Neural Network Predictions of Stock Price Fluctuations

By Wojciech Gryc (wojciech@gmail.com)

Table of Contents

Abstract

This research paper is the result of a three-month-long independent study focusing on the ability of feedforward and recurrent neural networks in predicting stock price fluctuations of companies within the "Consumer Discretionary" category of the Global Industry Classification Standard (GICS). The paper focuses on predictions for Circuit City Stores Inc. (CC) for 1-, 5-, and 20-business day periods through the use of input patterns based on fundamental and technical indicators from the stock market and economy. The project is unique in its focus on companies selling to consumers. The results of the project are promising, with statistically significant results appearing for 1-day single-layer feedforward and Elman networks, as well as networks based on fundamental inputs and predicting for 20-day periods.

Introduction

Within the last two decades, a great deal of attention has been focused on the idea of predicting stock prices and price fluctuations through the use of neural networks. Such research, if successful, has the potential to bring numerous rewards – both academic and financial – to researchers or analysts that find any patterns. There are numerous reasons why neural networks offer an advantage in the quest to predict stock price fluctuations. Many theories focusing on economics and finance often begin with a set of assumptions, and hope to build rational and believable models of the world. At this time, however, there is only one widely accepted theory on stock prices and markets, called the Efficient Markets Hypothesis (EMH), which ultimately concludes that such price changes may never be predicted.

What is unique about neural networks is that the amount of assumptions made through their use is minimized, and focuses solely on the data being fed into them. Researchers can avoid assumptions about perfectly rational human beings, used by the EMH, and simply watch if and how their networks converge. Whether random or fully predictable, a correctly designed neural network will theoretically converge on the optimal result. Most other models do not have this luxury.

With this in mind, the research project presented below aims to test how well neural networks predict stock price fluctuations by presenting a number of diverse architectures using two very different sets of inputs and combinations thereof. Specifically, the work aims to shed light on the differences

between using fundamental and technical analysis combined with neural networks to predict stock price fluctuations within 1, 5, and 20-business day time frames, starting in the year 2000 and ending in 2005. A second aim of this project is to compare how well predictions fare in both feedforward and recurrent networks.

This paper is organized into three parts, beginning with a theoretical overview of the psychology of the stock market, and why stock prices may be predictable. The paper then goes into some discussion of of the theory behind neural networks, and how they work. Finally, it ends with the results of the project, based on over 125 different combinations of network architectures and training settings, and their implications on the predictive ability of neural networks within the stock market.

Part 1: The Stock Market

Stock Market Psychology

A great deal of work has gone into analyzing the psychology of the stock market. Simply put, the stock market is a place, be it abstract or real, where large groups of people gather to buy and sell stocks. The dynamics of such a place is often linked with that of group psychology and many have tried to build theories of how the stock market functions based on the psychology of large groups.

The only widely accepted theory of how the stock market works is called the Efficient Market Hypothesis (EMH) (Wärneryd, 2001), and in its simplest form, states that stock prices always reflect all the available information about the market and companies in question (Shleifer, 2000). The theory rests on three assumptions, all focusing on investors themselves. These are:

- Investors are fully rational and respond rationally to all changes within the market. This may go so far as saying that investors subconsciously use tools like Bayesian probability to analyze probabilities and make decisions.
- Irrational investors trade randomly, and as such, all random investments cancel themselves out, leaving no effect on the market.
- Even if investors' irrational decisions do not cancel out through probabilistic methods, other rational investors will take advantage of these irrational decisions to make a profit. Such schemes bring the market back to an efficient state.

The EMH has been widely accepted, though it does have shortcomings. On a simple behavioural level, it is clear that people do not use Bayesian probability to learn from their mistakes (Shleifer, 2000) and are rarely fully rational. The theory also assumes that every investor has access to all the required information to value a stock. This problem of "perfect information" is its interpretation – while financial statements and economic statistics are widely available, one cannot assume that such information is interpreted correctly by all investors (Dyckman & Morse, 1986). Indeed, this interpretation often depends on how information is framed and where it comes from (Shleifer, 2000). Furthermore, well-documented stock market anomalies (Dimson, 1988) such as an increase in share prices in January and a continuation of Friday price trends on Mondays run counter to the idea of unpredictable and perfectly efficient markets.

In terms of large groups, one of the biggest problems with the EMH is that the assumptions

imply investors act independently of each other: a naive investor making mistakes will be taken advantage of by a rational investor. However, it is often the case that investors act in unison (Shleifer, 2000), as can be seen during periods of irrationally increasing prices, like the late 1990s' dot-com stock craze. Indeed, a common stock trading strategy, especially among technical analysts, is the "contrarian strategy" (Siegel, 1998): by observing the markets and seeing how large groups of investors act, one can make a great deal of money by doing the opposite of the group. This, however, depends on the assumptions that investors act in a herd-like way and that their group-based decisions, whether conscious or not, are incorrect.

Within economics, there are two important time frames: the long and the short run. Supporters of the EMH theory argue that in the long run, patterns and anomalies are often noticed by investors and eventually cancel themselves out. For example, if enough investors learn about increasing share prices in January, many investors will try to sell their stocks in that month to make a profit, thus lowering the price and canceling out the effect. However, there is a difference between efficiency in the long- and short-terms. It is possible for the market to be efficient in the long run while exhibiting irrational and potentially predictable patterns in the short term (Malkiel, 2003). Thus, it is may be possible, even under the EMH, to have a neural network predict short-term patterns within the stock market or individual companies' shares. Within the context of this project, it is these short-term irregularities and collective mistakes by investors that may be extremely useful, especially for a neural network. By focusing on a time frame of at most one month, any patterns that may arise are not long-term, and a neural network may learn to predict them before investors do.

To make such predictions, a neural network must have access to information. Even if one acknowledges that stock prices may be predicted in a given span of time, there are multiple strategies and forms of analysis that may or may not work. When predicting share price fluctuations, there are two major camps: technical and fundamental. Both are used within this project, and an analysis of their effects and usefulness based on the accuracies of their respective neural networks is given near the end.

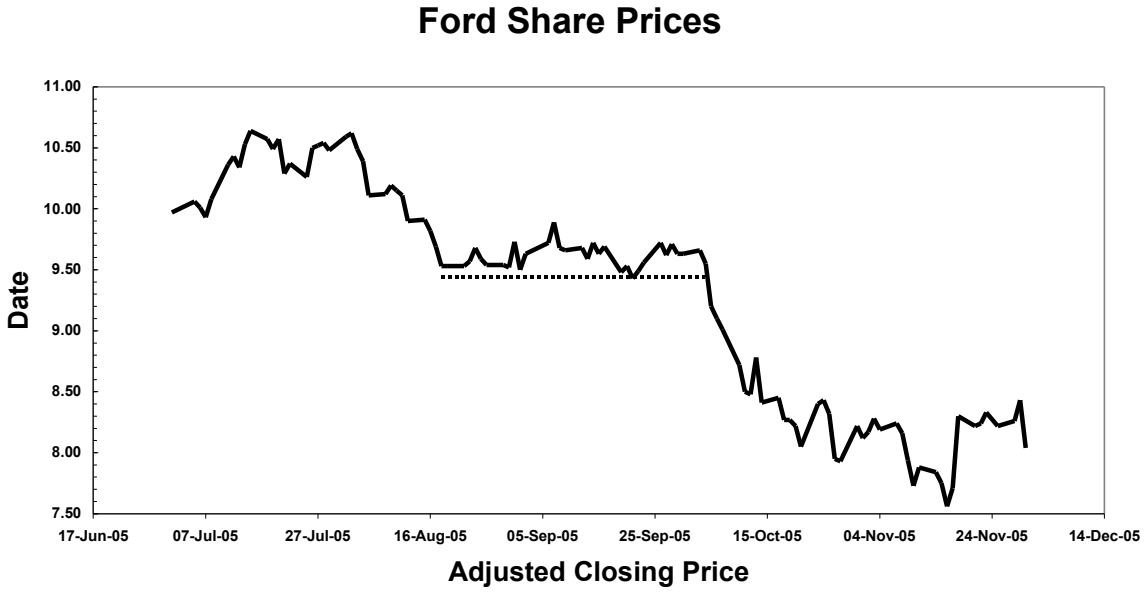
Technical Analysis

Whereas fundamental analysis looks at many different aspects of a company and economy to predict how well a stock will perform, the assumption inherent in technical analysis is that all such required information is available in a share's past prices. Specifically, technical analysis is often defined as “The process of analyzing a security’s historical prices in an effort to determine probable

future prices." (Achelis, 2001, pt. 1) Some forms of technical analysis also incorporate volume – how often a security is traded during a day or within another time period – to help predict prices.

The way technical analysis helps predict whether a stock will increase or decrease in value is by signaling trend reversals. According to technical analysts, many securities exhibit different types of growth over time, and the key to making a profit in the stock market is to predict when a stock's tendency to increase in value will rise, or a falling stock price will stop and begin to increase. By viewing a stock's past performance, one may begin to understand how people investing in that stock act and which prices they are comfortable with. Indeed, some (Wärneryd, 2001; Iihara, Kato & Tokunaga, 2001) have compared the theory behind technical analysis to herd behaviour in the stock market.

