An Extension of ETF Arbitrage to Sector Trading Using ANN

Ramnik Arora, Utkarsh Upadhyay
Indian Institute of Technology, Kanpur

Abstract—We design and deploy a trading strategy that mirrors the Exchange Traded Fund (ETF) arbitrage technique for sector trading. Artificial Neural Networks (ANNs) are used to capture pricing relationships within a sector using intra-day trade data. The fair price of a target security is learnt by the ANN. Significant deviations of the true price from the computed price (ANN predicted price) are exploited. To facilitate arbitrage, output function of the trained ANN is locally linearly approximated. The strategy has been backtested on intra-day data from September 2005. Results are very promising, with a high percentage of profitable trades. With low average trade durations and ease of computation, this strategy is well suited for algorithmic trading systems.

Index Terms—ETF Arbitrage; Neural Networks; Sector Trading; Statistical Arbitrage

1 Introduction

ANNs have been used for several years in the selection of investments because of their ability to identify patterns of behavior that are not readily observable. These include business forecasting, credit scoring, bond rating and business failure prediction. Of particular interest to financial researchers is the deployment of neural networks for forecasting economic time series data, and as predictors for future share prices.

Since most of the development and deployment of trading strategies is undertaken for commercial use and is thereby proprietary, there are not many statistically profitable strategies in the public domain that deploy ANNs efficiently. We design and deploy such a strategy which is based on intra-day data and closely mirrors ETF arbitrage. This approach hinges on the assumption that *there exist* complex, non-linear relationships between price movements of related securities. Any mispricing or deviations from this relationship can be exploited to financial advantage. Furthermore, due to high correlation in corporate models, we have assumed that stocks from a particular sector can be reasonably priced using others from the same sector.

2 Review of Literature

ANNs have been shown to be an efficient tool for non-parametric modeling of data in a variety of different contexts. Hornik et al.(1989)(Hornik, Stinchcombe, and White, 1989) showed that standard neural network model using an arbitrary transfer function can approximate any measurable function in a precise and satisfactory manner, if a sufficient number of hidden neurons are used.

In the context of financial forecasting, Kuan and Liu (1995)(Kuan and Liu, 1995) discuss forecasting of foreign exchange rates using ANNs. They show that a properly designed ANN has lower out-of-sample mean squared prediction error relative to the random walk model. Jasic and Wood(2004)(Jasic and Wood, 2004), discuss the profitability of trading signals generated from the out-of-sample short-term predictions for daily returns of S&P 500, DAX, TOPIX and FTSE stock market indices evaluated over the period 1965-99. The out-of sample prediction performance of neural networks was compared against a benchmark linear autoregressive model. They found that the buy and sell signals derived from neural network predictions were different from unconditional one-day mean return and were likely to provide significant net profits for reasonable decision rules and transaction cost assumptions.

Huang et al. (2004)(Huang, Chen, Hsu, Chen, and Wu, 2004) report a comparative study of application of Support Vector Machines (SVM) and Back propagation Neural Networks (BNN) for an analysis of corporate credit ratings. They report that the performances of SVM and BNN in this problem were comparable and both these models achieved about 80 per cent prediction accuracy. Pendharkar (2005)(Pendharkar, 2005) discusses the application of ANNs for the bankruptcy prediction problem. He reported that ANNs perform better than the statistical discriminant analysis both for training and hold-out samples.

Hence, ANNs have a history of being used for financial predictions, but yet, haven't been seen as a possible regressors for arbitrage strategies. We investigate the usefulness of ANNs in this area.

- *R. Arora is with the Department of Mathematics and Scientific Computing, Indian Institute of Technology, Kanpur.*
  Homepage: see http://home.iitk.ac.in/˜ramnik.
- *U. Upadhyay is with the Department of Electrical Engineering, Indian Institute of Technology, Kanpur.*
  Homepage: see http://home.iitk.ac.in/˜utkarshu

3 Trading Strategy And Deployment

Arbitrage are a series of long-short trades that yield risk-free profit. Arbitrage is of two forms: *deterministic arbitrage*, whereby risk free profit is assured on each set of trades, and *statistical arbitrage*, where market pricing inefficiencies are exploited under the assumption of convergence to historical or predicted normal prices and it generates profit with a probability more than random walk. Developed as early as 1980 as *pair trading*(Gatev, Evan, Goetzmann, William, Rouwenhorst, and Geert, 2006), arbitrage has received much attention in recent times owing to its easy extension to automated trading systems. A very common strategy based on the idea of arbitrage is ETF Arbitrage, which we extend to sectoral trading.

