Training a Value vs. Growth Stock Classification Neural Network

By Alex Kaufman,
Parallax Financial Research
Redmond WA
akaufman@pfr.com

Project Goal

The goal of this project is to build a neural network that will be able to look at a company's fundamental data and determine whether it should be classified as a value stock or a growth stock, or maybe somewhere in between. Investment professionals as a way of characterizing risk/reward potential routinely classify stocks as value or growth. Value stocks are thought to be safer, growing slowly, but having solid intrinsic value and often paying dividends to compensate investors for the slower growth. Growth stocks often do not pay dividends, may have little intrinsic value, but have been growing very fast. Training data was taken from the Russell 3000 value and growth indices and fundamental data from Zacks. The neural net needed to be created and trained with BioComp's NeuroGenetic Optimizer software. It is useful for Parallax to have a consistent method of classifying stocks as value or growth because it enables us to build asset allocation predictors, build our own indices, evaluate client portfolios, and design stock screening tools for the investment industry. The Russell method includes I/B/E/S earnings estimates in their formula, which Parallax has found to be misleading. We expect that by leaving expectations out of our training, that the classification system produced may even outperform the Russell technique.

Neural Network Background

The NeuroGenetic Optimizer (NGO) facilitates the creation of artificial neural networks by employing genetic algorithms in order to modify neural net structure. The term neural network could refer to the real life networking of neurons in various biological organisms, but it is used here (and in most cases) to reference the concept of an artificial neural network, which is a model created inside the computer that mimics the most fundamental functions in a biological neural network.

In real world neural networks, it is believed that "learning" takes place when the strength of an axon connection is increased by virtue of its usage. For instance, when a small child is learning how to define a tree, he could be shown a pine tree, and the connections supporting a tree as a tall, green, spiky, thing with brown bark are reinforced and strengthened. As the concept of "tree" evolves, green will be given more weight, and spiky will be given less weight, because, while most trees he is shown are still green, they do not all have spikes (i.e. – a maple tree). This is a real-world example of a neural

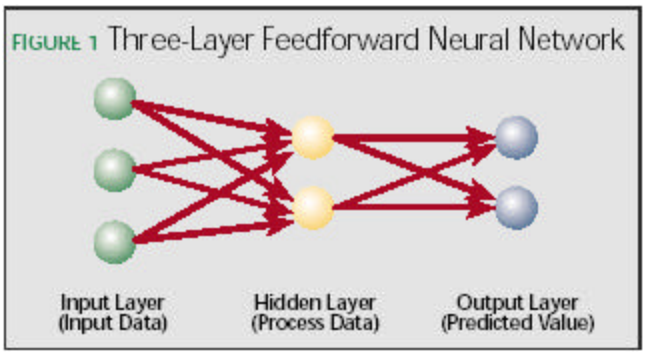
network, in order for the boy to accurately predict what is and is not considered a "tree"; he must have seen enough *examples* of *different kinds of trees*. This way, he can determine the connection between the various characteristics of the tree (independent variables) and whether or not it is a tree (classification output).

This real-world example fits nicely with the project, because both were "classification" neural networks. For the young boy's learning, he had to classify objects as 'tree' or 'not tree' by looking at the variables that are important. He knew which variables were important by experience and training with many different sorts of trees. If he had only seen pine trees all his life, and classified them as 'tree', then he would not be able to accept that a palm tree is also a 'tree'. So with this project, the computer was simply asked to learn what a growth stock is and what a value stock is, based upon company fundamentals.

The Russell 3000 value and growth indices, which are the industry standard, provide the value vs. growth information for training our network. Clearly, it is easier and more cost-effective to classify the stocks by computer than by hand.

The NeuroGenetic Optimizer allows the user to train a number of different application types: function approximation, diagnosis, clustering, time series prediction, and classification. Classification neural networks look at examples with multiple outputs or categories, and correlate that to the input data provided. Once a classification net has been properly trained, it will be able to look at a set of data and determine what category it falls into. Neural networks are powerful tools, but nobody has placed biological neurons inside the computer, and they are certainly not comparable to human consciousness, so how exactly is this machine learning achieved?

In order to understand neural networks it is important to understand the 'hidden layer' of nodes in between the input and output layers. Input nodes could be compared to the neurons in the eyes, and output nodes could be the neurons leading to the vocal chords where an answer or result is spoken aloud. The hidden layer simulates the set of neurons in between, where both learning and problem solving occur. All the input nodes send information to all the hidden nodes, and all the hidden nodes send data to all the output nodes. Here is a simplified diagram of just such a basic three-layer neural network:



Picture provided from Louis Francis' article "The Basics of Neural Networks Demystified" in *Contingencies* magazine, available at <http://www.contingencies.org/novdec01/workshop.pdf>

In the diagram, the term "feedforward" simply references the direction of data flow from input layer to hidden layer to output layer. A quote from the magazine article from which

this diagram was taken goes a long way to summarize the "learning" that goes on in the hidden layer:

> Neural networks "learn" by adjusting the strength of the signal coming from nodes in the previous layer connecting to it. As the neural network better learns how to predict the target value from the input pattern, each of the connections between the input neurons and the hidden or intermediate neurons and between the intermediate neurons and the output neurons increases or decreases in strength… A function called a threshold or activation function modifies the signal coming into the hidden layer nodes… Currently, activation functions are typically sigmoid in shape and can take on any value between 0 and 1 or between -1 and 1, depending on the particular function chosen. The modified signal is then output to the output layer nodes, which also apply activation functions. Thus, the information about the pattern being learned is encoded in the signals carried to and from the nodes. These signals map a relationship between the input nodes (the data) and the output nodes (the dependent variable(s)). (3)

Just as with our analogy of the young boy and the trees, the neural network takes each independent variable (input) and weights them according to their importance in discerning the dependent variable(s) (output). The activation functions that are triggered in each hidden node of the neural network record the signal strength and thereby encode the patterns correlating input and output data. In summary, this program "evolves" to a point where it can continually solve similar problems over and over again. These tools are extraordinarily useful, because they can "learn" about how to solve a problem that may be highly non-linear, such as stock market prediction.

A result of this learning is that the network *is* the answer to the problem. If one were to distill the neural network down to a corresponding equation it probably would be pages in length and be too confusing to comprehend. The weights and transfer functions of the hidden layer can be viewed by the user, since they are written into the file, but only when each of the weights and interconnections are taken together in their entirety can the pattern be discerned by the neural network. Understanding how and why the chosen variables, the activation functions, and the connection weights interact to produce the correct answer is next to impossible for a human being to discern by simply looking at the net. Neural networks are often referred to as "black box" modeling, because the correlation that's determined is not easily discernible, but the neural network produced will allow the user to look at "response curves." These response curves show a graphed correlation between data variables used for input, and allow the user to judge whether or not the neural network has learned about the independent input variables adequately. There will be an elaboration on response curves in the results section.

The last bit of information that is important to know concerns the transfer functions. The NGO utilizes three different types of transfer functions, which are the functions in the hidden layer that specify the relationship between inputs and outputs. The different types of functions used by the NGO are as follows:

1. "Lo" – Logistic Sigmoid
2. "T" – Hyperbolic Tangent
3. "Li" – Linear

These functions adjust during training so that the neural net as a whole can simulate the learning process. Depending on the problem being solved, different combinations of these transfer functions will be employed, and the sum of the dynamics of these functions will produce the trained network.

Data Preparation using Excel 2002

The project was conducted with the Parallax Financial Research training data, which contains fundamental information about hundreds of stocks since 1991. This data is divided by sector into 16 separate .DAT files. The files were saved in the "DAT files" folder in the Pfr directory. Since the Russell 3000 data was only available for June 30, 2001, the stock data for that particular date was copied out of the .DAT files into 16 separate Excel spreadsheets.

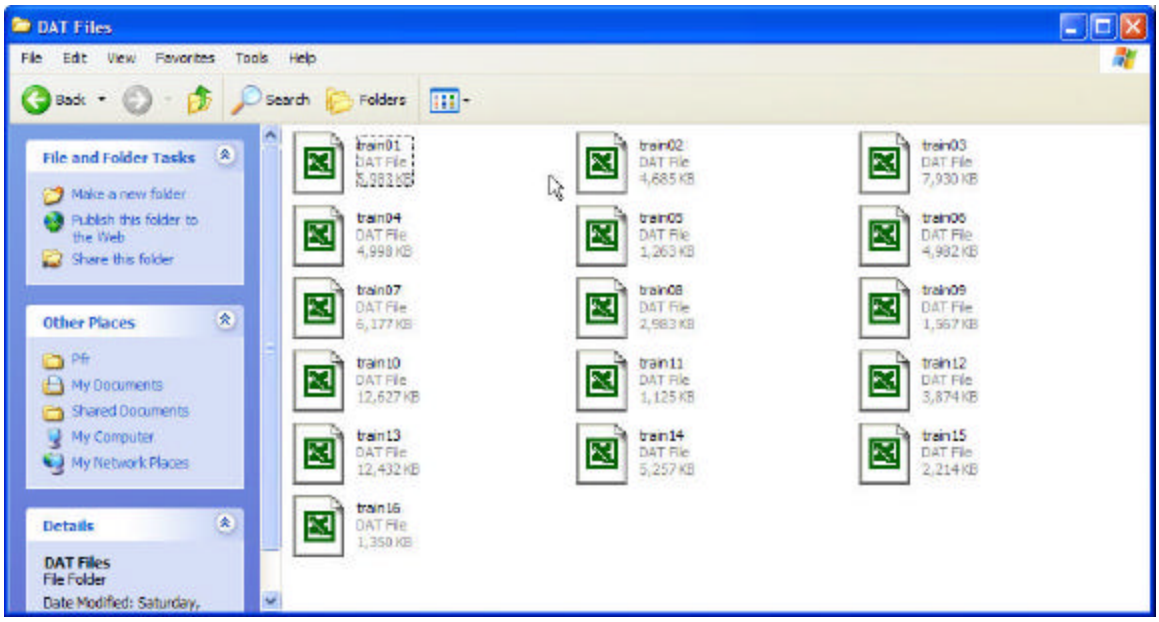


Diagram 1 - Screen capture of original 16 .DAT training files

The excel spreadsheets were saved as "train01vg," "train02vg," etc… up to "train16vg" in the "June 30, 2001 Train files" folder, as seen below.