For example, one technical indicator, called the Trading Range Breakout, assumes that there is an upper and lower price that investors will tolerate for their stocks. If one considers a lower bound for the price, investors may often allow the price to fall to the lower bound and then bounce back, but the group has a psychological barrier to letting the price fall below this bound. One can predict a trend reversal – in this case, a stock's price tending to fall within the coming days – when this lower bound is finally broken and the stock's price falls lower. When this happens, technical analysts argue that one can be sure the price change will be significant, and in the downward direction. Observing such patterns after they occur is easy, and an example with share prices of Ford is shown below. In this case, one can see a lower bound in starting mid-August 2005 and lasting until the end of September 2005. After the lower bound is breached, the price drops significantly.



The use of technical analysis within investment decisions has been controversial. According to Efficient Market Theory, investors act completely rationally and have access to all required information to value a stock's price during the present – as such, a stock's current price reflects its value at that time, and has no connection with past prices. Such assumptions have been criticized, and statistical analysis of technical indicators has shown their usefulness in predicting trend reversals and share price fluctuations (Brock, Lakonishok & LeBaron, 1992). Interestingly, historical technical analysis of pre-1929 stock market data shows that technical analysis would have helped investors predict the stock market crash and the Great Depression (Siegel, 1998).

Within the context of this research project, technical analysis is extremely useful when combined with neural networks. This type of analysis depends largely on past prices of share sales, which are often available instantaneously and in real-time. The data is often quite extensive as well, including daily high and low prices, adjusted closing prices to account for dividends and split shares, and numerous other variables. Indeed, the problem for technical analysis is that there is so much data and forms of analysis that one does not know what to do with it all. For example, *Technical Analysis from A to Z* (Achelis, 2001), a reference guide for technical analysts, lists 135 different tools for doing such analysis. By entering all this information into a neural network, the network may be able to discern relevant patterns without having an analyst do it. Furthermore, because everything depends on the analysis of past prices, it is theoretically possible to use prices as inputs and have the neural network develop its own technical indicators.

One of the problems with technical analysis is that while so many tools exist, many of the details are subjective: it is still up to the investor to choose values for the variables within each tool, and it is the investor's decision to follow a buy or sell signal from a technical indicator. With so many options, neural networks can play a crucial role in deciding which technical variables to follow when analyzing past prices. Within this project, neural networks are presented with a number of technical variables and buy/sell signals, and it is up to the network to learn which indicators are useful and which are not.

Based on discussions with financial experts and reading into prior research (Brock, Lakonishok & LeBaron, 1992), six types of indicators were chosen as inputs for neural networks using technical analysis.

Variable Moving Average (VMA)

While technical analysis posits that past stock prices are not random, any technical analyst will admit that there is a great deal of noise within the data. To smoothen the noise, a variable moving average is used, using the following equation:

$$\frac{\sum_{i=1}^{k} p_i}{k}$$

where p_i is the price for the i^{th} day, and the moving average extends for k days. As defined by Brock, Lakonishok & LeBaron (1992), a VMA indicator uses two separate moving averages, one for 200 days and one for 1 day. When the price of the 1-day moving average is 1% higher than the 200-day moving average, the stock is considered overbought, and its price is likely to fall in the future. Thus, a sell signal (-1) is given to the network for that day. Likewise, if the 1-day moving average is 1% below the 200-day moving average, a buy signal (1) is given, because the stock is oversold and has a low price.

A second variation on the above is also used as an input to the network, with a 200-day moving average and a 2-day moving average.

Trading Range Breakout (TRB)

The premise behind the TRB is that investors holding securities have a psychological barrier to extremely high and low prices. As shown earlier with Ford's price drop, share prices often tend to bounce off of this psychological barrier. Once the barrier is broken, however, and a stock's price falls below the historical minimum, a sell signal is generated, while a buy signal is generated when a stock's price rises above a historical maximum.

The implementation of the TRB is simple: find the lowest and highest prices for the last k days, and if the stock's closing price rises above the maximum or falls below the minimum, the stock should be bought or sold, respectively. In this project, 50-day, 100-day, and 200-day TRBs are used, with buy signals generated when a price rises 1% above the maximum, and instructions to sell are sent when the stock's price falls 1% below the minimum. This 1% difference is significant, as it avoids regular buy/sell signals if a price fluctuates around a maximum, and was shown to work well in previous investigations (Brock, Lakonishok & LeBaron, 1992).

Bollinger Bands

Bollinger Bands are used to measure a stock's price volatility and then provide insights as to whether or not the stock should be bought, sold, or held. The buy/sell signal depends on three different lines, the first and simplest of which is a variable moving average, as presented earlier. Oftentimes, the moving average is set for the past 20 days. Two other bands are drawn around the moving average, the first being a multiple of the standard deviation added to the middle band, and the other being a multiple of the standard deviation subtracted, as shown below:

$$M \pm d\sqrt{\frac{\sum_{i}^{k}(p_i - M)^2}{n}}$$

where M is the moving average, p_i is the closing price (or in the project's case, the adjusted close), and n is the number of days of the moving average. d is the number of standard deviations that one would like to use. For the data being analyzed, $d = 2$, as recommended by Technical Analysis from A to Z (Siegel, 2001).

While there are many ways to interpret the information given by Bollinger Bands, the information supplied for the neural networks follows a pattern similar to the TRB: if a stock's price dips below standard deviations of the moving average, a buy signal is sent, while a sell signal is sent if the stock price rises two standard deviations above the moving average.

Money Flow Index (MFI)

MFI is the only technical variable within this project that incorporates volume. The MFI analysis how much money is being spent on purchasing securities and how much is made by selling them, and depending on the ratio between the two, an index is created. Calculations begin by creating a Money Flow (MF) variable:

$$Money\ Flow = \frac{p_{high} + p_{low} + p_{close}}{3} * Volume$$

The MF is different from the MFI. In this case, one adds all the days and prices thereof where a security's price increased to make the positive MF, while all the negative days' prices are added to get a negative money flow. This allows one to calculate the Money Ratio (MR) which is then used to calculate the MFI.

$$MR = \frac{MF_{positive}}{MF_{negative}}$$

$$MFI = 100 - \frac{100}{1 + MR}$$

If the MFI dips below 20, a buy signal is created, while an MFI rising over 80 generates a sell signal (Siegel, 2001).

Temporal Data

Data on the month of the year and day of the week is also included in the data set, as some stock analysts have noted fluctuations in security prices based on the day of the week and month of the year (Dimson, 1988). Two notable effects are called the "Monday Effect", where stocks tend to follow the patterns exhibited on Fridays, and the "January Effect", when stock prices are more likely to rise in this month than other months.

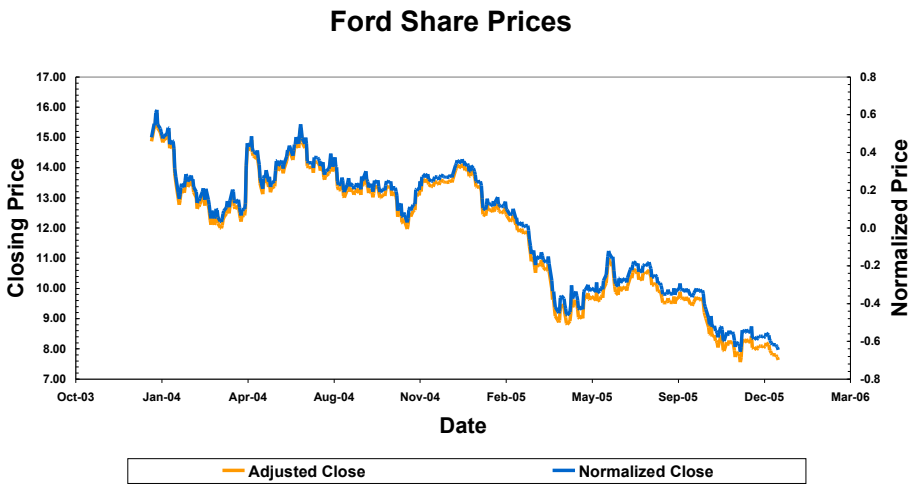
Due to the nature of the stocks being analyzed, this data can be considered both technical and fundamental – while the temporal patterns above are technical, because the stocks being analyzed tend to represent goods and services that are often perceived as luxuries and exhibit cyclical patterns, consumption decisions made by households (for example, to go to luxury hotels more often in the summer than winter) may affect certain stock prices. Since no assumptions about such events are made in the data and because the data changes on a daily basis while all other fundamental variables do not, this was treated as a technical input.

Prices

Finally, prices were still included in the inputs. In this case, prices were normalized as follows:

$$p_{norm} = \frac{p_{original} - p_{average}}{3d}$$

where p_{average} represents the average price for the entire data set and d is the standard deviation of the entire data set. Such a system puts virtually all prices into a range between -1 and 1, and allows the graph to retain its former shape, as shown below for Ford shares.



