***Neural Networks Basics***

Table of Contents

[Introduction 2](#_Toc349233778)

[Biological and Artificial Neurons 4](#_Toc349233779)

[Network Training 5](#_Toc349233780)

[Gradient Descent Algorithm 6](#_Toc349233781)

[The Back-propagation Algorithm 9](#_Toc349233782)

[Appendix – Primer on Calculus 12](#_Toc349233783)

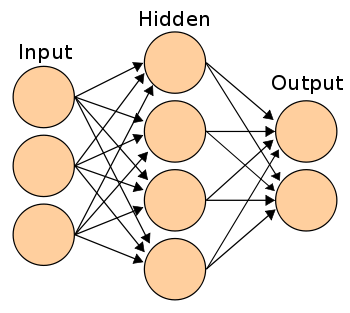
[Power Rule 13](#_Toc349233784)

[Derivative Chain Rule 13](#_Toc349233785)

[References 14](#_Toc349233786)

# Introduction

Neural networks are members of family of computational architectures inspired by biological brains. Our brains have billions of cells called neurons. Each neuron forms a complex network with other neurons. This network allows signals to flow from one neuron to thousands of other neurons. As a result, we are able to synthesize information from different neurons. Our ability to synthesize information from different sources is the basis of intelligence. The computational equivalent of brain neural network follows a similar model. A computational neural network typically consists of 3 conceptual layers. Each layer has several nodes which are similar to neurons. Each node in a layer is connected to nodes in the subsequent layer. Each connection represents a weight which is applied to the output of the source node. Since only forward connections are allowed, this network is known as feed forward neural network. The process of learning in an artificial neural network consists of arriving at the optimal values of connection weights so that observed error is minimized.



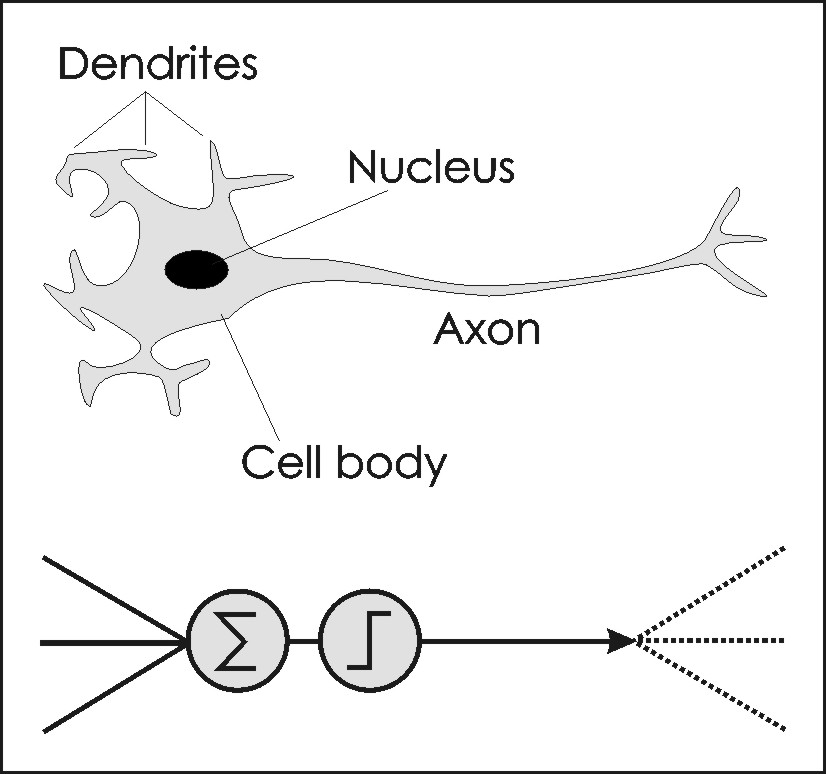
Neural networks have advantages when dealing with data that does not

adhere to the generally chosen low order polynomial forms, or data for which there is little *a priori* knowledge of the appropriate mathematical model to select for regression modeling2. For classification problems, neural networks also work generally better as compared to regression3. Fitting data using regression does not involve time consuming training and hence should be the preferred method when a simple model is desired. The following situations are ideal for the application of a neural network:

1. One expects correlations between data but one does not know any rules or laws that could be used for a prediction.
2. The relationship between input and output variables is complex and no simple and reliable model currently exists.