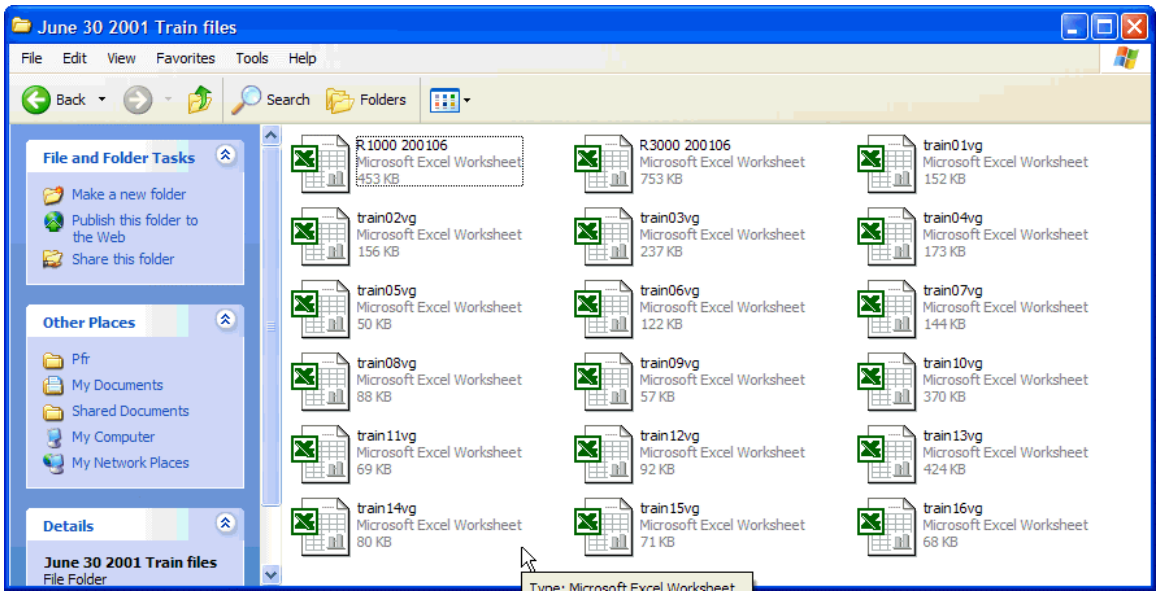


Diagram 2 - Screen capture of 16 Excel training files with training data for June 30, 2001 only.

The above screen capture shows that the Russell 1000 and Russell 3000 data were also kept in this folder for the project. By having the Russell data in the same folder, it was easier to combine the training data (inputs) with the value vs. growth classifications provided by Russell (outputs). When merged together, these files will provide the NGO with the training data necessary for creating an appropriate classification neural network. Neural networks were actually created for both the Russell 1000 and Russell 3000 during this project, which is why there are R1000 and R3000 files in the above folder, but the Russell 3000 is the only file that this report will cover, since it contains everything the R1000 has and more.

In order to merge the input and output data in the "June 30 Training files" folder, the R3000 file and training data files had to open simultaneously in Excel. Using the MATCH and OFFSET functions, each of the training data files were able to grab the appropriate values for "value" and "growth" (which always add to 1) from the R3000 file. The value number varies from 0 to 1 as an indication of how strongly Russell feels about a stock belonging to the value class. If value =1, then it is a value stock. Likewise, the growth class is simply 1-the value number, so if value=0, then it is a growth stock. The following screen capture shows the value and growth columns of data that need to be appended to the training files.

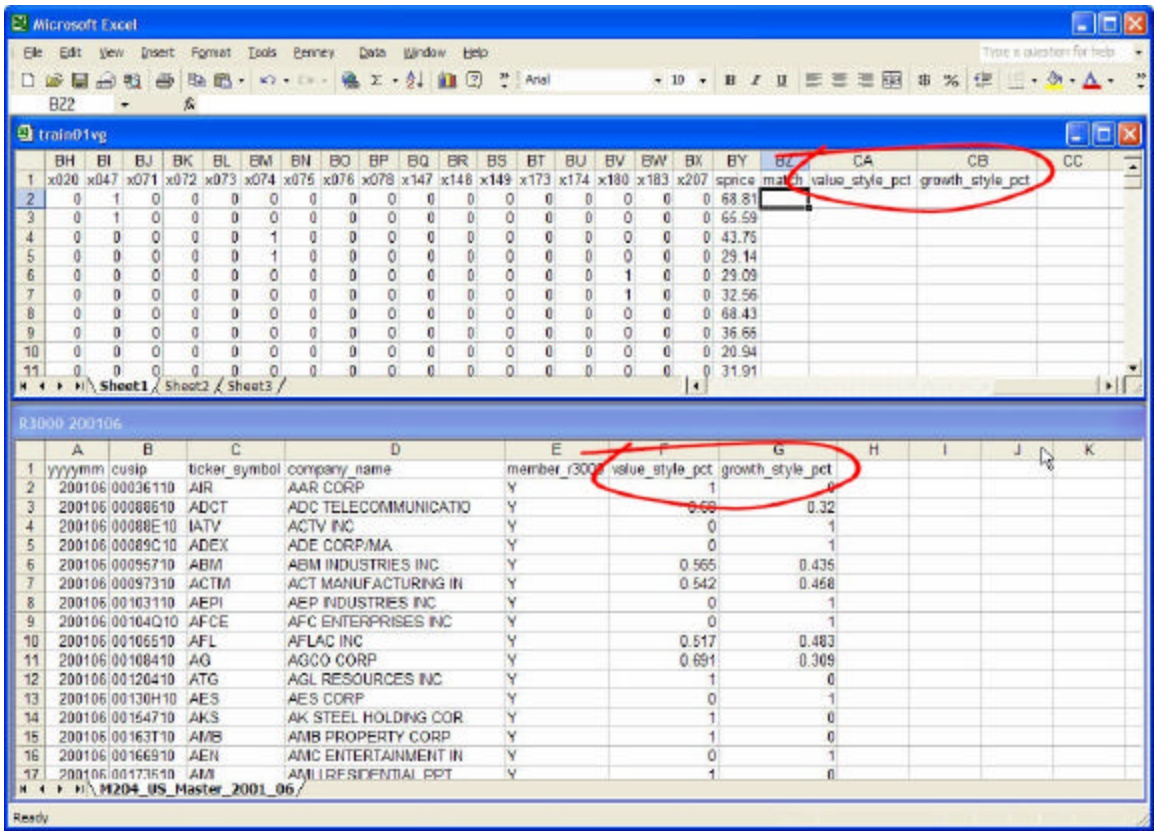
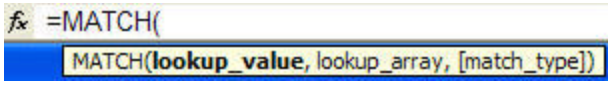
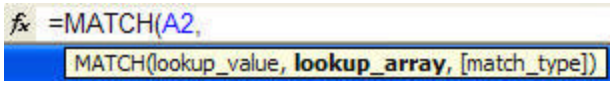


Diagram 3 – This screen capture shows the two files opened simultaneously, and it has red circles around the headers for the value and growth data columns that need to be transferred for the appropriate stock ticker.

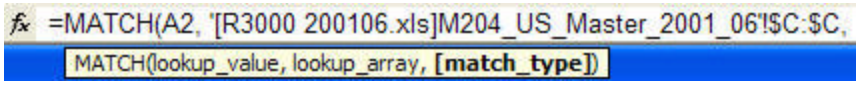
The MATCH and OFFSET functions are needed for this transfer (as opposed to cut and paste) because the R3000 file has a list of 3000 tickers listed alphabetically, while each of the training files has ticker data broken up by sector (in each file) and within each file the tickers for that sector are arranged alphabetically, so they are *not in the same order*, and won't necessarily correspond entirely anyway. In Excel, the MATCH function can be used to return the *location number* of a particular number or character string. The following equation needs to be written out for the MATCH function to return the location number of the desired ticker.



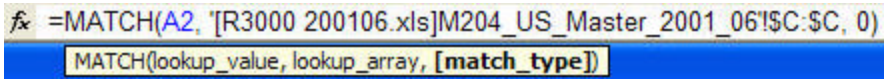
The first value needed is the "lookup value", which specifies what number or character string the MATCH function will be searching for. In this case, the ticker character string is required, and since the first ticker symbol of the training file starts at cell A2, that is what is entered for the lookup value.



The next value needed is the "lookup array." This is a lot simpler than it sounds. The lookup array is simply the range of cells in which the MATCH function is going to look for your lookup value. The function needs to find the ticker symbol from the R3000 file and return the number of its location. Since this array is in a different file, it is important that the two files are both open at the same time and arranged horizontally, as they are in Diagram 3. To fill in the correct value for the lookup array, the user must simply click on the desired column of cells. In this case, it is the C column in the R3000 file, because that is the column with ticker symbol listings.



The seemingly long file name before the C column specification is to show that it is the C column in "R3000 200106.xls" worksheet on the sheet entitled, "M204_US_Master_2001_06." It is important to remember that the MATCH function will return an error message if the R3000 file is not open. The final value necessary is the match type. Different match types correspond with different numerical values. For an exact match to the lookup value a '0' is used, so that is what is placed here to complete the MATCH function.



