Using Artificial Neural Networks to generate trading signals for crude oil, copper and gold futures

Lukas Schulze-Roebbecke, 23291

A Project carried out under the supervision of:

Professor Joao Pedro Pereira

07.01.2016

Abstract

In this thesis, a feed-forward, back-propagating Artificial Neural Network using the gradient descent algorithm is developed to forecast the directional movement of daily returns for WTI, gold and copper futures. Out-of-sample back-test results vary, with some predictive abilities for copper futures but none for either WTI or gold. The best statistically significant hit rate achieved was 57% for copper with an absolute return Sharpe Ratio of 1.25 and a benchmarked Information Ratio of 2.11.

Table of Contents

All Appendices are to be found in the accompanying separate file.

List of figures and tables

List of abbreviations

ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
BDI	Baltic Dry Index
CL1	Generic first WTI future
CME	Chicago Mercantile Exchange
COMEX	Commodity Exchange
FX	Foreign Exchange
GA	Generic Algorithm
GC1	Generic first gold future
HG1	Generic first copper future
ICE	Intercontinental Exchange
LMEX	London Metal Exchange Index (base metals)
NYME	New York Mercantile Exchange
PCA	Principal Component Analysis
RBF	Radial Basis Function
S&P 500	Standard & Poor’s 500 Index
S&P GSCI	Standard & Poor’s (Goldman Sachs) Commodity Index
T-Bill	Treasury Bill
TOPIX	Tokyo Stock Price Index
US	The United States of America
WTI	West Texas Intermediate

1 Introduction

1.1 Scope of the thesis

Over the past thirty or so years, Machine Learning techniques have increasingly been applied in the field of financial forecasting, albeit to varying success. Machine Learning, although being a field of computer science, involves many statistical and mathematical procedures and can be defined as follows.

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" (Tom M. Mitchell 1997).

This implies a program that will learn, both from its mistakes and its successes, without being explicitly programmed to do so by a human being, a very compelling property for financial forecasting. One of the most widely used of these techniques are Artificial Neural Networks. Being extremely efficient at classification and pattern recognition, neural networks exhibit many of the properties of a potentially successful financial forecasting model. They are flexible, can approximate highly non-linear functions and are able to proficiently process huge data sets, both in set length and number of features.

WTI, gold and copper are by far the most traded futures contracts in their respective categories (CME Group 2015). WTI acts as one of the leading global benchmarks in the energy sector. Gold, as a precious metal, is mainly used as wealth storage in the form of bullions or jewellery but possesses several unique properties that make it attractive for industrial applications[1]. Copper on the other hand forms one of the most essential

[1] Gold is an excellent conductor of electricity, doesn't corrode, is chemically unreactive and reflects electromagnetic radiation (RSC, 2015).

industrial metals and is used widely in areas ranging from electronics and wiring over industrial machinery to construction (RSC 2015).

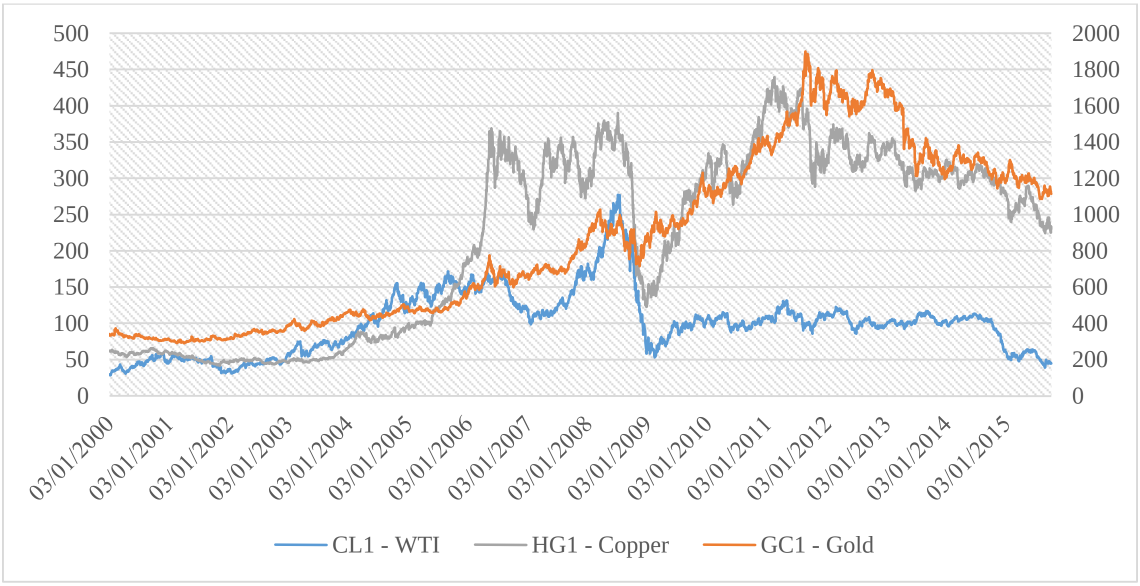

Figure 1 - The development of WTI, gold and copper (generic first) futures, 2000 - 2015. Gold on the secondary axis.[2]

Historically, many commodities are lowly or negatively correlated with both stocks and bonds, making them an attractive asset in any portfolio allocation (Gary Gorton and K. G. Rouwenhorst 2006). However, a traditional long-only roll-over investment in the above three commodities might not be the best approach. Not only have futures prices been highly volatile over the last decade or so and declined significantly in the recent past (Figure 1). But commodities prices, in particular copper and oil, are expected to remain at low levels for the foreseeable future due to supply gluts and weak demand (IMF 2015). It therefore stands to reason to investigate the use of an advanced active trading approach such as neural networks to the commodities futures market to still be able to profit from its liquidity and asset-correlations while maximizing returns.

[2] Figure: Own, data retrieved from Bloomberg.

1.2 Structure of the thesis

First, the history of ANNs as well as research of their use in the field of financial forecasting will be discussed. This is followed by a description of the mathematical principles of neural networks and in particular the feed-forward multilayer perceptron. Potential shortcomings and solutions will also be examined. Finally, the specific model used in this thesis as well as its parameters, the data and the back-test procedure will be explained. Subsequently, the results will be discussed and potential suggestions for improvement presented.

