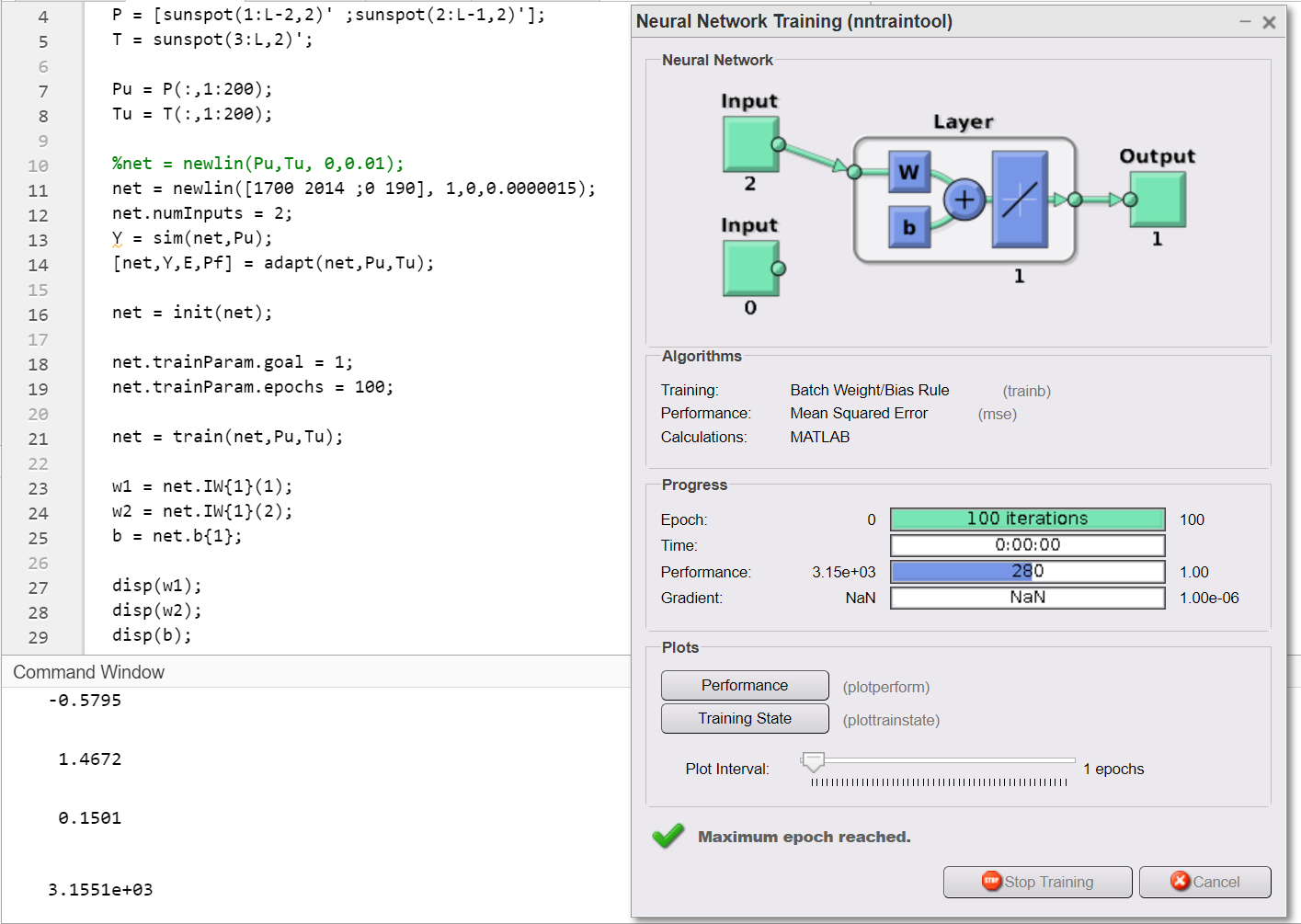
And this is the **best** result that I could get by changing the LR. LR has only increased **0.0000005**.

**Lr = 0.0000015 - Goal = 1 - Epochs = 1000;**



Changing the Goal parameter doesn’t affect the performance.

**Lr = 0.0000015 - Goal = 1 - Epochs = 100;**



As expected, decreasing the number of epochs, increases the mse.

