Chongqing University of Technology

# 《MACHINE LEARNING》

## ALGORITHOMS

## NEURAL NETWORKS

Academic Term： 2020-21

Class：

Student ID： 62017010084

Student Name： Md Anower(an hao ming)

Teacher： Tan Shu qui

Score：

## Table of Contents

# 1. Abstract

Nowadays, large amount of data is available everywhere. Therefore, it is very important to analyze this data in order to extract some useful information and to develop an algorithm based on this analysis. This can be achieved through data mining and machine learning. Machine learning is an integral part of artificial intelligence, which is used to design algorithms based on the data trends and historical relationships between data. Machine learning is used in various fields such as bioinformatics, intrusion detection, Information retrieval, game playing, marketing, malware detection, image deconvolution and so on. This paper presents the work done by various authors in the field of machine learning in various application areas.

# 2. Introduction to Machine Learning

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks. A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

# 3. Neural Network Algorithm

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1. These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

## 3.1 Definition and History

Neural networks are mathematical models that use learning algorithms inspired by the brain to store information. Since neural networks are used in machines, they are collectively called an ‘artificial neural network.’ Nowadays, the term machine learning is often used in this field and is the scientific discipline that is concerned with the design and development of algorithms that allow computers to learn, based on data, such as from sensor data or databases. A major focus of machine-learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data. Hence, machine learning is closely related to fields such as statistics, data mining, pattern recognition, and artificial intelligence. Neural networks are a popular framework to perform machine learning, but there are many other machine-learning methods, such as logistic regression, and support vector machines. Similar to the brain, neural networks are built up of many neurons with many connections between them. Neural networks have been used in many applications to model the unknown relations between various parameters based on large numbers of examples. Examples of successful applications of neural networks are classifications of handwritten digits, speech recognition, and the prediction of stock prices. Moreover, neural networks are more and more used in medical applications. Many different types of neural networks exist. Examples of various types of neural networks are Hopfield network, the multilayer perceptron, the Boltzmann machine, and the Kohonen network.

## 3.2 Introduction

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

## 3.3 Application of Neural Network

Neural networks are broadly used, with applications for financial operations, enterprise planning, trading, business analytics and product maintenance. Neural networks have also gained widespread adoption in business applications such as forecasting and marketing research solutions, fraud detection and risk assessment. A neural network evaluates price data and unearths opportunities for making trade decisions based on the data analysis. The networks can distinguish subtle nonlinear interdependencies and patterns other methods of technical analysis cannot. According to research, the accuracy of neural networks in making price predictions for stocks differs. Some models predict the correct stock prices 50 to 60 percent of the time while others are accurate in 70 percent of all instances. Some have posited that a 10 percent improvement in efficiency is all an investor can ask for from a neural network.

## 3.4 Architecture of Neural Network

Humans and other animals process information with *neural networks*. These are formed from *trillions* of neurons (nerve cells) exchanging brief electrical pulses called action potentials. Computer algorithms that mimic these biological structures are formally called artificial neural networks to distinguish them from the squishy things inside of animals. However, most scientists and engineers are not this formal and use the term *neural network* to include both biological and no biological systems.

Neural network research is motivated by two desires: to obtain a better understanding of the human brain, and to develop computers that can deal with abstract and poorly defined problems. For example, conventional computers have trouble understanding speech and recognizing people's faces. In comparison, humans do extremely well at these tasks. Many different neural network structures have been tried, some based on imitating what a biologist sees under the microscope, some based on a more mathematical analysis of the problem. The most commonly used structure is shown in Fig. 3.4. This neural network is formed in three layers, called the input layer, hidden layer, and output layer. Each layer consists of one or more nodes, represented in this diagram by the small circles. The lines between the nodes indicate the flow of information from one node to the next. In this particular type of neural network, the information flows only from the input to the output (that is, from left-to-right).