2 Artificial Neural Networks

2.1 History

The first theoretical foundations for Artificial Neural Networks were laid by Warren S. McCulloch and Walter Pitts (1943) who developed a mathematical model of neural activity in the brain based on threshold values of the individual neurons. Shortly thereafter, in 1949, in his highly influential book "The Organization of Behaviour", D.O. Hebb introduced the idea of neural plasticity, i.e. that neural connections in the brain are non-static. Webb inferred that learning consists of a strengthening of these connections (D. O. Hebb 1949).

Both ideas, although originally intended to serve in the field of neuropsychology, were essential for the development of Artificial Neural Networks. Based on this research, F. Rosenblatt developed the first single-layer neural network, which he coined a 'perceptron', in 1958 and built a physical machine used for image recognition (F. Rosenblatt 1962). However, a single-layer perceptron can only learn linearly separable problems, a limitation which was famously proofed by M. Minsky and S. A. Papert (1969) using the XOR function as an example. Although they pointed out that three-layered perceptrons are indeed capable of learning non-linear functions, specifically convex decision boundaries, public interest in and funding for research in the area declined subsequently.

Throughout the 1980s, with the advent of the greater computational power needed in the calculations of multi-layer neural networks and the invention of the backpropagation algorithm using gradient descent (P. Werbos 1974), research interest in ANNs increased again. Notable applications were character and speech recognition and autonomous driving (Dean A. Pomerleau 1991). Eventually, Kurt Hornik (1991) proofed that "standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and non-constant activation function are universal approximators […] provided only that sufficiently many hidden units are available". More recently, research has centred on Deep Learning, i.e. the use of many-layered Artificial Neural Networks, and Artificial Intelligence e.g. by Google Inc. (Tom Simonite 2012).

2.2 ANNs in Finance

Due to their role as universal approximators and their effectiveness in pattern recognition, ANNs have been widely used in the financial industry since ca. the early 1990s. "[Apart] from the U.S. Department of Defence, the financial services industry has invested more money in neural network research than any other industry or government body" (Robert R. Trippi and Efraim Turban 1993).

Notable research includes economic time series forecasting, both qualitatively and quantitatively, and stock selection (Joarder Kamruzzaman, Rezaul Begg, and Ruhul A. Sarker 2006). T. Kimoto et al. (1990) develope an ANN for buying and selling signal predictions for the TOPIX and achieve a 18.6% profit premium to a buy-and-hold strategy. H. Ahmadi (1990) uses a backpropagation ANN with gradient descent algorithm to estimate the factors of an Arbitrage Pricing Model for various securities. More recently, Ray Tsaih, Yenshan Hsu, and Charles C. Lai (1998) use an ANN approach to predict the direction of daily price changes of S&P 500 futures and Manfred Steiner and Hans-Georg Wittkemper (1997) apply an ANN to portfolio optimization, specifically to predict stock return distributions. While all of these research publications succeed in producing statistically significant predictions, results are often less than outstanding. This may in part be due to the extremely noisy nature of financial data and their inherent non-stationarity (Kyoung-jae Kim and Ingoo Han 2000). The same authors also argue that backpropagation coupled with gradient descent, the optimization procedure used in the majority of early research papers, poses a significant problem by often finding only a local as opposed to the global minimum. On the other hand, although not specifically referring to the financial field, Andrew Ng (2015) states that local optima are often less of a problem in practice than stipulated in theoretical works. Nonetheless, current research in financial time series forecasting now often favours more advanced optimization procedures over simple backpropagation / gradient descent learning algorithms such as combined GA/ANN approaches (Kim and Han 2000; Kyoung-jae Kim 2006) or fuzzy neural networks (Guido Deboeck 1994; Arnold F. Shapiro 2002; Mehdi Khashei, Seyed Reza Hejazi, and Mehdi Bijari 2008).

2.3 ANNs and the forecasting of commodities prices

There has been some research into the use of ANNs for forecasting the prices or directional movements of selected commodities. In an early study, Nowrouz Kohzadi et al. (1996) compare a feed forward neural network and traditional ARIMA processes at forecasting a large sample of monthly live cattle and wheat prices in the US. They conclude that the neural network is superior to ARIMA approaches in both the prediction error rates and the statistical validity of directional forecasts. Interestingly, they circumvent the argument of the efficient market hypothesis by arguing that, according to chaos theory, even a seemingly random time series might be generated by a deterministic function, requiring non-linear approaches such as ANNs. By far the most studied commodity market is oil. William E. Shambora and Rosemary Rossiter (2007) use lagged moving averages to predict the directional movement of oil futures and achieve a 53% hit rate, with the ANN outperforming alternative strategies such as buy-and-hold or a simple T-Bill investment. Concentrating on the prediction of crude oil spot prices, Imad Haidar, Siddhivinayak Kulkarni, and Heping Pan (2008) reach high hit rates of above 70% with, however, quite low R^2s. Other researchers use cascaded neural networks (Farooq Malik and Mahdi Nasereddin 2006) or an crisis index in an effort to eliminate the significant price volatility of oil (A. Alizadeh and Kh. Mafinezhad 2010). More sophisticated approaches include Empirical Mode Decomposition with an ensemble of neural networks used for forecasting (Lean Yu, Shouyang Wang, and Kin K. Lai 2008) and, similarly, a wavelet decomposition method (Rania Jammazi and Chaker Aloui 2012). Both show impressive results[3]. In summary, it can be said that most research on the use of artificial neural network for forecasting commodities prices find the method superior to traditional approaches while individual results, in particular hit rates of directional movement prediction, vary widely. It should also be mentioned that many papers on the topic use

[3] The latter also argue for the use of a simple hyperbolic tangent activation function, citing Occam's razor, and emphasize the importance of an ANNs topography as well as it's sensitivity towards the size of training and test windows.

spot prices as opposed to futures prices, possibly hampering an actual trading implementation.

3 The working of ANNs

3.1 Principles

Artificial Neural Networks try to mimic the way in which the human brain operates[4]. They consist of an input layer, an output layer and any number of hidden layers in between. A layer consists of a certain number of neurons and each neuron is connected to the neurons of the following layer. The size of the input and output layers equal the lengths of the feature vector and output vector respectively while the size of the hidden layer is chosen by the programmer. Each neural connection is represented by a weight and at each neuron the signal is subjected to an activation function. ANNs are routinely categorized by their topography and the learning paradigm used. The by far most common type is the feed-forward backpropagation neural network. This means that signals pass through the network in one direction only, from input layer towards output layer. The network is actively taught by presenting data with known output. This is called supervised learning.

