Predictive Modeling in the Economic Sphere

K.M. Carper - T.J. Hobson-Lowther - M.B. Mason

***Abstract*—In econometrics, stock prices are traditionally modeled as white noise. From a predictive modeling standpoint, this presents an interesting challenge. Often times, short-term prediction is adequate. In particular, linear regressive models show promise for predicting short-term behavior in the FOREX market. Through the application of several filters, including a novel k-previous neighbors smoothing technique, these predictive models are further strengthened. Time-varying frequency analysis of exchange rates provides a way to recognize market volatility. A more complex predictive scheme, the neural network, is given a tutorial treatment and evaluated in terms of its applicability for predicting exchange rates.**

I. Introduction

The focus of this paper is the prediction of exchange rates in the FOREX market. The market reflects a principle that any international traveler is familiar with: one currency can be traded for another. The exchange rate of currency pairs is governed by several factors, most notably the actions of the huge number of investors in the FOREX market and the interactions between national economic markets. FOREX has many properties that lend it well to computational analysis. FOREX is the most liquid market in the world, with multiple trillions of dollars worth of currency traded per day. This extremely high trading volume allows for an investor (or an algorithm) to trade virtually any amount of currency at any time. The market is also notoriously volatile. The constant fluctuations of the market make predicting a challenge, but provide proportional opportunity. This oportunity is magnified when coupled with the ability to trade at very high rates. Furthering its suitability for analysis is the fact that there is a plethora of easily accessible data for use in evaluation and training of algorithms[1].

II. Linear Regression

A. Long-term Prediction with Linear Regression

Fitting economic data with a linear (or any practical polynomial) function is inherently coarse, but still has some predictive merit. We will use the following procedure:

1) Assume the FOREX data follows a linear trend and will continue to do so
2) Use linear regression on a subset of the data to determine a best fit line
3) Use the computed line for prediction of future prices
4) Compare predictions to reality

While this algorithm is not intended to predict small fluctuations, the hope is that this will pick out larger scale trends that could be useful for trading. To demonstrate, half of a month of USD/JPY exchange rate data is fit with a line with minimum ℓ_2 error, using the procedure described in [1, pp. 147–149]. The resulting line and its extension to the rest of the month can be seen in Figure 1.

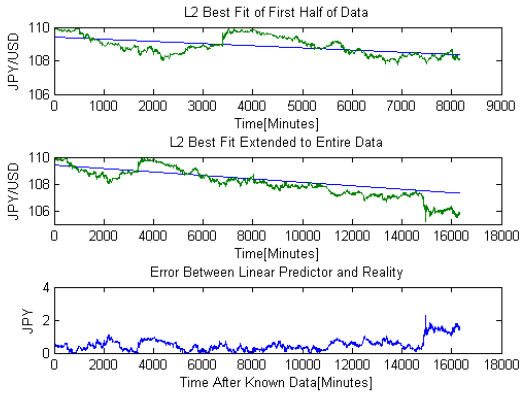

Figure 1. Fit of exchange rate data, maintains approximate error between the predictive and fitting periods

This data had a definite downward trend that persisted throughout the month, resulting in a reasonably accurate linear fit. Near the end of the month, however, there was an abrupt fall in the exchange rate which was not, and could not, be predicted with linear regression. This inherent inaccuracy is further illustrated in Figure 2. Here, although still roughly linear, the trend changes almost immediately after the fitting period, resulting in and increasing error as time goes on.

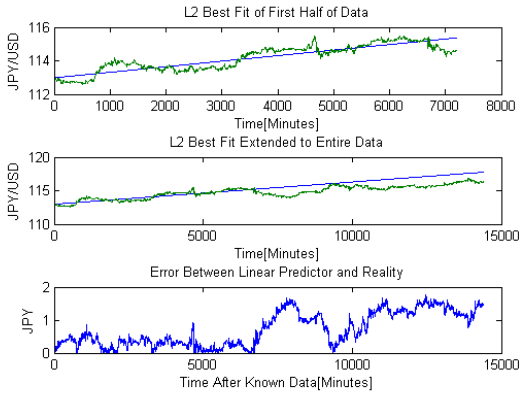

Figure 2. Change in trend causes linear prediction to become increasingly innacurate

B. Short-term Prediction with Linear Regression

As was discussed in the previous section, long term prediction with linear regression was relatively unsuccessful. Over

[1]We primarily used minute data from histdata.com

shorter intervals the data can be approximated as linear much more accurately. By imposing a minimum linearity condition, ,we can make predictions with a higher degree of certainty.