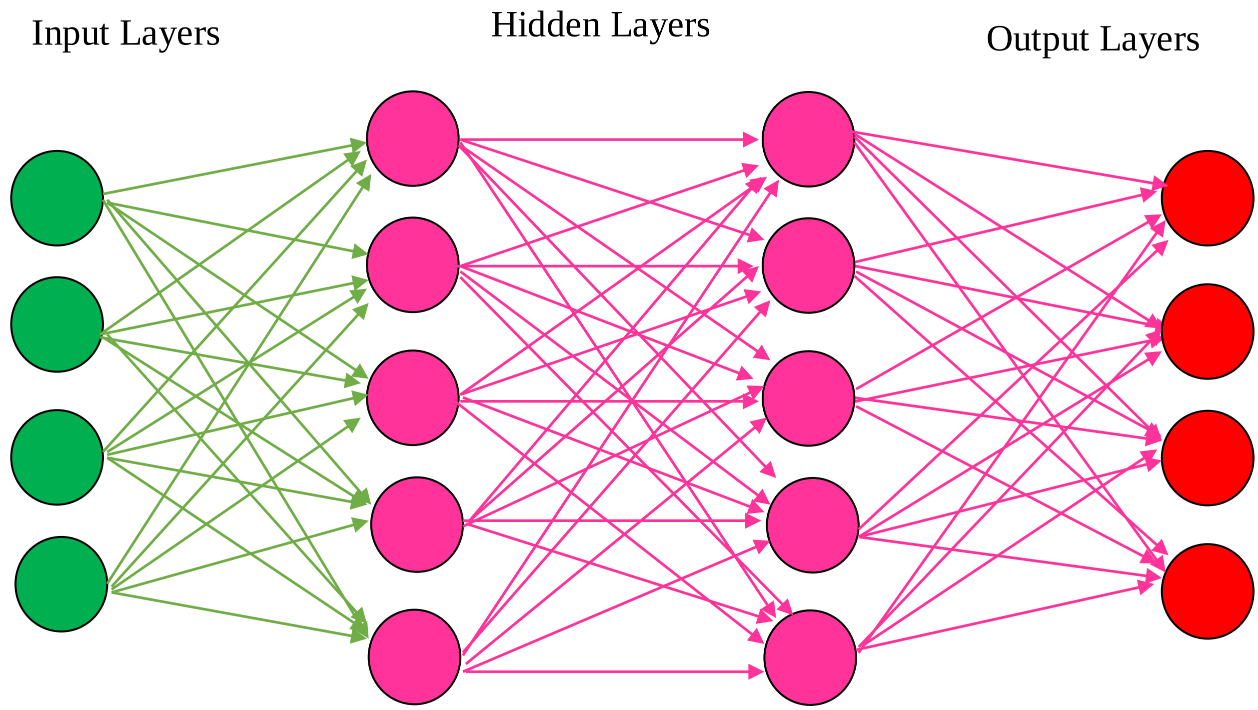



Figure 3.4: Sample Neural Network

# 4. Algorithms Descriptions

In a Neural Network, the learning (or training) process is initiated by dividing the data into three different sets:

- ❖ Training Dataset
- ❖ Validation Dataset
- ❖ Test Dataset

## 4.1 Training dataset

A training dataset is a dataset of examples used during the learning process and is used to fit the parameters of, for example, a classifier. For classification tasks, a supervised learning algorithm looks at the training dataset to determine, or learn, the optimal combinations of variables that will generate a good predictive model. The goal is to produce a trained (fitted) model that generalizes well to new, unknown data. The fitted model is evaluated using “new” examples from the held out datasets (validation and test datasets) to estimate the model’s accuracy in classifying new data. To reduce the risk of issues such as overfitting, the examples in the validation and test datasets should not be used to train the model. Most approaches that search through training data for empirical relationships tend to over fit the data, meaning that they can identify and exploit apparent relationships in the training data that do not hold in general.

## 4.2 Validation dataset

And a validation dataset is a dataset of examples used to tune the hyper parameters of a classifier. It is sometimes also called the development set or the. An example of a hyper parameter for artificial neural networks includes the number of hidden units in each layer. It, as well as the testing set, should follow the same probability distribution as the training dataset. In order to avoid overfitting, when any classification parameter needs to be adjusted, it is necessary to have a validation dataset in addition to the training and test datasets. For example, if the most suitable classifier for the problem is sought, the training dataset is used to train the candidate algorithms, the validation dataset is used to compare their performances and decide which one to take and, finally, the test dataset is used to obtain the performance characteristics such as accuracy, sensitivity, specificity, F-measure, and so on. The validation dataset functions as a hybrid: it is training data used for testing, but neither as part of the low-level training nor as part of the final testing.

Since our goal is to find the network having the best performance on new data, the simplest approach to the comparison of different networks is to evaluate the error function using data which is independent of that used for training. Various networks are trained by minimization of an appropriate error function defined with respect to a training data set. The performance of the networks is then compared by evaluating the error function using an independent validation set, and the network having the smallest error with respect to the validation set is selected. This approach is called the hold out method. Since this procedure can itself lead to some overfitting to the validation set, the performance of the selected network should be confirmed by measuring its performance on a third independent set of data called a test set.

## 4.3 Test dataset

A test dataset is a dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset. If a model fit to the training dataset also fits the test dataset well, minimal overfitting has taken place (see figure below). A better fitting of the training dataset as opposed to the test dataset usually points to overfitting. A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier. To do this, the final model is used to predict classifications of examples in the test set. Those predictions are compared to the examples' true classifications to assess the model's accuracy.

In a scenario where both validation and test datasets are used, the test dataset is typically used to assess the final model that is selected during the validation process. In the case where the original dataset is partitioned into two subsets (training and test datasets), the test dataset might assess the model only once. Note that some sources advise against such a method. However, when using a method such as cross-validation, two partitions can be sufficient and effective since results are averaged after repeated rounds of model training and testing to help reduce bias and variability.