In this thesis, the type of network used is a Multilayer Perceptron, i.e. a feedforward network that utilizes the backpropagation algorithm. Consequently, this topography and the gradient descent learning algorithm are described in the following paragraphs. A simplified two-layer perceptron is illustrated in Appendix E.

[4] See Appendix A for further information.

3.2 The Multilayer Perceptron[5]

The application of an Artificial Neural Network typically consists of two phases. First the network is taught using a training data set containing instances of known features and their respective outcomes (supervised learning). Afterwards, the resulting optimized weights are used to predict the outcomes of a test data set. The learning phase is divided into forward propagation and back propagation. During forward propagation, the inputs for each neuron are multiplied by their respective weights, summed up and fed through the activation function. This output then serves as the input for the neurons of the next layer and the process is repeated until the output layer is reached (compare Equation 1). The final output is also called the hypothesis of the input vector x.

$$a^1 = x \rightarrow z^2 = \Theta^1 a^1 \rightarrow a^2 = g(z^2) \rightarrow [\ldots] \rightarrow z^L = \Theta^{L-1} a^{L-1} \rightarrow a^L = g(z^L) = h_\Theta(x) \qquad (1)$$

$a, z = feature\ vectors$

$\Theta = weight\ matrix\ of respective\ layer$

$g(z) = activation\ function$

$h_\Theta(x) = hypothesis\ function$

$L = number\ of\ layers$

The hypothesis is compared to the desired or target output and its mean squared error or cost is computed.

$$cost = J(\Theta) = \frac{1}{2M} * \sum_{k=1}^{K} (h_\Theta(x)_k - y_k)^2 \rightarrow \min_\Theta J(\Theta) \qquad (2)$$

$K = number\ of\ output\ neurons$

$y = desired\ output\ /\ target$

During backpropagation, the cost function $J(\Theta)$ is minimized over the set of weights Θ for the training set using the gradient descent algorithm. The gradients, i.e. the partial

[5] The procedures and formulas described in this chapter follow Ng (2015) if not otherwise stated.

derivatives of the cost function with respect to every weight $\frac{\partial J(\Theta)}{\partial \Theta_{ij}^{l}}$, are multiplied by a learning rate and subtracted from the initial weights and the process is repeated. Intuitively, the gradients point towards a decrease of the cost function surface in the multi-dimensional space spanned by the model parameters, i.e. the weights. Therefore, the gradient descent algorithm repeatedly takes minute steps in that direction to reach a minimum. In practice, the gradients are calculated by computing an ‘error signal’ at every output neuron and back-propagating it through the network. The error at the output layer consists of the gradient of the cost function weighted by the gradient of the activation function. Therefore, the error at each node k of the output layer can be written as follows.

$$\delta_k^L = g'(z_k^L) * J'(\Theta) = g'(z_k^L) * (a_k^L - y_k) \qquad (3)$$

$\delta_k^l = error\ at\ neuron\ k\ in\ output\ layer\ L$

$z = (unactivated)\ input\ into\ node\ k\ in\ layer\ L$

Note that the final output a_k^L is a function of the previous layer’s activated output and can therefore be written as the hypothesis function $h_\Theta(x)$. The errors at the hidden and input neurons can be written as follows.

$$\delta_i^l = g'(z_i^l) * \sum_{j=1}^{J} \Theta_{i,j}^l \delta_j^{l+1} \qquad (4)$$

$J = number\ of\ neuron\ in\ the\ following\ layer$

$\Theta_{i,j}^l = weight\ of\ connection\ from\ neuron\ i\ to\ neuron\ j\ in\ layer\ l$

Thus, for every neuron in layer i, the error at every (connected) neuron in the following layer is multiplied by the corresponding weight, summed up and, as above, weighted with the gradient of the activation function. This process is repeated until the input layer is reached. Given these errors, the gradient of the cost function with respect to every weight can then be calculated as follows.

$$\frac{\partial J(\Theta)}{\partial \Theta_{ij}^{l}} = a_j^l * \delta_i^{l+1} \tag{5}$$

The weights are then updated by subtracting the gradient multiplied with a learning rate determined by the programmer (Equation 6).

$$\Theta_{ij}^{l} \leftarrow \Theta_{ij}^{l} - \eta * \frac{\partial}{\partial \Theta_{ij}^{l}} J(\Theta) \Rightarrow \Delta\Theta_{ij}^{l}(t) = -\eta * \frac{\partial}{\partial \Theta_{ij}^{l}(t)} J\big(\Theta(t)\big) \tag{6}$$

$\eta = learning\ rate$

In batch learning, the method used in this thesis, the gradients are summed up and averaged over the whole training set before the weights are updated. This process, also called an Epoch, is then repeated surreptitiously. While batch learning can be very computing-expensive for large datasets with many different features, it has the advantage that the error is strictly decreasing with every epoch until convergence.

3.3 Problems with ANNs and the gradient descent algorithm

Neural networks and in particular the gradient descent algorithm face a variety of issues. ANNs in general are by nature prone to overfitting. That is, given a sufficiently complex network, the weights will be optimized for the given training data during the learning phase, but the network will fail to generalize well to unknown data. Colloquially, the ANN is said to 'memorize' the training set. On the other hand, under fitting the network has to be avoided. Appendix B illustrates the problem for a one-dimensional feature vector and target value. Two methods were applied in this thesis to prevent overfitting, regularization and cross-validation.

Regularization involves adding a regularization term to the cost function. This term consists of the sum of all weights multiplied by a regularization parameter lambda to effectively penalize large weights. When the cost function is minimized with respect to the weights, the constant lambda prevents excessive values for the weights. Following Ng (2015), the cost function from Equation 2 is therefore amended as follows.

$$cost = J(\Theta) = \frac{1}{2M} * \sum_{k=1}^{K} (y_k - h_\Theta(x)_k)^2 + \frac{\lambda}{2M} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ij}^{l})^2 \qquad (7)$$

$\lambda = regularization\ parameter$

$L = number\ of\ layers$

$s_l = number\ of\ neurons\ in\ layer\ l$

Lambda constitutes a global parameter and is to be determined by the programmer. Note that an excessively large value for lambda will result in high bias and the network will not produce reasonable results.