Fundamental Analysis of Stocks

Unlike its technical counterpart, fundamental analysis posits that companies that do well in their line of work, be it by having high profits, a good managerial structure, a successful focus on research and innovation, or any other similar factors, will do well in the stock market. Many argue that this type of analysis is much more reliable than technical analysis for longer term investments, as stocks are supposed to reflect the value of the company they represent.

The stocks within this project were chosen based on their potential responses to fundamental variables within the macroeconomy. The “Consumer Discretionary” (Morgan Stanley Capital International Inc., 2005) categorization of stocks, from which the list of chosen stocks comes from, is known for its responsiveness to business cycles, including ones that affect the entire economy. This makes it possible to find patterns within the stocks based on how certain economic indicators and variables are doing.

There are two caveats to be mentioned with regards to using the variables above, or any others available to one using neural networks for applications within the stock market. First and foremost, many fundamental variables are released long past the period they represent. Even something as simple as a variable representing whether an economy is in a recession could lag by months, if not years: the recession of 1990 to 1991, for example, was only announced by the National Bureau of Economic Research (NBER) after it was over (Kacapyr, 1996). Similarly, aside from the leading indicators, the variables chosen for this project may come long after the period which they discuss.

Secondly, while stock prices change on a regular basis, the same is not true for fundamental variables. Many of the variables below are released monthly, though corporation earnings information is released quarterly – incorporating data that appears from numerous time periods within a neural network is difficult, especially when one has access to a nearly-continuous stream of information, like share prices.

As with technical variables and indicators, there are numerous options for analysts, and one must be extremely careful in choosing them. The indicators were chosen with this in mind, and are listed and explained below.

Consumer Price Index (CPI)

There are multiple ways of measuring inflation within an economy, and one of the most widely used is the CPI. There are two reasons why the CPI was chosen as an input for the neural network. Changing prices can often affect how consumers spend their money, and how well companies do. This is especially true for companies that deal directly with consumers, such as those selling products. The companies in the Consumer Discretionary category all deal with goods that are seen as luxuries rather than necessities, and thus may be sensitive to price fluctuations – if people have less money to spend, luxuries are often the types of goods that are avoided.

The CPI is measured by taking a fixed basket of goods that consumers usually buy (household goods, food, and so on) and comparing the price of these goods during the current year or month to a base value, normalized to a value of 100. Thus, if the CPI is 150 today, it means prices rose by 50% since the base year (Mankiw & Scarth, 2004). All of the data being used is based on the US economy, and was acquired from the Bureau of Labor Statistics (n.d.).

Consumer Sentiment Index

The University of Michigan (2006) conducts a monthly survey of consumers that gauges a number of different variables, and based on these variables, the University compiles an index allowing one to see the relative changes in consumer sentiment each year. Again, due to the nature of the Consumer Discretionary category of companies, how consumers feel about their future finances, employment, and other economic and financial factors may affect what they spend money on, and thus, how well the companies do.

The index is based on a number of variables from the university's surveys, and includes

(University of Michigan, 2006):

- Current Personal Finances
- Expected Personal Finances
- Expected Business Conditions in 5 Months
- Expected Business Conditions in 12 Months
- Buying Conditions

Consumer Confidence Index

Similar to the CSI, the Consumer Confidence Index (Conference Board, 2006a) is based on a monthly survey of 5000 representative households in the United States, and is conducted by the Conference Board of the United States. The survey is meant to learn how consumers see the current and future economic and financial situation, and how they feel it will affect them personally. This is very similar to the CSI, though the historical values do differ, and there are times when one value rises and another falls.

New Orders Diffusion Index

The Institute for Supply Management (ISM) is a an international association composed of businesses working for the "identification, acquisition, access, positioning and management of resources that an organization needs or potentially needs in the attainment of its strategic objectives." (Institute for Supply Management, n.d.). Specifically, this includes manufacturing, transportation, and inventory management. The ISM's New Orders Diffusion Index is an indicator that measures the level of new inventory orders from businesses, and how dispersed these new orders are between industries. This index can be used as an indicator of supplier sentiment and expectations – businesses order new goods only if they expect to sell them, and the higher the diffusion of new orders, the more likely it is that suppliers within the entire economy have a positive outlook.

Leading Indicators Index

Indicators are economic variables that are correlated with the general state of the economy, and come in three types: lagging, coincident, and leading (Granger, 1989). Lagging indicators are variables that shift in a similar direction as the economy, but do so after the economy has experienced the

change, while coincident indicators do so at the same time. Leading indicators are extremely useful for forecasting, because they often experience changes before the economy as a whole experiences similar shifts. This makes them very useful for predicting how the economy will look in the near future.

An example of a leading indicator is the average workweek in the United States (Glosser & Golden, 1997). In this case, when employers begin to experience higher demand for products, they will have their employees work longer hours to meet production demand. This increases the average workweek, and the economy only experiences an increase in activity once the employees are finished working and the products have been shipped. The ISM Orders Diffusion Index may be a leading indicator for a similar reason.

This index is produced by the the Conference Board and is comprised of ten leading indicators (Conference Board, 2006b):

- average weekly initial claims for unemployment insurance (inverted)
- average weekly manufacturing hours
- building permits
- index of consumer expectations
- interest rate spread
- manufacturers' new orders for consumer goods and materials
- manufacturers' new orders for non-defense capital goods
- real money supply
- stock prices
- vendor performance

Interest Rate of the 30-Year Conventional Mortgage

The Federal Reserve publishes weekly statistics on the interest rates given for fixed-rate first mortgages (Federal Reserve Board, 2006b). Such interest rates often affect major buying decisions within families, especially for things such as new homes. It is assumed that this interest rate also affects smaller purchases indirectly, by affecting buying habits with regards to automobiles and other goods.

Federal Funds Rate

This is another interest rate, but is charged by the Federal Reserve to banks and other lending

institutions when they need money to balance their own reserves and accounts. The Federal Reserve uses this rate to promote price stability and sustainable economic growth (Federal Reserve Board, 2006a), and often, this rate influences consumers using banks and credit.

Price-Earnings (P/E) Ratio

The only stock specific fundamental variable in this data set, the P/E Ratio for a stock is the value of all shares being traded divided by a company's annual earnings. Mathematically,

$$PE\ Ratio = \frac{Share\ Price * Number\ of\ Shares}{Annual\ Earnings}$$

The generally accepted rule in investing is that as the PE ratio grows, the likelihood that a stock's price will fall increases. While there are numerous factors affecting a stock's price and a low PE ratio does not necessarily mean one should buy shares of a specific stock, it does play a role in many investors' decisions.

Target Stocks

There are numerous options available to those interested in researching neural networks and their applications to the stock market. For this research, an initial set of 18 stocks were chosen for analysis, and the choice was based on a number of factors. The method to choosing the stocks began by observing the S&P 500. This index is used as a standard gauge for models and predictions, and the index contains some of the largest and well-reputed companies on the market.

The S&P 500, as well as many other stocks, uses the Global Industry Classification Standard (GICS) (Morgan Stanley Capital International Inc., 2006) to classify its stocks into specific categories, industries, and other sectors. While one can focus on a large amount of diverse stocks, this project has chosen to focus on the subset labeled “Consumer Discretionary”. According to Morgan Stanley Capital International Inc.,

> *“The GICS Consumer Discretionary Sector encompasses those industries that tend to be the most sensitive to economic cycles. Its manufacturing segment includes automotive, household durable goods, textiles & apparel and leisure equipment. The services segment includes hotels, restaurants and other leisure facilities, media production and services, and consumer retailing and services.”*

This category was judged to be the most promising for predictions. If the economy or stock market as a whole is exhibiting an cycles or patterns, this category of stocks is most likely to show it, assuming that such cycles would be reflected in share prices. Furthermore, fundamental variables often depict business cycles and may even be used to predict them, making them more useful than more stable and unchanging categories.

There are, however, more than 18 stocks in the S&P 500 falling into this category. The list of stocks was further narrowed to ensure quality of data and to ensure that stocks are traded often and by large amounts of people, to even out any discrepancies that may be caused in the short term. As such, stocks with a high percentage (10% or more) of insider owners, or with more than 95% ownership by institutional investors, were not included. As were stocks that split after 2000, when the data is collected. While an adjusted closing prices are used in most data to ensure dividend payments do not distort predictions, splits have a much larger effect (often halving the stock price) and were convenient for narrowing the list of stocks.

Finally, stocks with an average volume of less than 1 million shares, and with a difference between their 52-week high/low prices greater than 80% and less than 20% were also not included to ensure high volumes of trading and a larger potential for technical indicators to appear based on groups of investors acting in unison. Stocks that were relatively constant in price and highly volatile were ignored to avoid any distortions they may introduce into the neural networks picking up patterns. Below is a list of the final 18 stocks:

AutoNation, Inc.	Hilton Hotels	RadioShack Corp
Circuit City Group	Home Depot	Sherwin-Williams
Clear Channel Communications	Mattel, Inc.	Starwood Hotels & Resorts
Ford Motor	McDonald's Corp.	Tribune Co.
Gannett Co.	New York Times Cl. A	Walt Disney Co.
Goodyear Tire & Rubber	Newell Rubbermaid Co.	Wendy's International

While it would have been ideal to focus on all 18 stocks, due to a lack of time (the project lasted about three months), it was not possible to test networks on all of these stocks. By chance, the stocks that were focused on were Circuit City and Ford, with a much greater emphasis on the former. It is recommended that further research be taken to explore how other stocks function in these research settings, though one should stick to the 18 chosen, or at most, deviate only within the Consumer Discretionary category.