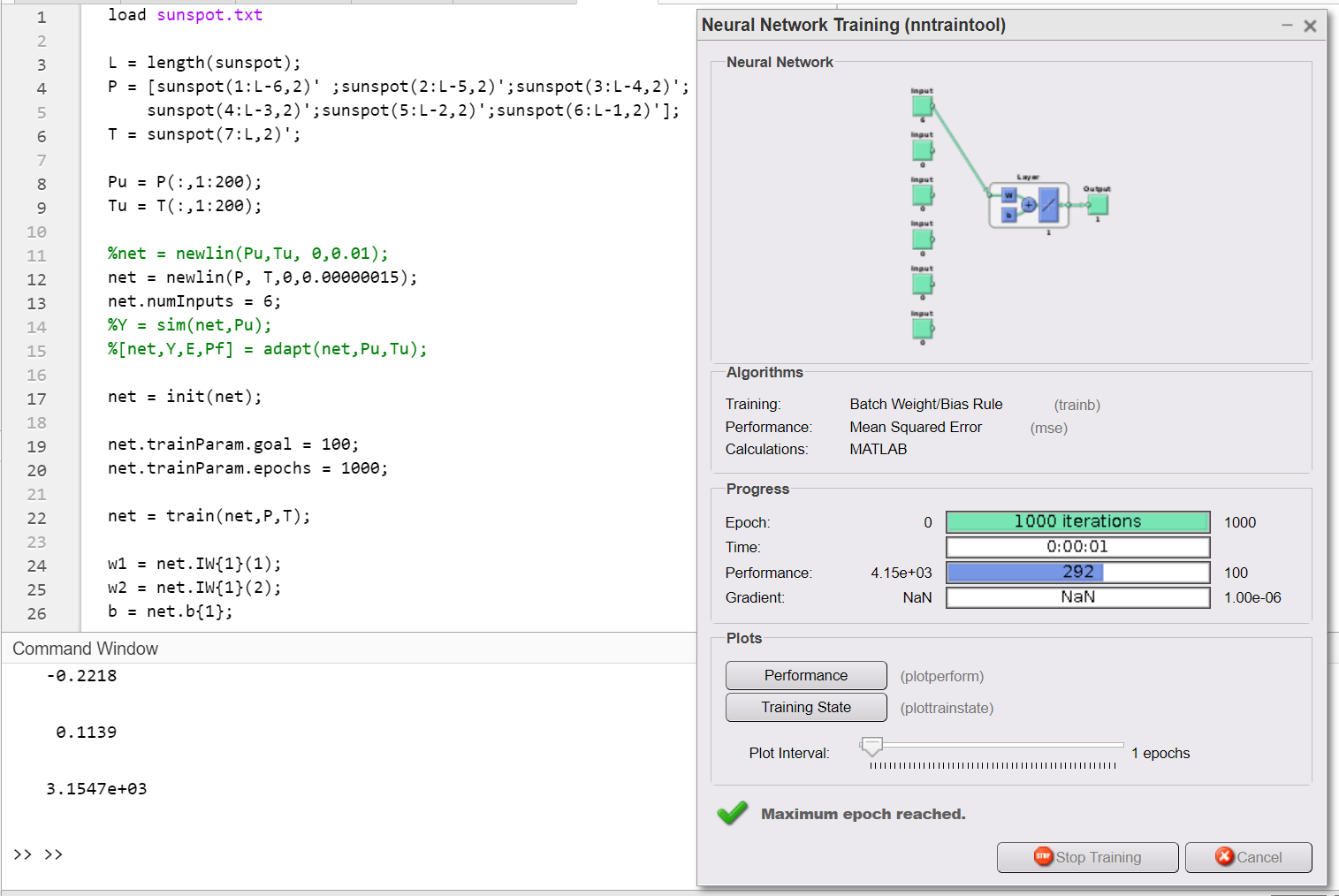
### 21

**Lr = 0.0000015 - Goal = 100 - Epochs = 1000 - INPUTS = 6;**

### 

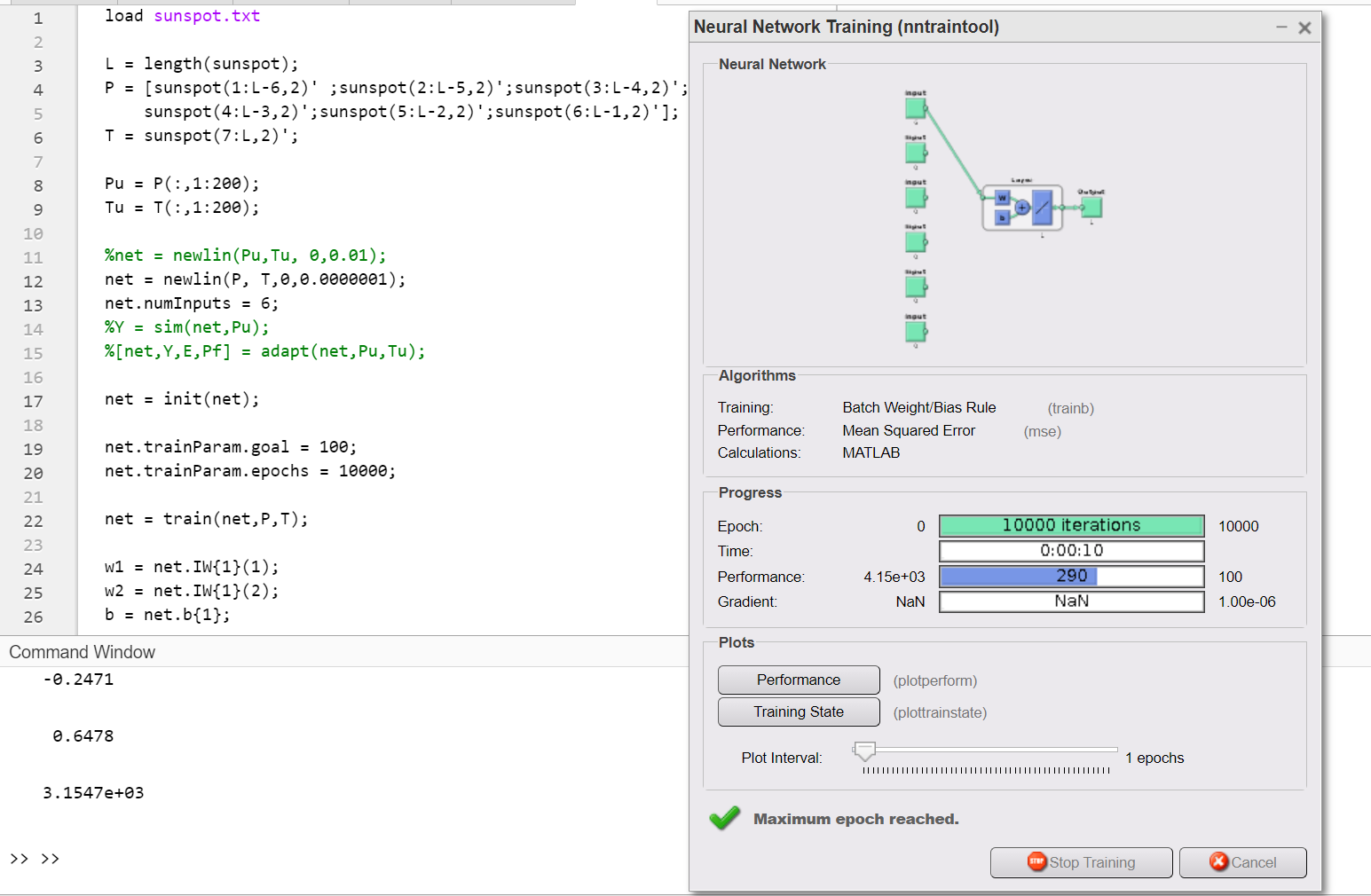
MSE is higher than expected.

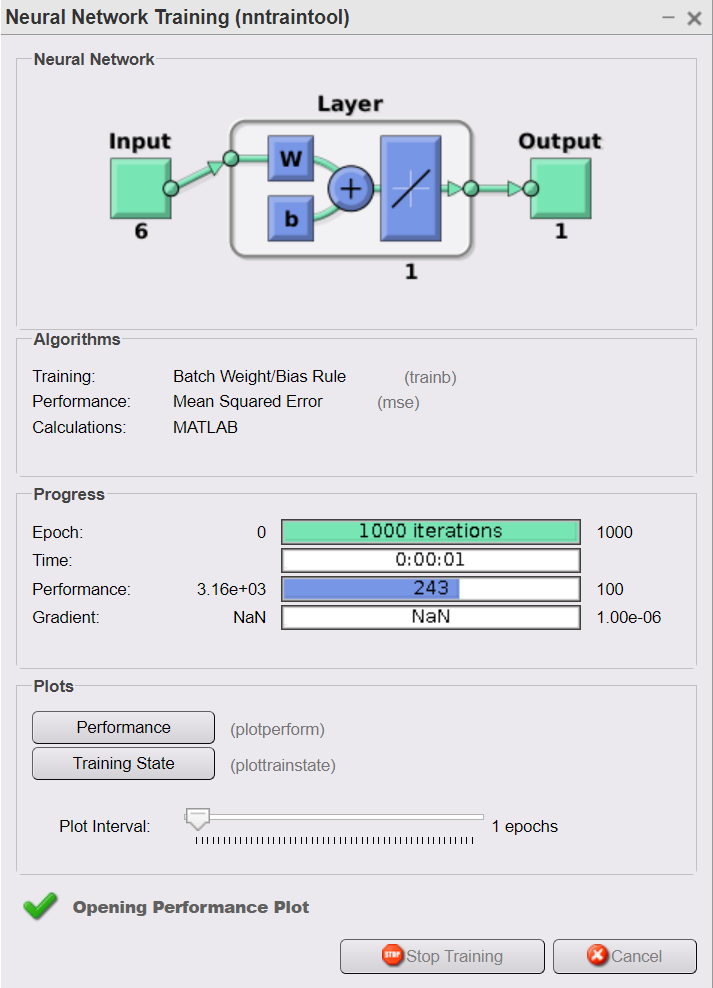
**Lr = 0.00000015 - Goal = 100 - Epochs = 1000 - INPUTS = 6;**