In addition to regularization, a cross-validation approach, specifically K-fold cross-validation, was applied. The training data was randomized and split into K folds. K-1 folds were then used to train the network while one fold was used to test its predictive power on the unknown validation data set. Training was aborted if the error of the validation set started to increase. In particular, training was aborted as soon as the error of the validation set exceeded the error of 50 epochs prior. This first-difference approach was chosen over a moving average approach to accommodate for local minima. Otherwise, training was aborted after a maximum of 5000 epochs. Ng (2015) recommends a value of K equal to five while Kevin L. Priddy and Paul E. Keller (2005) suggest a value between five and ten. While the training set was relatively small, $K = 6\frac{2}{3}$ was chosen to ensure a sufficiently large validation set.

The gradient descent algorithm attempts to move towards the global minimum of the cost function in regards to the weights. If, however, the error surface is complex and the cost function possesses local minima, the algorithm will get ‘trapped’ and terminate early even though no global minimum has been reached. In this thesis, a momentum strategy was applied to prevent the network from converging to a small local minimum. At each weight update, a fraction of the previous epoch's update term is added to the sum. Following Genevieve Orr (1999) and Tijmen Tieleman (2014), a momentum term is added to Equation 6 as follows.

$$\Delta\Theta_{ij}^{l}(t) = -\eta * \frac{\partial}{\partial\Theta_{ij}^{l}(t)} J\big(\Theta(t)\big) + \alpha * \Delta\Theta_{ij}^{l}(t-1) \qquad (8)$$

$\alpha = momentum\ parameter$

Adding momentum has the further advantage of speeding up convergence as it dampens the oscillation of successive gradients with opposite signs (Orr 1999). In this thesis, to avoid another global parameter and accompanying trial and error, alpha is initialized with a value of 0.1. It is then increased incrementally by 0.1 after every 100 epochs to a maximum value of 0.99. This approach follows suggestions by Tieleman (2014).

The second drawback of the gradient descent algorithm is the need to determine a suitable value for the learning rate. If the rate is too small, convergence will take very long. If it is too large, the weights will oscillate but eventually converge, again with a high demand of time and computational power. If the learning rate is twice the optimal rate, the weights will diverge (Genevieve Orr and Klaus-Robert Müller 1998) (See Appendix C for a graphical representation of these three cases as well as the case of an optimal learning rate).

As the magnitude of the gradients of the individual weights can have a very high variance, most practitioners recommend to use individual and adaptive rates instead of a global parameter (Orr and Müller 1998). In this thesis, following Tieleman (2014), an initial global learning rate was multiplied during each epoch by an empirically determined local gain. The weight update term from Equation 8 is therefore amended as follows.

$$\Delta\Theta_{ij}^{l}(t) = -\eta * g_{ij} * \frac{\partial}{\partial\Theta_{ij}^{l}(t)} J\big(\Theta(t)\big) + \alpha * \Delta\Theta_{ij}^{l}(t-1) \qquad (9)$$

$g_{ij} = gain\ for\ neuron\ i \rightarrow j$

The gain is initialized as one and then calculated as follows for each epoch.

$$if \quad \left(\frac{\partial}{\partial\Theta_{ij}^{l}(t)} J\big(\Theta(t)\big)\right) * \big(\alpha * \Delta\Theta_{ij}^{l}(t-1)\big) \ > \ 0 \qquad (10)$$

$$then \quad g_{ij}(t) = g_{ij}(t-1) + 0.1 \qquad (11)$$

$$else \quad g_{ij}(t) = g_{ij}(t-1) * 0.9 \qquad (12)$$

Note that above equation uses an agreement in sign of current gradient and momentum of each weight to determine the gain. Intuitively, the learning rate is slightly increased if the weights move into the ‘right’ direction and significantly decreased if momentum and gradient have different signs, i.e. momentum points toward the ‘wrong’ direction.

Finally, an often voiced critique of ANNs is their black box nature. For sufficiently complex neural networks, it can be very hard to reconstruct how the algorithm reaches an output value y for a given input vector x. However, one of the purposes of an algorithmic trading system is to decouple human emotions and whims from trading decisions. A black box algorithm might therefore be useful in preventing existing knowledge and opinions of the trader from interfering with the trading decisions. Furthermore, the assumption of non-stationary financial markets and changes in their correlation structure over time leads by definition to non-constant decision-making rules of the network. Knowing the exact process for one week might be of little use in the following week. Nonetheless, there exist several methods to extract knowledge from a trained ANN such as rule extraction (Robert Andrews, Joachim Diederich, and Alan B. Tickle 1995). None of these methods were applied in this thesis in line with above arguments and due to the complexity of rule extraction in general.

4 The ANN model

4.1 The Activation Function

Following Orr and Müller (1998), a sigmoidal function, specifically the modified hyperbolic tangent function shown in Equations 13, was chosen as the activation function. Being symmetric over the origin, it has a higher probability to produce outputs close to zero than the standard logistic function. The constants are added to result in outputs with a variance of close to one (Orr and Müller 1998).

$$g(z) = 1.7159 * tanh(\frac{2}{3} * z) \tag{13}$$

$$g'(z) = 1.1439 * \ sech(\frac{2}{3} * z)^2 \tag{14}$$

Note that the maxima of the second derivative of the Activation Function occur at one and negative one[6], ensuring that the output is not maximized along the asymptotes during training. This would drive the weights towards increasingly large values, resulting in smaller and smaller weight updates (Orr and Müller 1998).

[6] a plot of the function and its first derivative can be found in Appendix D.

4.2 The Data and target values

The dataset consisted of a mixture of fundamental features and technical signals of daily frequency spanning a timeframe from the 31.08.1992 to the 01.10.2015. A daily frequency was chosen to provide a large enough data set while avoiding the noisiness of intra-day data. The timeframe was limited by the availability of data. The fundamental features were (generic first) futures on a US dollar currency basket (DX1 at ICE) and on the S&P 500 stock index (SP1 at CME) as well as a US government three-month treasury bill index (USGG3M). Additionally, for gold and copper, the Baltic Dry Index[7] (BDIY) as well as the respective COMEX warehouse stocks (COMXGOLD and COMXCOPR) were included. The Baltic Dry Index measures the cost of marine transport of a variety of bulk commodities while the COMEX warehouse indices measure the inventory levels of the respective commodity at the NYME, formerly the COMEX. These time series were de-trended by taking the one-day log-returns and five and 20-day moving averages were added. The technical signals consisted of either negative one (sell), zero (no action) or positive one (buy). They included the five and 20-day momentum as well as 20/2 Bollinger bands of the three investigated futures contracts, i.e. the (generic first) futures for WTI (CL1 at NYME), gold (GC1 at COMEX) and copper (HG1 at COMEX). All data was retrieved from Bloomberg Terminal using the above tickers. Overall, this resulted in 14 and 20 features for WTI and copper and gold respectively, spanning a timeframe from the 01.10.1992 to the 30.09.2015 (6000 instances).