Part 2: Neural Networks

Neurons and Neural Networks

Like the brain, the basic building block of any neural network is the neuron. From a biological perspective, a neuron is “a cell in the brain whose principal function is the collection, processing, and dissemination of electrical signals.” (Russell & Norvig, 2003, pg. 736) Its function in neural networks is very similar. Like biological neurons, artificial neurons are connected to other neurons – they can accept information from others, as well as send it to them. Rather than receiving electrical signals, artificial neurons receive a number from other neurons, and process these numbers accordingly.

Each artificial neuron is fundamentally the same. In and of themselves, neurons can perform simple calculations based on the inputs they receive. What is unique and most powerful about them, however, is their ability to perform calculations collectively: by combining groups of neurons, one can perform extremely complex operations, even while each neuron acts independently of others. For this reason, neural networks have often been described as “parallel distributed process” (PDP) networks. Such networks are unique because rather than being set up to solve a specific problem, the networks are trained to solve problems by modifying how they receive information from other neurons.

The main drawback of such networks is their “black box” quality: after a network is trained, it may be extremely difficult to interpret its state to learn which inputs and qualities are important to the solution received. However, analysis of weights and networks in the past has been successful in learning the relationships between inputs and outputs of a neural network (Yao, Teng, Poh & Tan, 1998; Dawson, 2004). This is exciting in the context of stock markets, because if a neural network is trained to predict stock price fluctuations, there is a chance – albeit a small one – that the network's state could be analyzed to learn which factors are useful in forecasting changes in stock prices.

Neurons are all fundamentally the same, though may differ in specific details, such as connections and functions used to analyze data. When neurons are combined in such a way, they often take one of three roles: input, hidden, and output neurons. Input neurons are ones that are fed data from an external source, such as a file, rather than other neurons. Hidden neurons accept their information form other neurons' outputs. Output neurons are like hidden neurons, but rather than passing the processed information to a new set of neurons, they save the information and allow it to be read and interpreted by an external actor.

For explanatory purposes, a neuron may be broken down into three parts:

- input connections
- summing and activation functions
- output connections

Input Connections

Unless the artificial neuron is an input neuron, a neuron is connected to other neurons and depends on them to receive the information that it processes. There is no limit to the amount of connections a neuron may receive information from.

The information that a neuron receives from others is regulated through the use of weights. When a neuron receives information from other neurons, each piece of information is multiplied by a weight with a value between -1 and 1, which allows the neuron to judge how important the information it receives from its input neurons is. These weights are integral to the way a network works and is trained: specifically, training a network means modifying all the weights regulating information flow to ensure outputs are correct.

Summing and Activation Functions

The second portion of a neuron is the summing and activation functions. The information sent to the neuron and multiplied by corresponding weights is added together and used as a parameter within an activation function. In a biological context, a neuron becomes activated when it detects electrical signals from the neurons it is connected to (O'Reilly & Munakata, 2000). If these signals are sufficient, the neuron will become “activated” - it will send electrical signals to the neurons connected to it. An activation function is similar: the artificial neuron will output a value based on these inputs. It is almost always the case that a neuron will output a value between [0, 1] or [-1, 1], and this normalization of data occurs by using the summed inputs as a parameter to a normalizing function, called an “activation function”.

Numerous activation functions exist, but within this project, three types of activation functions were explored:

- *Threshold Function*: a simple function that compares the summed inputs to a constant, and depending on the result, may return a -1, 0, or 1. Specifically, for summed input

$$f(x)=\begin{Bmatrix} 1 & if\ x>0 \\ 0 & if\ x=0 \\ -1 & if\ x<0 \end{Bmatrix}$$

- *Piecewise-Linear Function*: if the summed inputs are in the range [-0.5, 0.5], the value is kept as before, while anything else will return a -1 or 1:

$$f(x)=\begin{Bmatrix} 1 & if\ x\geq\frac{1}{2} \\ 0 & if\ -\frac{1}{2}<x<\frac{1}{2}=0 \\ -1 & if\ x\leq-\frac{1}{2} \end{Bmatrix}$$

- *Hyperbolic Tangent Function*: a continuous function with a domain of (-∞,∞) and a range of (-1, 1). Specifically (Weisstein, 1999),

$$\tanh(x)=\frac{e^{2x}-1}{e^{2x}+1}$$

Within this project, the tanh(x) function is most used. By providing a function with a limitless domain and a range of (-1, 1), it is perfect for predicting whether or not a stock will rise (tanh(x) = 1) or fall (tanh(x) = -1). It should also be noted that the above functions, while having ranges of [-1, 1] also have corresponding versions with ranges [0,1]. For example, the sigmoid function, below, is a replacement for the hyperbolic tangent:

$$f(x)=\frac{1}{1+e^{-ax}}$$

where *a* allows one to set the slope of the function.

Output Connections

Finally, once the activation function returns a corresponding value for the summed inputs, these values are sent to the neurons that treat the current neuron as an input. The process repeats again, with the current neuron's output being summed with others, and more activation functions accepting the sum of these inputs. The only time this may be ignored is if the current neuron is an output neuron. In this case, the summed inputs and normalized sum is sent as an output and not processed again.

Feedforward Networks

While each neuron is, in and of itself, a computational unit, neurons may be combined into layers to create complex and efficient groups that can learn to distinguish between patterns within a set of given inputs. Indeed, by combining multiple layers of such groups, it is theoretically possible to learn any pattern.

There are many combinations of neurons that allow one to create different types of neural networks, but the simplest type is a single-layer feedforward network. In this case, a network is composed of three parts: a layer of input nodes, a layer of hidden neurons, and a layer of output nodes. Within this network, there is a neuron for each input variable in an input pattern, which is then propagated to the layer of hidden neurons. As mentioned earlier, each neuron in the hidden layer has the inputs multiplied by a weight, and all inputs are summed. After this is done, the value is passed to an activation function, and each neuron in the hidden layer then passes the output on to neurons in an output layer, which also multiply the values by weights and sum them.

A multilayer feedforward network is similar to a single-layer one. The main difference is that instead of having a hidden layer pass its calculated values to an output layer, it passes them on to another hidden layer.

Both types of networks are typically implemented by fully connecting each layer's neurons with the preceding layer's neurons. Thus, if *Layer A* has k neurons and sends its information to *Layer B*, with n neurons, each neuron in *Layer A* has n connections for its calculated output, while each neuron in *Layer B* has k input connections.

Interestingly, such a network can be represented mathematically in a simple manner. Supposing there are k neurons in *Layer A*, let $\underline{a}$ represent a vector, where a_i is the i^{th} neuron's activation function output. Let $\underline{b}$ represent the input values to neurons in *Layer B*, with b_j be the j^{th} neuron. Let $\mathbf{W}$ be a n by k matrix where w_{ji} represents the weight affecting the connection from a_i to b_j. Keeping this in mind, we can see that for a single-layer feedforward network, we can mathematically represent the flow of information by,

$$\mathbf{W}\underline{a} = \underline{b}$$

and the learning thus becomes a modification of each w_{ji} in $\mathbf{W}$. A similar mathematical analogy applies to multilayer feedforward networks, but in this case, there is a $\mathbf{W}$ for every layer and $\underline{b}$ is used as the value for $\underline{a}$ when moving to subsequent layers.

The most popular type of learning within a single-layer feedforward network is the Delta Rule, while multilayer feedforward networks implement the Backpropagation algorithm, which is a generalization of the Delta Rule.

Backpropagation

Both the feedforward and recurrent networks in this project implement the backpropagation algorithm, one of the most common learning methods within neural networks. While widely used, the algorithm has its pros and cons, and it helps to understand how it works.

The backpropagation algorithm is useful in that it generalizes the Delta Rule for single-layer feedforward neural networks (O'Reilly & Munakata, 2000) to multilayer feedforward ones. One of the biggest problems in training neural networks is the Credit Assignment Problem (Haykin, 1999): how does one assign credit to individual neurons for errors in the final outputs of a neural network?

The Delta Rule is one solution to this problem, but only applies to single-layer feedforward networks. While its derivation is beyond the scope of this project, the Delta Rule may be summarized with the following equation:

$$\Delta w_{ij} = -\epsilon \delta_j x_i$$

In this case, Δw_{ij} represents the change of the weight connecting the i^{th} neuron with the j^{th} output neuron, x_i is the output value of the i^{th} neuron, ε is the learning rate, and δ_j is the error term in the output layer, defined as:

$$\delta_k = -(t_k - o_k)$$

where t_k is the expected output, while o_k is the actual output.