3.1 ETF Arbitrage

An ETF's Net Asset Value (N.A.V.) is dictated by the equation:

$$x = \sum_{i=1}^{i=n} a_i \, y_i \tag{1}$$

x	is the true NAV
y_i	is the price of the i^{th} underlying
a_i	is the weight of the i^{th} component

This NAV decomposition is publicly known. However, owing to the demand and supply of either the ETF or the underlying components, often a mispricing between the ETF price and its NAV may occur. *Authorized participants*, who are privileged traders, can exchange the ETF with the underlying or vice-versa. At this point of mispricing, they can buy the underpriced security (or short the overpriced security), exchange it with the other, and then sell the overpriced one (or buy the underpriced one) to obtain risk free profit from a *self-financing portfolio*.

Thus, if we short one unit of the security whose price is x, we would need to long a_i units of the i^{th} security. For example, if the equation governing the NAV of the ETF were: $x = 3y_1 + 4y_2$, then trades would need to be made in a ratio of $1 : -3 : -4$, where $-d$ would indicate short selling of d shares.

However, with stocks from a particular sector, such a relationship does not exist. Nevertheless, it can be assumed that broad news affects entire sector in a more-or-less uniform way. Thus, the stocks within a sector should have a high correlation, though it would be subject to market change.

Our strategy attempts to learn this pricing relationship in the following manner:

3.2 Pricing relationship

As will become clear in section 4, we intend to use ANNs for the task of computing the pricing relationship. Although ANNs can be readily used for non-parametric regression, they generally find non-linear relationships between the input and the output variables. However, in accordance with ETF arbitrage requirements we would like to have a linear relationship between the input and the output variables as was seen in the equation 1.

ANNs invariably yield a non-linear relationship for which the simple analysis as done for ETF arbitrage will not hold. We prefer obtaining a *linear approximation* instead of the true relationship between the input and the output variables. This approximation is made using the Taylor series expansion of the non-linear function of the ANN. By providing slight perturbations in the input variables, we can obtain the directional derivatives of the output variable with respect to each input variable (see appendix: A). We arrive at the following form (where γ_i are the directional derivatives along coordinate i):

$$dx = \sum_{i=1}^{i=n} \gamma_i \, \partial y_i \tag{2}$$

Now integrating both sides, we arrive at the following equation, which is of the desired form.

$$x - \bar{x} = \sum_{i=1}^{i=n} \gamma_i \, (y_i - \bar{y_i}) \tag{3}$$

On comparison with equation 1 we can easily see that trades need to made in ratio of $1 : -\gamma_1 : -\gamma_2 : \ldots : -\gamma_n$.

3.3 Profit/Loss Analysis

To calculate the profit/loss of the strategy, consider a simplified scenario where we have two underlying of a composite stock. Another simplifying assumption is of negligible transaction costs.

If we trade only at two points (See figure 1), our accounts would look as follows:

At time t_s:

We short the target security (overpriced security) and long the composite security (underpriced security). Thus, our total account currently is:

$$A(t_s) = -\lambda_1(t_s)S_1(t_s) - \lambda_2(t_s)S_2(t_s) + S_{target}(t_s) \tag{4}$$

$A(t)$	is the investment at time t
$S_{target}(t)$	is the value of the target stock at time t
$\lambda_i(t)$	is the weight of the i^{th} component at time t
$S_i(t)$	is the price of the i^{th} stock at time t

Let us say that the composite stock price and the true target price converge at time t_f.

At time t_f:

Here we reverse our trades i.e: long the target security and short the composite security. These trades maybe summarised as:

$$A(t_f) = \lambda_1(t_f)S_1(t_f) + \lambda_2(t_f)S_2(t_f) - S_{target}(t_f) \tag{5}$$