The training data was pre-processed by mean-normalization and feature scaling (Equations 15 and 16). Convergence is faster if each feature's mean is close to zero and

[7] In terms of WTI, no data was available for the Baltic Dirty Index until December 2001. Furthermore, storage levels data as published by the US Energy Information Agency are of weekly frequency. Consequently, these time series were omitted in the model.

all features, assuming an equal (or unknown) importance, are of roughly the same scale (Orr and Müller 1998).

$$x_{norm} = x - \frac{\sum_{i=1}^{m} x_i}{m} \tag{15}$$

$$x_{scale} = \frac{x_{norm}}{\sigma_{norm}} \tag{16}$$

Following Ng (2015), Principal Component Analysis (Equations 17 and 18) was then applied to remove linear correlations from the input data. This has the further advantage that the dimensionality of the data is reduced which improves computational speed.

$$\Sigma = VCV = \frac{1}{n} \sum_{j=1}^{n} x_{scale}^{j} x_{scale}^{j^T} \tag{17}$$

$$S = U^{-1} \Sigma U \tag{18}$$

$U = matrix\ of\ eigenvectors$

$S =\ diagonal\ matrix\ containing\ eigenvalues$

The data is reduced to K dimensions such that variance retention is above a threshold determined by the programmer. 99% was used in this thesis (Equation 19).

$$\frac{\sum_{i=1}^{K} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99 \tag{19}$$

$U_{reduced} = first\ K\ columns\ of\ U$

$x_{reduced} = U_{reduced}^{T} * x_{scale}$

Note that these steps were repeated for each training window of the data set. Given that the properties of the test set are unknown during the training, the test set was normalized, scaled and reduced by using the mean, the standard deviation and the matrix of eigenvectors determined from the training set.

The target values of the ANNs were binary signals for buy (1) and sell (-1) as well as a signal for no trading (0). The holding period was one day. To train the networks, the one-day log-return of the three futures contracts for each instance's next period was calculated and the desired values were set to one and negative one for returns above and below zero.

4.3 The network predictions and the output signal

One of the properties of artificial neural networks is that the quantitative value of the network output allows inference of the 'strength' of the signal. That is, the larger the absolute value, the 'surer' the network is of its prediction. Unfortunately, the specific sigmoid activation function used in this paper has the disadvantage that its predictions cannot be directly interpreted as probabilities. Therefore, to still be able to profit from this property of neural networks, the output signal was set to positive one (buy), negative one (sell) and zero (no trade) for predicted values above, below and between two thresholds respectively. These thresholds were set to (negative) multiples of 0.1 up to 0.4 and -0.4. Figure 2 shows an example of the raw network output and the various thresholds in coloured horizontal bars. The higher the threshold, the lower the number of predictions made and, presumably, the higher the accuracy of the network's prediction.

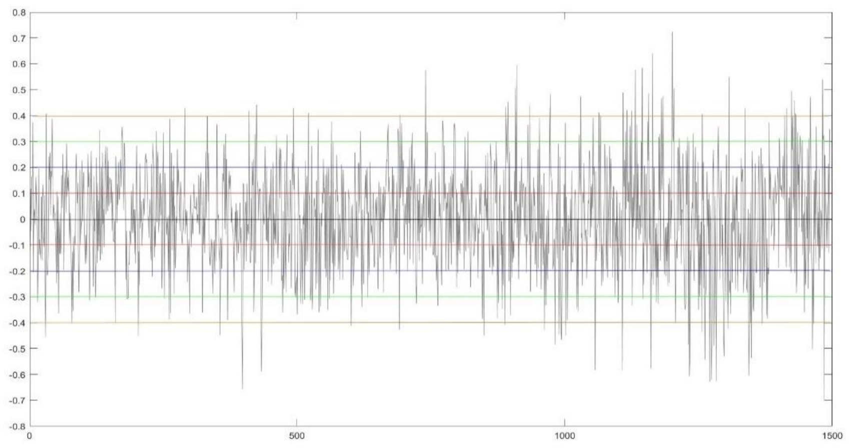

Figure 2 - Raw network output (20 nodes, window of 3000 days and lambda of 0.3) for HG1 with threshold bars

4.4 Global parameters

As discussed in Chapters 3.2 and 3.3, programming an efficient ANN involves determining an optimal value for several global parameters of the model. These are the general topography of the network, i.e. the number of hidden layers and the number of neurons per hidden layer, the length of the training and test windows and the regularization parameter lambda. There exists no analytical method to determine the optimum value for these parameters and therefore their value must be found either experimentally or by using general rules-of-thumb. In the case of the sliding learning and test windows, the values are also influenced by the nature of the data used. In this thesis, assuming a non-constant relationship of the features over time, a too large training window might result in contradictory training data[8] while a too-short window will not include enough information for reliable and stable predictions. In order to reduce the number of global parameters, the number of hidden layers and the length of the training window were set to fixed values of two and five days (one trading week) respectively.

To determine an optimized value for the other parameters, the total dataset was split roughly in two. The first half (3000 days) was then divided into training and cross-validation sets and the network was retrained weekly for the first six years (1500 days) of the second half of the data (the evaluation period). Its mean hit rate of all five thresholds was recorded and the process was repeated for different values of the global parameters[9]. This approach for finding optimized global parameters essentially follows Wei Cheng, Lorry Wagner, and Chien-Hua Lin (1996) who developed an ANN to trade US treasuries. Overall, these procedures resulted in the evaluation of 192 different network

[8] This assumes that financial markets evolve over time. For example, data that might have indicated an up-move ten years ago might indicate the opposite in present economic conditions.
[9] An overview over all tested global parameters can be found in Appendix F.

configurations over 1500 instances. The best performing network configuration for each commodity was then chosen to be used in the out-of-sample back test over the last 1500 instances of the data set.