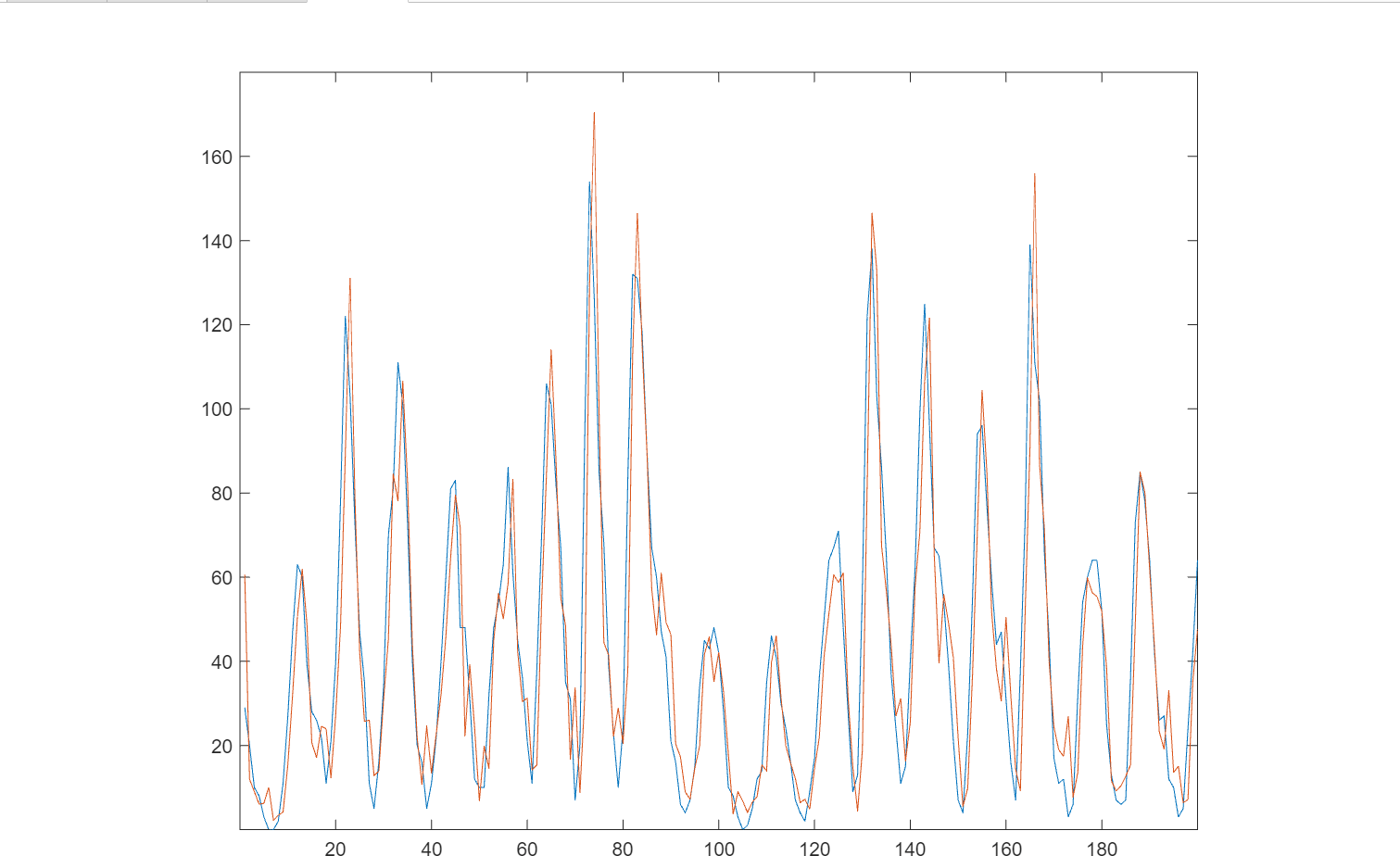


**LR** is decreased to **0.00000015** from **0.0000015** and it is improved the performance.

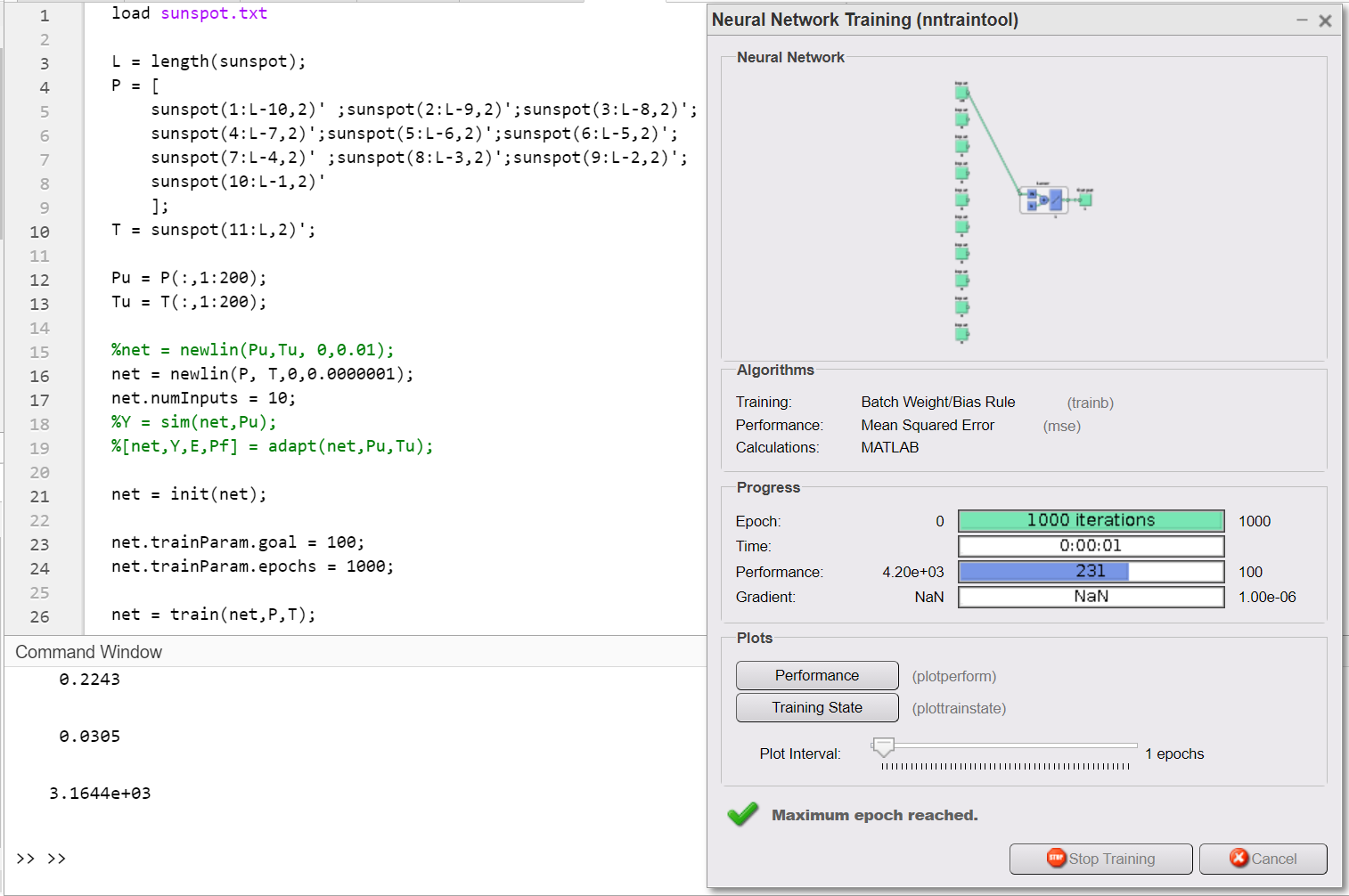
**Lr = 0.0000001 - Goal = 100 - Epochs = 10000 - INPUTS = 6;**