Once this equation has been entered, it can be extended to all of the cells in the designated column in the training file. In training file 1, that column is BZ, here is the result of extending the MATCH function down the column:

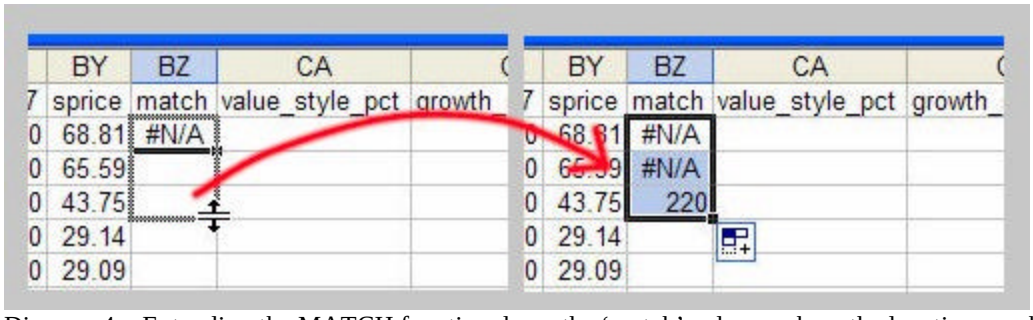
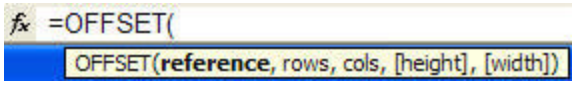


Diagram 4 – Extending the MATCH function down the 'match' column where the location number of the ticker will be returned if it can be found. The value '#N/A' means the ticker was not found in the R3000 listing.

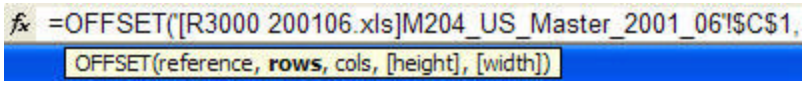
The diagram shows the MATCH function only partially extended down the 'match' column, but in order for all ticker symbols to be found in the R3000 file it must be extended the rest of the way down the column. Some of the cells shown above in Diagram 4 display '#N/A', and this indicates that the ticker symbol was not found. The third cell in the example diagram returned a value of 220. This indicated that the desired

ticker symbol for that row in the training file will be found at cell C220 on the R3000 file.

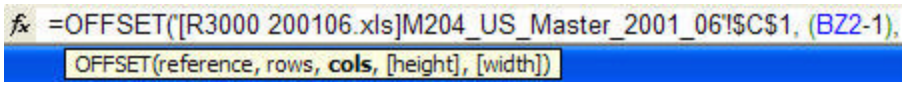
The MATCH function would not be of much use to the project if it was not paired with the OFFSET function, which will now deliver the value and growth numbers into the two columns neighboring the 'match' column. The OFFSET function will take the location value provided by the MATCH function, and use it to find the value and growth numbers from the R3000 file. OFFSET will be used in both the 'value_style_pct' and 'growth_style_pct' (refer to spreadsheet layout in Diagram 3) columns in the training files as follows:



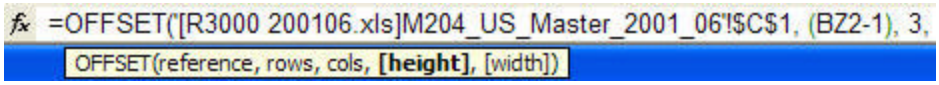
The 'reference' value tells the OFFSET function where it needs to start from as it searches for a specific value. The OFFSET function retrieves a value much as you would retrieve a phone number from a phone book. In retrieving the phone number, a person uses the last name in order to find the number. The OFFSET function uses a reference value, which is similar to a 'last name', and then it moves over or up a specified number of cells in the spreadsheet and retrieves the value it finds there.



In this case the reference point will be the first cell in the C column in the R3000 file. Once it finds this spot, the function can use the values returned in the 'match' column to determine how far down to go in order to retrieve the desired value. Now the OFFSET function needs to know how many rows down it needs to move in order to find the desired value. This value is provided by the MATCH function in the 'match' column of the training file. OFFSET needs to know how many cells to move down from the original reference point at the top of the C column in the R3000 file, but the MATCH function returned the *row number* of the desired tickers. In order for the OFFSET function to find the correct ticker, the value returned by the MATCH function needs to have 1 subtracted from it. The next screen capture shows how the 'rows' value is satisfied by taking the 'match' column value and subtracting 1.



The 'cols' value specifies the number of columns the OFFSET function needs to travel in order to find the desired value. In order to retrieve the 'value' number in the R3000 file the function needs to move 3 columns over, so 3 is the fixed value here. When programming the OFFSET function to retrieve the 'growth' number it needs to move over one extra column, so 4 is the fixed value. Here the OFFSET function for retrieving the 'value' number is shown.



The height and width values simply specify the size of the array that needs to be retrieved from the specified location. Since just one cell is being retrieved, both of these values are 1 as shown.

The final OFFSET function looks like this, and it needs to be applied to both the 'value_style_pct' and 'growth_style_pct' columns in the training file, the only difference being that value column should have a 'cols' value of 3 while the growth column has a 'cols' value of 4. When applying this function by extending it to other cells, it should look like this:

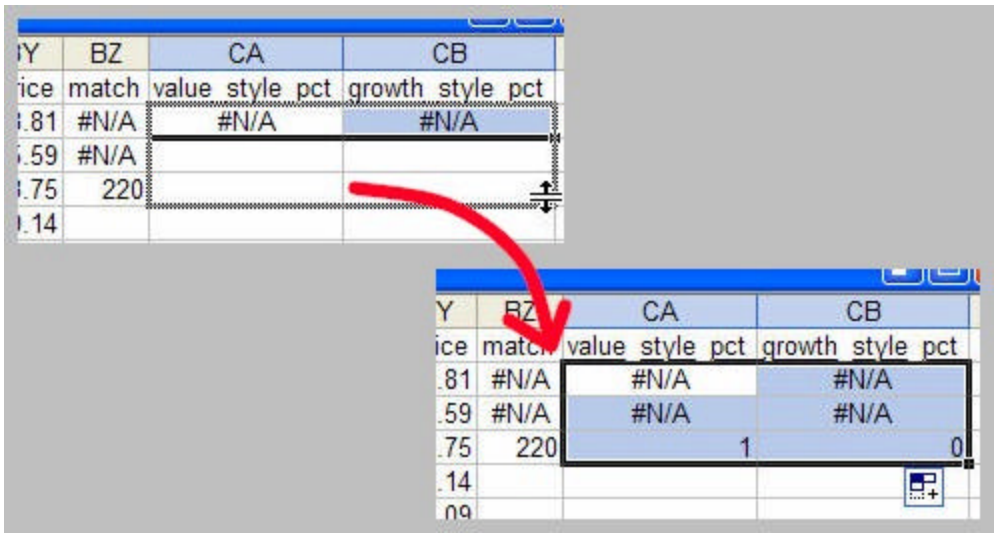


Diagram 5 – The OFFSET functions that have been programmed for the two columns are dragged down and extended to other cells. Once the function is applied to a row that has a 'match' value, the value and growth numbers that correspond to that ticker are returned, and as is required, the values add up to 1.

This process now needs to be applied to all 16 of the training files so that the merge between training data and R3000 value vs. growth data can be completed. Before all of the data is ready to be merged into one training file for the neural network though, it needs to have all the ticker data that doesn't correspond to an R3000 value vs. growth set (i.e. – all of those rows that returned #N/A values) extracted from it. This is accomplished by going through each of the 16 files and applying Auto filter to the 'match' column as shown below.

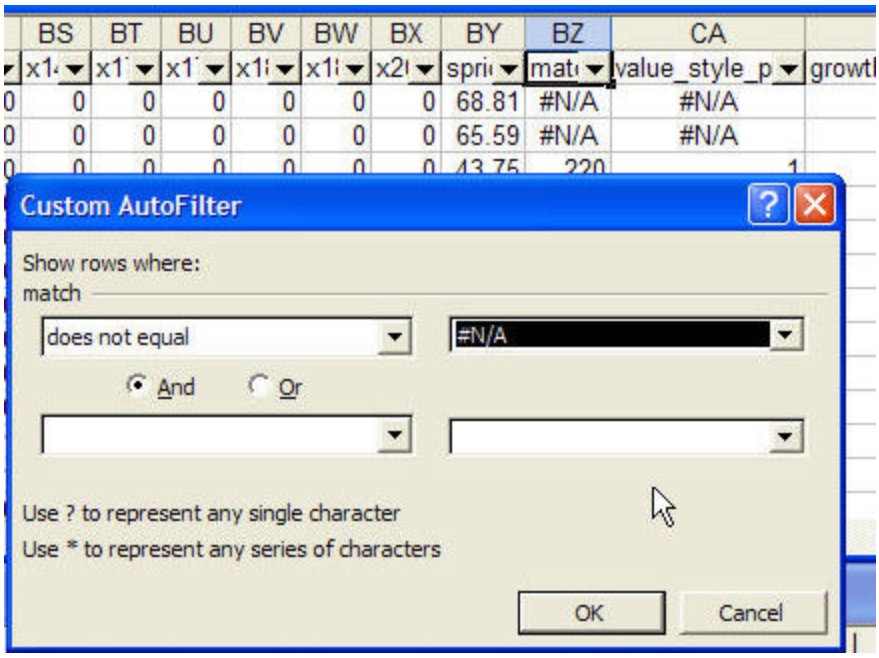


Diagram 6 – The custom AutoFilter tool can filter out any rows of data that have a '#N/A' value returned in the 'match' column.

