Artificial Neural Networks Using Microsoft Excel for Windows 95†

Dr. Bruce E. Segee, Michael D. Amos
University of Maine

Abstract

Artificial Neural Networks have been researched now for decades. The standard method of implementing Artificial Neural Networks is by using C++, Fortran, Pascal, or other high level computer language to develop a system able to take a set of inputs and generalize to produce a satisfactory output. The property of generalization allows Artificial Neural Networks to respond reasonably to novel inputs. With the high level of computer expertise needed to program such a network, the power of Artificial Neural Networks has been restricted to those with the ability to spend weeks in developing code for such an application. The purpose of this project is to research the feasibility of constructing a Neural Network in Excel™ for Windows 95™. This will bring the power of Neural Networks to the average computer user with a working knowledge of Excel for Windows 95.

1. Introduction

This section explains the general organization of this paper and discusses a brief outline of the main objectives of this project.

Section 2 contains a general background of Artificial Neural Networks. Explanations will be given to explain how Artificial Neural Networks "learn" and generalize for a given set of data and how that helps in solving complicated systems measurement and control problems. Section 2 also discusses the significant understanding and knowledge of programming and Neural Network fundamentals needed to use the traditional C++ Neural Network.

Section 3 discusses the methods used to program the Neural Network using VBA (Visual Basic for Applications) in Excel. The programming knowledge required to use this interface is minimal and the only computer experience needed is a good working under-standing of Excel for Windows 95.

Section 4 provides conclusions and information that demonstrate that this Neural Network in Excel for Windows 95 performs as well, if not better, than its standard C++ counterpart.

2. Introduction to Artificial Neural Networks

2.1 Why Neural Networks?

One may ask, "Why do we need Neural Networks? Why can't we derive solutions analytically?" The explanation is simple. When using a large data set, the computation time is astronomical. To give an idea of how much time is involved, please consider the following analogy: Given a small data set with a two dimensional input, one has the equivalent of solving two equations with two unknowns. Any person with a good knowledge of algebra could solve such a problem in a few minutes. A computer could solve the same problem in a few milliseconds.

† This research was supported in part by National Science Foundation grant EEC-9531378

By comparison, a data set with 100 items, each with an eight dimensional input, is comparable to solving 100 equations with eight unknowns. Solving an equation of this magnitude, is a virtual impossibility to do by hand and difficult and time consuming even with a computer.

Even if one could wait for the computer to return a complete solution, this solution may not be the best generalizing solution. In other words, the goal is to create a system that can tolerate a substantial amount of error(noise) in the input values.

When properly set up, Neural Networks have the ability to generalize, that is, find a solution for a set of inputs that gives an answer with low error values even for inputs not in the training set. Generalization is a very important topic, since the network can still provide reasonable output even if the input contains error and is new to the system. This is how Neural Networks appear to "think", by developing a system that gives the best answer for a given set of inputs and adapting the method in which the decision is made. [Wasserman, 1989]

With this brief explanation of the necessity of Neural Networks, the focus of this paper will now change to show how Neural Networks are designed, and how the architecture is derived from the organization of neurons in living creatures.

2.2 The Brain

The Brain would be considered a Natural Neural Network, comprised of hundreds of billions of neurons, each interconnected with thousands of others. Scientists believe that this interconnection between neurons is what enables us to think, reason, and rationalize solutions to problems that cannot be solved analytically. No one knows how brains actually function and how humans have the ability to reason and learn, but one can only design models that are vaguely similar. [Wasserman, 1989]

2.3 Artificial Neural Networks (ANN)

Artificial Neural Networks are constructed in a way that somewhat resembles the neurons in a biological entity. A program is written that has a number of elements, or artificial neurons (also called nodes), that exhibit certain behaviors based on parameters set by the programmer. By changing the parameters, the programmer can change the behavior of the neuron, effectively changing the way that the ANN interprets data and how it "thinks".
There are several different architectures of Artificial Neural Networks based on various aspects of neuron behavior. Because of the nature of this project, the architecture that was developed and studied was Radial Basis Function Artificial Neural Network, or RBF.