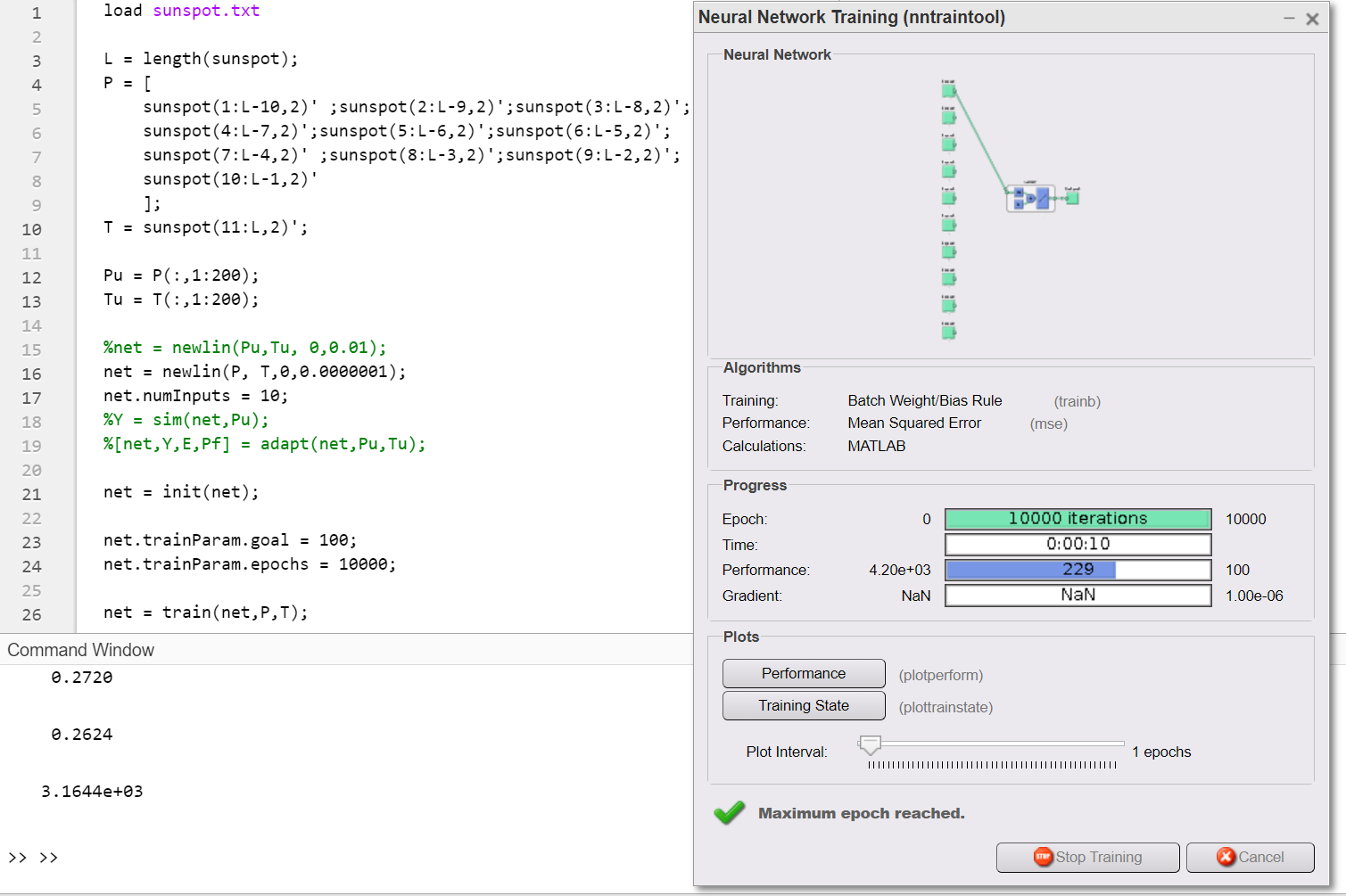


****

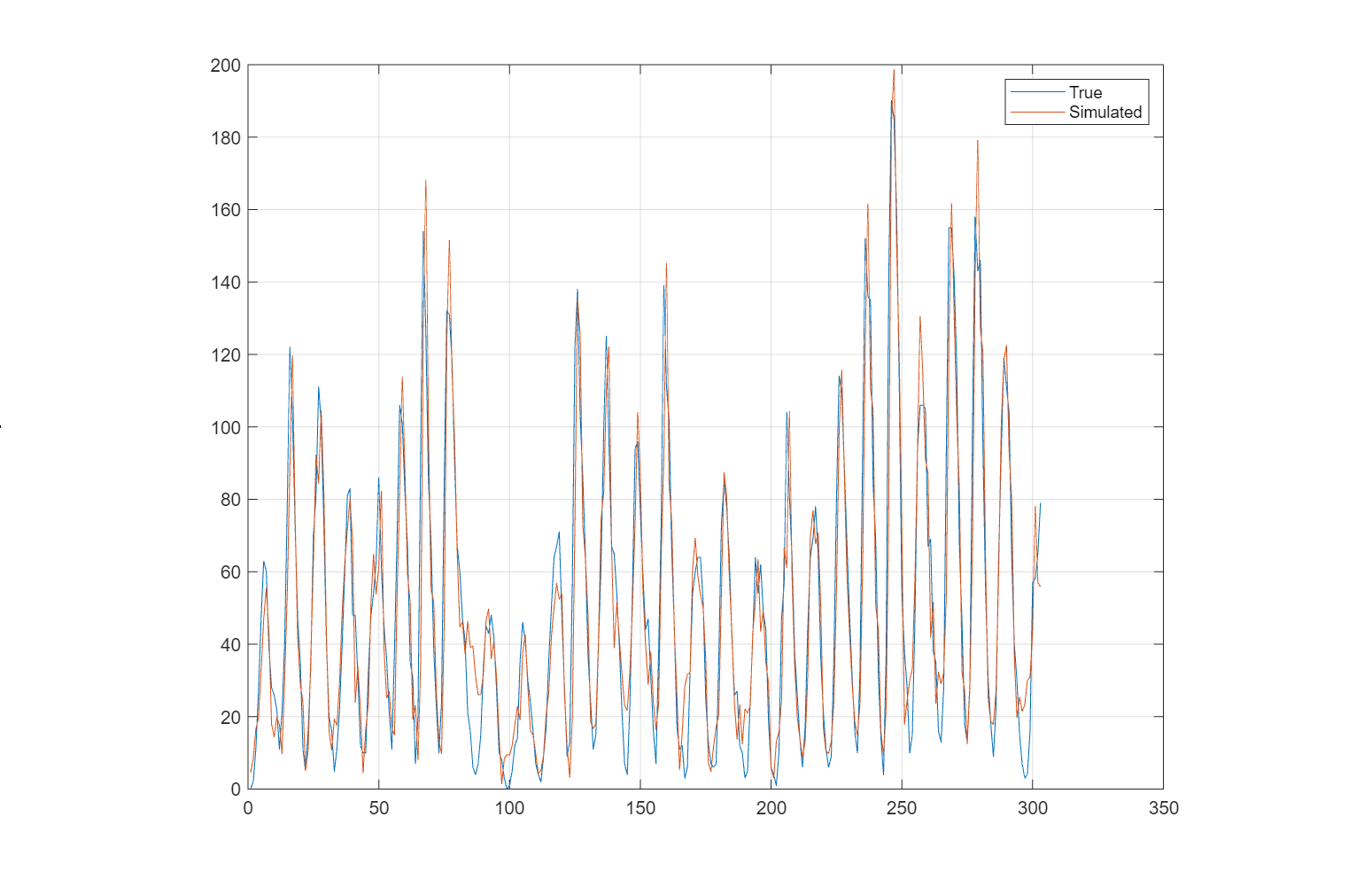
**Lr = 0.0000001 - Goal = 100 - Epochs = 1000 - INPUTS = 10;**



**Lr = 0.000001 - Goal = 100 - Epochs = 10000 - INPUTS = 10;**



As expected, the lowest MSE in all those tests with the highest number of epochs and inputs.



# Part C

### DATASET 1

**Selected Dataset:** Terrorism in Turkey

|  |  |
| --- | --- |
| **Link**: | https://www.kaggle.com/START-UMD/gtd |
| **Description**: | Statistics of actual terrorist attacks in Turkey |
| **Format**: | A data frame with 4292 observations on the following 15 variables. |

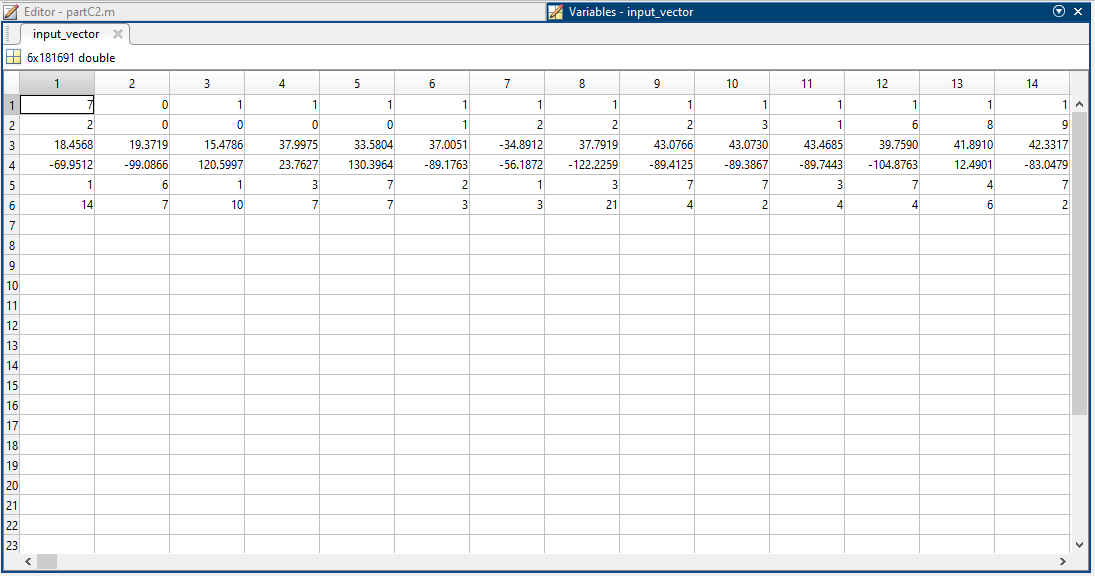
|  |  |
| --- | --- |
| Year  This field contains the year in which the incident occurred. | Month  This field contains the number of the month in which the incident occurred. |
| Day  This field contains the numeric day of the month on which the incident occurred. | Country  This field identifies the country where the incident occurred. |
| City  This field identifies the city where the incident occurred. | Latitude  The latitude of the city in which the event occurred |
| Longitude  The longitude of the city in which the event occurred. | Attack Type  The general method of attack and broad class of tactics used. |
| Killed  The number of total confirmed fatalities for the incident | Wounded  The number of total confirmed wounded for the incident |
| Target  The specific person, building, installation that was targeted and/or victimized | Weapon Type  General type of weapon used in the incident |
| Group  The name of the group that carried out the attack | Target Type  The general type of target/victim |