1) Variable Window: An intuitive alteration to long-term linear prediction lies in using a variable-length window. Rather than running linear regression on the whole dataset, we may systematically reduce the size of our window until the total error of our best-fit line falls below some user-defined tolerance, ϵ.

The process is delineated below:

1) Select a maximum total error tolerance, ϵ
2) Specify an initial window size, n
3) Perform linear regression on the vector $\vec{\mathbf{x}}$, which consists of the n values previous to the data point whose value we wish to predict
4) Generate a best-fit line, $\vec{\mathbf{v}}$. Calculate $\varepsilon = \|\vec{\mathbf{v}} - \vec{\mathbf{x}}\|_2$
5) If ε is below ϵ, then we are done. Otherwise, reduce the window size from the left by 1. Repeat steps 3 and 4
6) When condition 1 is met, output the vector $\vec{\mathbf{v}}$

We can use this method as a cut-and-dry indicator of current risk in our model's predictive capability. If a relatively large window length n is capable of producing results within our error tolerance, we can be ensured that our prediction lies on a long-term trend, and predictive risk is low. However, if the required window size is relatively small, then our predictive capability is applicable only for a small length of time, and this can be seen as a higher-risk predictive environment.

2) Fixed Window: We also investigated using a fixed window, wherin the data satisfies:

$$\|\vec{\mathbf{y}} - \vec{\mathbf{x}}\|_2 < \epsilon \tag{1}$$

To find these regions we use the following algorithm:

1) Choose ϵ and the size of the subset to examine
2) Perform linear regression on a (small) subset of data to find $\vec{\mathbf{y}}$ that minimizes (1) in that region
3) Check if ℓ_2 norm is less than ϵ
4) If (1) not met, shift the window to the right, i.e. advance in time
5) Repeat 2,3,4 until constraint is met or the entire data set is examined
6) Output best fit line, $\vec{\mathbf{y}}$

Newly accumulated data is appended to the window and the oldest point is discarded. Once a suitable window is determined, prediction can proceed with confidence, as seen in Figure 3. The prediction maintains fair accuracy for about 500 data points, or half the window length. This behavior is typical for most data sets tested. When the window is made very small, few points are used in the prediction of the next point. Under these conditions, we can obtain a sense of the immediate opportunity present in this market. This is done by using the slope of the fitted line as an indicator to trade out currency. The best results were obtained when only the two previous points were used, and a trade was made when the slope changed from positive to negative. Using this technique, we managed to generate theoretical returns of over 4% monthly.

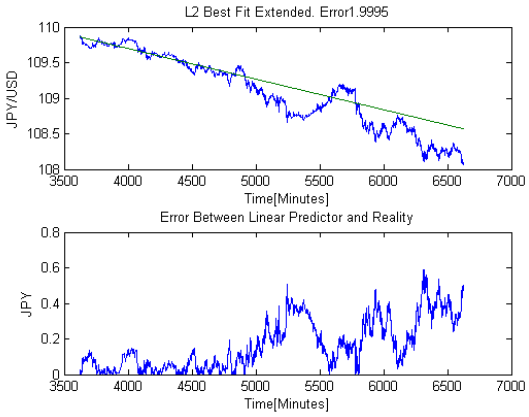

Figure 3. Algorithm run, window length 1000. A tight linearity constraint finds the left-most portion of graph suitable

C. Filtering

Filtering techniques are often beneficial before running analysis on large data sets. This can be especially true when attempting to apply feature extraction on noisy data. Here, we inspect the benefits and drawbacks of using three types of filters on our FOREX currency pairs: the median filter, the mean filter, and a k-previous neighbors filter.

1) Standard Filters: Median Filters are commonly used in smoothing algorithms for image processing. A sample window of size $\mathbf{n}$ data points is selected. This window is centered around the data point to be filtered, such that there are (n-1)/2 points leading and lagging the central point. We calculate the median value of the window for each point $\vec{\mathbf{x}}_i$ in the data set $\vec{\mathbf{x}}$. This median value is now placed into a filtered vector, $\vec{\mathbf{v}}$.