Once this superfluous data is removed, all of the good data is placed in 16 separate new files for further processing before it is used in the neural network. Before the data is ready to be sent through a neural network for training, each of the 16 files needs to be configured so they can all be merged together seamlessly. Each of the files represents data from a particular sector, but within sectors there are also different categories of companies. For example, in the technology sector there are different categories for internet companies versus microchip companies, but these vary from sector to sector. The problem these categories present in the training data is that each of the files has a different number of categories. In order to merge them all together for an effective training file, the categories had to be deleted in favor of a sector specification value. The following screen capture shows how the sector specification value was added to each file in place of the individual categories.

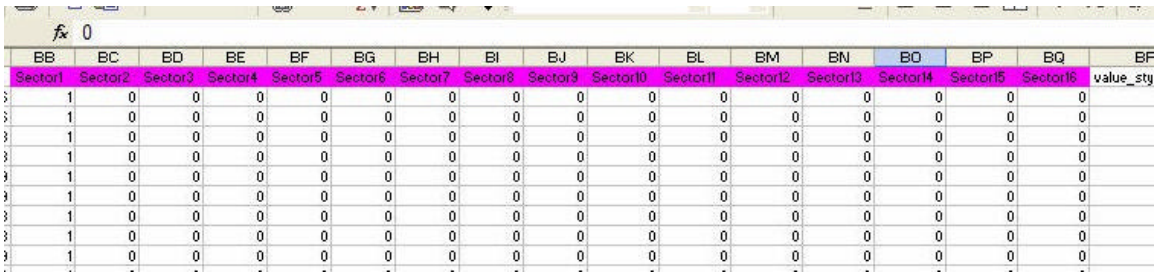


Diagram 7 – This is a screen capture of the Sector classification value that was inserted in order to let the neural net judge whether or not the sector of a stock has any relevance to its value vs. growth rating.

All the data for the first training file was from Sector 1, so for that data the 'Sector1' column was given a 1 value while the rest of the sectors were given a 0 value. The binary value changes according to sector such that all of the stocks have a 1 value in just one of

the 16 columns. This data will allow the neural net to learn whether or not the sector of a stock has any impact on the value vs. growth rating of that stock.

Once this was accomplished on all 16 spreadsheet files, the data was all merged together into the 'Merged Data Set R3000' file, which provided the neural network precisely 1747 examples of ticker data and its corresponding value vs. growth rating. This large file was then saved in the .CSV format, a comma-delimited format that is required by the NGO for processing.

Fundamental Inputs Provided to the Neural Net

The 'Merged Data Set R3000' file contained a total of 42 values that were used as inputs for the neural network training run. The table below is a list of those values that ended up being used as inputs, but first is a screen capture of the official Zacks Current Database from which all of the values and their explanation were gleaned.

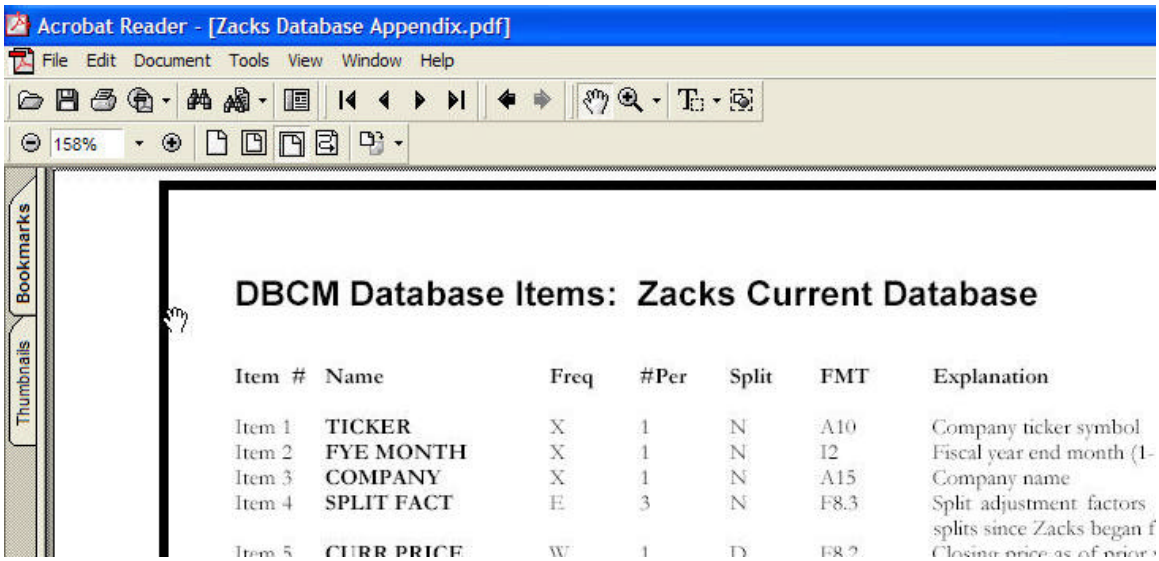


Diagram 8 – Screen capture of DBCM Database Items reference page for the Zacks current database. The table below was compiled from this .PDF document.

Inputs	Input Description
stale	This is an integer value representing the number of months old the data is
roi	Return on Invested capital. Calc: (Income before extras & discontinued operations)/(Long-term debt, convertible debt + non-current capital leases + mortgages + book value preferred stock + common equity)
sales_q	Sales quarterly per share
inc_bnri12	Twelve month Income before non-recurring items per share
book/share	Book value of common equity per share
trend_bvsg	Trend book value per share growth rate calculated from beta of an exponential regression on the last 20 quarters' of fiscal book value per share
cash/share	Cash flow per common share. Calc: CASH FLOW/ SHARES OUT
op_margin	(Income before extraordinary items and discontinued operations for the previous 12 month period/Sales for the previous 12 month period) X 100
net_margin	(Net income for the previous 12 month period/Sales for the previous 12 month period) X 100
pretax_mrg	Trailing four quarter pretax profit margin. Calc: (PRETAX INC/SALES) x 100
curr_ratio	Current Ratio: Current Assets/Current liabilities
payout_rat	Dividend payout ratio. Calc: INDICATED ANNUAL DIVIDEND/ACT EPS 12
inventory	Inventory value per share
tot_c_asst	Total current assets per share

tot_c_liab	Total current liabilities per share
beta	Stock return volatility relative to the S&P 500 over the last 60 months (includes dividends)
roe_12m	Return on Equity; 12-month EPS/Book Value per Share
roa_12m	Return on Assets; 12-month EPS/Total Assets per Share
div_yield%	Dividend yield, based on IND AN DIV and PRICE
ttl_lt_dbt	Total Long-term debt quarterly; Debt due more than one year from balance sheet date; long-term debt, convertible debt, non-current capital leases and mortgages
act_eps_q	Diluted quarterly actual EPS before non-recurring items.
act_eps_12	Diluted quarterly actual earnings per share before non-recurring items
sales_12mo	Sales during the last 12 months
12msls/-4q	Last 12 months of sales divided by the last 12 month sales 4 quarters ago
12meps/-4q	Last 12 months of earnings divided by the last 12 month earnings 4 quarters ago
quick_rati	Quick Ratio – Most Recent (Current Assets- Inventory)/Current Liabilities
Sector1	A value of 1 indicates that the stock is in Sector 1 – Consumer Staples; a value of 0 indicates that it is not.
Sector2	A value of 1 indicates that the stock is in Sector 2 – Consumer Discretionary; a value of 0 indicates that it is not.
Sector3	A value of 1 indicates that the stock is in Sector 3 – Retail/Wholesale; a value of 0 indicates that it is not.
Sector4	A value of 1 indicates that the stock is in Sector 4 – Medical; a value of 0 indicates that it is not.
Sector5	A value of 1 indicates that the stock is in Sector 5 – Auto/Tires/Trucks; a value of 0 indicates that it is not.
Sector6	A value of 1 indicates that the stock is in Sector 6 – Basic Materials; a value of 0 indicates that it is not.
Sector7	A value of 1 indicates that the stock is in Sector 7 – Industrial Products; a value of 0 indicates that it is not.
Sector8	A value of 1 indicates that the stock is in Sector 8 - Construction; a value of 0 indicates that it is not.
Sector9	A value of 1 indicates that the stock is in Sector 9 – Multi-Sector Conglomerates; a value of 0 indicates that it is not.
Sector10	A value of 1 indicates that the stock is in Sector 10 – Computers and Technology; a value of 0 indicates that it is not.
Sector11	A value of 1 indicates that the stock is in Sector 11 - Aerospace; a value of 0 indicates that it is not.
Sector12	A value of 1 indicates that the stock is in Sector 12 – Oils/Energy; a value of 0 indicates that it is not.
Sector13	A value of 1 indicates that the stock is in Sector 13 - Finance; a value of 0 indicates that it is not.
Sector14	A value of 1 indicates that the stock is in Sector 14 - Utilities; a value of 0 indicates that it is not.
Sector15	A value of 1 indicates that the stock is in Sector 15 - Transportation; a value of 0 indicates that it is not.
Sector16	A value of 1 indicates that the stock is in Sector 16 – Business Services; a value of 0 indicates that it is not.

Table 1 – This table shows all of the data selections that were present in the 1746-example Merged Data Set file, and were available for the neural network to train off of in NGO.

The NeuroGenetic Optimizer

Now that the arduous task of preparing the training data has been accomplished, it's time to sit back and let the raw computing power of BioComp Systems' NeuroGenetic Optimizer take control. First off, a little background information should be provided here on how the NGO functions and why it employs the methods that it does.