The RBF network uses a function that creates an output for each neuron that is a function of the Euclidean distance of the data point from the node center. This output is called the activation value. The function used is a Gaussian described using the following formula:

$$\frac{e^{-\sum_{i=1}^{n}(x_i - c_i)^2}}{w^2}$$

where x_i is the coordinate of the input in dimension i, c_i is the coordinate of the node center in dimension i, w is the width, and n is the number of input dimensions. This function creates a shape (see Figure 1) that looks like a bell curve. Each node in the RBF network has several parameters that can be adjusted to change the way the network behaves. [Goel, 1995]

The first parameter is the center of the node. The center is the point of symmetry in the input space. In a one dimensional input space, the center is a point somewhere on a number line. Each input value falls on this number line somewhere and depending on the distance from the node will cause that node to have an activation value. As you get further from the center of a node, the activation value gets smaller.

The next parameter of the node is its "width". A larger width has an effect as shown in Figure1. The width parameter is very important for learning and generalization. If the width is too large for the nodes in a network, then significant overlap of the Gaussians will occur and the network may not train to get the lowest possible RMS (root mean square) Error. If the width is too narrow, then the Gaussians do not overlap enough, causing the training network to converge

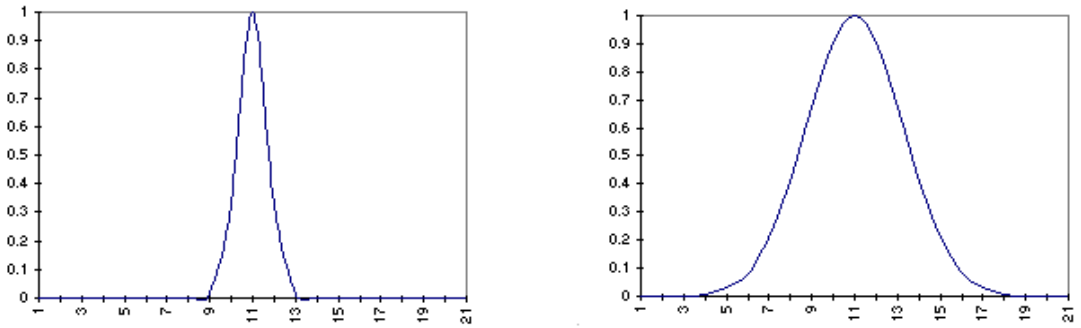

Figure 1. The graph on the left shows a Gaussian with a narrow width while the graph on the right has a much wider width.

quickly, but not getting a good generalization of the entire testing data set. Although it may seem contradictory, zero training error is not the goal of an ANN. Good generalization over the entire data set and any new data that may be entered is the ultimate goal of any Neural Network.

The final parameter of each node, is the weight. The activation value of each node is multiplied by the weight to get the node output for that data point. It is then summed with the weighted outputs of the other nodes to produce the network output for the given data point.

As the network is trained, there is an iterative process where analysis is done on the error of each of the training points and each node. After analysis, it is known which "direction" to move all of the weights in the network to have the overall error decrease by a small amount. After new weights are calculated, the entire network is then recalculated using the new weights. This process continues for hundreds or thousands of iterations until the overall RMS error of the entire system is satisfactory.

When the network is trained, a portion of the data set is used to evaluate error and calculate new weights. The remaining data is used as testing data, that runs separately and produces an RMS error as well. The purpose of the testing network is to see how well the network is generalizing. As mentioned before, the purpose of ANNs is not to find a solution with zero error, but find the solution with the best generalization characteristics. Because of this, the network must be monitored to know when the best generalized solution for the given data set is encountered.

As seen by this brief description of how ANNs operate, there is a high opportunity for error when programming a large project such as a Neural Network. A significant amount of time is also consumed in the writing of code for doing the calculations as well as memory management and speed considerations. After programming the entire network, the programmer is still left with a program that is probably text based, difficult to modify, and cumbersome for the average computer user with no programming experience to use.

The next section will describe how this project works as well as the advantages and disadvantages of using such a system.

3. Neural Networks Using Excel

3.1 Objectives

There were several objectives in mind when designing the interface for this software package. The first and most important was to make it easy to use. The goal was to make it easy enough so that someone with little or no programming experience could sit down, read the instructions, and within 10 minutes have a working Neural Network. The intended audience was the many computer users such as teachers, business people, and even students that could benefit from using Neural Networks.