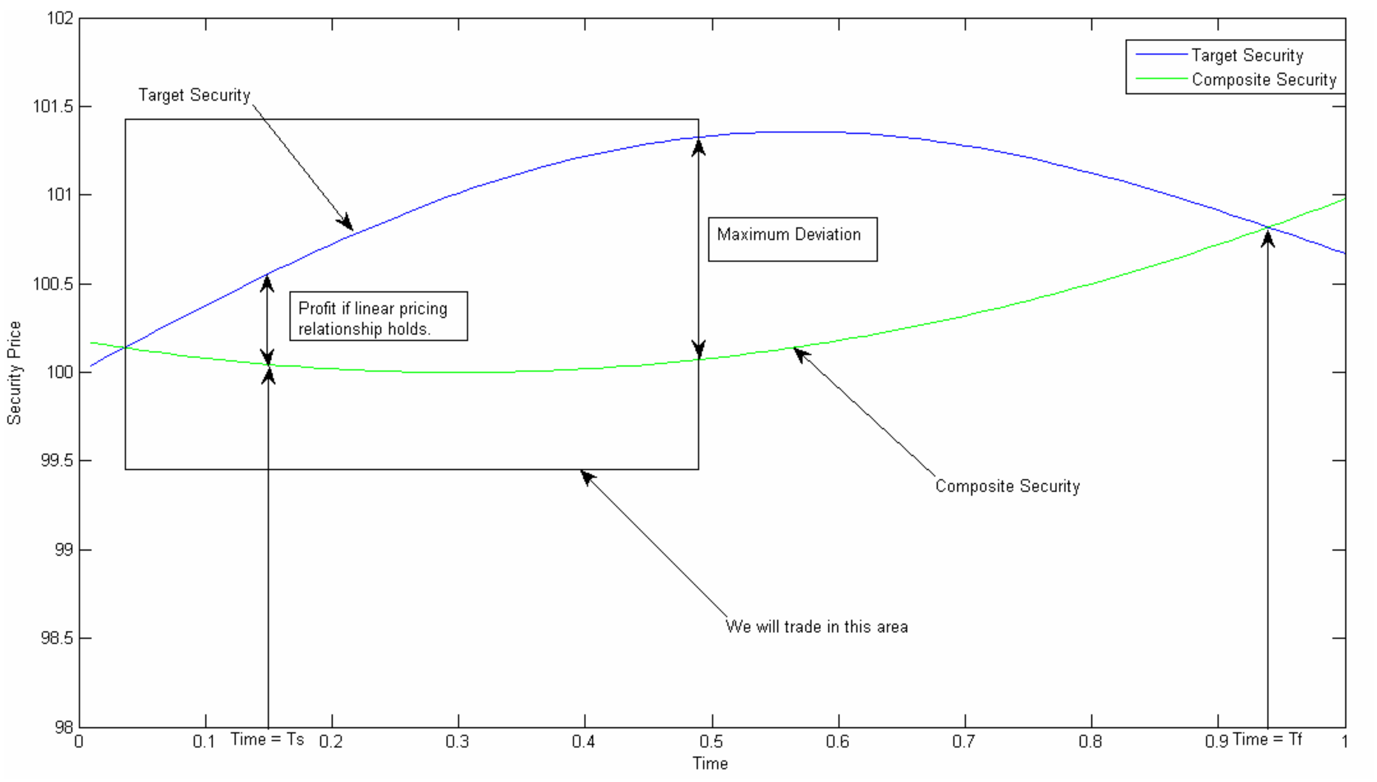

Fig. 1. We short the target security and long composite security at time t_s and reverse the trades at t_f.

Analysis

At this point, we have:

$$Total\ Cash\ Flow = A(t_s) + A(t_f) \tag{6}$$

Looking at figure 1, $A(t_f) = 0$. Hence, equation 6 simplifies to:

$$Total\ Cash\ Flow = A(t_s) \tag{7}$$

Calculating our open positions or Outstanding Stock Value (OV) at time t_f is

$$[\lambda_1(t_s) - \lambda_1(t_f)]\, S_1(t_f) + [\lambda_2(t_s) - \lambda_2(t_f)]\, S_2(t_f) \tag{8}$$

Now the importance of the linear approximation comes into play. If the linear approximation holds good, then $\lambda_1(t_s) = \lambda_1(t_f) = \lambda_1$ and, similarly, $\lambda_2(t_s) = \lambda_2(t_f) = \lambda_2$. Under these conditions, the profit would truly be only $A(t_s)$, since the terms with $S_1(t_f)$ and $S_2(t_f)$ would completely cancel each other out. However, in light of equation 8 and non-linearity of the market, the new profit/loss at time t_f is:

$$[\lambda_1(t_s){-}\lambda_1(t_f)]\, S_1(t_f){+}[\lambda_2(t_s){-}\lambda_2(t_f)]\, S_2(t_f){+}A(t_s) \tag{9}$$

It is easy to see that the differences $[\lambda_1(t_s){-}\lambda_1(t_f)]$ and $[\lambda_2(t_s){-}\lambda_2(t_f)]$ may take positive as well as negative values with equal probabilities if the linear approximation fails to hold. However, if the linear approximation holds, the total profit as seen in equation 9 would be positive as $A(t_s)$ would always be positive. Hence, this strategy would be statistically profitable.