For those of us who ran to dictionary.com when they heard the term 'neurogenetic' the record shall be set straight, 'neurogenetic' is not an actual word, but it is a term that makes explicit the type of function the software performs. An overview of neural networks has already been provided, and as one might expect, the 'neuro' in 'neurogenetic' refers to the NGO's ability to craft custom neural network systems for its users. 'Genetic' refers to the manner in which the neural networks are created, and this, incidentally, is what makes the NGO so unique when juxtaposed to other neural network creation software. The NGO utilizes genetic algorithms to "evolve" high quality neural network architectures. To understand why this is beneficial one must first understand what causes neural networks to have varying architectures.

Neural networks come in all shapes and sizes. For example, the 'hidden layer' that was described in the background information above can contain any number of nodes. Typically, the greater the number of nodes given to a neural network, the more specific its answer will be. Returning once again to the metaphor of the small child learning about trees, this is equivalent to the child needing to learn what the definition of a palm tree is. If he needed to learn what a palm tree was, he would need to know a few more things about it than if he were going to classify it simply as a tree. So depending on what kind of problem is being solved, the neural network that works best will be able to provide a sufficiently broad or specific answer. Another important factor is the number of nodes in the *input* layer, or in other words, which of the inputs (42 in this case – see Table 1) are most useful in obtaining an accurate answer from the neural network? It may very well be that a neural network that doesn't use all the inputs available, since some outputs might be misleading or functionally unrelated to the desired outputs creates the best answer to the problem.

All of these factors make neural networks difficult to optimize for the specific problem, since there are so many different combinations of inputs and node numbers to combine together. So the number of combinations possible is enormous, but there are only a handful of really excellent networks that will solve the problem; this is where genetic algorithms come in handy.

Genetic Algorithms

The NGO help file contains an introductory explanation of how genetic algorithms progress when used to optimize neural network architectures for a given problem.

> The NGO uses genetic algorithms to perform a combinatorial search across all provided input variables and neural network configurations (within user specified constraints) and then creates, trains and tests these networks to determine their accuracy. These networks can be saved to disk for later use in other applications.
>
> …The NGO then...
>
> - builds and validates training and test data sets,
> - creates a population of candidate input variables and neural structures,
> - builds the neural networks,
> - trains them,
> - evaluates them,
> - selects the top networks,
> - pairs up the genetic material representing the inputs and neural structure of these networks,
> - exchanges genetic material between them,
> - throws in a few mutations for a flavor of random search and
> - goes back into the training/testing cycle again.
>
> (NGO Help file)

Notice how this process parallels biological evolution by operating on a 'survival of the fittest' paradigm. The term "combinatorial" is also used in the quote above to refer to the combinations of architectures made when best networks are found and spliced together in the hope of getting better networks. The architecture of the neural network is recorded as the network's "genetic material" and then it is combined with other networks, adding in some random mutations in the hope that it might randomly find even more successful neural net designs.

One of the most important parts of this process is the training and evaluating of individual neural net designs, which allows the NGO to identify the best networks as it searches. The fitness of each network is determined by randomly selecting half of the sample data and training the network on it, and then taking the remaining half of the sample data and seeing how accurately the model predicts the provided outputs given the provided inputs. After a specified number of generations the NGO will return a list of the top ten neural networks found by the genetic algorithm search. The best network can then be saved and used as the final product of the project.

Running the Training Data through the NGO

This section will provide the reader with a run-through of how this project's data was loaded into the NGO and subsequently used to train a highly successful neural network.

First, the data has to be opened from the NGO's File menu by clicking File>Open>Train/Test Data as shown here. The 'Merged Data Set R3000' can be seen being opened in this screen capture:

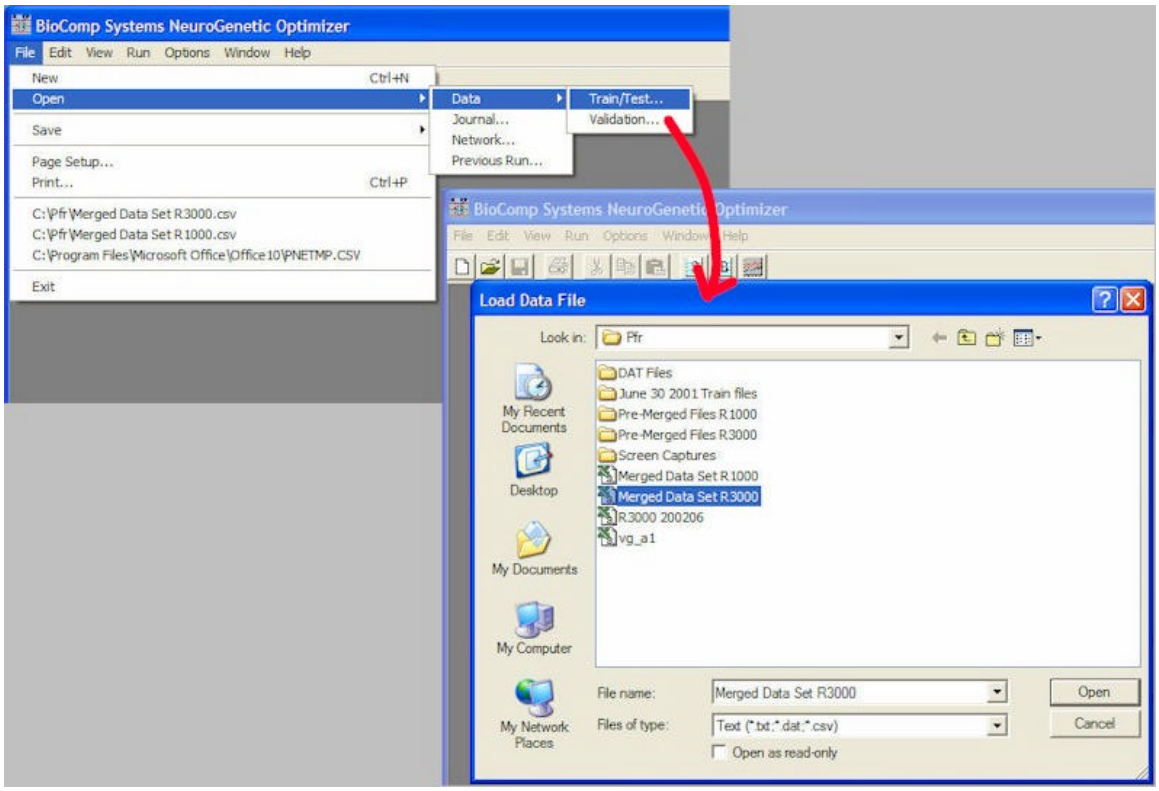
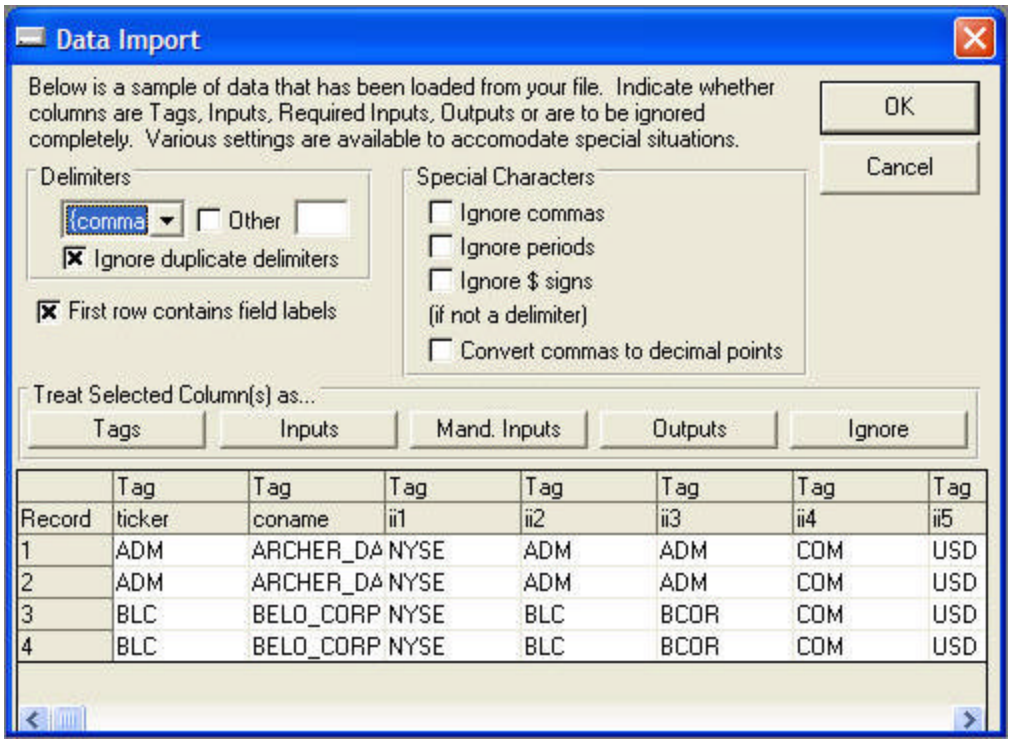


Diagram 9 – Opening 'Merged Data Set R3000' in the NGO. The NGO will then go through a series of dialogue boxes in which it receives specifications on how to handle the data.



	Tag	Tag	Tag	Tag	Tag	Tag	Tag
Record	ticker	coname	ii1	ii2	ii3	ii4	ii5
1	ADM	ARCHER_DA	NYSE	ADM	ADM	COM	USD
2	ADM	ARCHER_DA	NYSE	ADM	ADM	COM	USD
3	BLC	BELO_CORP	NYSE	BLC	BCOR	COM	USD
4	BLC	BELO_CORP	NYSE	BLC	BCOR	COM	USD

Diagram 10 – Data Import box that appears when the desired data file is being prepared for the training run.