# Biological and Artificial Neurons

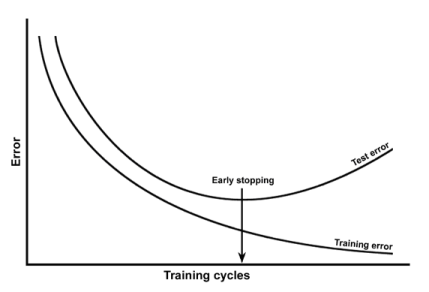
The figure below shows the comparison between artificial and biological neurons. A biological neuron receives inputs from dendrites and passes the received inputs using axon to other neurons in its network. Similarly, an artificial neuron first forms a weighted sum of the inputs received and then passes the resulting input into an activation function. The output of the activation function is then sent to other neurons for further processing.



Despite the apparent similarity between biological an artificial neural networks, it must be emphasized that biological networks are way more complex as compared to artificial neural networks. In an adult brain there are 100 billion neurons and each neuron forms connections with 50,000 other neurons! An artificial neural network is much smaller and may contain only a few hundred connections.

# Network Training

A neural network needs a teacher to tell what the desired output should be. This is called supervised learning. During training, features extracted from the historical data enter the neural network's input layer of neurons. The neuron activation’s travel through the network to the final outputs where comparison with the known solution occurs. A training algorithm alters the weights on the network's internal connections in accordance with the difference between the network outputs (the predicted solution) and the known solution, so the next time the network receives a similar example its output is closer to the required pattern. This process repeats until the network has learned to produce the required response to each input. Once a neural network has been trained it must be evaluated to see if it is ready for actual use. This final step is important so that it can be determined if additional training is required. To correctly validate a neural network, validation data must be set aside that is completely separate from the training data. The following graph shows how error (the difference between actual and predicted value) stabilizes after a certain number of training cycles have been completed.



# Gradient Descent Algorithm

The reader must be wondering how error reduction happens in an artificial neural network. An artificial neural network employs a popular approach called gradient descent to alter weights and minimize the error. Suppose you are skiing on a hill and need to reach the valley quickly. You will take the steepest edge in the downhill direction. This is the idea of gradient descent. The following graph shows the relationship between error and weight. If we take steps proportional to the gradient or slope of tangent at a point, we can construct a convergent algorithm to reach the minimum error denoted by Emin. In our initial position, we will fall quickly as gradient is steep. As we get closer to minimum, our step size will reduce and will be zero when we reach the minimum (slope is zero at minimum). We now have an algorithm which will eventually converge. If α denote a small number, our step can be governed by equation:

------ (1)

W2

W4

W3

Weight

W1

Error

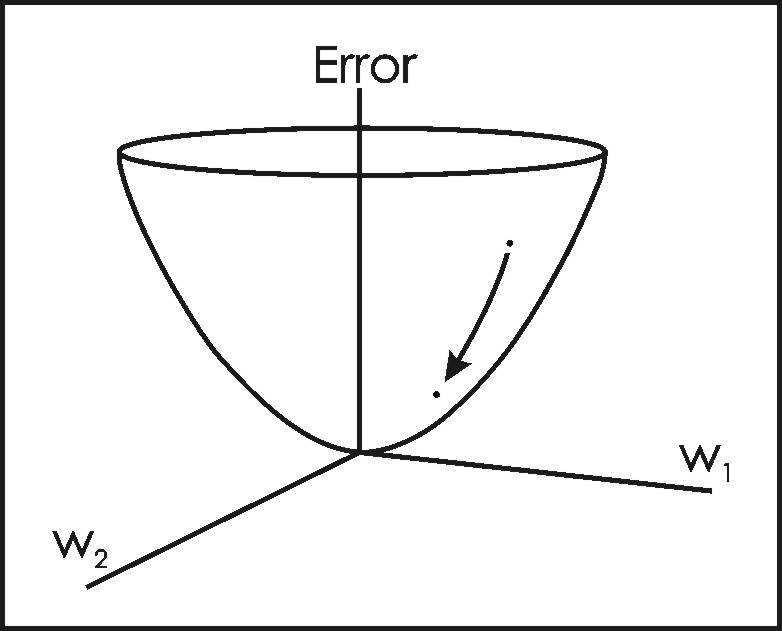
Emin

If our initial point is W1 our slope will be negative. As a result, we will end up incrementing our initial position and reach W2. If we are at W3 our slope at W3 is positive, so we will end up decrementing our initial position and reach W4. The parameter α denotes the size of our steps and is known as learning rate. If our learning rate or simply step size is reasonable, we may eventually reach the minimum possible error. Since slope is zero at the minima, the new position will be same as the initial position. In other words, we will see the error stabilize.