4 Artificial Neural Networks

For the design on the neural networks, we rely heavily on the seminal work of Kaastra and Boyd(Kaastra and Boyd, 1996). We adapt and fine tune their approach to arrive at optimal neural network for our work.

4.1 Variable selection

In George Soros(Soros, 1994) words:

One of the major problems faced in modeling financial market movements is the fact that information comes in from a very large number of sources.

The pricing of a particular asset can depend on the specific asset related news and broad news(or macro-economic news). Thus, for example, share prices of the steel industry would be directly affected by news pertaining to the steel industry but also by a change in interest rates or a change in the cement prices. Owing to the large volume of broad news that could impact the stock prices, we need a reliable proxy. The impact of broad news on other elements in the target securities sector will be used to this effect in our study. The impact of a development or news, in general, would have a *similar* impact on the all elements of the sector. As input variables, we select the prices of those companies stocks which closely resemble the business model and size of our target company.

It is interesting to note that we cannot use lagged stock prices or any other inputs which cannot be traded upon. For example, if we had made our system dependent on any variable other than the current prices of the other underlying, while attempting to perform linear decomposition of the target price into its components, we would have encountered a components which corresponded to an entity (a previous price, or difference of two prices) which we cannot trade on, thereby making arbitrage impossible. Therefore, for stock price forecasting, we have used real-time prices of companies of the same sector that have a similar structure and size as the target company.

4.2 Data Preprocessing

Deployment issues, relating to arbitrage, forbid us from taking technical indicators and ratio of stock prices for forecasting. We have used feasible data vectors to the neural network as inputs (see appendix B). For optimal results, inputs to neural networks must be normalised. There exist multiple ways to normalise the input data(Kaastra and Boyd, 1996) which could be further explored.

4.3 Neural Network Topology

The properties of an individual neuron such as its transfer function and how the inputs are combined along with the structure including the number of neurons per layer and the number of hidden layers define the network topology. There is no accepted literature that defines the optimal topology of an arbitrary neural network and most networks are fine tuned with experimentation. It is widely believed that one or two hidden layers with *sufficient* number of hidden neurons are enough to approximate any function. However, many suggestions for this 'sufficient number' have been proposed; varying from half the number of input neurons(Katz, 1992) to iterative doubling to the point of deterioration of results on testing data set(Ersoy, 1990). We keep one output neuron, the forecasted price of target security based on the independent input prices.

4.4 Neural Network Training

We do not intend to model our neural networks in the conventional sense of being a regressor which can best fit all data to date; they will be trained and deployed on each day independent of previous days.

As can be seen in figure 2, the data that we have will be segmented into different sections on a day-to-day basis. To evaluate the performance of our network on the i^{th} day, one would need to train it on the initial subset of the data available for the i^{th} day and test it on the remaining data. Daily training is necessary since the exact relationship between the input variables and the output variable may differ from day to day.

This way, we perform hyper parameter tuning to come up with the parameters of a model that captures well the daily relationship between the input and the output variables, rather than attempting to realise a universal

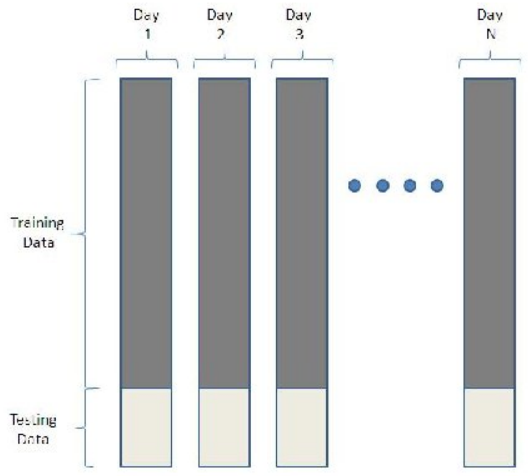

Fig. 2. Division of training data

relationship that holds across days. This tuning theoretically can be performed everyday using the immediately collected data but this task can be computationally arduous and suitable adjustments maybe made while deployment.