In the Data Import box the different columns of data are specified to be in one of 5 categories: Tags, Inputs, Mandatory Inputs, Outputs, and Ignore. Here is a brief definition of each input type:

1. **Tags** – Any input column that is listed as a Tag will not be used as an input, but will be conserved by the NGO as data that remains associated with that row of inputs and outputs. In this case, the ticker and company names are good tags because they allow the user to identify which company is being processed later on after a file is created by the NGO.
2. **Inputs** – These columns will be designated as potential input nodes for the neural network; they may or may not be used in the input layer of the final neural network.
3. **Mandatory Inputs** – Inputs that must be included in the input layer, usually because the user knows ahead of time that any decent answer absolutely must be linked to a particular input.
4. **Outputs** – In a Classification neural network, such as the one being created in this project, there is no limit on the number of outputs, but in most other neural networks only one output is required.
5. **Ignore** – The NGO will completely ignore any columns marked as such, and they will not be available as tags in the end product.

The data is broken down so that all of the columns listed as inputs in Table 1 were designated as Inputs, the value vs. growth numbers were designated as Outputs, and any remaining values were set as Tags (such as the ticker name, etc…)

The NGO then asks the user to provide settings specifications for the type of neural network desired as well as the type of genetic algorithms that will be used for optimization. Here is a screen capture of the Data Preparation dialogue box.

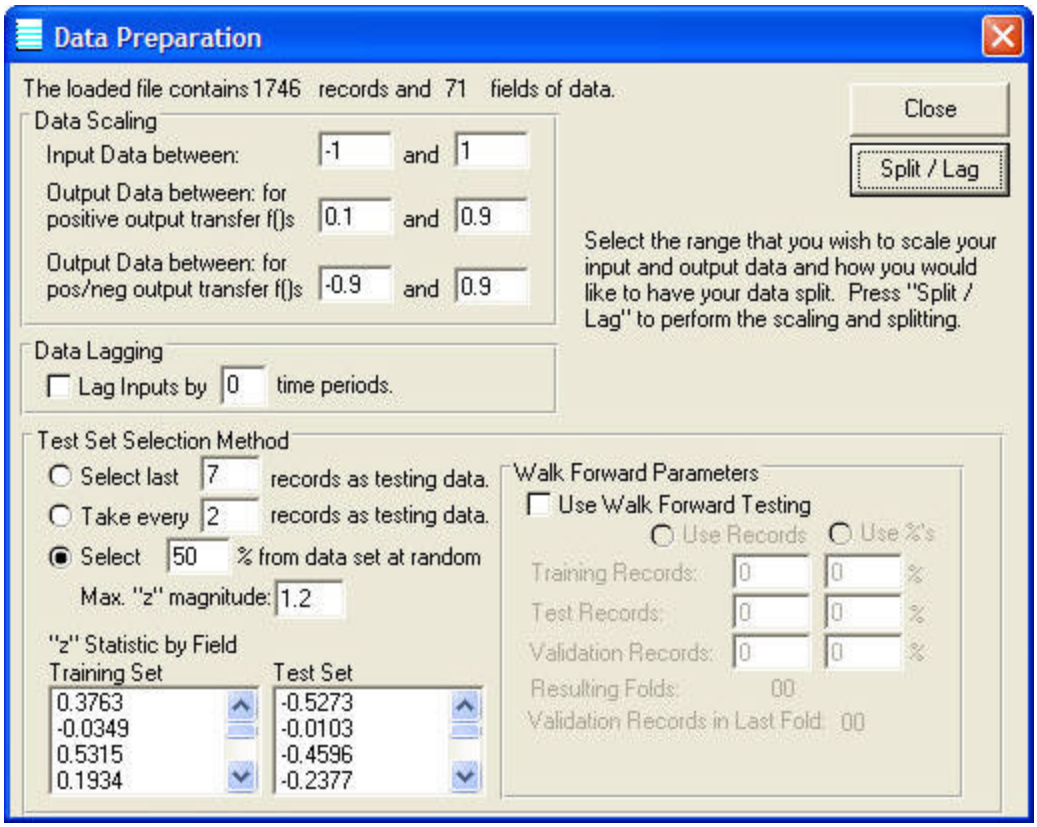


Diagram 11 – The Data Preparation dialogue box is where the Split/Lag function can be applied to the incoming data. The Test Set selection method is also determined here.

The Data Scaling capability of the NGO scales all of the variables in the data set between -1 and 1. This is done by identifying the minimum and maximum values in each column and scaling within a given range, this makes neural net training a less computation-intensive task. The Test Set Selection Method is important for determining the overall 'fitness' of the neural networks as they are evolved, and the best way to determine fitness is to select half of the data set at random and test the current neural network on it. The Split/Lag function will go ahead and split data in this way and apply the "z" statistic to it, which will show what the average deviations are between the data sets. Different randomly selected sets will continue to be checked for statistical similarity until an adequate split is found.

Now the data is ready for a run through the NGO and for neural network creation. All of the settings for genetic algorithms and neural networks are all pre-set, but can be changed from the Options menu. Since these settings are already set to an optimal profile, the data is ready to be run by either hitting F5 or clicking Start>Run. Once the run is going, a number of fascinating properties can be observed. First, the training neural networks and the accuracy of their predictions can be observed as graphs. One of the graphs is shown in the screen capture below; the solid black line in the background of the graph represents the values that the neural network is trying to predict, and the blue line represents how close its predictions are coming once fully trained.

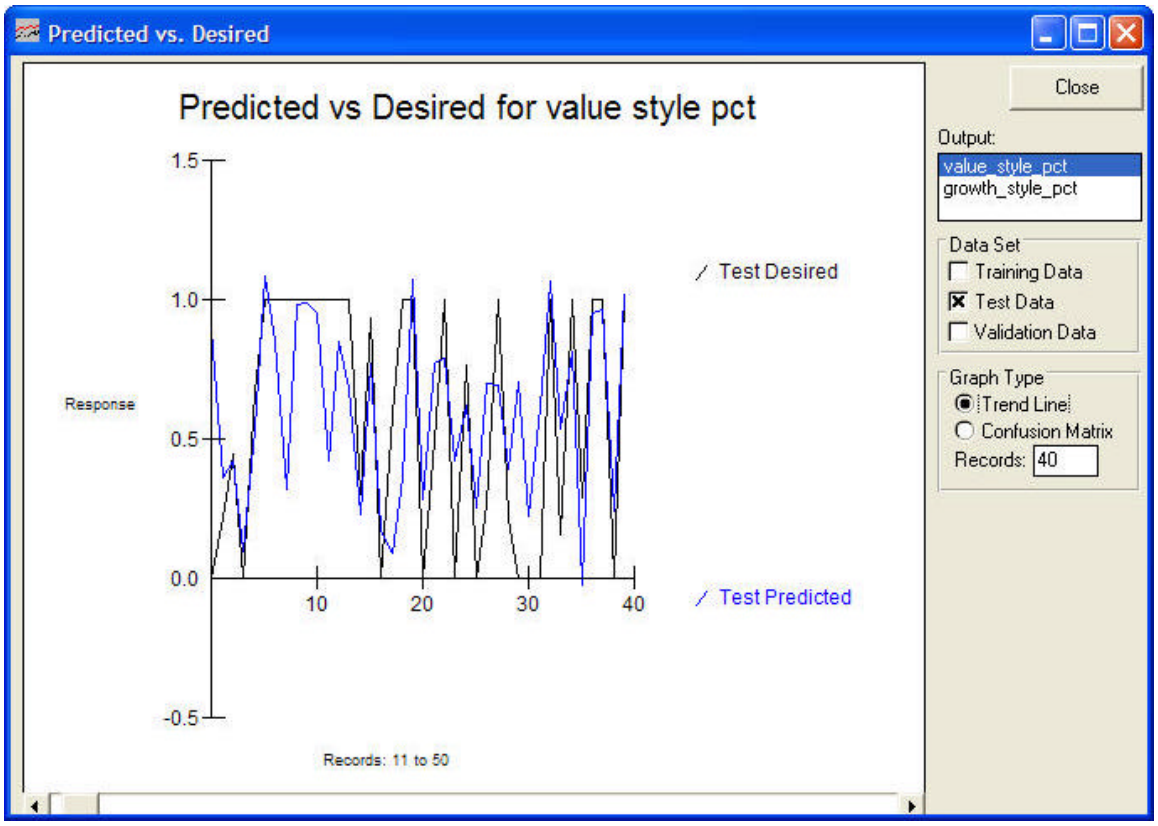


Diagram 12 – This is a Predicted vs. Desired graph for one of the neural networks during the training run. This isn't necessarily the best predictor, this is just the graph of one possible neural network in a stream of mutating, evolving, mating, and training neural network architectures.

Another important dialogue box during the run is shown below. It is the window where the report is given about the current best network, which is selected from the many networks being created by the computer.

