Chapter 1

Introduction

1.1 Commodities, Markets and Profits

A commodity is a good, especially an agricultural or mining product, that can be processed and resold. Some examples of such goods are: soybeans, wheat, crude oil and gold. Large volumes of commodities are bought and sold around the world each day. Farmers and miners need to sell their products and manufacturers need to purchase raw materials. This can be done on the spot market, where commodities are bought and sold "on the spot". The spot market is actually a term that denotes many decentralized locations at which the good may be bought or sold.

However, commodities trading is actually much more complex than simply buying and selling goods on the spot market. Most commodities trading is actually done on the futures market as futures contracts. That is, at time t_0 an individual promises to sell x bushels of soybeans at price y at a time t_1 in the future [Hul97]. Someone else would enter into the other side of the contract, promising to buy that quantity of the commodity at the agreed upon price at the proposed time. The development of futures can be traced to the middle ages when they were developed to meet the needs of farmers and merchants [Hul98]. Futures make a great deal of sense for both parties because they reduce risk. The farmer knows that he will sell his crop for a certain price and the manufacturer can rest assured that his business will be able to continue production with an ample supply of affordable raw materials.

Commodity futures contracts were first traded on an organized exchange, the Chicago Board of Trade (CBOT), in 1850. The first contract called for the delivery of 3,000 bushels of corn. The corn would be exchanged for a cash payment at a pre-specified future time and location. In 2002, over 260 million contracts for 47 different products were traded on the CBOT. Comparable trading volume exists on the Chicago Mercantile Exchange and the New York Mercantile Exchange. The enormous size of the futures market has sparked researchers to examine the motivations of traders.

Commodity futures contracts originated as a way for farmers to reduce their price risk. Prior to these contracts, a farmer would have to bear the cost of planting and cultivating a crop with no guarantee of the price for which that crop could be sold. By contracting to sell his crop at a certain price before planting it, the farmer did not have to be concerned with future price changes. His only risk was the actual production of his crop. Therefore, a farmer's compensation would depend on his ability to produce a crop rather than his ability to produce a crop and future changes in the price of the commodity being produced. The farmer, as is anyone else who seeks to reduce risk, is called a hedger.

The farmer's need to hedge is fulfilled by speculators. Speculators are willing to bear the price risk in hopes of favorable price changes. Since farmer's hedge their position by contracting to sell in the future (aka shorting the contract) speculators take long positions (agree to buy). Speculators will only take a long position if they expect the price of the commodity to be greater at time of delivery than the price specified in the futures contract [Key78]. Then they can buy from the farmer at a set price and immediately sell the commodity for a greater price. The difference between the price stated in the futures contract and the speculators expected price at that future time is the risk premium. The hedger is willing to allow speculator to have this premium in exchange for the reduced risk of a guaranteed price. The speculator's

compensation is this risk premium. Since it is the speculator's belief about the probability distribution of future spot prices that determines the size of the risk premium that he is willing to accept, the speculator's success is dependent on the validity of his beliefs. A comparative advantage can be gained by acquiring superior information about the underlying dynamics of the commodity's price behavior.

The use of futures contracts to hedge is not limited to farmers. A manufacturing firm may require large amounts of a commodity to produce its final product. Airlines must buy large amounts of jet fuel. Any entity which is exposed to the price risk of a commodity that underlies a traded futures contract may reduce their risk by taking the appropriate position futures contracts. Those seeking to reduce risk are hedgers [CWS95].

The futures market has grown well beyond the previous examples but the logic still applies. It should be noted that most futures contracts do not result in physical delivery. A trader can realize the financial gain or loss of his position by taking a position in the same futures contract that is opposite to his current position.

The effects of various factors on a particular commodity's price are unique to that particular commodity. As with any economic good the ultimate arbiter of price is supply and demand. Efficient markets imply that all information is fully reflected in the commodity's price and no one can make a economic profit by trading futures contracts.

If a pattern can be found in past price data that accurately predicts future price movement, then markets are not efficient and an economic profit can be made by those who recognize the pattern. NN and GAs are used to find such a pattern. In an attempt to predict commodity futures prices, people attempt to use historical price data for a commodity future to predict the value of that commodity future. Mathematically, values which run in a series and vary over time are known as a time series.