4.4.1 Training, testing and validation sets

Levenberg-Marquardt algorithm(Levenberg, 1944; Marquardt, 1963), implemented using MATLAB Neural Network Toolbox has been used for training the ANN. The testing datasets would consist of a set of observations immediately following the training set from the same day, much like real-time deployment would entail(see figure 2). Each intra-day segment of data can be divided into training and testing data in two ways:

1) **Trades limiting:** Up to a minimum number of trades, say 100,000, train the system and then test on all ensuing data. This also ensures that on days when the data is not enough (days of low volume), the model does not trade.
2) **Time limiting:** Use the intra-day data collected unto a pre-determined time of the day for training and deploy on remaining trading day.

4.5 Evaluation Criterion

ANNs are usually evaluated on root mean square error (RMSE), and a minimisation of the sum of squared errors is desired. Hereby, the different models will be compared on their average RMSE and we will choose the model that gives us minimum average RMSE over the testing dataset, the average being taken across all training days. These different models are chosen from different predefined network topologies and normalization schemes.

5 Experimental Results

5.1 Data and Neural Network

Back-testing is performed using intra-day data from September 2005. Stocks under consideration are Apple

TABLE 1
Neural Network Topology

No.	Hidden Layers	Neurons	Transfer Functions
1	3	[7 5 1]	['tansig', 'purelin', 'purelin']
2	3	[6 10 1]	['tansig', 'tansig', 'purelin']
3	2	[6 1]	['tansig', 'purelin']
4	3	[5 7 1]	['purelin', 'purelin', 'purelin']

(AAPL), Amazon (AMZN), Cisco (CSCO), E-Bay (EBAY), Google (GOOG), IBM (IBM), Microsoft (MSFT), News Corporation (NWSA), Oracle (ORCL), Time Warner (TWX) and Yahoo (YHOO). Consolidated trades from New York Stock Exchange have been used. Data has been downloaded from Wharton Research Data Services. The target security for our study is YHOO. Input and target data is component-wise linearly scaled to [0, 1]. We have chosen the neural network amongst those in table 1 which gave us the minimum mean squared error on the training set.

There is nothing sacrosanct about the above chosen neural network topologies nor the linear normalisation used.

5.2 Trading Rules

Each trade consists of a combination of long and short positions based on the whether the target security is overpriced or underpriced with respect to the ANN computed fair value. Standard deviation of the target security price is calculated(say σ). As explained in section 3.3 trades are opened only once and we refrain from trading continuously. The volume of trade would be restricted to only one unit of target security and the corresponding components. 80% of a day's data is used for training while the ANN is deployed for the remaining 20% of the day. Only unique feasible vectors are used in the training. Fixed point approximation (see appendix C) is used. The two models tested are:

5.2.1 Model 1

Trades are opened when the difference between the target price and the computed fair value exceeds 1.25σ and closed when the computed fair price and the target price come within 0.75σ.

5.2.2 Model 2

Trades are opened when the difference between the target price and the computed fair value exceeds 1σ and closed when the computed fair price and target come within 0.75σ.

5.3 Results

Results are presented without transaction costs for both short and long trades. Furthermore, for sake of convenience, it is assumed that quantity of shares bought may take continuous real values instead of integer values. Results consist of the *number of trades, average profit per trade, standard deviation of profit, percentage of profitable trades, maximum profit and loss, average length of trade, maximum draw-down, maximum money put in, and the neural network chosen*. Results are compiled in tables 2 and 3 for model 1 and 2 respectively.

It is noted that the strategy generates net profit on most number of days, albeit the maximum money put in is fairly high. Also, as we make the conditions to be met for trading more and more stringent, the number of trades reduce, which explains the greater number of trades in model 2. 22nd September, 2005 is particularly noticeable because of the high volume of trades committed. On this particular day, the target security (YHOO) was very volatile and, hence, the conditions for trading were met several times. A threshold on the maximum tolerable volatility can be employed to keep such excessive trading in check. Moreover, another interesting observation is that the neural network with parameter set 4 (see table 1) is, in effect, a linear regressor, as all its transfer functions are *'purelin'*. It is the network with the minimum MSE for several days, which indicates that on some days, there exists a strong linear relationship between the price movements of the target security and the rest of the sector throughout the day.

