Chapter 2

Error Calculation Methods

- Understanding Error Calculation
- The Error Function
- Error Calculation Methods
- How is the Error Used

In this chapter we will see how to calculate errors for a neural network. When performing supervised training a neural network's actual output must be compared against the ideal output specified in the training data. The difference between actual and ideal output is the error of the neural network.

Error calculation occurs at two levels. First there is the local error. This is the difference between the actual output of one individual neuron and the ideal output that was expected. The local error is calculated using an error function.

The local errors are aggregated together to form a global error. The global error is the measurement of how well a neural network performs to the entire training set. There are several different means by which a global error can be calculated. The global error calculation methods discussed in this chapter are listed here.

- Sum of Squares Error (ESS)
- Mean Square Error (MSE)

- Root Mean Square (RMS)

Usually you will simply use MSE. MSE is the most common means of calculating errors for a neural network. Later in the book we will use ESS. The Levenberg Marquardt Algorithm (LMA), which will be covered in Chapter 8, requires ESS. Lastly, RMS can be useful in certain situations.

2.1 The Error Function

We will start with the local error. The local error comes from the error function. The error function is fed the actual and ideal outputs for a single output neuron. The error function then produces a number that represents the error of that output neuron. Training methods will see to minimize this error.

This book will cover two error functions. The first is the simple linear error function that is most commonly used. The second is the arctangent error function that is introduced by the Quick Propagation training method. Arctangent error functions and Quick Propagation will be discussed in Chapter 6, "Quick Propagation". This chapter will focus on the standard linear error function. The formula for the linear error function can be seen in Equation 2.1.

$$E = (i - a) \tag{2.1}$$

The linear error function is very simple. The error is simply the difference between the ideal (**i**) and actual (**a**) outputs from the neural network. The only requirement of the error function is that it produce an error that you would like to minimize.

To see an example of this, consider a neural network output neuron that produced 0.9 when it should have produced 0.8. The error for this neural network would be the difference between 0.8 and 0.9, which is -0.1.

In some cases you may not provide an ideal output to the neural network and still use supervised training. In this case you would write an error function that somehow evaluates the output of the neural network for the given input. This evaluation error function would need to assign some sort of a score to the neural network. A higher number would indicate less desirable output; a lower number would indicate more desirable output. The training process would attempt to minimize this score.

2.2 Calculating Global Error

Now that we have seen how to calculate the local error, we will move on to global error. Because MSE error calculation is the most common, we will begin here. You can see the equation that is used to calculate MSE in Equation 2.2.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} E^2 \tag{2.2}$$

As you can see from the above equation, it makes use of the local error (E) that we defined in the last section. Each local error is squared and summed. The resulting sum is then divided by the total number of cases. In this way the MSE error is similar to a traditional average, except that each local error is squared. The squaring negates the effect of some errors being positive and others being negative. This is because a positive number squared is a positive number, just as a negative number square is also a positive number. If you are unfamiliar with the summation operator, shown as a capital Greek letter sigma, refer to Chapter 1.

The MSE error is typically written as a percentage. The goal is to decrease this error percentage as training progresses. To see how this is used consider the following program output.

```
Beginning training...
Iteration #1 Error:51.023786% Target Error: 1.000000%
Iteration #2 Error:49.659291% Target Error: 1.000000%
Iteration #3 Error:43.140471% Target Error: 1.000000%
Iteration #4 Error:29.820891% Target Error: 1.000000%
Iteration #5 Error:29.457086% Target Error: 1.000000%
Iteration #6 Error:19.421585% Target Error: 1.000000%
Iteration #7 Error:2.160925% Target Error: 1.000000%
Iteration #8 Error:0.432104% Target Error: 1.000000%
Input=0.0000,0.0000, Actual=0.0091, Ideal=0.0000
Input=1.0000,0.0000, Actual=0.9793, Ideal=1.0000
Input=0.0000,1.0000, Actual=0.9472, Ideal=1.0000
Input=1.0000,1.0000, Actual=0.0731, Ideal=0.0000
Machine Learning Type: feedforward
Machine Learning Architecture: ?:B->SIGMOID->4:B->SIGMOID->?
Training Method: lma
Training Args:
```

The above shows a program learning the XOR operator. Notice how the MSE error drops each iteration? Finally, but iteration eight the error is below one percent, ant training stops.

2.3 Other Error Calculation Methods

Though MSE is the most common method of calculating global error, it is not the only method. In this section we will look at two other global error calculation methods.

2.3.1 Sum of Squares Error

The sum of squares method (ESS) uses a similar formula as the MSE error method. However, ESS does not divide by the number of elements. As a result, the ESS is not a percent. It is simply a number that is larger depending on how severe the error is. Equation 2.3 shows the MSE error formula.

$$\text{ESS} = \frac{1}{2}\sum_{p} E^2 \tag{2.3}$$

As you can see above, the sum is not divided by the number of elements. Rather, the sum is simply divided in half. This results in an error that is not a percent, but rather a total of the errors. Squaring the errors eliminates the effect of positive and negative errors.

Some training methods require that you use ESS. The Levenberg Marquardt Algorithm (LMA) requires that the error calculation method be ESS. LMA will be covered in Chapter 8.

2.3.2 Root Mean Square Error

The Root Mean Square (RMS) error method is very similar to the MSE method previously discussed. The primary difference is that the square root is taken of the sum. You can see the RMS formula in Equation 2.4.

$$\text{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} E^2} \tag{2.4}$$

Root mean square error will always be higher than MSE. The following output shows the calculated error for all three error calculation methods. All three cases used the same actual and ideal values.

```
Trying from -1.00 to 1.00
```

```
Actual:[-0.36,0.07,0.55,0.05,-0.37,0.34,-0.72,-0.10,-0.41,-0.32]
Ideal:[-0.37,0.06,0.51,0.06,-0.36,0.35,-0.67,-0.09,-0.43,-0.33]
Error (ESS): 0.00312453
Error (MSE): 0.062491%
Error (RMS): 2.499810%
```

RMS is not used very often for neural network error calculation. RMS was originally created in the field of electrical engineering. I have not used RMS a great deal. Many research papers involving RMS show it being used for waveform analysis.

2.4 Chapter Summary

Neural networks start with random values for weights. These networks are then trained until a set of weights is found that provides output from the neural network that closely matches the ideal values from the training data. For training to progress a means is needed to evaluate the degree to which the actual output from the neural network matches the ideal output expected of the neural network.

This chapter began by introducing the concept of local and global error. Local error is the error used to measure the difference between the actual and ideal output of an individual output neuron. This error is calculated using an error function. Error functions are only used to calculate local error.

Global error is the total error of the neural network across all output neurons and training set elements. Three different techniques were presented in this chapter for the calculation of global error. Mean Square Error (MSE) is the most commonly used. Sum of Squared Errors (ESS) is used by some training methods to calculate error. Root Mean Square (RMS) can be used to calculate the error for certain applications. RMS was created for the field of electrical engineering for waveform analysis.

The next chapter will introduce a mathematical concept known as derivatives. Derivatives come from Calculus and will be used to analyze the error functions and adjust the weights to minimize this error. In this book we will learn about several propagation training techniques. All propagation training techniques use derivatives to calculate update values for the weights of the neural network.