1.2 Time series and financial analysis

Technical analysis focuses on the past data of an asset, such as a commodity futures contract, in order to predict its future behavior [BKM98]. Thus, in essence, technical analysis is devoted to studying time series of data related to financial instruments, including commodities, futures, stocks, etc. Technical analysts examine this historical data and attempt to formulate rules which explain the behavior that they observe. They look for patterns in the data and attempt to generalize from those patterns in order to predict the future behavior of the variable in question such as the price of the asset. Some technical analysts specialize in stocks, others in futures or commodities, yet others in foreign exchange rates.

Technical analysts use a variety of methods in their attempts to extract valuable information from time series. Some of the methods, such as Box-Jenkins methods [BJ76] or other linear regression approaches rely on their assumption of a linear relationship among the variables. However, most experts have found that the relationships among time series values for various financial instruments are not linear [BLB92, DGE93]. Other, non-linear methods have also been used to perform financial time series analysis [Sav89, Sch90].

1.3 Methodology

1.3.1 Experiments

A neural network is an artificial intelligence technique that is especially useful for recognizing patterns in complex data sets and generalizing from those patterns in order to work with new data. Thus, a neural net would seem to be an ideal technique to use in time series analysis. The neural net should be able to take n values of the time series as inputs and estimate the next value as an output. The network should be able to find any existent relationship between these variables. The neural net

should be able to recognize any pattern in the data and predict future values. The theoretical limit to the accuracy of a neural network is the amount of noise present in the data.

Neural networks are quite useful for recognizing patterns and generalizing from them. However, optimal network performance is quite dependent upon making the proper choices when constructing the network. Furthermore, the construction of a neural network is very much an art form and the ideal network architecture can be quite elusive. It is possible to combine the neural net with another artificial intelligence technique such as a genetic algorithm. Genetic algorithms excel at minimization / maximization problems. Thus, a genetic algorithm can be used to determine the proper architecture to maximize the accuracy of the neural network.

In addition to the architecture problem, the methods used to train a neural net can fall prey to local minima. Genetic algorithms can be used once more to find optimal values for the weights of the neural networks as well. Thus, local minima can be avoided and optimal neural network results can be obtained.

The goal of the current research is to find a preferred artificial intelligence approach to predict future values for commodities futures. Project objectives are:

1. Develop a neural network model to predict the next value in the commodity future time series.

2. Use a genetic algorithm to select the proper architecture for a neural network model to predict the next value in the commodity future time series.

3. Use a genetic algorithm to search for weight assignments for a neural network model to predict the next value in the commodity future time series.

Chapter 2 examines objective 1, while chapters 3 and 4 address objectives 2 and 3, respectively. Chapter 5 compares results from all of the objectives and discusses overall conclusions.

1.3.2 Data

The data consist of daily commodity data for soybeans at the Chicago Board of Trade starting with January 1, 1980 and ending with August 30, 2002. Different portions of this data are used in the various experiments. The data are obtained from Datastream. In order to more easily optimize neural network performance, the data are normalized before being used as input for the neural networks. Each data set is divided by its largest member to provide values between zero and one. In turn, the predicted, output values are multiplied back to their original values and compared to actual observed values in their original form. Error statistics are generated using these non-normalized values.

Chapter 2

Pure Neural Network Approach

2.1 Background on Neural Networks

The brain is made up of neurons. These neurons are connected to each other by synapses which vary widely in strength. Each neuron receives input from a number of other neurons. This input may be excitatory, i.e. positive, or inhibitory, i.e. negative. If the input activation level is sufficient, then the neuron will "fire". The firing neuron will then send activation to those neurons which are connected to its output. In this manner activation spreads through a network of neurons. The human brain contains between 10^{10} and 10^{11} neurons. Each of these neurons is connected to hundreds or thousands of other neurons[And95]. Thus, the human brain understands speech, recognizes a loved one's face, and accomplishes abstract thought, etc. by spreading patterns of activation throughout this complex neural network. Though each neuron is making a simple calculation, complex learning is accomplished by altering the connection strengths between neurons. The calculation performed by each neuron remains the same, however the weight given to the various inputs changes. This enables the marvelous complexity of function shown by the human brain.

Neural networks are based on this system. There are nodes which have an activation function, most commonly the sigmoid function, $1/(1 + e^{-x})$. This function determines if the input activation is sufficient to cause the node to fire. Biological synapses are modeled by weighted links between the nodes. These weights can be

positive or negative, modeling the excitatory and inhibitory function found in biological neural networks. In general, there are one or more nodes which receive the data as input. These are called input nodes. In most neural network implementations, the input nodes do not use an activation function. These nodes simply function as storage bins for the input being fed into them. There are also usually one or more nodes which provide the output, or answer from the network. These are called output nodes. In many neural network architectures, including all of those used in this research, there are additional nodes which reside between the input and output nodes. These nodes are called hidden nodes and they provide much of the computing power for the network.