6 Conclusion and Further Work

Strategy similar to ETF arbitrage has been developed for sector trading, where a linear pre-specified pricing relationship does not exist. Local linear pricing relationship has been developed between the input and target security. The experimental results (see tables 2 and 3) are encouraging though extensive back-testing is advised. Even after accounting for transaction cost, like commissions and slippage costs, this strategy is likely to be profitable. With low average trade duration, high percentage of profitable trades, and ease of deployment this strategy is well suited for automated trading systems.

Further work may look into optimal network topology or use of more reliable proxies for macro-economic news impact. Neuronal parameters of the model which adapt best to the intra-day pricing relationships, using parameter space exploration may be further investigated. Clearly, the strategy can be applied to any collection of stocks bearing strong correlation with the target security.

Appendix A
Calculation of derivative using discrete data

Let F: $\Re^n \to \Re$ be the neural network function. Directional derivative at a point u, in the direction of ψ is defined as:

$$dF(u,\psi) = \lim_{\tau \to 0} \frac{F(u+\tau\psi) - F(u)}{\tau} \tag{10}$$

TABLE 2
Results: Fixed Point Approximation - Model 1

Date	1	2	6	7	8	9	12	13	14	15	16	19	20	21	22	23	26	27	28	29	30
Trades	3	2	0	0	1	2	1	1	1	2	0	1	5	4	209	1	6	2	1	0	1
Avg Profit/Trade	0.12	0.03	NaN	NaN	1.24	0.30	0.62	0.06	0.06	0.15	NaN	0.43	0.19	0.58	0.86	0.14	0.07	-0.08	0.04	NaN	0.01
Std. Profit $\times 10^{-3}$	2.24	0.37	0.00	0.00	9.10	3.97	4.13	0.45	0.43	1.58	0.00	3.18	3.22	9.04	111.16	1.03	1.25	1.39	0.29	0.00	0.09
Percentage Profitable	100	100	NaN	NaN	100	100	100	100	100	100	NaN	100	100	100	98.56	100	100	50	100	NaN	100
Max Profit	0.25	0.03	0.00	0.00	1.24	0.44	0.62	0.06	0.06	0.20	0.00	0.43	0.32	0.82	2.24	0.14	0.10	0.02	0.04	0.00	0.01
Max Loss	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.30	0.00	0.00	0.18	0.00	0.00	0.00
Average Trade Duration (sec)	351.67	46.50	NaN	NaN	0.00	5.00	0.00	14.00	39.00	0.00	NaN	0.00	0.20	56.25	1.18	2.00	54.00	49.50	3.00	NaN	7.00
Max Drawdown	0.10	0.02	0.00	0.00	0.00	0.10	0.00	0.05	0.17	0.13	0.00	0.00	0.00	0.15	1.56	0.04	0.04	0.84	0.01	0.00	0.01
Max money	130.39	48.41	0.00	0.00	29.88	42.37	167.60	137.97	138.46	607.62	0.00	41.42	11.85	122.87	125.55	59.97	29.80	180.41	84.34	0.00	48.99
ANN chosen	1	2	4	2	4	4	1	3	3	3	1	1	4	2	4	4	4	4	1	3	2

TABLE 3
Results: Fixed Point Approximation - Model 2

Date	1	2	6	7	8	9	12	13	14	15	16	19	20	21	22	23	26	27	28	29	30
Trades	7	2	1	0	46	2	7	0	0	2	3	1	5	9	311	1	12	3	4	0	0
Avg Profit/Trade	0.10	0.03	0.06	NaN	0.07	0.30	0.71	NaN	NaN	0.08	-0.01	0.43	0.19	0.33	0.73	0.14	0.03	-0.07	0.03	NaN	NaN
Std. Profit $\times 10^{-3}$	3.51	0.56	0.47	0.00	11.32	3.97	21.90	0.00	0.00	0.89	0.42	3.18	3.22	9.76	121.53	1.03	0.99	1.59	0.45	0.00	0.00
Percentage Profitable	85.71	100.00	100.00	NaN	86.96	100.00	71.43	NaN	NaN	100.00	66.67	100.00	100.00	77.78	98.71	100.00	91.67	33.33	100.00	NaN	NaN
Max Profit	0.38	0.06	0.06	0.00	1.47	0.44	2.70	0.00	0.00	0.12	0.02	0.43	0.32	0.83	2.24	0.14	0.08	0.03	0.04	0.00	0.00
Max Loss	0.00	0.00	0.00	0.00	0.12	0.00	0.33	0.00	0.00	0.00	0.05	0.00	0.00	0.16	1.30	0.00	0.01	0.20	0.00	0.00	0.00
Average Trade Duration (sec)	36.29	1.00	24.00	NaN	20.96	5.00	18.29	NaN	NaN	0.00	6.00	0.00	0.20	60.78	1.64	2.00	56.83	38.00	15.50	NaN	NaN
Max Drawdown	0.06	0.01	0.00	0.00	0.26	0.10	0.55	0.00	0.00	0.12	0.05	0.01	0.00	0.41	1.56	0.04	0.12	0.78	0.03	0.00	0.00
Max money	181.79	46.03	73.71	0.00	197.80	42.37	1041.59	0.00	0.00	290.30	100.46	26.90	11.85	236.89	125.55	59.97	29.75	180.45	12.47	0.00	0.00
ANN chosen	3	3	4	2	2	4	2	3	3	3	3	2	4	2	4	4	4	4	1	3	1