Many features of Excel were used to accomplish the easy to use interface. The main feature used was the Excel programming language called VBA, or Visual Basic for Applications. This programming language, a subset of Visual Basic, can easily interface with spreadsheets and perform any task that a user could do by using the menu bars and the mouse.

3.2 Design

A spreadsheet was designed that would hold all of the information that was necessary for the network to operate. Code was written that would take an input file in any standard Excel format, and use the data there to construct an ANN automatically. Following the expected format, the input file must have the actual data in the first columns, while the final column must always contain the desired output for the previous columns' input data. The data has a limit of up to 10 input dimensions and one output dimension. The user simply highlights the desired area for the Neural Network setup with the mouse or the keyboard and then runs the VBA code, constructing the network in the format shown in Figure 2.

The new Network is constructed with the centers for each node placed at each of the Training Data points. For example, if the first training point has inputs of 2.34, 3.45, 4.56, then the center coordinates of the first node will have coordinates of 2.34, 3.45, 4.56. With this method of placing node centers at each training point, it is extremely important to have a good random sample of the total data set. Otherwise, the network may contain an uneven coverage of the input space with the nodes. This can lead to poor generalization and high RMS Error.
The code then places the appropriate formulas in each of the areas of the spreadsheet. The following list indicates the function of the sections in Figure 2:

- The Training Data section contains the training data inputs and outputs
- The Node section contains the coordinates of the Node in x-dimensional space, the width of the Gaussian, and the weight that has been calculated for that Gaussian. The nodes are separated by column (column 1 contains node 1, column 2 contains node 2, etc.) with the weight being the number at the bottom of the column, and the width being the one just above that for all nodes.

- The Node Activation section contains the activation values for all the nodes at all of the training sets.
- The Network Output section contains the output value of the network for all training sets.
- The Absolute Error section contains the Desired-Actual value of the network output compared to the desired training value for all training sets.
- The Cumulative Error section contains the cumulative error of all training sets for each node.
- The Calculated Weight section shows the new value for the weights of all training sets as calculated from the Cumulative Error
  and the Learning Rate.
- The RMS Error section shows the RMS error value of the entire network.

As seen from the layout in Figure 2, all of the numbers associated with the network are easy to find and accessible in an easily read format.

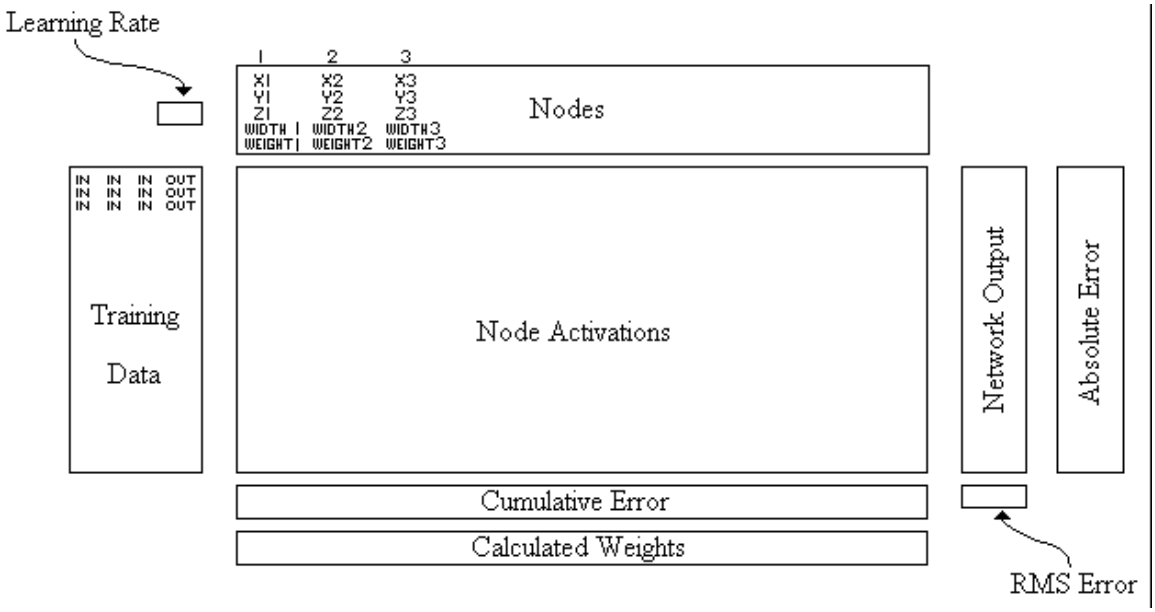

Figure 2

3.3 Training

Network training is accomplished by setting up an iterative or circular reference in the spreadsheet. This was done because the VBA code is too slow and time consuming to do all of the intensive calculations needed to train the network. The circular reference is made because the network output is calculated with the node weights. The cumulative error uses the network output. The new weights are calculated with the cumulative error. The new weights change the network output and the process repeats.