I have used Terrorism in Turkey dataset. For the input data I have choosed: month, day, latitude, longitude. Attack type, target type. And for the output data I have choosed: Number of kills.

However, whatever I have tried the results was not good.

The results I have got.

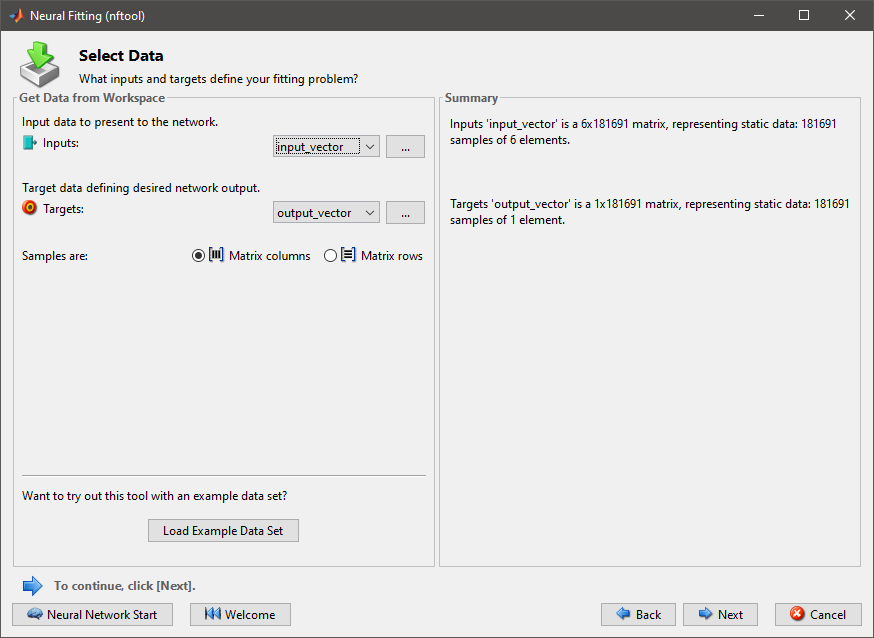
Input Vector:



There were NaN values. I have changed them with the mean values of that row.

**10 Hidden Layers:**

Summary of the dataset

****

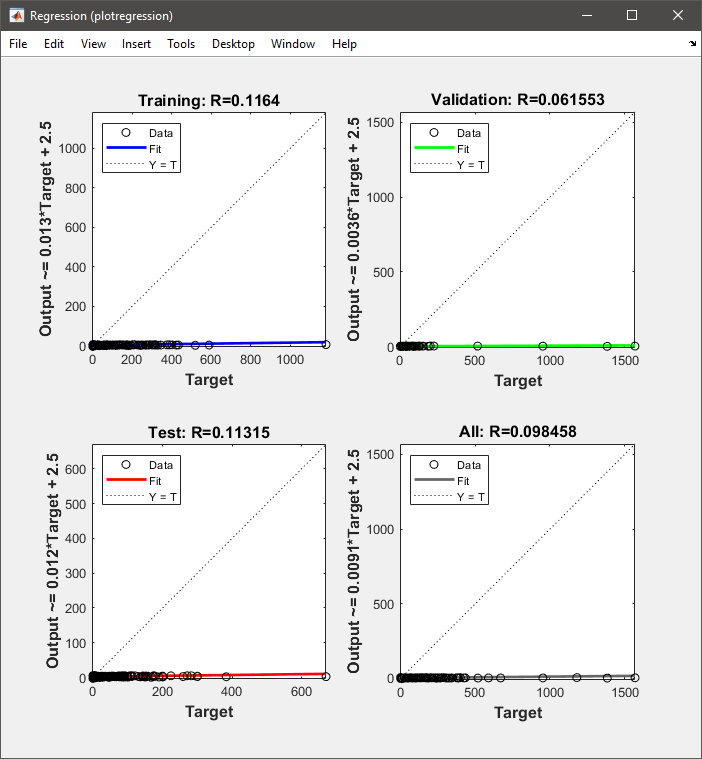
Percentages of Traning, Validation and Testing



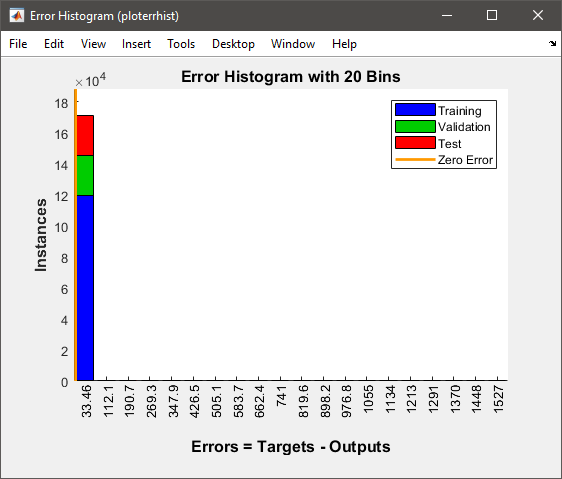
Training Algorithm and Results



Regression

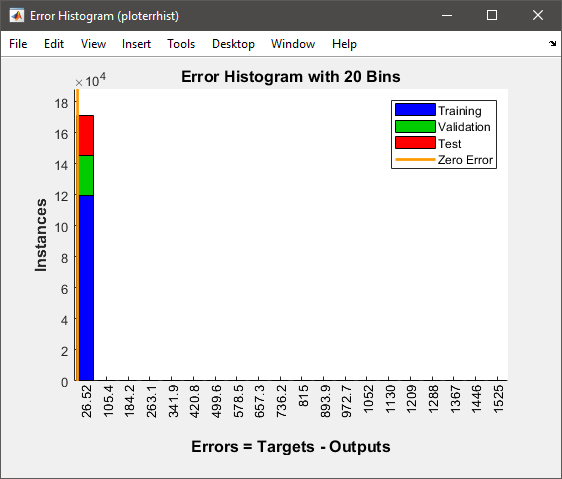


Error Histogram



I was not happy with the results so I increased the hidden layers from 10 to 100.

**Hidden Layers 100:**



Also I am not happy with these results. I have tried to increased hidden layer or changed the method to train. But I could not get good results. Because of that, I have used another dataset.