# 5. Implementation Neural Network

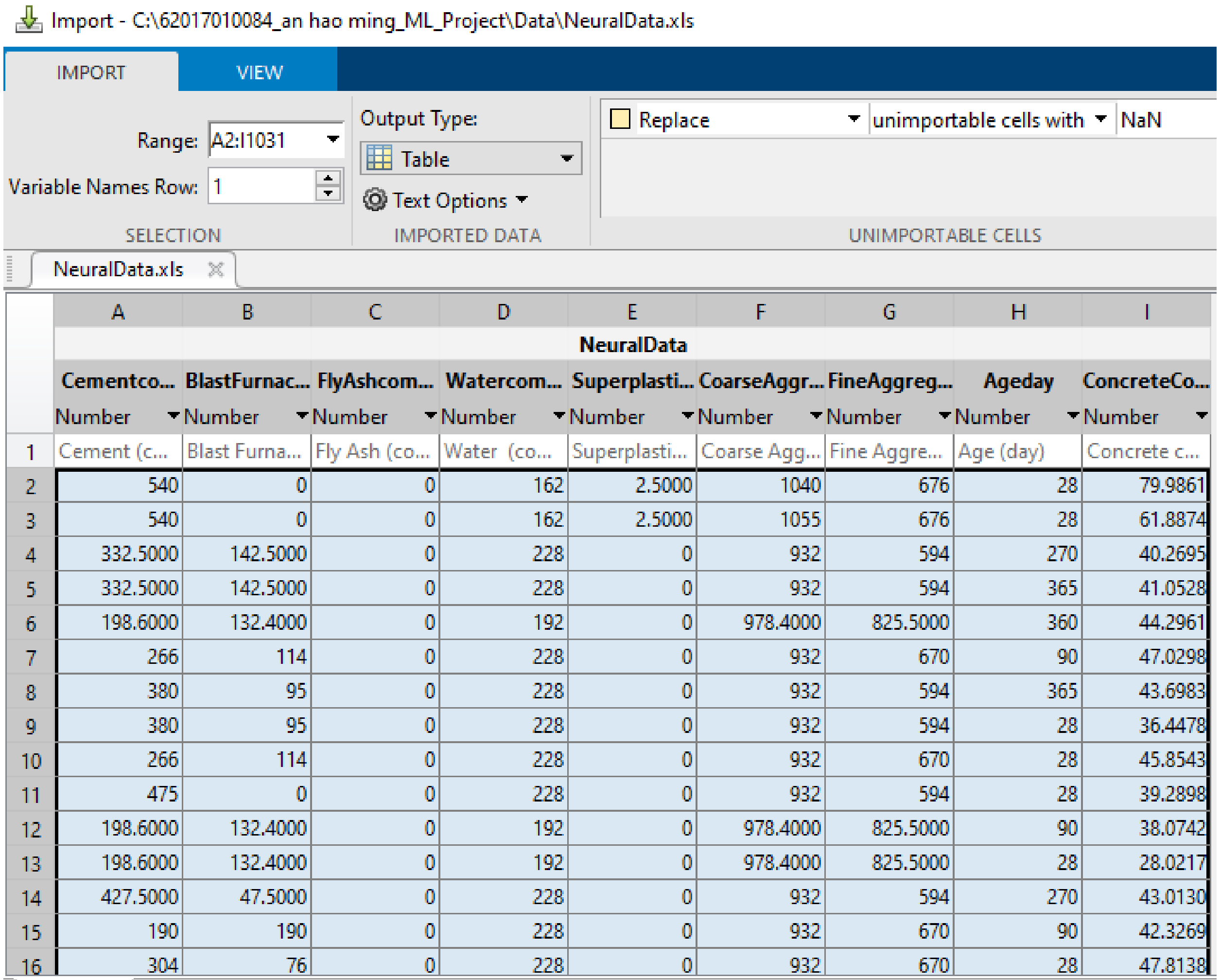
It’s time to implement the code for Neural Network here I will use MATLAB programming language. For this Neural network implementation, I will use the dataset " **NeuralData.xls”**.

## 5.1 Import Data

Before Implement the Neural Network we need to import data. we need it in a specific way so we need to process it accordingly. The preprocessing of data depends on the type of data.

```
clear
clc

ReadData = readtable('Data/NeuralData.xls');
summary(ReadData)
X = table2array(ReadData(:,1:8));
Y = table2array(ReadData(:,9));
plotmatrix(X,Y)
X = X';
Y = Y';
```




| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| | NeuralData | | | | | | | | |
| | Cementco... | BlastFurnac... | FlyAshcom... | Watercom... | Superplasti... | CoarseAggr... | FineAggreg... | Ageday | ConcreteCo... |
| | Number | Number | Number | Number | Number | Number | Number | Number | Number |
| 1 | Cement (c... | Blast Furna... | Fly Ash (co... | Water (co... | Superplasti... | Coarse Agg... | Fine Aggre... | Age (day) | Concrete c... |
| 2 | 540 | 0 | 0 | 162 | 2.5000 | 1040 | 676 | 28 | 79.9861 |
| 3 | 540 | 0 | 0 | 162 | 2.5000 | 1055 | 676 | 28 | 61.8874 |
| 4 | 332.5000 | 142.5000 | 0 | 228 | 0 | 932 | 594 | 270 | 40.2695 |
| 5 | 332.5000 | 142.5000 | 0 | 228 | 0 | 932 | 594 | 365 | 41.0528 |
| 6 | 198.6000 | 132.4000 | 0 | 192 | 0 | 978.4000 | 825.5000 | 360 | 44.2961 |
| 7 | 266 | 114 | 0 | 228 | 0 | 932 | 670 | 90 | 47.0298 |
| 8 | 380 | 95 | 0 | 228 | 0 | 932 | 594 | 365 | 43.6983 |
| 9 | 380 | 95 | 0 | 228 | 0 | 932 | 594 | 28 | 36.4478 |
| 10 | 266 | 114 | 0 | 228 | 0 | 932 | 670 | 28 | 45.8543 |
| 11 | 475 | 0 | 0 | 228 | 0 | 932 | 594 | 28 | 39.2898 |
| 12 | 198.6000 | 132.4000 | 0 | 192 | 0 | 978.4000 | 825.5000 | 90 | 38.0742 |
| 13 | 198.6000 | 132.4000 | 0 | 192 | 0 | 978.4000 | 825.5000 | 28 | 28.0217 |
| 14 | 427.5000 | 47.5000 | 0 | 228 | 0 | 932 | 594 | 270 | 43.0130 |
| 15 | 190 | 190 | 0 | 228 | 0 | 932 | 670 | 90 | 42.3269 |
| 16 | 304 | 76 | 0 | 228 | 0 | 932 | 670 | 28 | 47.8138 |

Figure 5.1: Imported Data

## 5.2 Train Data

When we start off with our neural network we initialize our weights randomly. Obviously, it won't give you very good results. In the process of training, we want to start with a bad performing neural network and wind up with network with high accuracy. In terms of loss function, we want our loss function to much lower in the end of training. Improving the network is possible, because we can change its function by adjusting weights. We want to find another function that performs better than the initial one.

```
trainFcn = 'trainlm';
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize,trainFcn);
view(net)
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};
net.divideFcn = 'dividerand';
net.divideMode = 'sample';
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
net.performFcn = 'mse';
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', 'plotregression', 'plotfit'};
[net,tr] = train(net,X,Y);
figure, plotperform(tr)
figure, plottrainstate(tr)
```

## 5.3 Test & Validation

An application of this process is in early stopping, where the candidate models are successive iterations of the same network, and training stops when the error on the validation set grows, choosing the previous model (the one with minimum error).

```
Ytest = net(X);
e = gsubtract(Y,Ytest);
performance = perform(net, Y,Ytest);
figure, ploterrhist(e)
trOut = Ytest(tr.trainInd);
vOut = Ytest(tr.valInd);
tsOut = Ytest(tr.testInd);
trTarg = Y(tr.trainInd);
vTarg = Y(tr.valInd);
tsTarg = Y(tr.testInd);
plotregression(trTarg, trOut, 'Train', vTarg, vOut, 'Validation', tsTarg, tsOut, 'Testing',Y,Ytest,'All')
```

# 6. Result

## 6.1 Function Fitting Neural Network

Neural networks provide a new tool for the fast solution of repetitive nonlinear curve fitting problems. In this article we introduce the concept of a neural network, and we show how such networks can be used for fitting functional forms to experimental data. The neural network algorithm is typically much faster than conventional iterative approaches. In addition, further substantial improvements in speed can be obtained by using special purpose hardware implementations of the network, thus making the technique suitable for use in fast real-time applications.

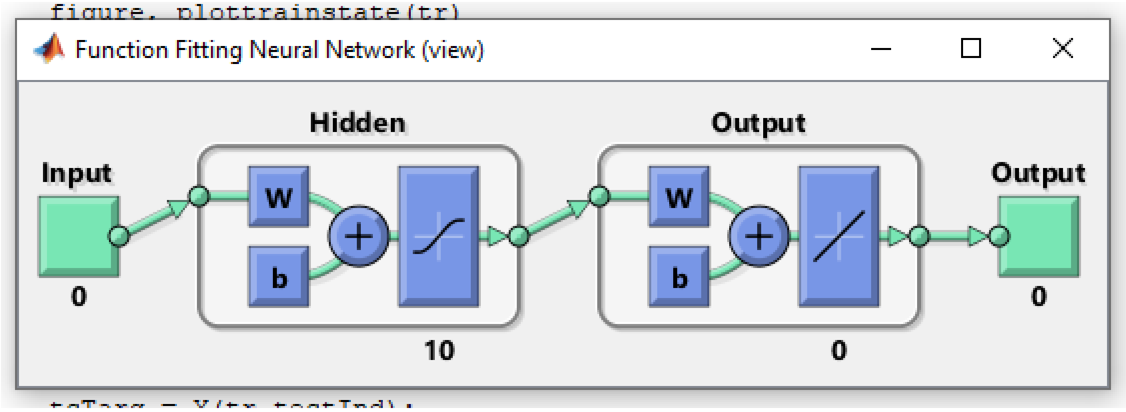



Figure 6.1: Function Fitting Neural Network

## 6.2 Neural Network Train Tool

There are four ways you can use the Neural Network Toolbox software. The first way is through the four graphical user interfaces (GUIs) that are described in this chapter. These provide a quick and easy way to access the power of the toolbox for the following tasks:

- ❖ Function fitting
- ❖ Pattern recognition
- ❖ Data clustering
- ❖ Time series analysis

The second way to use the toolbox is through basic command-line operations. The command-line operations offer more flexibility than the GUIs, but with some added complexity. This chapter introduces some of the command-line functions, but the next seven chapters cover command-line operations in more detail. The next two chapters are important to understanding the use of the command line, and the fundamentals of training neural networks. You should read them before advancing to later topics. Here you know I have used Function fitting.