A wide variety of neural network architectures have been used. However, this research focuses on the most commonly used architecture, the multilayer perceptron, also known as a three layer, feed-forward neural network. Hereafter "network" or "neural network" will refer to a network of this type. In this sort of network, nodes in one layer may only influence those residing in a layer closer to the output layer. Thus, activation may only feed forward through the network from input nodes to hidden nodes and then on to output nodes. No feedback loops are allowed.

A properly constructed network can recognize patterns in the input data. Different patterns of input produce different output. The network can also generalize by responding to similar patterns in a similar manner. However, these desirable behaviors are dependent upon the weights in the network being properly set. Calculating the proper weights for a small network quickly becomes a daunting task. Directly calculating the weights for a large network is practically impossible. However a system of using supervised learning called back propagation of errors[RHW86] provides an answer to this problem. This technique uses a gradient decent method to establish values for the network weights which minimize the output errors. Thus, both the input and output must be known for the network to be trained.

2.2 Neural Networks and Financial Time Series Analysis

In addition to the more traditional approaches, a artificial intelligence techniques have been applied to a number of financial instruments. Garcia and Gençay[GG00] use a neural network to estimate a generalized option pricing formula. Walczak[Wal01] uses neural networks to forecast foreign exchange rates for a variety of currencies. Trippi and DeSieno[TD92] examine a day trading system for Standard & Poors 500 index futures contracts which is based on a neural network combined with rule-based expert system techniques. Hutchinson, et al.[HLP94] construct neural network models for predicting Standard & Poors 500 futures options prices. Castiglione[Cas00] uses neural networks to predict a variety of financial time series. Yao, et al.[YTP99] use neural networks to forecast the Kuala Lumpur Composite Index.

2.3 Software used / Methodology

For this research, a variant form of backpropogation called resilient propagation (RPROP)[RB93] was used. RPROP utilizes a local adaptation of the weight updates according to the behavior of the error function in order to overcome some of the disadvantages of a pure gradient decent approach. The RPROP algorithm is not effected by the influence of the size of the error derivative, since it relies only on the sign of the error derivative. Instead, the size of the weight change is determined by a weight-specific update value $\triangle_{ij}^{(t)}$:

$$\triangle w_{ij}^{(t)} = \begin{cases} -\triangle_{ij}^{(t)} & , \quad \text{if } \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ +\triangle_{ij}^{(t)} & , \quad \text{if } \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ 0 & , \quad \text{else} \end{cases} \tag{2.1}$$

The new update-values $\triangle_{ij}^{(t)}$ must also be determined. This is based on a sign-dependent adaptation process.

$$\triangle_{ij}^{(t)} = \begin{cases} \eta^+ * \triangle_{ij}^{(t-1)} & , \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ \eta^- * \triangle_{ij}^{(t-1)} & , \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ \triangle_{ij}^{(t-1)} & , \text{ else} \end{cases} \tag{2.2}$$

$$\text{where } 0 < \eta^- < 1 < \eta^+$$

Thus, when the last update is too large and the algorithm has jumped over a local minima, the partial derivative of the corresponding weight w_{ij} changes its sign and the update value $\triangle_{ij}^{(t)}$ is decreased by the factor η^-. When the derivative retains its sign, the update value is increased by the factor η^+. Additionally, in the case of a change in sign, there is no adaptation in the succeeding learning step. This is achieved by setting $\frac{\partial E}{\partial w_{ij}}^{(t)} = 0$ in the above adaptation rule.

In order to reduce the number of freely adjustable parameters, the increase and decrease factors are usually set to fixed values of $\eta^- = 0.5, \eta^+ = 1.2$.

This research also utilized weight decay. Weight decay is a modification that may be used with neural network's backpropagation learning technique in order to improve generalization. Using a weight decay term (α) can assist in preventing over training and preserving generalization by performing a local reduction of the weights, discouraging large weights. This research used the weight decay variant of the RPROP algorithm. Thus, the composite error function is:

$$E = \sum (t_i - o_i)^2 + 10^{-\alpha} \sum w_{ij}^2$$

For implementation of the neural nets, this research used the Stuttgart Neural Network Simulator, version 4.2. This software was run on a Pentium 4 PC with the Linux operating system. A number of different network architectures were used along with a number of variations of the network parameters. The networks were evaluated by comparing predicted versus actual values on the data sets. For each

network, the sum of square errors (SSE) was collected and used for evaluation. Each combination of variables was set up in SNNS and trained until either:

- the test data error began increasing
- there was no further improvement in the training data error.