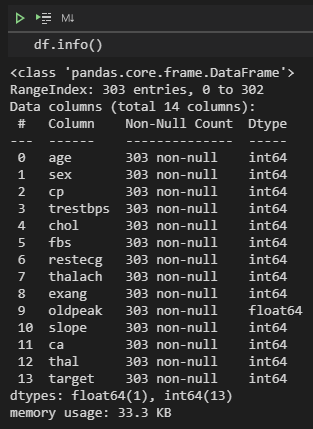
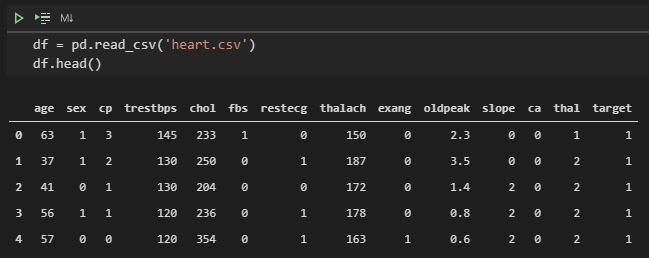
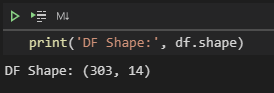
### DATASET 2

##### 1

**Selected Dataset:** Heart Disease UCI

|  |  |
| --- | --- |
| **Link**: | https://www.kaggle.com/ronitf/heart-disease-uci |
| **Description**: | This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. |
| **Format**: | A data frame with 303 observations on the following 14 variables. |

|  |  |
| --- | --- |
| age  age in years | cp  chest pain type |
| sex  (1 = male; 0 = female) | Trestbps  resting blood pressure (in mm Hg on admission to the hospital) |
| chol  serum cholestoral in mg/dl | fbs  (fasting blood sugar &gt; 120 mg/dl) (1 = true; 0 = false) |
| restecg  resting electrocardiographic results | thalach  maximum heart rate achieved |
| exang  exercise induced angina (1 = yes; 0 = no) | oldpeak  ST depression induced by exercise relative to rest |
| slope  the slope of the peak exercise ST segment | ca  number of major vessels (0-3) colored by flourosopy |
| thal  3 = normal; 6 = fixed defect; 7 = reversable defect | target  1 or 0 |

#### 2



Figure 1

As can be seen from the Figure 1, performing rearrangements on age, trstbps, chol and thalach attributes needed.

Data Rearrangements:

Age:

x <= 39 = 0,

39 < x <= 48 = 1,

48 < x <= 58 = 2,

58 < x <= 68 = 3,

68 < x = 4.

trstbps:

x <= 90 = 0,

90 < x <= 120 = 1,

120 < x <= 150 = 2,

150 < x <= 190 = 3,

190 < x = 4.

chol:

x <= 100 = 0,

100 < x <= 150 = 1,

150 < x <= 200 = 2,

200 < x <= 250 = 3,

250 < x <= 300 = 4,

300 < x <= 350 = 5,

350 < x <= 400 = 6,

400 < x = 0.

thalach:

x <= 110 = 0,

110 < x <= 130 = 1,

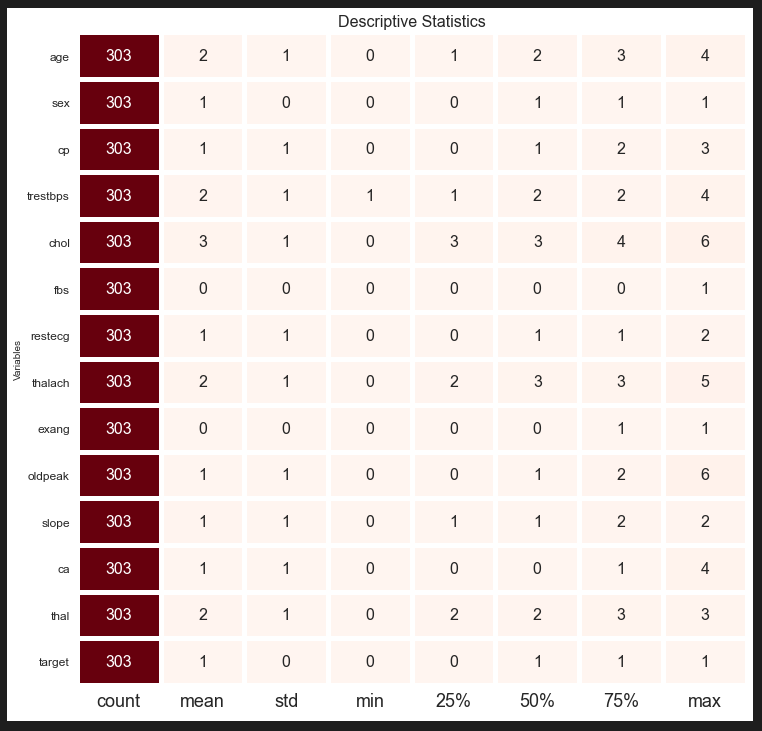
130 < x <= 150 = 2,

150 < x <= 170 = 3,

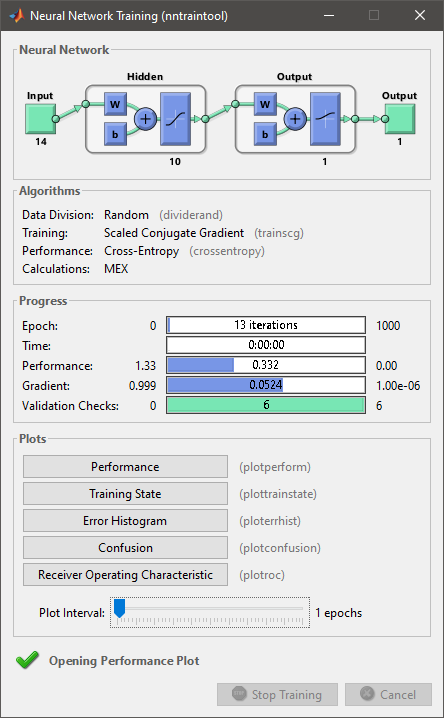
170 < x <= 190 = 4,

190 < x = 5.

After rearrangement:



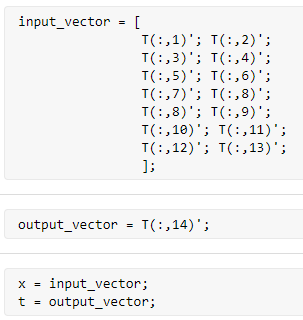
#### 3 - Architecture of ANN including values of parameters (learning rate, activation function)



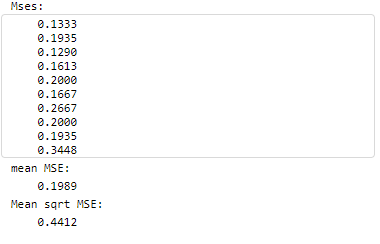
Learning Rate = 8.340315978212614e-05

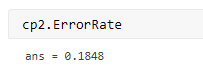
Training = Scaled conjugate gradient backpropagation