While this rule works well when there is only one hidden layer, credit assignment becomes much more difficult with multiple hidden layers. For this, one must use a new equation:

$$\delta_j = (\sum_k \delta_k w_{jk})(h_j(1 - h_j))$$

In this case, one uses the same equation for Δw_{ij} but uses the term above instead, with k representing the neurons receiving information from the current neuron being modified. $\delta_k w_{jk}$ is the error term of the k^{th} neuron in the receiving layer, with w_{jk} being the connecting weight. The activation functions of all the neurons in a network implementing backpropagation must be differentiable, because:

$$h_j = \sigma'(z_j)$$

with z_j being the net input for the neuron.

Finally, if biases are present, they are treated like regular neurons, but with their x values equal to 1:

$$\Delta B_j = -\epsilon \delta_j$$

When implemented, this algorithm has two phases. The first deals with having the network evaluate the inputs with its current weights. Once this is done, and all the neuron and output values are recorded, phase two begins. The algorithm begins this phase by applying the original Delta Rule to the output neurons, modifying weights as necessary. Then the generalized Delta Rule is implemented, with the previous δ values sent to hidden neurons, and their weights changing as required.

Drawbacks

Essentially, backpropagation is a learning algorithm that allows a network to find a state that minimizes the amount of error the network exhibits (Churchland & Sejnowski, 1994). It does this by modifying weights connecting neurons using a gradient descent (Arbib, 2003), which allows the weights to be modified in a manner following a negative slope along the "error landscape" – a n-dimensional surface representing all possible errors within the network.

The process, however, has a few drawbacks. First and foremost, backpropagation does not guarantee to find the global network minimum. While it does minimize error, there is a chance the weights will be changed to fit a local minimum in the error landscape, but the network will not be optimized.

With the above in mind, the learning coefficient (ε) must be chosen with great care – one too small will force the network to take an excruciatingly long time in optimizing itself, while a large one will force a network to oscillate around a minimum value. Another potential solution is using a *momentum term* α (Haykin, 1999). Such a term can allow the delta and generalized delta rules to modify weights at different magnitudes, rather than have them depend on a constant learning parameter.

This drawback is especially important for this project because of the complexity of the data being dealt with. Haykin writes that for complex networks modeling real world phenomena, "It is generally agreed that it is inadvisable for a multilayer perceptron to be fully connected." (1999, pg. 233) This proved to be a challenging issue in this project, mainly because the number of weights in complex, fully-connected networks was often much greater than that of the total number of training patterns.

Backpropagation Applied

Within the research project, backpropagation was used within all the feedforward networks. The algorithm is extremely effective at learning patterns, and is proficient in adapting to various types of data. The Java Neural Network Simulator (JNNS) allows for the creation of neural networks implementing backpropagation, and allows one to modify a number of settings, including (Zell, Mamier, Vogt, Mache, Hübner & Döring, n.d.):

- η: the learning rate (ε above).
- δ_{max}:the maximum error rate backpropagated in the network.
- cycles: the number of times JNNS should run through the training data.
- steps: the number of times each input should be propagated through the network during a cycle. For networks implementing backpropagation, 1 step is sufficient.
- shuffle: asks whether or not each input pattern should be shuffled. If unchecked, the inputs are sent to the network in the same order each time.

The various options above give one an unlimited amount of choice in setting up a network, and choosing the settings must be done with care: incorrect settings can stop a network from converging on the training data and patterns held therein.

Within this project, the learning parameter takes values of 0.01, 0.003, and 0.0005. Initially, values were set higher, near 0.1, but it appears that the networks using such a learning rate were ineffective at learning input patterns, and the value was abandoned to conserve computing time and focus resources on more accurate settings.

Due to limitations with computing, cycles were often kept at 25000 and lower. Again, almost all networks being analyzed within this project were tested at 25000, 10000, 5000, 1000, and 500 cycles. In the examples presented below, only the best examples are shown, but statistics and information on all such trained networks is available.

The learning rate and number of cycles were the only two settings that were experimented with. Feedforward networks are not dependent on previous or forthcoming input patterns and as such, the set was shuffled to avoid overtraining of the network to the specific order being presented. Furthermore, the number of steps was kept constant to avoid any discrepancies in the data. The number was kept at 100.

δ_{max} was kept at 0.0 at all times as well. Due to the nature of backpropagation, it is important to

keep the outputs in a network continuous to allow for the error rate between outputs and correct patterns to get smaller. As such, predictions from the neural networks range from -1 to 1, and are often in the middle. While such predictions allow the network to exhibit confidence in predictions (i.e. a prediction of 1.0 is more likely to be correct than one of 0.9), this confidence is relative, and it is impossible for one to know whether an output of 0.65 means the stock is likely to rise, or if it means the network is unsure. By setting δ_{max} to a value greater than 0, one makes an assumption that a prediction other than 1 or -1 can still be correct.

Such an assumption, however, is not necessarily incorrect. Rather than setting δ_{max} from JNNS, however, an analysis was undertaken of predictions to see how accurate predictions are at various threshold values.

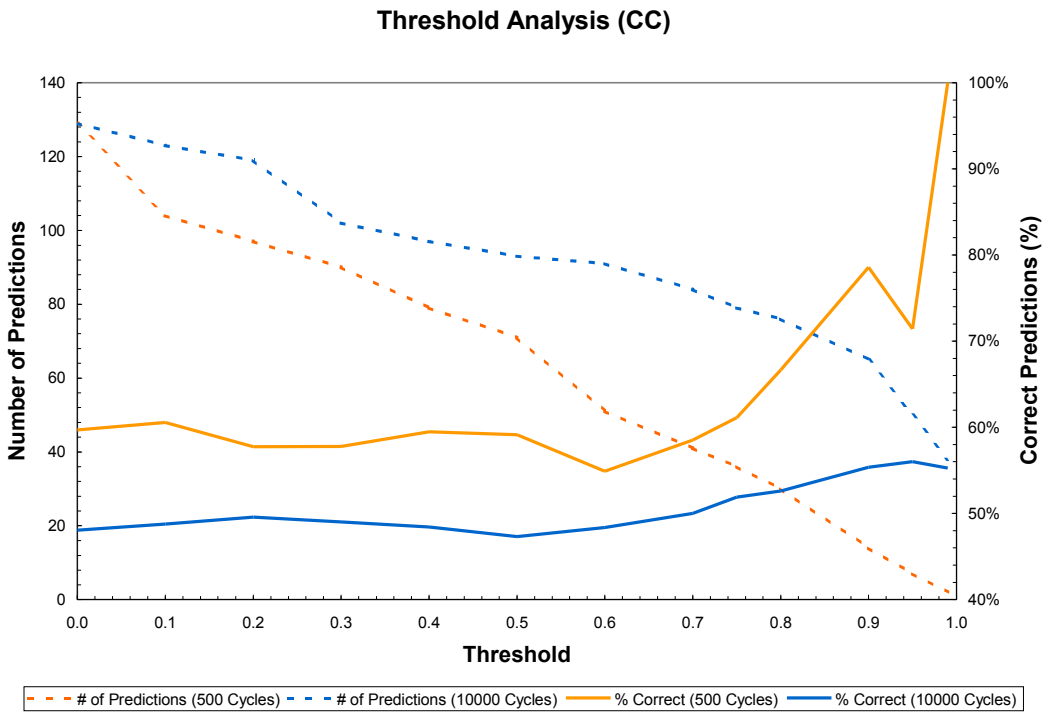
Threshold Analysis

By leaving δ_{max} equal to 0, JNNS keeps training a network until all neurons make correct predictions of 1 or -1 for all inputs, or until the number of training cycles is complete. Due to the amount of training data (over 1,000 inputs in all cases), the network never converged on the training data. As such, many predictions for future prices were not equal to ±1 but were somewhere in between. It was assumed that the closer a prediction was to ±1, the more sure the network was that the prediction was correct. Again, this was purely relative: a value of 0.65 was more accurate than 0.90, but one cannot be sure how much more accurate.

To test whether or not this was true, a threshold analysis was performed on predictions to see how accurate they get. A threshold, ϕ, was selected, and the following function applied:

$$f(x)=\begin{Bmatrix} -1 & if\ x<-\phi \\ 1 & if\ x>\phi \\ 0 & otherwise \end{Bmatrix}$$

with -1 meaning the stock will fall, and 1 meaning the stock will rise. A number of different thresholds were chosen, and the graph below shows the threshold analysis for Circuit City (CC).



In the case above, it is clear that as the ϕ approaches 1, the accuracy of the network increases, while the number of predictions decreases. It is clear that by having setting $\phi = 0.9$, one can have a meaningful number of accurate predictions. From the above chart, it was decided that $\phi = 0.75$ was an acceptable threshold value.

Recurrent Networks

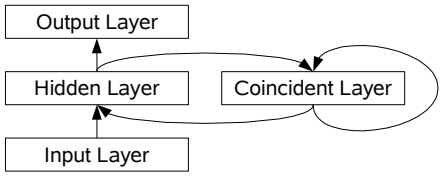
While feedforward networks are extremely efficient at learning static patterns, it is difficult to have such a network learn patterns that change over time. The most important reason for this is that feedforward networks do not take into account earlier input patterns within their analysis of a pattern presented to them. To deal with this issue, feedforward networks were presented with a few days' worth of historical data. However, this approach is limited to the number of input patterns one presents, and also drastically increases the amount of inputs.