Note that the initial weights, although not global parameters in the above sense, have a significant impact on the learning phase and therefore the effectiveness of any neural network. Ng (2015) recommends small, randomly chosen non-zero values while Orr and Müller (1998) recommend intermediately sized, randomly chosen non-zero values along the linear part of the sigmoid function. In this thesis, the weights were initialized randomly with values between negative one and one.

The Artificial Neural Network was programmed and implemented in Matlab 2015 based on the functions structure from Ng (2015). The complete code can be found in Appendix G. The data was pre- and post-processed in MS Excel.

5 Results

5.1 Global parameter optimization

As discussed in the previous chapter, the hit rates of the various networks over the evaluation period were used to determine the optimized global parameters. Intuitively, the hit rates depict the percentage of correct predictions. They are calculated by dividing the number of correct predictions (prediction equals one and actual return is above the positive threshold and prediction equals negative one and actual return is below the negative threshold) by the number of total predictions made for that threshold. To be able to compare the 64 network configurations for each commodity using a single indicator, the hit rates for the five thresholds, i.e. 0, ± 0.1, ± 0.2, ± 0.3 and ± 0.4, were then averaged.

The highest of these mean hit rates during the evaluation period as well as their global parameters are shown in Table 1.

Commodity	*Mean Hit rate*	*Neurons / layer*	*Window length*	*Lambda*
WTI	0.53	10	1500	0.30
Gold	0.53	5	2250	0
Copper	0.54	20	2250	0.03

Table 1 - Results of the global parameters optimization

5.2 Back test

The actual back tests for the last 1500 instances of the data set were conducted using the global parameters determined above. To test for the significance of the achieved hit rates, a version of Merton's test was applied. Based on Kohzadi et al. (1996), the probabilities of the forecasts were defined as follows.

$$P_1 = Prob[F_t = 1 \mid A_t = 1] \qquad (20)$$

$$P_2 = Prob[F_t = 0 \mid A_t = 0]^{10} \qquad (21)$$

$F_t = forecasted\ signal$

$A_t = actual\ signal$

According to Robert C. Merton (1981), there is sufficient evidence for a forecasting ability if the sum of P_1 and P_2 is larger than one. Accordingly, the hypothesis and the regression equation used for the test, based on Robert E. Cumby and David M. Modest (1987) and Kohzadi et al. (1996), are as follows.

$$H_0: P_1 + P_2 - 1 \leq 0 \qquad (22)$$

$$H_1: P_1 + P_2 - 1 > 0 \qquad (23)$$

$$F_t = \alpha_0 + \alpha_1 A_t + \varepsilon_t \qquad (24)$$

$\alpha_1 = P_1 + P_2 - 1$

[10] Note that the forecasted values of the model were 1, -1 and 0. As no trade was executed for 0, the respective values were dropped in the test. Furthermore, to accommodate for Merton's test, all values of -1 (both for the forecast and the target) were substituted with 0.

The results of the back-tests as well as their p-values are shown in tables two to four. The hit-rates and respective number of trades are ordered by thresholds.

WTI	*0*	*0.1*	*0.2*	*0.3*	*0.4*
Hit rate *(p-value)*	0.49 (0.61)	0.50 (0.54)	0.48 (0.71)	0.46 (0.30	0.49 (0.89)
# of trades	1495	910	473	197	85
Hit rate (buy)	0.50	0.50	0.48	0.46	0.49
Hit rate (sell)	0.48	0.50	0.50	0.44	0.53

Table 2 - Results of the back-test for WTI

Gold	*0*	*0.1*	*0.2*	*0.3*	*0.4*
Hit rate *(p-value)*	0.50 (0.13)	0.48 (0.65)	0.49 (0.86)	0.57 (0.46)	0.40 (0.78)
# of trades	1495	463	111	30	5
Hit rate (buy)	0.49	0.47	0.46	0.55	0.50
Hit rate (sell)	0.51	0.50	0.56	0.60	0.33

Table 3 - Results of the back-test for gold

Copper	*0*	*0.1*	*0.2*	*0.3*	*0.4*
Hit rate *(p-value)*	0.48 (0.68)	0.50 (0.10)	0.52 (0.01)	0.53 (0.04)	0.57 (0.01)
# of trades	1495	1036	656	379	207
Hit rate (buy)	0.47	0.50	0.51	0.55	0.55
Hit rate (sell)	0.48	0.51	0.53	0.50	0.59

Table 4 - Results of the back-test for copper

As can be seen, none of the models show any significant predictive ability without an increased threshold for the output of the networks. This is in line with some of the existing research (Cheng, Wagner, and Lin 1996; Shambora and Rossiter 2007). When thresholds are increased, only the neural network for copper exhibits reliable hit rates above 50%. The best result achieved was a 57% hit rate for the prediction of copper futures with a threshold of 0.4 and 207 trades made (i.e. ca. 14% of the total number of instances). With a p-value of 0.01, the result is also highly significant. The other two networks appear to have no statistically significant predictive ability at all. For copper, the largest difference between positive (buy) and negative (sell) hit-rates is 5% but there is no evident pattern. Therefore, the direction of the return has presumably not any large impact on the predictive abilities of the network.

Intuitively, these results make economic sense. In the case of WTI, no features for either stock levels or shipping costs were included in the model due to a lack of data. Therefore, there is by definition less possible explanatory value in the model and weaker results are to be expected. Furthermore, as the by far most traded future of the three, it stands to reason that exploitable patterns or correlations are much harder to find. Copper, as an industrial metal, is rarely used for wealth storage and less heavily traded than WTI. As opposed to gold, a much larger importance of physical features mirroring supply and demand and less dependence of the futures price on economic policy not captured by the model is plausible.

5.3 Financial descriptive statistics

The cumulative returns for the three best performing trading strategies from chapter 5.2, i.e. copper futures with three different thresholds, are illustrated in Figure 3. As determined in chapter 5.1, the sliding window size was 2250 days. The holding period was one day (see chapter 4.2) and the future was bought/sold if the network prediction was above/below ± 0.2, ± 0.3 ± 0.4 (see chapter 4.3 and Figure 2). Also added to Figure 3 were the S&P GSCI and the LMEX non-ferrous base metal indices as well as a buy-and-hold strategy. As can be seen, the ANN vastly outperforms the benchmark indices over the back-test period.