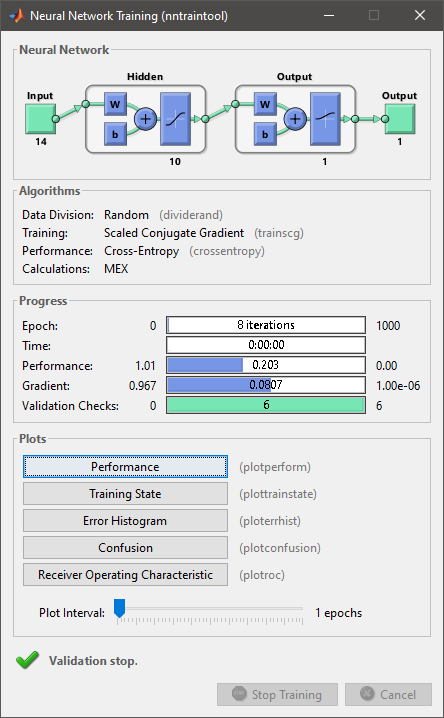
Activation Function =



#### 4 - Results of 10-fold cross validation experiments (cost function value at each fold, averaged value)







#### 

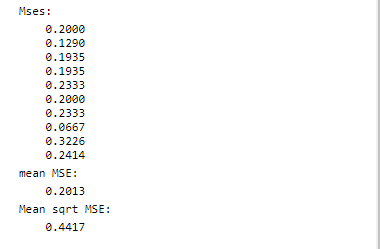
#### 

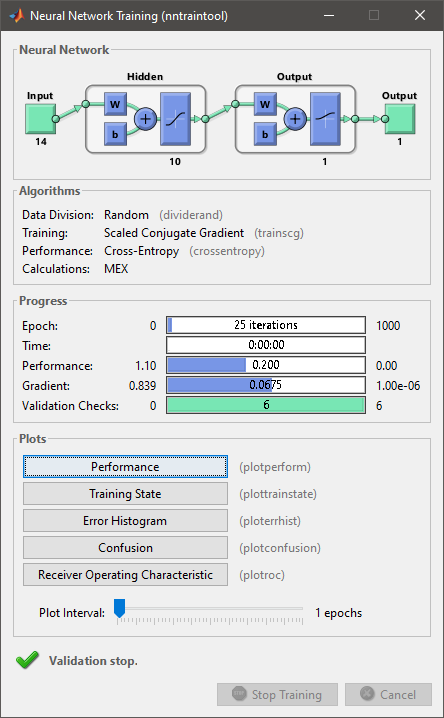
#### 

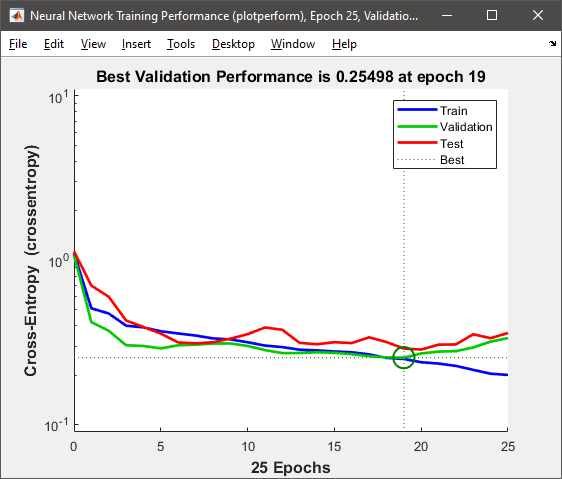
#### 

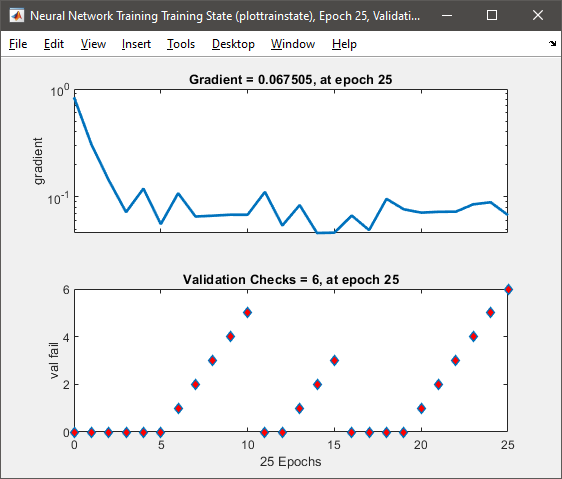
#### 5 - Description of measures taken in order to increase ANN performance. Repeat 4th step for documenting improved model performance.

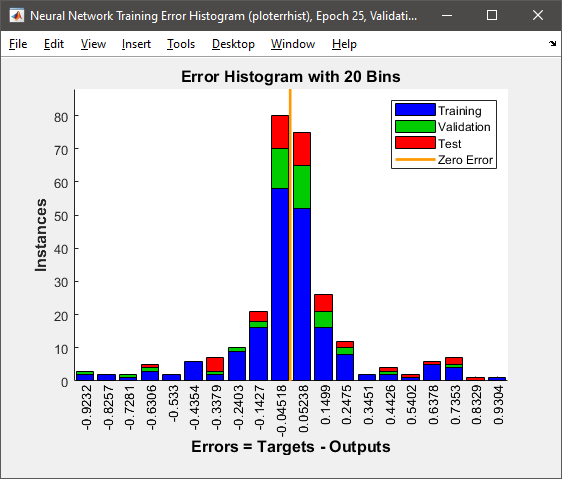
###### Learning rate is increased %5 percentage: 0.0000875733