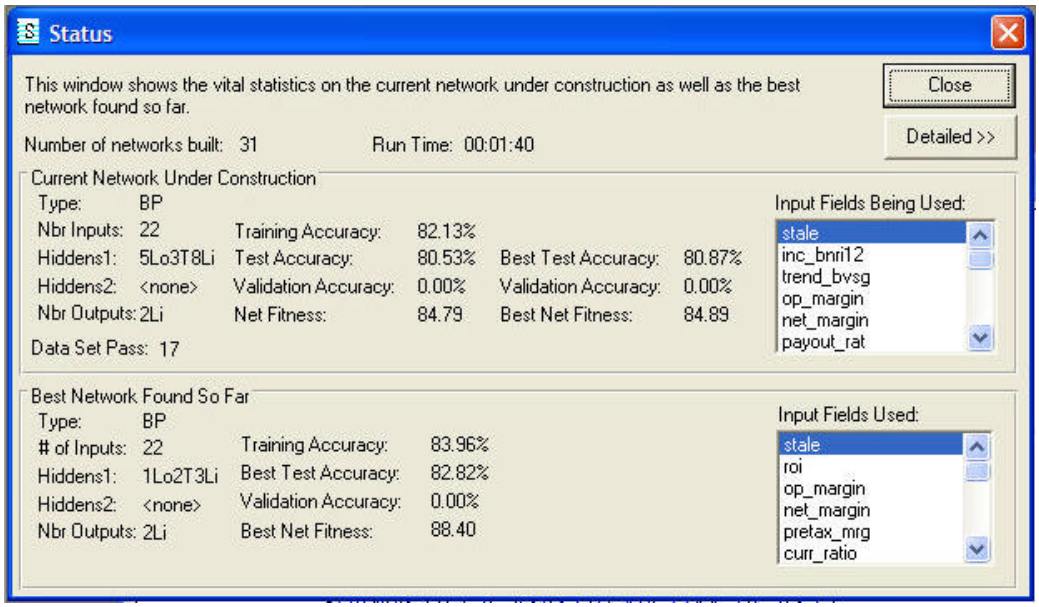


Diagram 13 – The Status dialogue box is the main window for watching the progress of the training neural network. In the above box, 31 networks have been created in a minute and 40 seconds, and the best network so far has an 83.96% training accuracy and an 82.82% Best Test accuracy. Also, the number of inputs in the best network is only 22, and the names of the input fields used is available in a scroll menu.

The Final Value vs. Growth Classifier Neural Network

The final neural network produced for the product was a result of this optimization procedure employed by the NGO. The final status of the Best Network is shown below in a screen capture similar to Diagram 13, but in this case the region for the Best Network Found So Far details the status of the final neural network that will be used as a value vs. growth classifier.

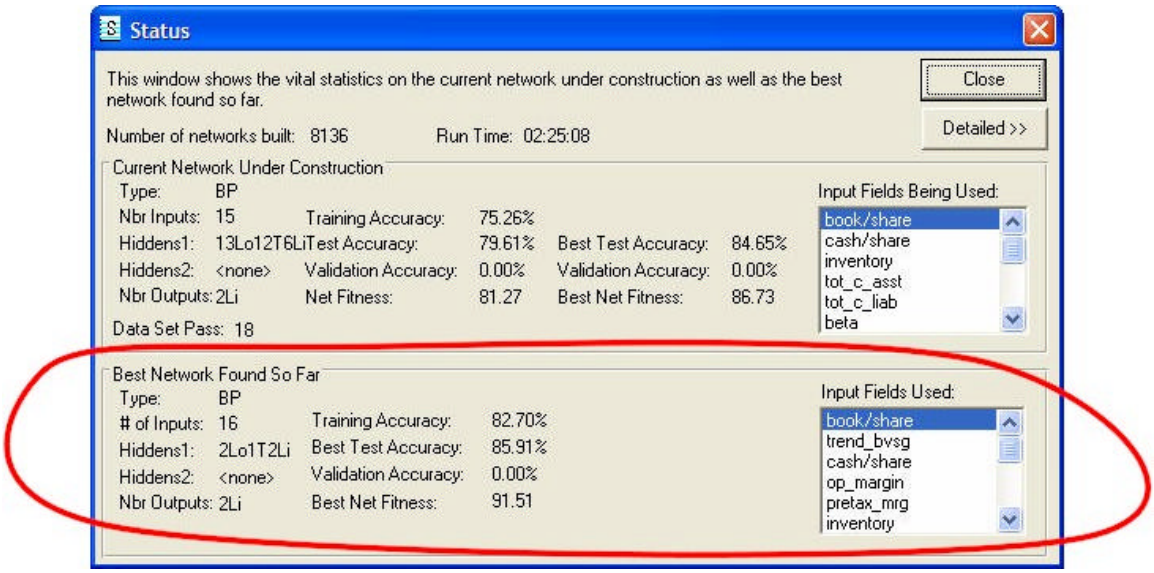


Diagram 14 – Here are the statistics for the neural network that resulted from almost 2 and a half hours of NGO training.

The network that appears to work the best is a Back Propagation network that uses 16 inputs, which are specified in a scroll down menu to the right. "Book/share" was set as a mandatory input for the net, because it is widely known to be a contributing factor to the value vs. growth designation. If it had not been set as a mandatory input, then the NGO might have gotten confused and left it out of the input set during training. Of the 42 inputs put before the NGO, the 16 that it uses for value vs. growth prediction are

1. **book value of common equity per share,**
2. **trend book value per share growth rate,**
3. **cash flow per common share,**
4. **operating margin,**
5. **pretax margin,**
6. **inventory value,**
7. **total current assets per share,**
8. **total current liabilities per share,**
9. **beta,**
10. **return on assets,**

11. dividend yield,
12. 12 month sales divided by 12 month sales 4 quarters back,
13. presence in Sector 1 (Consumer Staples),
14. presence in Sector 3 (Retail/Wholesale),
15. presence in Sector 6 (Basic Materials), and
16. presence in Sector 10 (Computers and Technology).

This network has a single hidden layer with 5 hidden nodes, and they use the three different transfer functions described in the Background section. There are 2 Logistic Sigmoid, 2 Linear, and 1 Hyperbolic Tangent transfer functions in the hidden layer. The network can then be exported to Excel by using the NGO's "Send to Excel" feature. This process is shown in greater detail with the following set of screen captures.

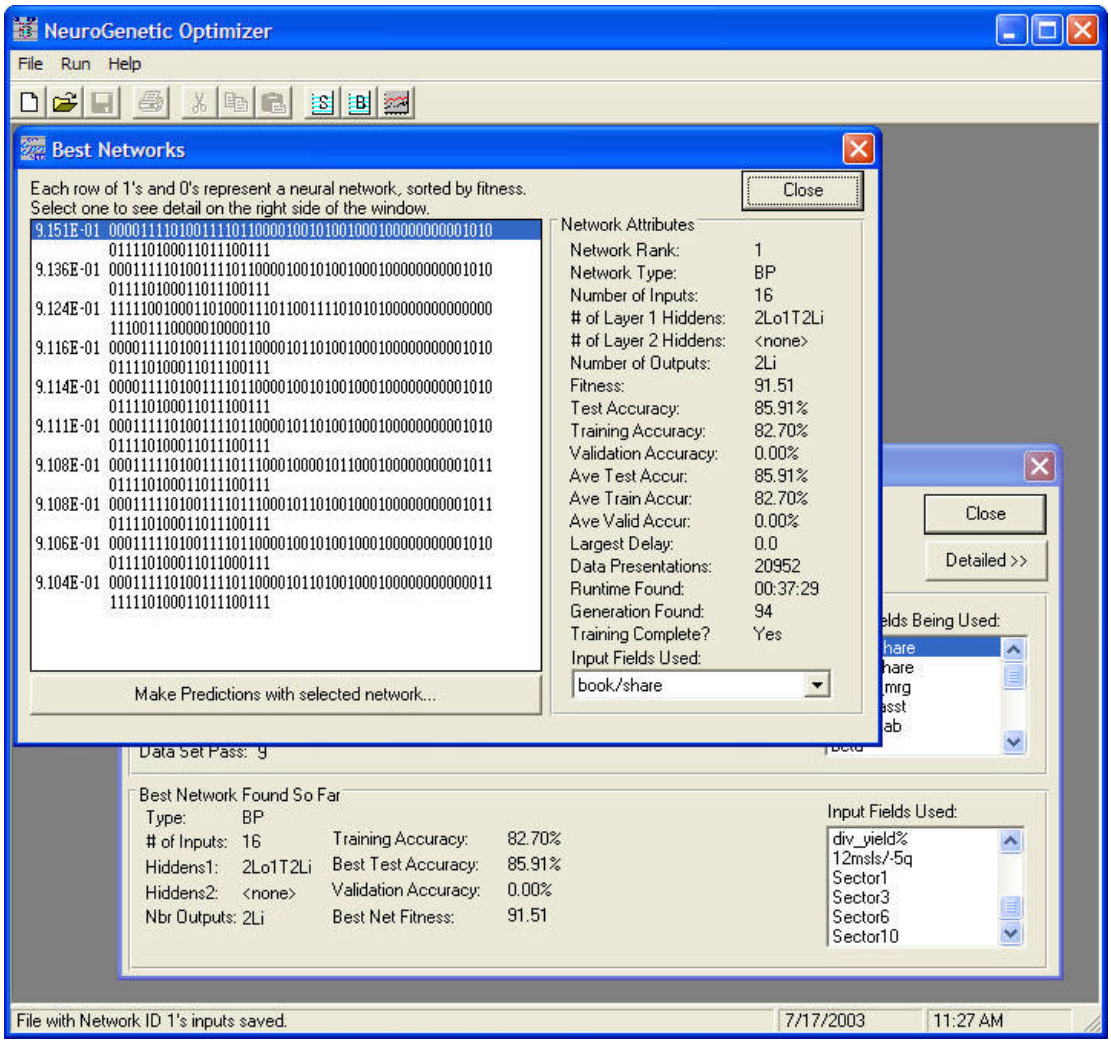


Diagram 15 – Here the list of the top ten networks produced by the NGO training run are shown. They are displayed in the Best Networks window, and the #1 network is selected at the top with its status shown on the right.