This data should provide insight into the problem as well as establishing a baseline for comparison with other techniques.

2.4 Results / Conclusions

One set of experiments uses one hundred days worth of input into the neural net. The other set uses only ten days worth of input. The longer data set should allow longer term patterns to be detected, while the shorter data set should focus the network on recent values for the variable in question.

Additional, longer time periods could be tested in an attempt to find still longer term patterns. A time period of approximately a year would be a fairly logical extension, especially for agricultural commodities such as the soybeans used in this research. However, longer time periods are outside the scope of this research.

The number of hidden nodes used were five, ten, and one hundred. The weight decay parameter was allowed to assume the values five, ten, and twenty. The initial weight range values were kept semetrical and had values of plus and minus: 1.0, 0.5, 0.25, 0.125, 0.0625.

The results of the experiment can be seen in Table 2.1 through Table 2.6. There is considerable variety in the quality of solutions found by the different networks. These tables contain the error calculations on the out of sample validation set. The mean squared error (MSE) was calculated by dividing the SSE by the number of observations in the data set. The mean error is simply the square root of the MSE.

These values are in cents per bushel. So, a mean error of 15.20 means that the network missed its prediction by an average of 15.20 cents per bushel. The average price per bushel was 651.37 cents. Even relatively small differences in error could mean a considerable difference in the profit made by using the network since the volume of the commodity traded can be rather large.

It appears that the most recent data is the most important for calculating the next value in the time series since there is no improvement gained by increasing the number of inputs from ten to one hundred. Indeed, there is evidence of decreased performance on the experiments which used one hundred data points as input. There seems to be a negative effect of including additional hidden nodes as well. The error values for the five and ten hidden node subsets are comparable, however the error generated by the one hundred hidden node sets is noticeably greater. This difference seems to only exist in the experiments which used one hundred input values. This seems to indicate that the greater number of hidden nodes combined with the larger number of input nodes allowed the network begin to over train despite the weight decay parameter and the stopping condition. Indeed, the results for a weight decay parameter of five are slightly better than the others. Since a lower weight decay parameter causes more severe weight decay and thus a more severe limit on over training, this discrepancy supports the hypothesis. These findings are in agreement with those of Gerson and Fuller[LF95]. They found that including additional input or hidden nodes reduced network efficacy for time series evaluation.

Table 2.1: Results with 100 input values, average price 651.37 cents per bushel

Hidden Nodes	Weight Decay	Initial weight range	SSE	MSE	Mean Error
100	5	-1.0–1.0	253563	329.30	18.14
100	5	-0.5–0.5	236477	307.11	17.52
100	5	-0.25–0.25	304617	395.60	19.88
100	5	-0.125–0.125	224618	291.71	17.07
100	5	-0.0625–0.0625	233060	302.67	17.39
100	10	-1.0–1.0	321903	418.05	20.44
100	10	-0.5–0.5	242407	314.81	17.74
100	10	-0.25–0.25	202006	262.34	16.19
100	10	-0.125–0.125	171655	222.92	14.93
100	10	-0.0625–0.0625	242307	314.68	17.73
100	20	-1.0–1.0	478583	621.53	24.93
100	20	-0.5–0.5	207835	269.91	16.42
100	20	-0.25–0.25	326325	423.79	20.58
100	20	-0.125–0.125	375068	487.10	22.07
100	20	-0.0625–0.0625	139092	180.64	13.44

Table 2.2: Results with 100 input values, average price 651.37 cents per bushel

Hidden Nodes	Weight Decay	Initial weight range	SSE	MSE	Mean Error
10	5	-1.0–1.0	152358	197.86	14.06
10	5	-0.5–0.5	127937	166.15	12.89
10	5	-0.25–0.25	150851	195.91	13.99
10	5	-0.125–0.125	328134	426.14	20.64
10	5	-0.0625–0.0625	194167	252.16	15.87
10	10	-1.0–1.0	170047	220.84	14.86
10	10	-0.5–0.5	134972	175.28	13.23
10	10	-0.25–0.25	108339	140.70	11.86
10	10	-0.125–0.125	114269	148.40	12.18
10	10	-0.0625–0.0625	160197	208.04	14.42
10	20	-1.0–1.0	159494	207.13	14.39
10	20	-0.5–0.5	164318	213.40	14.60
10	20	-0.25–0.25	125927	163.54	12.78
10	20	-0.125–0.125	120299	156.23	12.49
10	20	-0.0625–0.0625	112058	145.53	12.063