Excel conveniently can handle iterative or circular references. By setting up the spreadsheet this way, the power of Excel's very fast mathematics engine does all of the calculations necessary. Using the Tools|Options|Calculations menu option, the user can change the number of iterations to be done per calculation from 1 to over 20,000. The user may also specify the maximum change that they would like the spreadsheet to calculate before stopping the calculation process and returning to an idle state

Once the training worksheet is completely set up, the user may now construct a testing network worksheet. The testing data points must be in the same format that the training data. The user simply highlights the desired testing data set and starts the program with a simple mouse click. A testing network worksheet is then constructed using the dimensions and parameters of the training worksheet.
With both worksheets set up, it is possible to see how the network behaves and how well it generalizes to get results with the testing data.

3.4 Graphical Analysis of Nodes

Since Excel has an excellent graphing capability, this can be used to the user's advantage. It is often useful to graph the entire node activation area of the spreadsheet as a 3D-line graph. With this graph rotated in the proper orientation, it is easy to see when there are gaps in the input space, overlapping nodes, or all nodes have evenly spaced centers. See Figures 3 and 4. Often, when using data points that consistently have large absolute errors, the problem is noticeable in the graph. For example, if a graphical analysis is performed and there are training points that aren't affected by any of the nodes in the network, then you may have a large gap in the graph. If the training point isn't activating any nodes, then chances are you will never decrease the error in that region. As we can see from Figure 3, data point number twelve isn't activating any of the nodes in the network. This can lead to high error.

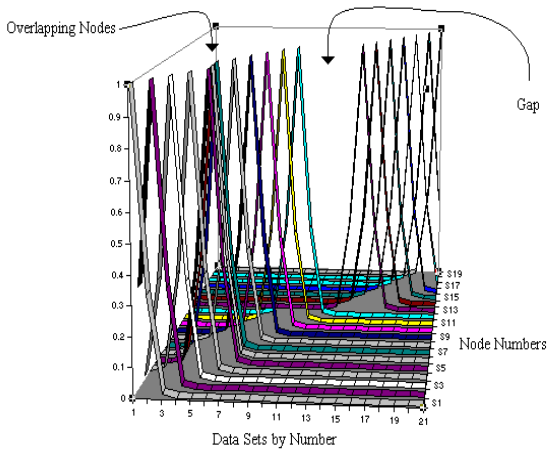

Figure 3

One the other hand, it usually isn't good to have many nodes which cover the same area in the input space. Figure 3 demonstrates that data points 4, 5, and 6 have several nodes that are affecting each data point. If these points have different desired output values, high absolute errors in this area can develop. This is because each data point is trying to affect the weight of the node to reduce its error. If data set 5 has a desired output of 5, and data set 6 has a desired output of 300, then no set of weights can satisfy them both. This results in high absolute and RMS errors for the system.

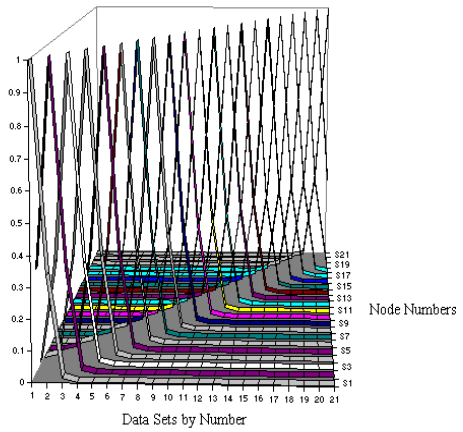

Figure 4

Figure 4 shows a system having all of the input space covered evenly. Each of the nodes overlaps only slightly in each direction. This setup could have the makings of a low RMS error network. Of course, every data set requires the network to be customized to the user's needs, so only experience and trial and error can develop the intuition required to tune the network to its optimal state.