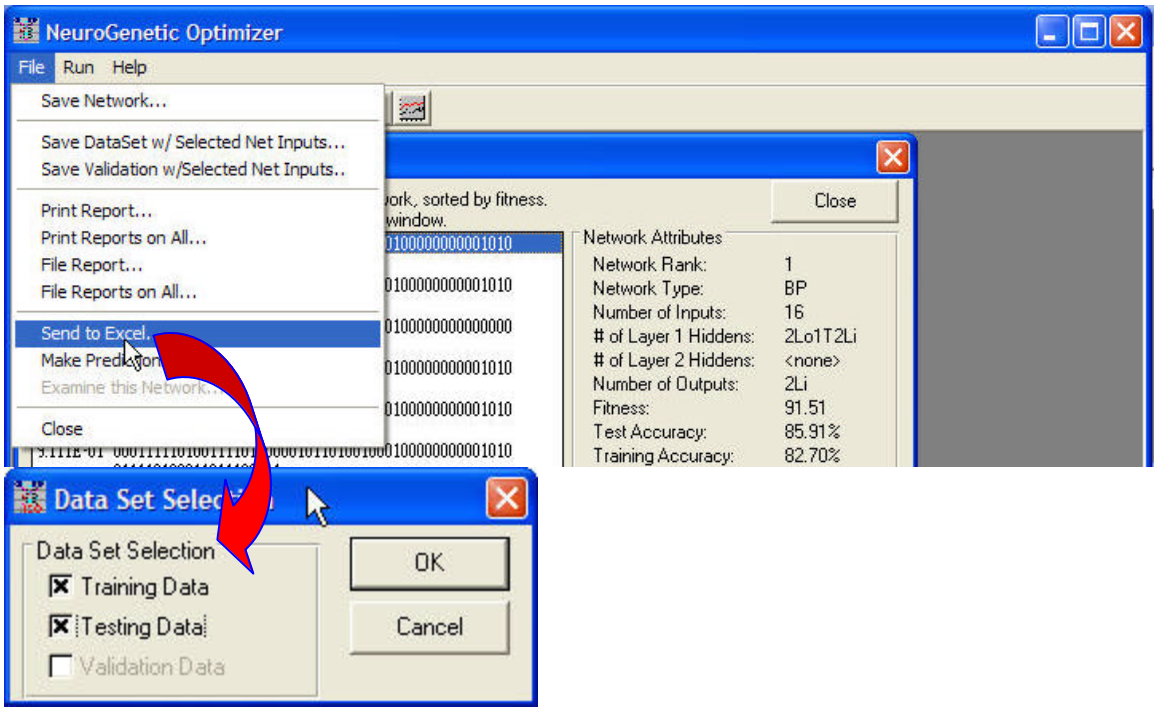


Diagram 16 – Excel must be open when the network is being exported to that program, and when the network is sent, it is important to send both the Training and Testing Data.

Once the data is sent to Excel, the Network will save in 2 different file formats for later access, until this is done though, it is not a saved network. In order to see that the neural network has been effective, response curves can be built in Excel by utilizing the Penney menu, which is a Visual Basic Application Add-In for Excel and is automatically installed with the NGO software. Since response curves are a central method for judging the efficacy of a neural network, this final section before the conclusion will be devoted to their explanation.

Response Curves

A response curve, or Neural Response Surface, provides information about how two or more variables relate to one another in the prediction of a particular output. These response curves are generated by isolating one or two of the inputs and changing them over a particular range of values in order to see what relationship the net perceives between the variables and the output. At this point a graph is produced so the relationship can be visualized.

This tool can be used to show whether or not the connections that the neural network has made between input and output variables make sense. One such relationship that can be easily verified with the response curves is that between book value per share, dividend yield per share, and the value stock designation. Any money manager could tell you that generally when a company has high dividend yield per share and high book value per share then it will definitely be designated as a value stock. However, if the opposite is true, and both the dividend yield and book value are low, then it is more likely going to be listed as a growth stock. Here is a picture of the response surface that shows that the neural network learned to make just that distinction.

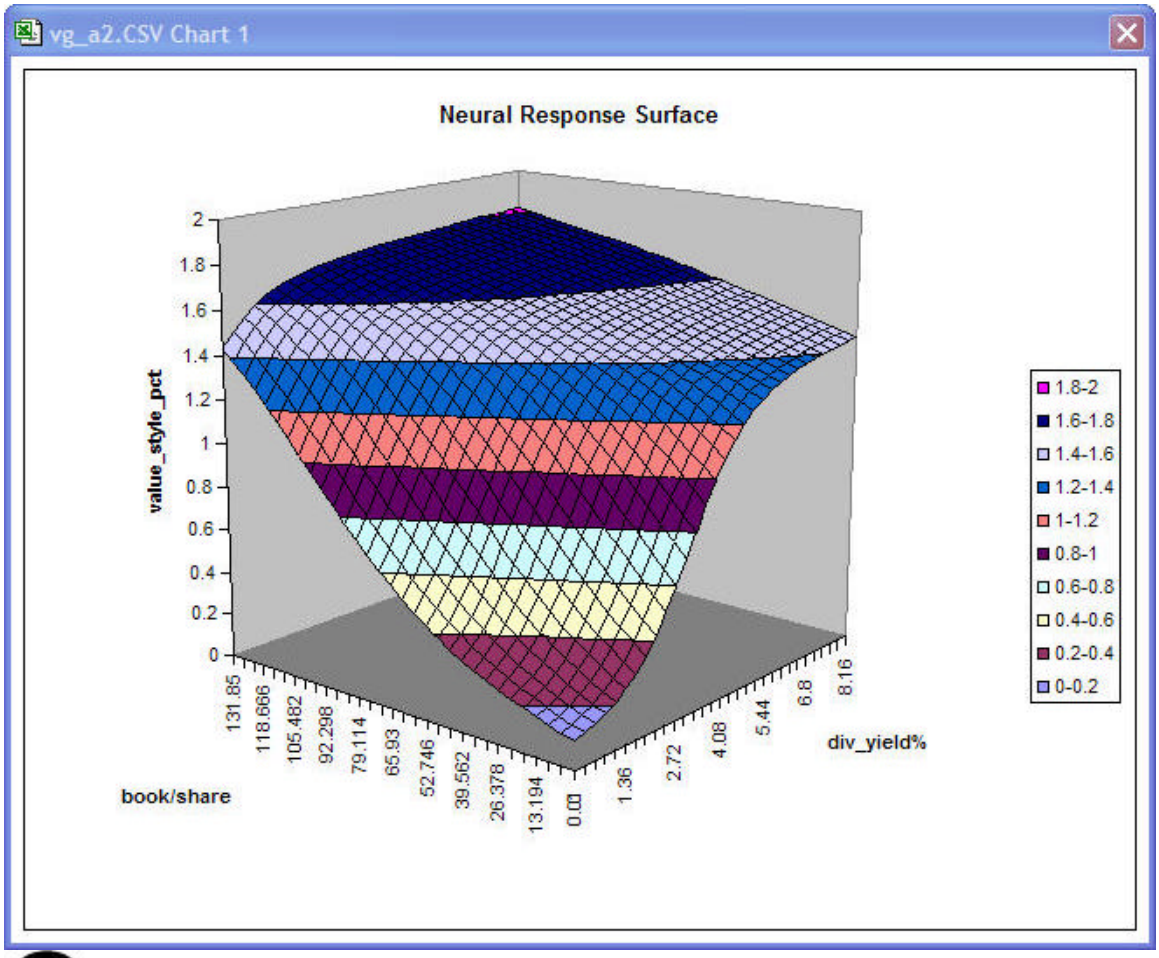


Diagram 16 – The response surface showing the relationship between dividend yield, book value per share, and the value stock designation.

A few other things can be inferred from the graph as well, since it shows 2 independent values in relation to the output. For instance, when dividend yield is really high, an increase in the book value will have little effect on the stock being a value stock. If dividend yield is low however, book value has a large impact on the classification. So a number of things can be said about these response curves and then checked against the common sense knowledge that any market analyst would have. In this way human beings can check the reasoning of the neural network and declare whether it is marred by confusion or has the same common sense perceptions about market variables as an analyst.

Conclusion

The neural network has performed at a high enough accuracy level and clearly made the appropriate connections between the 16 independent variables the NGO trained it to utilize. The ultimate test of the neural network produced by this project is yet to come; when it is used to predict the value vs. growth designations of stocks. According to the training and test data, it should be able to predict the Russell index value vs. growth designations with a high degree of accuracy without having to rely on I/B/E/S estimates.

Bibliography

1. Francis, Louis. "The Basics of Neural Networks Demystified." *Contingencies* magazine. <http://www.contingencies.org/novdec01/workshop.pdf>
2. NeuroGenetic Optimizer help file, BioComp Systems Inc. (Version 2.6), www.bio-comp.com
3. Zacks Database Appendix: Revised March 2001, Zacks Investment Research, Inc. Chicago, IL, www.zacks.com
4. Russell 3000 Value and Growth Indices, Frank Russell Company, Tacoma WA. www.russell.com
5. EXCEL help file, Microsoft Corporation, version for XP, www.microsoft.com

Parallax

Parallax Financial Research, Inc. (CTA) is a small research boutique in Redmond, WA., that is dedicated to delivering the finest stock and commodity forecasting tools to the professional money management community. These tools and forecasting models are all based on applications of chaos theory to the financial markets, and many incorporate neural networks as a final step to generate forecasts. The President & CEO of Parallax, Mr. Kris Kaufman, has degrees in math and physics, and spent 13 years as a geophysicist before incorporating Parallax in 1990. Clients past and present include Morgan Stanley Dean-Witter, Fidelity Investments, Goldman Sachs, Managed Quantitative Advisors, Corporate Consulting Group, Newport Financial Partners, Bankers Trust, Daiwa Securities, Schroders, Deutsche-Bank Securities, Paine Webber, Van Kampen American Capital, Marque Millennium Capital Management, Bank of New York, Ellis Island Partners, Vantage Consulting Group, HED Capital Management, McCarthy, Crisanti, & Maffei (MCM),RAB Capital, Millennium Capital, and Mathsoft.