Table 2.3: Results with 100 input values, average price 651.37 cents per bushel

Hidden Nodes	Weight Decay	Initial weight range	SSE	MSE	Mean Error
5	5	-1.0–1.0	170248	221.10	14.86
5	5	-0.5–0.5	116781	151.66	12.31
5	5	-0.25–0.25	123515	160.40	12.66
5	5	-0.125–0.125	120098	155.97	12.48
5	5	-0.0625–0.0625	234568	304.63	17.45
5	10	-1.0–1.0	135776	176.33	13.27
5	10	-0.5–0.5	220699	286.62	16.92
5	10	-0.25–0.25	115173	149.57	12.23
5	10	-0.125–0.125	114570	148.79	12.19
5	10	-0.0625–0.0625	118490	153.88	12.40
5	20	-1.0–1.0	117686	152.83	12.36
5	20	-0.5–0.5	112862	146.57	12.10
5	20	-0.25–0.25	129846	168.63	12.98
5	20	-0.125–0.125	371349	482.27	21.96
5	20	-0.0625–0.0625	111053	144.22	12.00

Table 2.4: Results with 10 input values, average price 651.37 cents per bushel

Hidden Nodes	Weight Decay	Initial Weight Range	SSE	MSE	Mean Error
100	5	-1.0–1.0	131555	152.97	12.36
100	5	-0.5–0.5	507729	590.38	24.29
100	5	-0.25–0.25	112058	130.30	11.41
100	5	-0.125–0.125	99294	115.45	10.74
100	5	-0.0625–0.0625	129746	150.86	12.28
100	10	-1.0–1.0	120902	140.58	11.85
100	10	-0.5–0.5	98591	114.64	10.70
100	10	-0.25–0.25	104118	121.06	11.00
100	10	-0.125–0.125	100701	117.09	10.82
100	10	-0.0625–0.0625	129947	151.10	12.29
100	20	-1.0–1.0	576371	670.19	25.88
100	20	-0.5–0.5	214066	248.91	15.77
100	20	-0.25–0.25	118490	137.77	11.73
100	20	-0.125–0.125	100802	117.21	10.82
100	20	-0.0625–0.0625	130751	152.03	12.33

Table 2.5: Results with 10 input values, average price 651.37 cents per bushel

Hidden Nodes	Weight Decay	Initial Weight Range	SSE	MSE	Mean Error
10	5	-1.0–1.0	410846	477.72	21.85
10	5	-0.5–0.5	107133	124.57	11.16
10	5	-0.25–0.25	134168	156.00	12.49
10	5	-0.125–0.125	103113	119.89	10.94
10	5	-0.0625–0.0625	108641	126.32	11.23
10	10	-1.0–1.0	98189	114.17	10.68
10	10	-0.5–0.5	328034	381.43	19.53
10	10	-0.25–0.25	461699	536.86	23.17
10	10	-0.125–0.125	137082	159.39	12.62
10	10	-0.0625–0.0625	247231	287.47	16.95
10	20	-1.0–1.0	102812	119.54	10.93
10	20	-0.5–0.5	101405	117.91	10.85
10	20	-0.25–0.25	96882	112.65	10.61
10	20	-0.125–0.125	104018	120.95	10.99
10	20	-0.0625–0.0625	117686	136.84	11.69

Table 2.6: Results with 10 input values, average price 651.37 cents per bushel

Hidden Nodes	Weight Decay	Initial Weight Range	SSE	MSE	Mean Error
5	5	-1.0–1.0	97787	113.70	10.66
5	5	-0.5–0.5	100400	116.74	10.80
5	5	-0.25–0.25	97485	113.35	10.64
5	5	-0.125–0.125	105827	123.05	11.09
5	5	-0.0625–0.0625	124017	144.20	12.00
5	10	-1.0–1.0	104319	121.30	11.013
5	10	-0.5–0.5	99294	115.45	10.74
5	10	-0.25–0.25	97083	112.88	10.62
5	10	-0.125–0.125	97586	113.47	10.65
5	10	-0.0625–0.0625	107837	125.39	11.19
5	20	-1.0–1.0	110550	128.54	11.33
5	20	-0.5–0.5	150750	175.29	13.23
5	20	-0.25–0.25	112962	131.35	11.46
5	20	-0.125–0.125	108641	126.32	11.23
5	20	-0.0625–0.0625	98892	114.99	10.72