The speed of convergence of a network can be improved by increasing the learning rate. Unfortunately, increasing will usually result in increasing network instability, with weight values oscillating erratically around the minimum. Instead of changing learning rate, most standard algorithms employ a ***momentum***term in order to speed convergence while avoiding instability. The momentum term is added to Equation 1, and is equal to the product of fraction β (0 <= β <= 1) and the change in weight that occurred in the previous iteration.

The momentum term helps convergence by avoiding the local minimum. This is illustrated by the analogy of a ball rolling down a hill as shown. When ball has more momentum, it can easily avoid getting stuck in the smaller valleys. Setting the momentum term too high can create the risk of overshooting the true minimum. This can cause the system to become unstable and should be avoided.

The example shown before presents a simplified view of a multi-connection neural network. In reality, error is a multi-variable function of weights as shown below.



However, principles are still the same. The slope which represents the simple derivative of the function changes to partial derivative evaluated at the initial position.

-------- (2)

To summarize, in each iteration (or epoch), we move a step closer to error minimum by adjusting our connection weight using the formula above. Our hope is that using this iterative approach, we will eventually get closer to our minimum and arrive at the optimal values of our weights. We can now call our network trained and use it for forecasting purposes.

# The Back-propagation Algorithm

In this example, we present details of back propagation algorithm. The back-propagation algorithm uses gradient descent to adjust connection weights by tracing backwards in a neural network. Some concepts will involve application of elementary calculus. A refresher on calculus is presented in the appendix. Readers not interested in the mathematical details may skip this section.

The figure below presents a small network with 2 nodes in the input layer, 2 nodes in the hidden layer and one node in the output layer. We will denote the connection weights between nodes i and j by symbol Wij.

**Output5**

**5**

**4**

**3**

**2**

**1**

If transfer function is denoted by f(x) and is differentiable, the final output can be summarized as follows:

-------- (3)

Let the target value (also known as actual value) of the output be T. Then error can be measured as follows:

. By substituting value from equation (3), we have

-------- (4)

We need to calculate how we need to change the weights to minimize error. From gradient descent method we know that we need partial derivative of error function with respect to a connection weight. Using derivative chain rule, we can easily compute this. For example, the partial derivative for is:

If transfer function is hyperbolic tangent (tanh), then

Let

Using equation (2), we can calculate the new value of weight to minimize the error.

------ (5)

Let us perform a similar calculation for weight W13.

We know that

Let us set . We will use old value of and assume it to be a constant.

This value can be substituted in the original equation to get . Similar to calculations for W35 , we have.

----- (6)

To summarize, we can use derivative chain rule to calculate the partial derivative of error function with respect to a connection weight. This partial derivative can then be used to compute adjustments to the current weight using equation (2).

# Appendix – Primer on Calculus

The field of calculus deals with functions. A function shows how things are related. For example, electric demand is related to temperature. We can capture this relationship using the following notation.

By using our knowledge of temperature, we can get demand. Now the question is when temperature changes, by how much demand changes. This is the basis of the idea of the derivative. The derivative of a function measures the relative change in output when input changes by an infinitesimal amount.

h->0

We can see that derivate is very similar to slope of line. The numerator represents change in y value when x is changed by h. The denominator is the change in x. When change in x is infinitesimal i.e. h is very close to zero, the secant line becomes a tangent line as shown. In other words, derivate evaluated at point X is the slope of the tangent line at X.

X+h

X

h

## Power Rule

Let y = c\* xn, where c is a constant. Then power rule states that

If y is a constant function i.e. y = c, then derivative is zero.

## Derivative Chain Rule

Derivative chain rule helps us to find derivative when a function is a composition of two or more functions. For example, y = (3x+1)2 is a composition of linear and quadratic function. The function g(x) = 3x+1 is a linear function and f(x) = x2 is a quadratic function. The function y = f(g(x)) is a composite function. So, how do we find derivative of such functions? Derivative chain rule can be used to find the derivatives of such functions. Let us substitute u = 3x+1 so that y = f(u). Mathematically, we can state the chain rule as follows:

We know that from elementary calculus rules that f’(u) = 2u and g’(x) = 3. So, derivative of this composite function is going to be 2\*(3x+1)\*3.

# References

1. Neural Network Basics: [http://www.webpages.ttu.edu/dleverin/neural\_network/neural\_networks.html](http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html%20)
2. Cost Estimation Predictive Modeling: Regression versus Neural Network <http://www.eng.auburn.edu/~aesmith/publications/journal/tony.pdf>
3. <http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-s06/www/nn.pdf>
4. [http://en.wikibooks.org/wiki/Artificial\_Neural\_Networks/Neural\_Network\_Basics#Momentum](http://en.wikibooks.org/wiki/Artificial_Neural_Networks/Neural_Network_Basics%23Momentum)