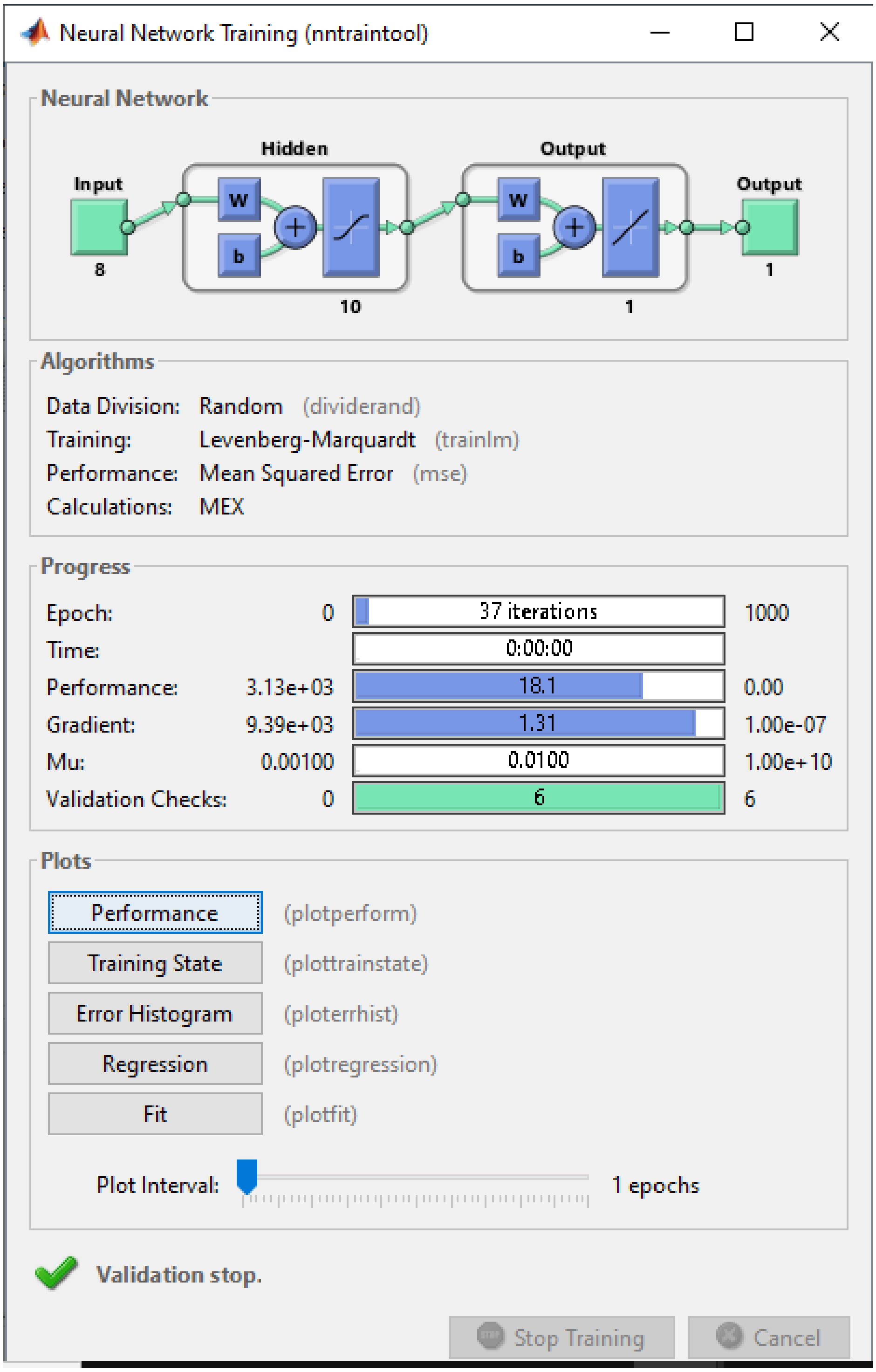



Figure 6.2: NN Training Tool

## 6.3 Performance

The performance of ANNs has also been compared with that of other types of field interpretation criteria based on localized loss. Disc topography data have also been added to visual field data to improve the diagnostic ability of ANNs.

Here the blue line indicated the train graph line and about train has expressed before. The another important result is validation which has been indicated by green line. the test case is also expressed before here the test line is red and the best cases are dotted.