A potential solution to this problem is the use of recurrent networks. Recurrent neural networks are networks with neurons that have feedback connections – connections to themselves, or send their outputs to a previous layer. These feedback connections can be a global or local type. For example, a neuron may send its output to itself, allowing it to incorporate this information in the next pattern's analysis, or the output layer may send its final answer as an input for the next pattern.

Elman Networks

There are numerous recurrent network architectures, but one of the simplest is the Elman network. Such networks are based on feedforward networks in that they have a typical input layer and hidden layers sending each other information in a sequential order. The main difference, however, is that every hidden layer sends its outputs to a coincident layer, where the information is stored and sent back to the hidden layer during the next pass-through in the network. Every neuron in this coincident layer (within SNNS) has a feedback loop to itself as well.

Such a setup allows Elman networks to retain information for longer than one pass-through in the network. After the first pass, subsequent passes will have information based on all earlier attempts, and it is assumed that this will allow the network to incorporate previous information into its own predictions.



Implementing such a network within JNNS is difficult, but its predecessor, the Stuttgart Neural Network Simulator (SNNS), contains a tool that builds an Elman network based on a list of variables, including number of inputs, outputs, and number of hidden networks. Training takes place through the use of a modified version of backpropagation that applies the generalized delta rule to coincident layers. Furthermore, all training function parameters are as in the case for backpropagation, but with one addition, a teacher forcing parameter (Zell, Mamier, Vogt, Mache, Hübner & Döring, n.d.). Rather than training based on the output of the network, one can use a weighted average between the expected output and actual output of the network. For example, if the parameter is set at 0.0, the training is based solely on the original output, while 1.0 means the expected output is used to train the network. The default value in SNNS is 0.5.

Part 3: Results

Statistical Analysis

With the two network types listed above and data for Circuit City (CC), Ford (F), and McDonalds (MCD), a total of 19 different trials were run that explored how various types of network architectures and data inputs affected predictions for stock prices within 1 day, 1 week, and 4 week periods. While numerous combinations were employed, all of which are explained in the appendix, it is important to note that this is by no means a comprehensive list. If a researcher constrains him or herself to just Elman and feedforward networks, the number of options he or she has are still extremely large, and impossible to analyze in a few months. However, it is hoped the results below shed light on what type of feedforward or recurrent network may provide the best results for predicting price fluctuations in share prices of specific stocks.

Due to the breadth of this project, a few important notes must be kept in mind when comparing data and results between networks. This project was meant as an academic study into neural networks and stock prices and as such, no attempts were made to develop a potentially profitable trading strategy based on these results. Furthermore, because of the many models explored, the data inputs as well as number of predictions differs between some networks.

Furthermore, longer time periods tend to focus on patterns that reach further into the past, making it difficult to compare short-term results and the actual numbers of predictions. This is especially true for networks solely using fundamental indicators: due to the constraints of the data (focusing between 2000 and 2005), training sets were built with only 55 training units and 10 validation patterns, while other sets had over 120 validation patterns and 1100 training ones.

A network's "correctness" is measured by the number of correct predictions above the threshold ϕ, as discussed earlier. It is assumed that when a value is below ϕ, the network is unsure of the data and does not make a reliable prediction, so those were ignored. As such, the percentage accuracies discussed later are:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions\ with\ Score\ Above\ \phi}{Number\ of\ Total\ Predictions\ with\ Score\ Above\ \phi}$$

For an individual network, the accuracy above is compared to two other values. The first of these is 50%, which signifies the accuracy of a perfectly random prediction system.

The second value θ, is based on the Bernoulli Distribution. Because every stock price either goes up or down in a day, a stock price can be treated as a variable $S \sim \text{Bernoulli}(\theta)$ that takes a value of -1 or 1. Based on the training data, the best estimate for θ, representing the probability a stock price will rise, is:

$$\theta = \frac{Number\ of\ Days\ A\ Stock\ Price\ Increased}{Total\ Number\ of\ Days}$$

In this case θ also represents the percentage that a stock price rises. This is important for comparisons, because if one uses a completely random predictor based on $S \sim \text{Bernoulli}(\theta)$, one can get a better result than an uninformed guess while still depending on a random variable.

The accuracy of individual networks was thus compared to both θ and 0.5, while groups of networks were compared to to the two values using t-tests. It is important to note that depending on the trial, θ took on different values.

One caveat remains in making comparisons. While a network may be accurate 100% of the time, due to the definitions above, this may not be significant. A number of networks (notably Trial 14), had extremely few predictions – sometimes as little as 2 or 3. Due to this, a network could have a 100% accuracy rate without being useful. It is important to check the number of total predictions if one sees such high (or low) rates of accuracy.

Results

The results of the various trials and networks tested are very promising, with a number of networks showing very good results. The analysis took place through two different groups – by focusing on the various trials and making generalizations about the types of networks therein, and by comparing specific networks. In all cases, a threshold value of 0.75 was used.

While numerous groups of networks were tested, the groups below show promising results. In this case, $p_{0.5}$ represents the p-value for the group of networks when testing for average accuracy of 0.5, while the p_θ is the p-value for when testing if accuracy is equal to θ. Detailed information about each trial and the meaning of the name may be found in the appendix.

Trial Name	Avg. Accuracy	$p_{0.5}$	θ	p_{θ}	Forecast*
Trial 3, n = 0.0005	55.20%	0.0324	0.5039	0.0419	
Trial 5, n = 0.0005	56.01%	0.0096	0.5078	0.0153	**1 day**
Trial 15, n = 0.01	61.06%	0.4110	0.5465	0.6110	
Trial 15, n = 0.003	60.39%	0.1928	0.5465	0.3278	
Trial 11, n = 0.01	54.93%	0.1356	0.6512	0.0144	**5 days**
Trial 12, n = 0.01	82.70%	0.0005	0.7000	0.0165	
Trial 12, n = 0.003	80.33%	0.0010	0.7000	0.0423	
Trial 13, n = 0.01	61.87%	0.0001	0.7442	0.0000	**20 days**
Trial 13, n = 0.003	57.85%	0.0299	0.7442	0.0037	
Trial 17, n = 0.003	57.13%	0.1385	0.7442	0.0166	

** business days*

Feedforward Networks and Types of Stocks

The results above are interesting for a number of reasons. First and foremost, Trial 3 represents a simple one-layer feedforward network trained with a learning parameter of 0.0005. It was noticed within many of the feedforward networks that as the learning parameter decreases, accurate results increase. Whether this is because the network avoids overtraining or because the data set (and stocks) were extremely sensitive to information flow was undetermined. Trial 3 focused on Circuit City (CC), while Trial 4, not shown above, focused on F. Interestingly, F was statistically significant for how poor its results were. From this, one can gather that some stocks are more inclined to follow predicted paths than others. For example, F's volume was 10 times as high as that of CC, and CC has twice as many institutional investors as F, in percentage terms. CC was also more volatile than F. This, along with the types of products (consumer retail versus automobiles) and investor psychology could all play a role in how predictable the stocks are.

What is interesting, however, is that while F did not work well with neural networks and CC did, Trial 5 was once again statistically significant. Trial 5 represents nothing more than a combination of the training and validation data of Trials 3 and 4. Doing so doubled the amount of information presented to each network without changing the network's structure, making overtraining less likely while also presenting more patterns to train the network.

With more time, it would be interesting to see how incorporating a larger group of stocks could help promote network convergence. While the “Consumer Discretionary” category of stocks was used, these also have subcategories – CC, for example, is in the Retailing category, and presenting a network

with a number of such stocks could allow it to learn general and industry-specific patterns.

An interesting observation with the data above also relates to the number of cycles required to obtain a good rate of training. While Trial 3 required 100000 cycles, Trial 5 required 1000. This may be a result of the randomness of weight initializations. Some work, presented in Appendix C, was done in this regard, and shows that the use of random weights can indeed skew results. The best solution to this problem is to run multiple training sessions on the same networks and see where most of them converge. However, this could not be done in this project due to a lack of time.

Other Network Architectures for 1-Day Predictions

What is significant about the 1-day predictions is that two-layer or three-layer feedforward networks are not present. While additional layers can help a network converge, they may also require more training and with the cycles and limited multilayer architectures used, useful results were not obtained. When increasing the number of layers, however, the number of nodes per layer was decreased to avoid overtraining. With more data, such a concern and limitation may not be necessary.

Recurrent networks were used for 1-day predictions as well, and proved to be interesting in how they work. A one hidden layer Elman network was constructed, where only the previous day's information was sent to the network. Due to the network's recurrent nature, it should theoretically take into account all previous data sets within its predictions. This network, however, did poorly.

A second feedforward / Elman hybrid was constructed, where inputs were provided for five days, as in the feedforward network, but the network also had a coincident hidden layer where previous hidden layer outputs were sent. This network proved to be most accurate, with an average accuracy rate higher than that of the feedforward networks. It should be noted however, that the validation sets were different because the Elman networks were recurrent and could not be easily tested on data at the middle of the training set, while feedforward networks could.