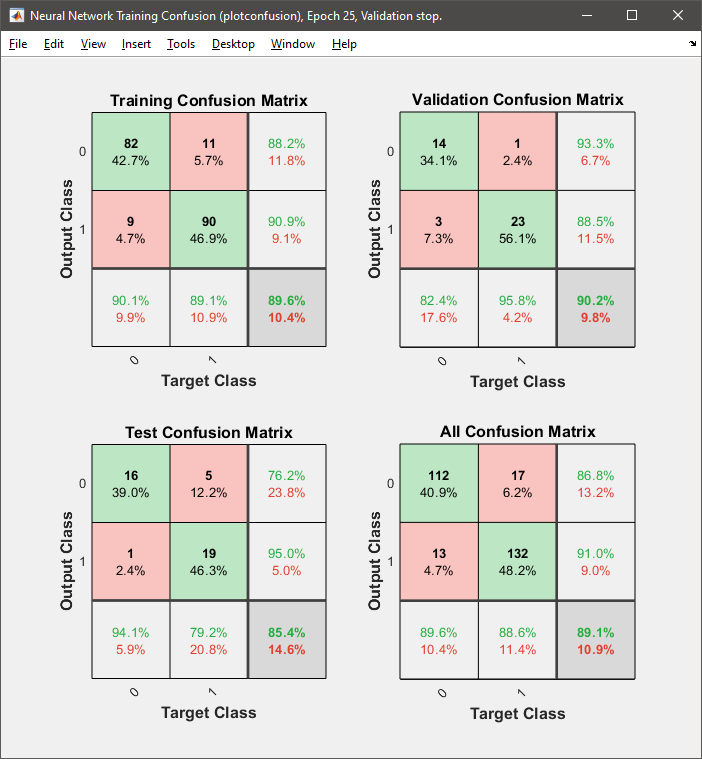


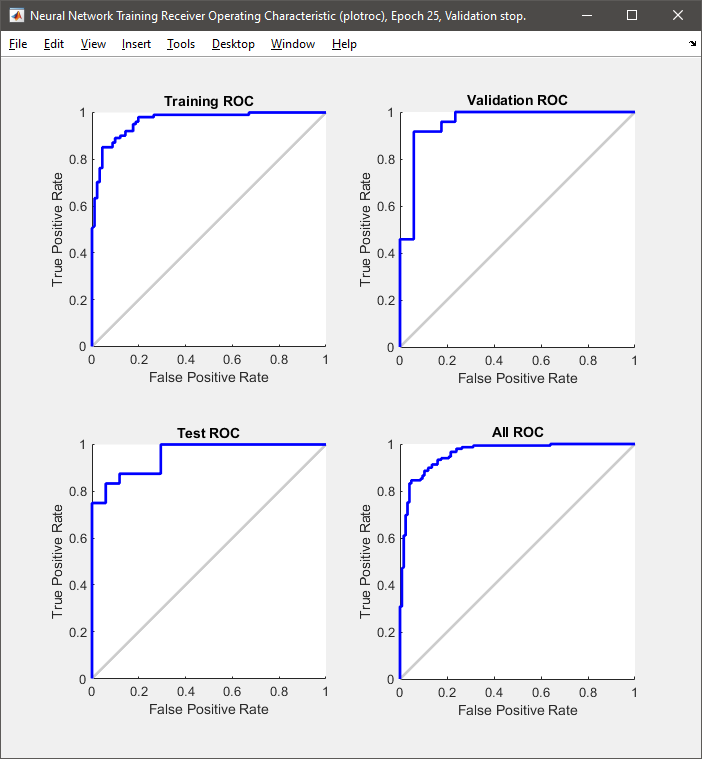




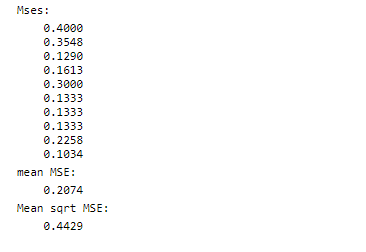


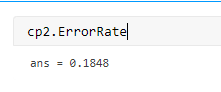


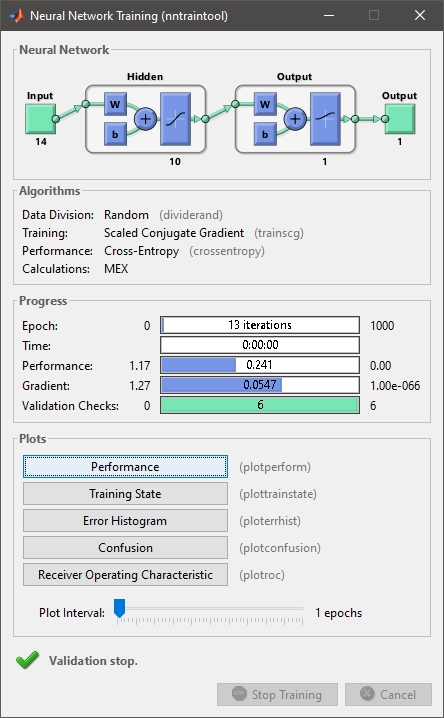


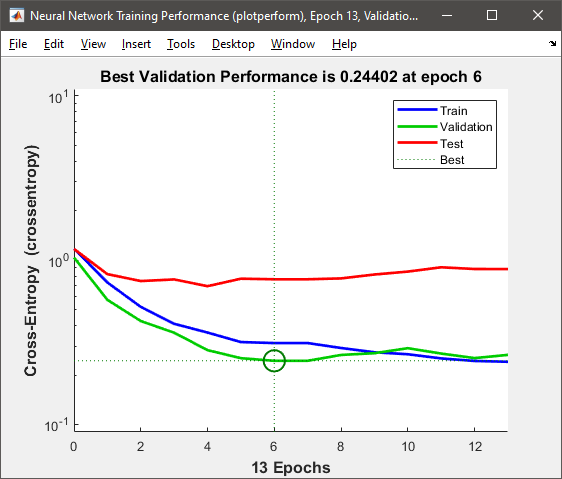
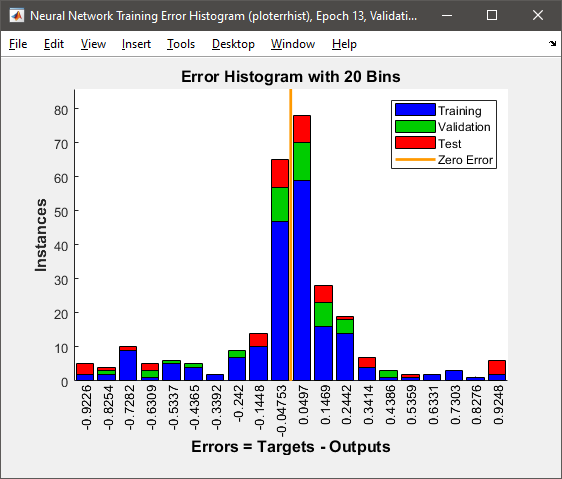
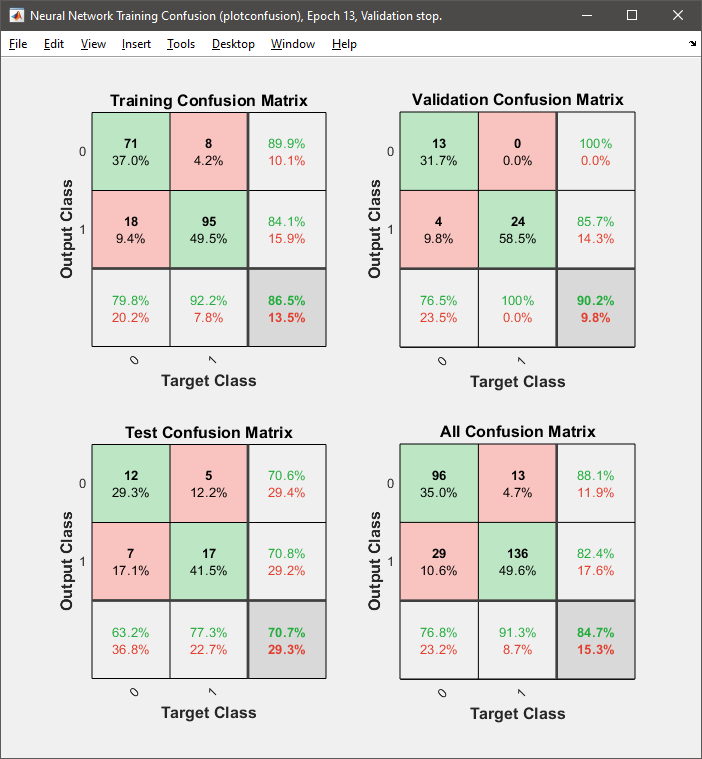
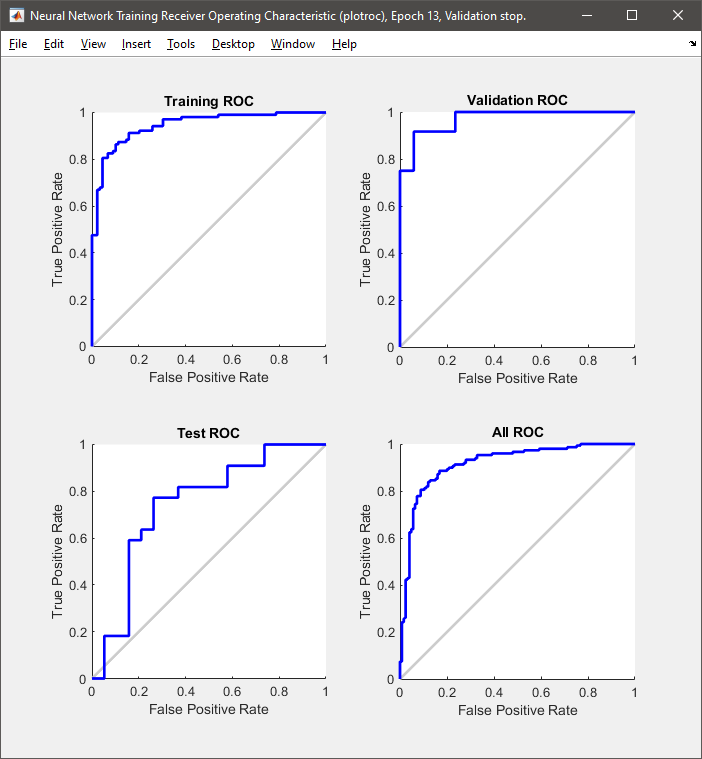


###### Learning rate is decreased %5 percentage: 0.000079233







##### Number of Hidden Layer Increased to 100 from 10

##### 

##### 

##### 

##### 

##### Changing the Training Function to Bayesian regularization backpropagation

##### 

##### 

##### 

##### 

##### 

## Conclusion

In this part;

I have tried to use Terrorism in Turkey dataset which is the dataset I’ve used for Lab 1. However, I haven’t got luck to get good results with that dataset. Because of that, I have changed and used Heart Disease UCI.

In Heart Disease UCI, there were some attributes that needs to be performed rearrangements, because wide range of numerical attribute values change by smaller number of intervals. As an example, I have decreased the range between ages to 5.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Index | Learning Rate | Training Func. | Best Validation Perf. | Gradient | Avg. MSE | Hidden Layer |
| **1** | 0.0000834032 | trainscg | 0.5011 | 0.081 | 0.1989 | 10 |
| **2** | 0.0000875733 | trainscg | 0.2549 | 0.068 | 0.2013 | 10 |
| **3** | 0.0000792333 | trainscg | 0.2440 | 0.055 | 0.2074 | 10 |
| **4** | 0.0000834032 | trainscg | 0.1919 | 0.536 | 0.2048 | 100 |
| **5** | 0.0000834032 | trainbr | 0.0074 | 0.023 | 0.2278 | 10 |

As can be seen on the table, increasing and also decreasing the learning rate increased average error. Increasing learning rate does not always means that it will give better performance because, optimality is much more important. In this example, 0.0000834032 learning rate is more optimal than the others.

Furhermore, according to the generated code by the Matlab,

The trainbr function defined like this:

`Takes longer but may be better for challenging problems.`

And also, the trainscg function defined like this:

`Uses less memory. Suitable in low memory situations.`

However, in this example, if you check 1st and 5th tests, the average MSE with trainscg function is less than the average MSE with trainbr function.

And finally, if you compare 1st and 4th tests, you will see that on the 1st test 10 hidden layers have been used. But, on the 4th test, 100 hidden layers have been used. And it can be seen on the table, gradient value of 4th test is huge comparing the other results. Also, 1st test has the highest Best Validation Performance.