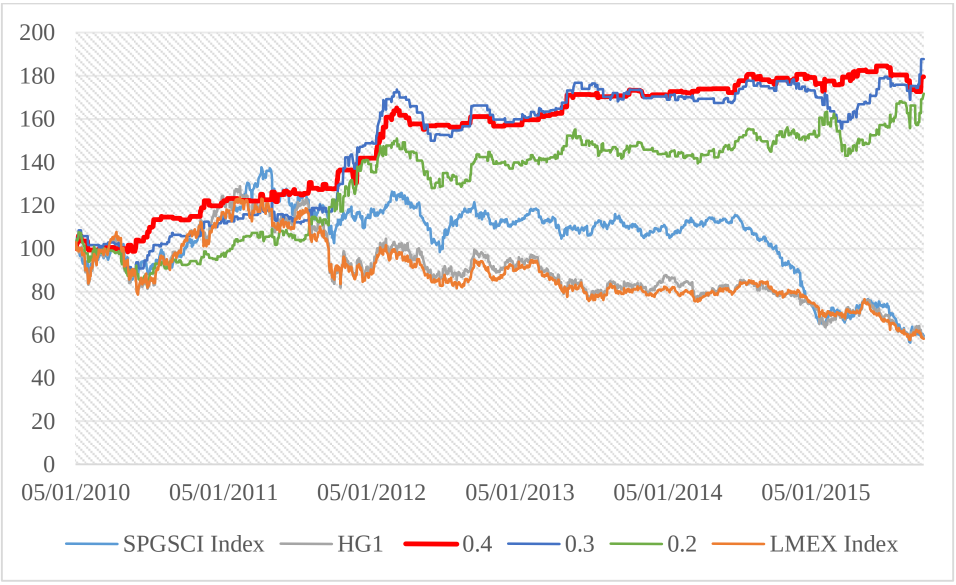

Figure 3 - Cumulative returns for 2 benchmark indices, a buy-and-hold strategy and the ANN copper trades with different thresholds

All three trading strategies possess average annual returns of around 10% with comparably low standard deviations of between 8% and 15% (Table 5). In terms of Sharpe Ratio[11], the top preforming ANN trading strategy scores 1.25.

	SPGSCI Index	*HG1*	*0.4*	*0.3*	*0.2*
Annual Return	-0.07	-0.06	0.10	0.11	0.10
Annual St. Dev.	0.19	0.23	0.08	0.11	0.15
SR	-0.37	-0.27	1.25	0.99	0.66

Table 5 - Financial Statistics and Sharpe Ratios for the top performing ANN trading strategies and two benchmarks

	ANN - GSCI			*ANN - LMEX*		
	0.4	**0.3**	**0.2**	**0.4**	**0.3**	**0.2**
Annual Return	0.17	0.18	0.17	0.17	0.18	0.17
Annual St. Dev.	0.08	0.11	0.15	0.08	0.11	0.15
SR	2.11	1.61	1.11	2.10	1.60	1.11

Table 6 - Information Ratios for the top performing ANN strategies and two benchmarks

[11] The risk free rate is omitted in this calculation, i.e. $SR = \frac{r_{ANN} - r_f}{\sigma_{ANN}}$ where $r_f = 0$.

When benchmarked against the GSCI and LMEX indices, the trading strategies fare even better, with Information Ratios[12] of up to 2.11 and 2.10 respectively (Table 6).

6 Suggestions for improvement

6.1 Improved feature selection

A more in-depth selection of relevant fundamental features might improve results significantly. Specifically, features that contain more information about actual supply and demand and storage costs and levels as well as shipping costs are promising candidates. Unfortunately, most of the freely available data, e.g. by the US Energy Information Agency in the case of WTI, are not of daily frequency and consequently an inclusion would significantly reduce the number of instances. Similarly, more momentum and mean reversal signals with varying window length might improve predictive abilities. However, it stands to reason that the type of market (trending or mean reverting) as well as the effective window length might change with time, resulting in the need to recalibrate the whole model when predictive abilities start to decline. This effect was observed by Cheng, Wagner, and Lin (1996). An elegant and efficient way to improve feature selection would be the use of an advanced optimization procedure such as a GA.

6.2 The use of intraday data

The use of intraday-bar or even high-frequency data would immensely increase the data set size available for training and cross-validation, reducing the risk of overfitting. Also, intraday data might exhibit patterns and correlations which are not present or apparent at lower frequencies. However, using intraday data would exponentially increase

[12] The Information Ratios are calculated as follows $IR = \frac{r_{ANN} - r_{benchmark}}{\sigma_{ANN}}$

computational demand. Nonetheless, this could be managed by using different learning techniques such as online or mini-batch learning.

7 Conclusion

In this thesis, a feed-forward Artificial Neural Network was developed to predict the daily directional movements of WTI, gold and copper futures. Special care was taken to avoid some of the pitfalls of neural networks, specifically overfitting and local minima. Furthermore, a system to optimize global parameters was applied to provide a viable back test and alleviate potential issues of data mining. Finally, the data was pre-processed extensively. The resulting models exhibit some predictive abilities, particularly for copper futures. However, overall hit rates are low, ranging from 47% to 57% with varying statistical significance. It is therefore the opinion of the author that a trading model based purely on feed-forward, back-propagating Artificial Neural Networks is not viable for WTI and Gold futures. However, there is evidence that such an algorithmic trading strategy could yield significant returns with low standard deviations for copper futures, particularly if benchmarked against the overall commodities and base metals markets. If over-fitting can be controlled, further improvements might be achieved by applying more complex architectures. In particular, the use of a large number of features and optimized feature selection are promising fields for further research.

References

Ahmadi, H., ed. 1990. *Testability of the arbitrage pricing theory by neural network*: IEEE.

Alizadeh, A. and Kh. Mafinezhad. 2010. “Monthly Brent oil price forecasting using artificial neural networks and a crisis index.” In *2010 International Conference on Electronics and Information Engineering (ICEIE 2010)*, V2-465-V2-468.

Andrews, Robert, Joachim Diederich, and Alan B. Tickle. 1995. “Survey and critique of techniques for extracting rules from trained artificial neural networks.” *Knowledge-Based Systems*, 8(6): 373–89.