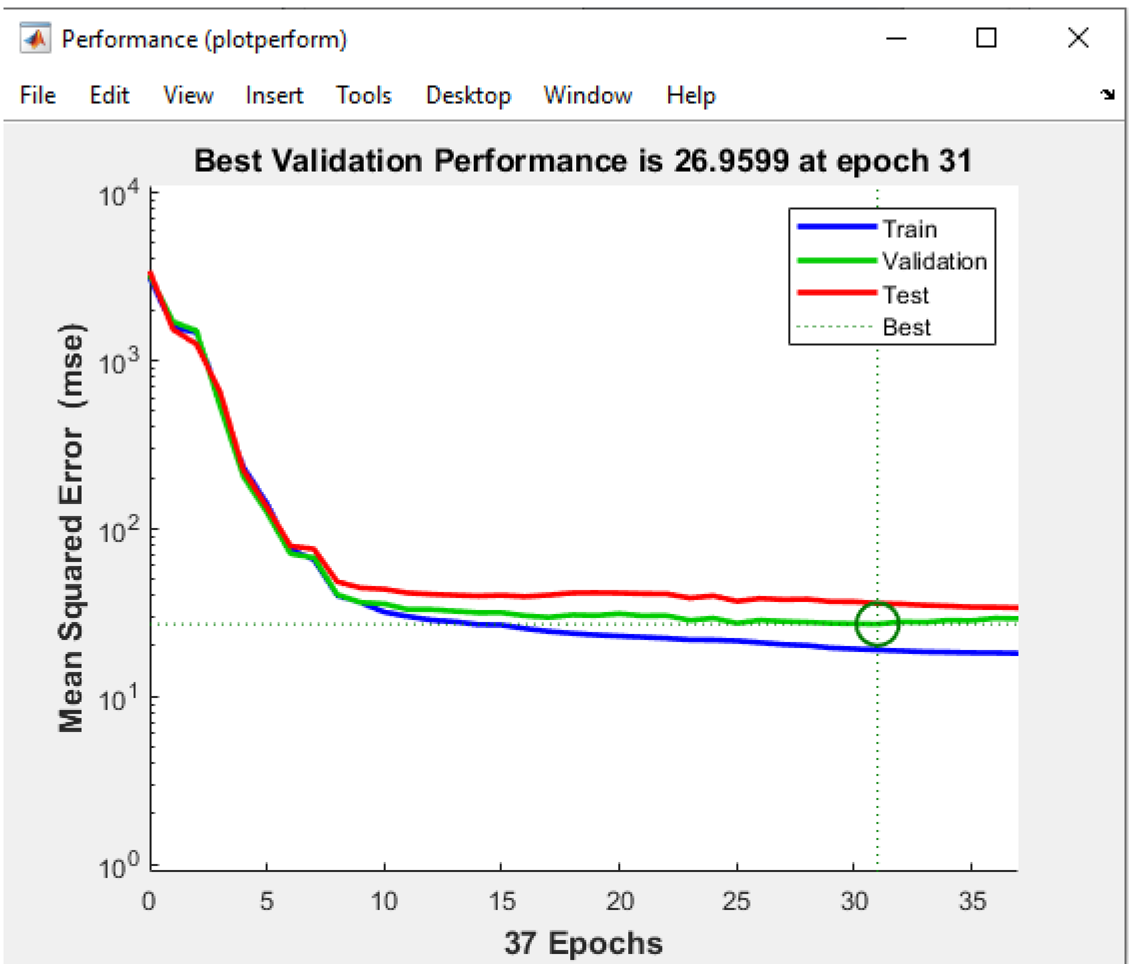



Figure 6.3: Performance

## 6.4 Training State

The result of Training State

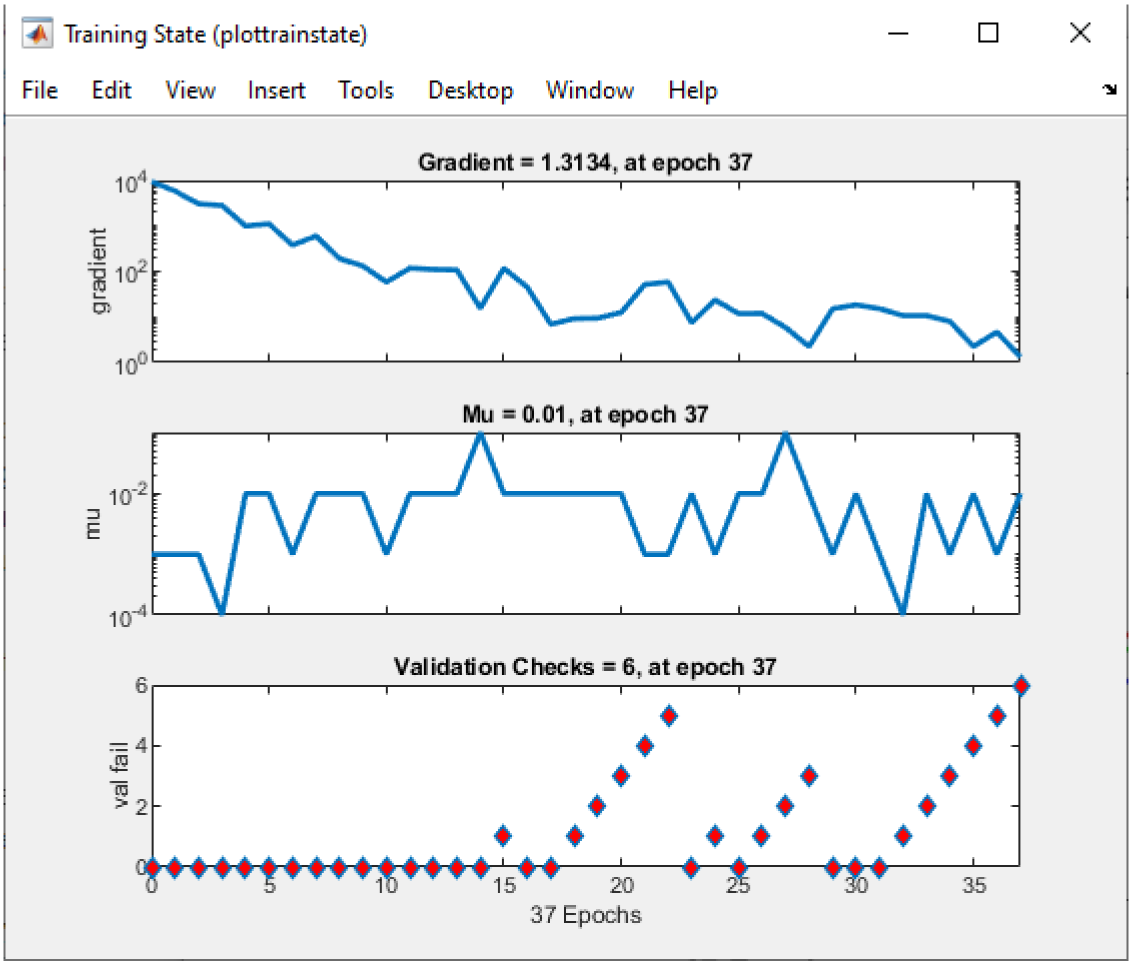



Figure 6.4: Training State

## 6.5 Regression

Neural networks are reducible to regression models. a neural network can "pretend" to be any type of regression model. For example, this very simple neural network, with only one input neuron, one hidden neuron, and one output neuron, is equivalent to a logistic regression. It takes several dependent variables = input parameters, multiplies them by their coefficients = weights, and runs them through a sigmoid activation function and a unit step function, which closely resembles the logistic regression function with its error term.