Prices

While most networks received inputs in the form of technical or fundamental indicators, Trial 19 was a unique trial in that it tested whether or not a network could make predictions after receiving nothing more than information on a stock's past prices. A single-layer feedforward network was presented with 50 days of historical prices and was trained to make predictions for 1, 5, and 20 business

days. For each time frame, a network was trained at both 25000 and 100000 cycles. For 1 and 5 day periods, the results were very poor, with networks making few, if any predictions over a threshold of 0.75. What is interesting, however, is the network's ability to make predictions for 20 days. At 100000 cycles, the network always predicts the stock will go up, which does happen the majority of the time, making the network correct 71.32% of the time. When trained at 25000 cycles, it makes 25 predictions, all of which predict upward movement, and the network is correct 96.00% of the time. While more study is warranted to judge whether such results are significant, it is possible that the network is generalizing its predictions to what the stock's long-term growth pattern appears to be – an upward direction.

Longer Term Predictions and Fundamental Analysis

While predicting for 1-day periods was met with some success when it came to the use of technical analysis, longer-term predictions did not seem to work so well. Indeed, there was no adequate network for 5-day predictions, and the one presented in the graph is statistically significant for its lack of accuracy. While it correctly predicted almost 55% of all attempts, a network just predicting an upward increase in stock prices every day would get better results. This is because a large portion of the validation data focused on a region of CC that grew regularly. It should be of some significance, however, that the network does exhibit some generalization.

While technical neural networks did poorly for 5-day and 20-day periods, fundamental-based networks in this period did extremely well, and were the only ones that had statistically significant average correctness rates. This was very surprising, as many discussions before this research focused on the idea that not only are fundamental neural networks difficult to implement, but are likely to fail. Within the table presented earlier, one should note that Trials 11, 13, and 17 are significantly above average for 50% accuracy, but below average compared to a network predicting a rise in stock prices all the time.

With this in mind, a neural network combining both technical and fundamental inputs was created. The results (Trial 17), while better than most attempts, were not as good as the fundamental networks. It should be noted that these networks were extremely large, with 360 inputs, and as such, a lack of computing ability severely detracted from efforts to train the networks. By increasing the number of cycles to 100000 or even higher and combining multiple stocks, such a network may prove to be the most useful in making predictions. Indeed, if nothing else, such a network could learn to

ignore everything but the fundamental variables.

Conclusion and Future Work

With the results above, it is clear that even simple networks like single-layer feedforward nets may be of some use in making predictions about future movements of stock prices. The trick, it seems, may lie in picking the correct stocks: while CC worked well with predictions, initial attempts at making predictions with F did not work well. While little time was spent on analyzing the applicability of various stocks for making predictions, it would be interesting to see a larger scale study – of all 18 chosen stocks, for example – on how well predictions work for various stocks.

One of the initial reasons for this project was to examine how well technical and fundamental analysis works in predicting future price movements. In the short-term of 1 or 5 business days, technical analysis is really the only tool one may implement. Longer time periods, however, show good results when implementing fundamental analysis, though the study could be exposed to a longer data set – all the research here was restricted to the year 2000 and onward.

It was found that combinations of data – be it through combining multiple stocks or multiple s of data – is a good way to strengthen a network. Due to computing limitations, this was not explored to the extent it should, and is an obvious direction for future work. A similar limitation was encountered in exploring the effects of random initial weights on the networks: some of the results showed networks providing useful results after as little as 1000 training cycles, while similar networks required 100000.

A second goal of this project was to compare recurrent and feedforward networks. The results here are interesting. The typical Elman network, providing only one day's historical data and using the coincident hidden layer to keep track of historical information, did not fare well. What is interesting, however, is a modified version of the Elman network, that still received multiple days' worth of historical data while keeping a coincident layer to keep track of longer-term historical information, did very well. It would be useful to examine this network architecture further, and see how well it works in other settings.

Overall, it is clear that neural networks applied to individual stocks can yield statistically significant predictions. The networks explored here are some of the most commonly used within the world of neural networks, but appeared to have good results within the stocks explored. Further research is warranted, and this work may have numerous implications within finance, as well as the theory of efficient markets and analysis thereof.

Appendices

Appendix A: Overview of Trials

This is a record of the various trials that took place during the project. Only the most general outline is given, so one can have an idea of what types of networks were trained, and at what settings. To obtain all the results, please consult the CD available with the paper. Unless otherwise noted, training took place with δ_{max} set at 0.0 and 100 steps per cycle. For all feedforward nets, the inputs were shuffled, while shuffling was not used in recurrent networks. Similarly, almost all trials focused on Circuit City and used solely technical inputs. Any reference to "days" should be noted as "business days".

Trial #1

A single-layer feedforward network with 50 inputs, one hidden layer with 50 neurons, and predicting ahead for one day based on five day historical data. The logistic activation function was used everywhere except the output, which was implementing the sign(x) function. This has a range of {-1, 0, 1} (Weisstein, 1999), making outputs easy to interpret. This format was abandoned because the result was either -1 or 1, which meant it was either right (error of 0) or wrong (error of 2 or -2), making it difficult for backpropagation to converge.

Trial #2

This trial used new activation functions and had the same network type as the first trial. No records were kept – validation patterns were used to ensure that there is some convergence among data, as early results regularly had the same predictions for each input.

Trial #3

Single-layer feedforward network with 50 inputs, one hidden layer with 50 neurons, and predicting ahead for one day based on five days of historical data. All activation functions were using the hyberbolic tangent. Various numbers of learning cycles were used.

Trial #4

Like Trial #3, but applied to stock ticker F.

Trial #5

Like Trial #3, but the data for stock tickers F and CC were combined.

Trial #6

A network exactly like that in Trial #3, but instead of technical indicators, 50 historical days worth of prices were put in and the network was trained at 100,000 and 25,000 cycles. Stock ticker MCD was used. The results were not extremely relevant, especially since MCD was not used in other analyses. See Trial #19 for further analysis.

Trial #7

Untrained one-, two-, and three-layer networks were created for stock ticker CC and validation pattern results saved. Likewise, a one-layer network for stock F was also created. This was used in some initial statistical analysis to see how different these networks are from their trained counterparts, as well as random predictions.

Trial #8

A two-layer feedforward network with 25 neurons in each hidden layer and hyperbolic tangent for its activation functions.

Trial #9

A three-layer feedforward network with 17 neurons in each layer and hyperbolic tangent for its activation functions.

Trial #10

Five-day predictions for a single-layer feedforward neural network with one hidden layer containing 50 neurons and 100 input neurons (10 days of historical data).

Trial #11

Like Trial #10, but the hidden layer had 100 neurons.

Trial #12

1 month (15^{th} of every month) predictions based on fundamental data. A single-layer feedforward network was used, receiving fundamental indicators for the three months prior to the month in question. Since the project only focused on the years 2000 to 2005, very little data was available, and only 10 different inputs for validation were available.

Trial #13

A single-layer feedforward network receiving 200 inputs (20 days worth of historical data) and predicting for 20 days ahead.

Trial #14

An Elman network with 10 inputs (for the current day), predicting the outcome of the price the following day. The hidden layer and coincident layers both had 10 neurons as well. Teacher forcing was left at 0.5, and Jordan Elman Backpropagation was used.

Trial #15

An Elman network with 50 inputs, or 5 days worth of historical data, predicting the next day's results of the stock market price.

Trial #16

Various Elman networks with Teacher Forcing set to 0.25, 0.50, and 0.75. The goal of this work was to see if Teacher Forcing had any real effect on learning.

Trial #17

Combination of fundamental and technical indicators. Data was put in for the past 20 days, which resulted in 360 inputs. The network was a one-layer feedforward network, with 50 neurons in the hidden network. Since the number of weights compared to number of input patterns was so high, δ_{max} was set to 0.1.

Trial #18

The same neural network as Trial #3, but 20 such networks were trained at 500 cycles with random weight initializations to test for the range of predictions and variance caused by the random weights. All networks above also had random weight initializations, and it was important to see how likely it was that the results achieved above were due to the initial weights.

Trial #19

Theoretically, the only thing one needs to have neural networks make predictions about share prices using technical analysis is past prices. This trial focused on how well a one-layer hidden network, like the one in Trial #3, would work on predicting for 1, 5, and 20 business days based on the past 50 days of prices. 150000 and 25000 cycles were used for training, with 5 steps and a learning parameter of 0.003.