We need directional derivatives along the coordinate axis. We would need a suitable value of h which we can substitute in equation 13 to compute γ_i.

$$\vec{e_i} = (\underbrace{0, 0, \ldots,}_{i-1} 1, \underbrace{0, 0, \ldots, 0}_{n-i})^T \tag{11}$$

$$\gamma_i = dF(\vec{u}, e_i) \tag{12}$$

$$= \lim_{h \to 0} \frac{F(\vec{u} + h\vec{e_i}) - F(\vec{u})}{h} \tag{13}$$

Appendix B
Creation of feasible data vectors

Feasible data vectors are a set of true prices that have at some point in time been achieved. Their construction is as follows:

1) Wait till at least one trade is committed for each input and output variable. No data vector would be available before that point in the day.
2) The first data vector would hold the latest traded price of each input and output data variable.
3) For each subsequent trade, where there is a change

in traded security price, we would generate a new data vector, with the updated value of the traded stock.

Appendix C
Linear approximation

- **Localised approximations**: The numerical derivatives are calculated at each point while trading. This results in the change of the derivative and thus, the calculated relationships are local to only the point. Between opening and closing of a series of trades we could be looking at different coefficients or a different local pricing relationship. Yet, at all points there has been a mispricing, allowing us to explore market inefficiency.
- **Fixed Point approximations**: The numerical derivatives are calculated only once for each arbitrage trade sequence, at the point where trade is opened, and the derivatives are kept constant until the trade is eventually closed. This assumes that the values of the derivatives remains constant for an entire trade duration. The fixed point approximation may not be the best pricing relationship for the entire duration of the trade; yet, this could lead to useful insights and simplifications.

References

Ersoy, O. (1990). Tutorial at hawaii international conference on systems sciences.

Gatev, Evan, Goetzmann, N. William, Rouwenhorst, and K. Geert (2006). Pairs trading: Performance of a relative-value arbitrage rule. *Review of Financial Studies 19*(3), 797–827.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural Networks 2*(5), 359–366.

Huang, Z., H. Chen, C.-J. Hsu, W.-H. Chen, and S. Wu (2004). Credit rating analysis with support vector machines and neural networks: a market comparative study. *Decision Support System 37*(4), 543–558.

Jasic, T. and D. Wood (2004). The profitability of daily stock market indices trades based on neural network predictions: Case study for the s&p 500, dax, topix and ftse in the period 1965-1999. *Applied Financial Economics 14*, 285–297.

Kaastra, I. and M. S. Boyd (1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing 10*(3), 215–236.

Katz, J. (1992, April). Developing neural network forecasters for trading. *Technical Analysis of Stocks and Commodities*, 58–70.

Kuan, C.-M. and T. Liu (1995, Oct.-Dec.). Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of Applied Econometrics 10*(4), 347–64.

Levenberg, K. (1944). A method for the solution of certain nonlinear problems in least squares. *Quarterly of Applied Mathematics* 2(2).

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Industr. Appl. Math. 11*(1), 431–444.

Pendharkar, P. C. (2005). A threshold-varying artificial neural network approach for classification and its application to bankruptcy prediction problem. *Comput. Oper. Res. 32*(10), 2561–2582.

Soros, G. (1994). *The Alchemy of Finance: Reading the Mind of the Market*. John Wiley & Sons.