Cheng, Wei, Lorry Wagner, and Chien-Hua Lin. 1996. “Forecasting the 30-year U.S. Treasury Bond with a System of Neural Networks.” *NeuroVe$t Journal*(1/2).

CME Group. 2015. Leading Products. Q3 2015. http://www.cmegroup.com/education/files/cme-group-leading-products-2015-q3.pdf (accessed November 18, 2015).

Cumby, Robert E. and David M. Modest. 1987. “Testing for market timing ability.” *Journal of Financial Economics*, 19(1): 169–89.

Deboeck, Guido. 1994. *Wiley finance editions, Trading on the edge. Neural, genetic, and fuzzy systems for chaotic financial markets*. New York: Wiley.

Gorton, Gary and K. G. Rouwenhorst. 2006. “Facts and Fantasies about Commodity Futures.” *Financial Analysts Journal*, 62(2): 47–68.

Haidar, Imad, Siddhivinayak Kulkarni, and Heping Pan. 2008. “Forecasting model for crude oil prices based on artificial neural networks.” In *2008 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 103–08.

Hebb, D. O. 1949. *The Organization of Behavior. A Neuropsychological Theory*. New York: John Wiley and Sons.

Hornik, Kurt. 1991. “Approximation capabilities of multilayer feedforward networks.” *Neural Networks*, 4(2): 251–57.

IMF. 2015. *World economic outlook, october 2015. Adjusting to Lower Commodity Prices*. [S.l.]: International Monetary Fund.

Jammazi, Rania and Chaker Aloui. 2012. “Crude oil price forecasting. Experimental evidence from wavelet decomposition and neural network modeling.” *Energy Economics*, 34(3): 828–41.

Kamruzzaman, Joarder, Rezaul Begg, and Ruhul A. Sarker. 2006. *Artificial neural networks in finance and manufacturing*. Hershey, PA: Idea Group Pub.

Khashei, Mehdi, Seyed Reza Hejazi, and Mehdi Bijari. 2008. “A new hybrid artificial neural networks and fuzzy regression model for time series forecasting.” *Fuzzy Sets and Systems*, 159(7): 769–86.

Kim, Kyoung-jae. 2006. “Artificial neural networks with evolutionary instance selection for financial forecasting.” *Expert Systems with Applications*, 30(3): 519–26.

Kim, Kyoung-jae and Ingoo Han. 2000. “Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index.” *Expert Systems with Applications*, 19(2): 125–32.

Kimoto, T., K. Asakawa, M. Yoda, and M. Takeoka, ed. 1990. *Stock market prediction system with modular neural networks*.

Kohzadi, Nowrouz, Milton S. Boyd, Bahman Kermanshahi, and Iebeling Kaastra. 1996. "A comparison of artificial neural network and time series models for forecasting commodity prices." *Neurocomputing*, 10(2): 169–81.

Malik, Farooq and Mahdi Nasereddin. 2006. "Forecasting output using oil prices. A cascaded artificial neural network approach." *Journal of Economics and Business*, 58(2): 168–80.

McCulloch, Warren S. and Walter Pitts. 1943. "A logical calculus of the ideas immanent in nervous activity." *The Bulletin of Mathematical Biophysics*, 5(4): 115–33.

Merton, Robert C. 1981. "On Market Timing and Investment Performance. I. An Equilibrium Theory of Value for Market Forecasts." *The Journal of Business*, 54(3): 363.

Minsky, M. and S. A. Papert. 1969. *Perceptrons. An Introduction to Computational Geometry*. Cambridge: M.I.T. Press.

Mitchell, Tom M. 1997. *McGraw-Hill series in computer science, Machine Learning*. New York: McGraw-Hill.

Ng, Andrew. 2015. *Machine Learning, Advice for Applying Machine Learning. Model Selection and Train/Validation/Test Sets*. online.

Orr, Genevieve. 1999. *CS-449, Neural Networks. Momentum and Learning Rate Adaptation*. Salem, Oregon.

Orr, Genevieve and Klaus-Robert Müller. 1998. *Lecture Notes in Computer Science*. Vol. 1524, *Neural networks. Tricks of the trade*. Berlin, New York: Springer.

Pomerleau, Dean A. 1991. "Efficient Training of Artificial Neural Networks for Autonomous Navigation." *Neural Computation*, 3(1): 88–97.

Priddy, Kevin L. and Paul E. Keller. 2005. *Artificial Neural Networks*. 1000 20th Street, Bellingham, WA 98227-0010 USA: SPIE.

Rosenblatt, F. 1962. *Principles of Neurodynamics. Perceptrons and the theory of brain mechanisms*. Washington: Spartan Books.

RSC. 2015. Periodic Table. http://www.rsc.org/periodic-table/element/79/gold (accessed November 18, 2015).

Shambora, William E. and Rosemary Rossiter. 2007. "Are there exploitable inefficiencies in the futures market for oil?" *Energy Economics*, 29(1): 18–27.

Shapiro, Arnold F. 2002. "The merging of neural networks, fuzzy logic, and genetic algorithms." *Insurance: Mathematics and Economics*, 31(1): 115–31.

Steiner, Manfred and Hans-Georg Wittkemper. 1997. "Portfolio optimization with a neural network implementation of the coherent market hypothesis." *European Journal of Operational Research*, 100(1): 27–40.

Tieleman, Tijmen. 2014. *CSC321, Neural Networks for Machine Learning. Overview of mini--batch Overview of mini-batch gradient descent*. Toronto.

Tom Simonite. 2012. Google Puts Its Virtual Brain Technology to Work.

Trippi, Robert R. and Efraim Turban. 1993. *Neural networks in finance and investing. Using artificial intelligence to improve real-world performance*. Chicago, Ill.: Probus Pub. Co.

Tsaih, Ray, Yenshan Hsu, and Charles C. Lai. 1998. "Forecasting S&P 500 stock index futures with a hybrid AI system." *Decision Support Systems*, 23(2): 161–74.

Werbos, P. 1974. "Beyond Regression. New tools for prediction and analysis in the behavioural science." PhD thesis. Cambridge.

Yu, Lean, Shouyang Wang, and Kin K. Lai. 2008. “Forecasting crude oil price with an EMD-based neural network ensemble learning paradigm.” *Energy Economics*, 30(5): 2623–35.