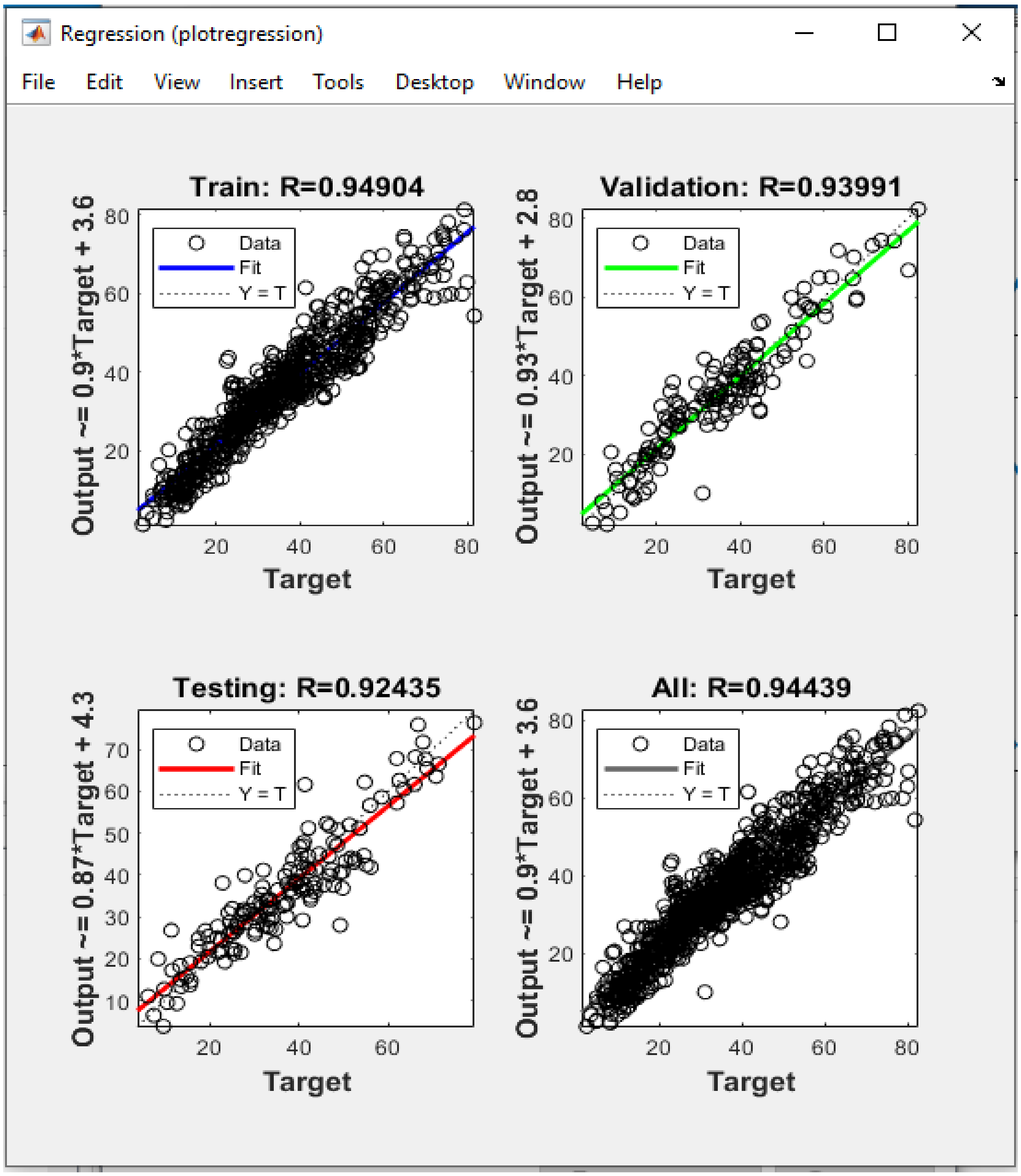



Figure 6.5: Regression

## 6.6 Training Error Histogram

Error histograms show how the errors from the neural network on the testing instances are distributed. In general, a normal distribution centered at 0 for each output variable is expected here. The next figure illustrates the histogram of errors made by a neural network when predicting sailing yachts' residuary resistance. Here the number of bins is 20.

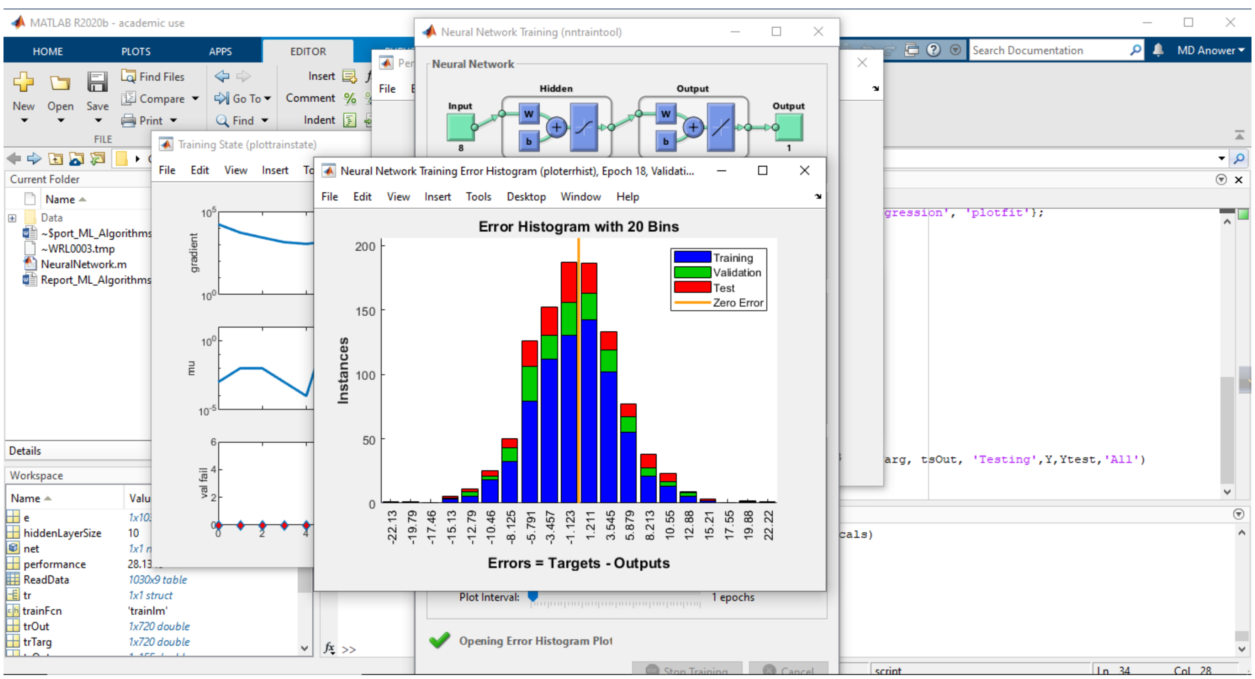



Figure 6.6: Error Histogram

# 7. Conclusion

We've covered some important theoretical information related to neural-network training data, and we did an initial training and validation experiment with our Python-language multilayer Perceptron. I hope that you're enjoying AAC's series on neural networks—we've made a lot of progress since the first article, and there's still much more that we need to discuss.

*** END ***