3.5 Modifying the Network

Once the graphical analysis is complete, it may be necessary for the user to change some of the parameters of the network. This is easily done since the information is laid out in an orderly and highly intuitive manner. The one parameter that often produces the greatest change in the network RMS error is the width.

Changing the width changes how the network responds to data points that are at a distance from the center of the node. A wider width will sometimes give a better generalization than a network with very narrow widths. A very narrow width tends to give very low training error, but at the same time responds with bad generalization and high testing error.

Another method of modifying the network is adding or subtracting nodes. For example, previously in Figure 3, there was a situation where there was a gap in the graph. In this example, it may help to lower error if a node was placed at the point where the error was high. On the other hand, removing overlapping nodes may also produce favorable results.

In both cases, Excel provides a means to perform such operations. Simply inserting and deleting rows is fully supported and Excel takes care of modifying the formulas to include the new columns or remove the deleted columns. This is a good example of the power of using Excel to help manage the large amount of numerical information. [McFedries, 1995]

Other methods of trying to improve network performance include adding or deleting training data points. Again, Excel supports both of these operations by adjusting all formulas to make room for the new data. The user has to do nothing except for the normal Excel functions, i.e., simply insert a row in the place where the data is desired, or delete the row containing the data set to be deleted.

4. Conclusions

Several conclusions can be drawn after constructing a Neural Network in the previously described way. The most outstanding feature is the ease with which a network can be constructed. It is possible for someone with absolutely no programming knowledge at all to sit down and construct a working Neural Network model within just a few minutes. Although this was the major objective of this project, there were several other advantages that revealed themselves during the project.

One of the most surprising facts about seeing this project operate, is the intuitive way that the node activations and node characteristics can be viewed using Excel. It is possible to look at a graph and see how the different nodes react with the data points, as well as noticing how some nodes may not even be activated by any of the data points. By using the graphing utility in Excel, the user is able to construct a picture of something that was previously just an array of numbers in the memory of a computer. The user can then use this picture to help tune the network parameters to give optimal performance.

Another very surprising outcome of this project lies in the speed with which Excel can do math calculations. This project performed at close to the same speed as the C++ Neural Networks. Speed comparisons were done comparing this project against C++ Neural Networks written by expert programmers. This can be attributed to the highly optimized Excel "number crunching" engine. This is very important since most Neural Networks need to train for thousands of iterations before developing acceptable generalizing characteristics.

In conclusion, this project proves without a doubt that constructing Neural Networks using Excel for Windows 95 is certainly feasible, desirable, and advantageous. It not only provides a mechanism for non-technical people to use Neural Networks, but also gives technical people a greater insight into how the network behaves.

5. References

Goel, 1995] Goel Shalabh (1995), Calibration of Solid State Gas Sensors Using Radial Basis Function Artificial Neural Networks.

[Lippman, 1988] Lippman R.P. (1988), An Introduction to Computing with Neural Nets, (reprinted from IEEE ASSAP, pp. 4-22, April 1987), Artificial Neural Networks: Theoretical Concepts, V. Vemuri (Ed.), IEEE Computer Society Press, pp. 36-54, Washington, D.C.

[McFedries, 1995] McFedries Paul (1995), Excel for Windows 95: Unleashed, SAMS Publishing, Indianapolis, IN.

[Wasserman, 1989] Wasserman Philip D., 1989, Neural Computing: Theory and Practice, Van Nostrad Reinhold publishing company, New York, NY.

Biography

Dr. BRUCE E. SEGEE received a Ph.D. in Engineering from the University of New Hampshire in 1992. He has been an assistant professor of electrical and computer engineering at the University of Maine since that time. At the University of Maine he heads the Instrumentation Research Laboratory, an organization dedicated to research and teaching involving instrumentation and automation. Work in the lab includes the use of PC's, PLC's, and embedded controllers for instrumentation, automation, and networking. Work also includes the use of fuzzy logic and artificial neural networks.

MICHAEL D. AMOS received an Associate of Applied Science Degree in Electromechanical Technology from Central Maine Technical College in 1991. He is currently pursuing a Bachelor of Science Degree in Electrical and Computer Engineering at the University of Maine.