Appendix B: Overview of Custom Software

While the Java Neural Network Simulator (JNNS) and Stuttgart Neural Network Simulator (SNNS) were used for creating and training the neural networks, and Excel and R were used to analyze data, a number of pieces of software were created to facilitate analysis and research. All of the custom software is described below and included in the accompanying CD. The software below was written in Java, using the Eclipse Integrated Development Environment (IDE). All the software must be run from the command line. Please note the software was created solely to serve it's short-term purpose, and is by no means efficiently coded.

getdata.jar

All of the required data was taken from finance.yahoo.com and a program was written that downloads the required data and puts it into a tab delimited file. To run the program, use the following command:

```
java -jar getdata.jar <ticker file> <target file>
```

where *ticker file* is a file with one stock ticker symbol per line, and the second parameter is the file name where the data will be stored, with data for each stock on a separate line.

getfundamentals.jar

Since fundamental data is released once a week, once a month, or another period, it is difficult to have a good idea of which data applies to what day when creating input patterns for a neural network. By specifying a date and data file to this program, the user will get a new file outlining what the values of various fundamental variables was on that day. To run this program, type:

```
java -jar getfundamentals.jar <data file> <date file> <target file>
```

where the *data file* is a file of data with the first column having dates organized in a dd/mm/yyyy format. The *date file* is a list of required dates in a similar format, and the *target file* is the comma separated values (CSV) file that the information will be saved at.

convert.jar

This program takes a CSV file and converts it into an SNNS / JNNS pattern file. The CSV file must be organized with the input patterns first and the outputs following, with one pattern per row. Enter parameters in the following format:

```
java -jar convert.jar
            <CSV file> <number of inputs per row> <include outputs> <target file>
```

where *include outputs* is either *TRUE* or *FALSE* and allows one to choose whether the expected output should be encoded with the input pattern.

combine.jar

Since some of the networks in the project had more than 256 inputs, Excel could not handle the amount of data and convert it to CSV format for the CSV to SNNS pattern converter. As such, a second program had to be written that combines two CSV files into one. The code below is used to run the program:

```
java -jar combine.jar <first CSV file> <second CSV file> <target file>
```

where the first two parameters are the required CSV files and the *target file* is the file to save the two others. This program will combine every line of each separate CSV file into one line.

results.jar

When you use the *Save Data* option in JNNS, the file is saved to a *RES* file with its own format. This program converts the *RES* file to a file with one result per line, so it can then be opened in Excel or another program. This program is used on entire directories, so that all *RES* files in a directory are converted to *CSV* files. The original files, however, are not modified. Use the code below to run this program:

```
java -jar results.jar <target directory>
```

Appendix C: Effects of Random Weights on Network Results

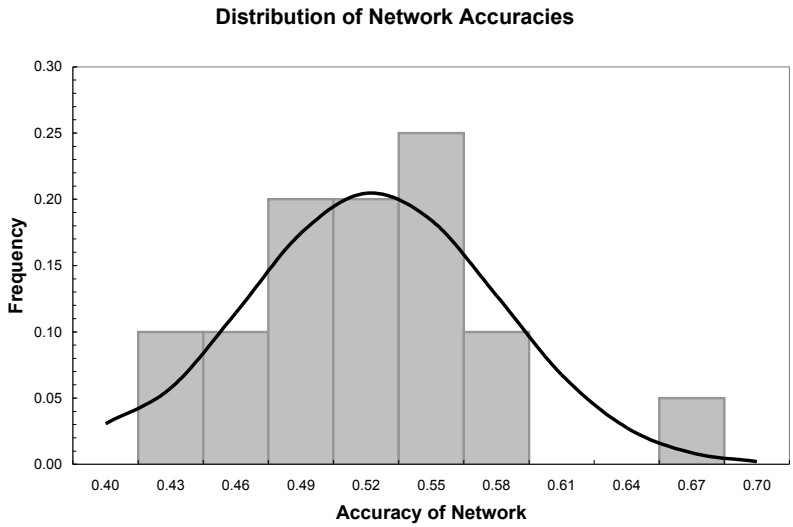
The purpose of Trial #18 was to see how well the results of the various networks tested adhere to the assumptions made in the statistical analysis, and whether the use of random weights had a major impact on results.

First and foremost were tests for normality. The assumption behind the t-tests used in the paper to compare networks to random predictions and Bernoulli distributions rests on the fact that a network's accuracy follows a normal distribution. Trial #18 consisted of 20 500-cycle training runs on a one-layer feedforward network, exactly like the one in Trail #3. This provided ample data to use the Shapiro-Wilk (National Institute of Standards and Technology, n.d.) test for normality. The following results were achieved:

Training Set: W = 0.9401

Validation Set: W = 0.9440

Within such a test, the higher the value for W, the more likely the data follows a normal distribution. For statistical interest, it was also found that the mean and standard deviation of the validation set was 50.81% and 5.78%, respectively. Below is a graph representing the results of the 20 networks.



Interestingly, one of the networks trained had an accuracy rate of 66.67%, making it a significant result if results follow a normal distribution as described above. Indeed, it seems random weights may actually play a major role in how well a network is trained, at least when the number of training cycles is low. While these are just preliminary results, it is useful to keep them in mind when analyzing results presented in the paper: poor network results may simply be a result of unlucky random weight initializations.

Works Cited

Achelis, S.B. (2001). *Technical analysis from A to Z*. United States: McGraw-Hill.

Arbib, M.A. (Ed.). (2003). *The Handbook of Brain Theory and Neural Networks*. Cambridge: MIT Press.

Brock, W., Lakonishok, L. & LeBaron, B. (1992). Simple Technical Trading Rules and the Stochastic Properties of Stock Returns. *The Journal of Finance*, 47, 1731-1764.

Bureau of Labor Statistics. (n.d.). *Consumer Price Index Home Page*. Retrieved on April 6, 2006 from http://www.bls.gov/cpi/home.htm

Conference Board. (2006a). *Consumer Confidence Index*. Retrieved on April 6, 2006 from http://www.conference-board.org/economics/consumerConfidence.cfm

Conference Board. (2006b). *Economic News*. Retrieved April 6, 2006 from http://www.conference-board.org/economics/press.cfm?press_ID=2840

Churchland, P. & Sejnowski, T.J. (1994). *The Computational Brain*. Cambridge: MIT Press.

Dawson, M. (2004). *Minds and Machines: Connectionism and Psychological Modeling*. United States: Blackwell Publishing.

Dimson, E. (Ed.). (1988). *Stock Market Anomalies*. Cambridge: Cambridge University Press.

Dyckman, T.R. & Morse, D. (1986). *Efficient Capital Markets and Accounting: A Critical Analysis*. United States: Prentice Hall.

Evans, M. & Rosenthal, J. (2004). *Probability and Statistics: The Science of Uncertainty*. United States: W. H. Freeman.

Federal Reserve Board. (2006a). *Open Market Operations*. Retrieved April 6, 2006 from http://federalreserve.gov/fomc/fundsrate.htm

Federal Reserve Board. (2006b). *Selected Interest Rates*. Retrieved April 6, 2006 from http://www.federalreserve.gov/releases/h15/Current/

Glosser, S.M. & Golden, L. (1997). Average work hours as a leading economic variable in US manufacturing industries. *International Journal of Forecasting*, 13, 175-195.

Granger, C.W.J. (1989). *Forecasting in Business and Econonomics* (2nd ed.). United States: Academic Press, Inc.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (2nd ed.). United States: Prentice Hall.

Iihara, Y., Kato, H.K., & Tokunaga, T. (2001). Investors' Herding on the Tokyo Stock Exchange. *International Review of Finance*, 2, 71-98.

Institute for Supply Management. (n.d.). *About ISM*. Retrieved April 6, 2006 from http://www.ism.ws/

Kacapyr, E. (1996). *Economic Forecasting: The State of the Art*. New York: M.E. Sharpe.

Malkiel B.G. (2003). The Efficient Market Hypothesis and Its Critics. *Journal of Economic Perspectives*, 17, 59-82.

Mankiw, G. & Scarth, W. (2004). *Macroeconomics: Canadian Edition* (2nd ed.). United States: Worth Publishers.

Morgan Stanley Capital International Inc. (2005). *Global Industry Classification Standard (GICS)*. Retrived April 6, 2006, from http://www.msci.com/equity/gics.html

National Institute of Standards and Technology. (n.d.). *Engineering Statistics Handbook*. Retrieved April 7, 2006, from http://www.itl.nist.gov/div898/handbook/

O'Reilly, R.C., & Munakata, Y. (2000). *Computational Explorations in Cognitive Neuroscience*. Cambridge: Massachusetts Institute for Technology.

Russell, S. & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach* (2nd ed.). United States: Pearson Education.

Siegel, J. (1998). *Stocks for the Long Run*. New York: McGraw-Hill.

Shleifer, A. (2000). *Inefficient Markets: An Introduction to Behavioral Finance*. Oxford: Oxford University Press.

University of Michigan. (2006). *Survey of Consumers*. Retrieved on April 6, 2006 from http://www.sca.isr.umich.edu/

Wärneryd, K.E. (2001). *Stock-Market Psychology*. United Kingdom: Edward Elgar.

Weisstein, E.W. (1999). *Mathworld*. Retrieved April 6, 2006 from http://mathworld.wolfram.com/

Yao, J., Teng, N., Poh, H., & Tan, C.L. (1998). Forecasting and Analysis of Marketing Data Using Neural Networks. *Journal of Information Science and Engineering*, 14, 843-862.

Zell, A., Mamier, G., Vogt, M., Mache, B., Hübner, R., & Döring, S. (n.d.). *Stuttgart Neural Network Simulator: User Manual, Version 4.2*. Retrieved April 6, 2006, from http://www-ra.informatik.uni-tuebingen.de/downloads/SNNS/SNNSv4.2.Manual.pdf