In a system that is both noisy and volatile, the median filter stands out as an excellent candidate for data smoothing because of its resistance to outliers. When a point exists with high deviation from its neighbors, it is almost guaranteed to be excluded from the median value. These filters tend to hide spikes and bumps in data, and the exclusion of these short-term features in FOREX currency pairs is conducive to the investigation of long-term trends. However, these hidden short-term trends can be quite lucrative to investors looking to take advantage of high-frequency trades for quick profit.

Another useful filter arises from taking the mean of the sample window described above, rather than the median. This technique, commonly referred to as moving average (MA) filtering, has wide applications in signal processing.

The most significant benefit of the MA filter is its ability to smooth out data sets, greatly reducing high-frequency components. However, this filter is notoriously susceptible to outliers. Therefore, some level of care needs to be taken when drawing conclusions from filtered data sets. As an example, say the USD/JPY exchange rate has been constant for some time. It then temporarily spikes upward and returns to its previous value in a couple of ticks. With a large enough sampling window, this will result in the rise of mean values for minutes to come. If the mean-filtered data is the only thing considered, this could be incorrectly perceived as a relatively long-term trend toward higher exchange rates.

2) K-previous Neighbors (KPN) Filtering: The previously described smoothing algorithms fail in predictive ability due to the position of the sampling window. The window acts over data points both behind and in front of $\vec{\mathbf{x}}_i$, whose mean or median value is being calculated. In cases where making predictions based on presently occurring data is the main concern, incorporating future values is clearly an impossibility.

One solution is to use a k-previous neighbor filter. This smoothing technique is an alteration of the mean filter, with the leading half of the window removed. A "property value", $\vec{\mathbf{v}}_i$, is assigned to each data point $\vec{\mathbf{x}}_i$ in the raw data set $\vec{\mathbf{x}}$ such that:

$$\vec{\mathbf{v}}_i = \frac{1}{k}\sum_{m=0}^{k-1} \vec{\mathbf{x}}_{i-m} \tag{2}$$

This property value is generated by taking the mean of only $\vec{\mathbf{x}}_i$ and the $\mathbf{k}$ previous neighbors. With this strategy, only data prior to the point in question is required to calculate the new filtered matrix. As the sample window size $\mathbf{k}$ grows, the new data vector $\vec{\mathbf{v}}$ tends to be smoother than the original data set, but a lag in response to trends also appears. This lag can be used as an indicator of economic growth opportunity, as demonstrated below.

The effects of KPN are shown in Figure 4. Using relatively small window size of 500 ticks, the algorithm smooths the noisier features, and property value lag is negligible. Increasing window size to 5,000 previous data points increases the smoothing effect, and also increases the lag between the data and KPN. This trend continues for $\mathbf{k}$=15,000.

Inspection of the difference between KPN-smoothed and unfiltered data reveals a pattern. When the FOREX data is trending up for an extended period of time, the KPN property value falls below the unfiltered value. Similarly, KPN values rise above the original data when a downward trend is occurring. We can use this information to create an algorithm that in some sense quantifies economic opportunity: as a general rule, if the property value $\vec{\mathbf{v}}_i < \vec{\mathbf{x}}_i$, USD should be exchanged for JPY. In this case we output a "1" to a binary opportunity vector at index i, $\vec{\mathbf{b}}_i$. Conversely, if $\vec{\mathbf{v}}_i < \vec{\mathbf{x}}_i$, JPY should be exchanged for USD, and we output a "0" to $\vec{\mathbf{b}}_i$.

As a test of this pattern's validity, we created an algorithm which starts with $1,000 and trades from USD to JPY or vice-versa based on the value $\vec{\mathbf{b}}_i$. With decreasing values of $\mathbf{k}$, we were able to generate monthly interest rates of up to 6%. Restrictions to capital lie in the trading frequency allowed by FOREX software. Our algorithm did not take into account buy/sell spread data.

III. Time-Frequency Analysis

There is often increased volatility in a market before a major swing. This volatility expresses itself as a more rapidly varying time series. To visualize this, we would like to transform our time series to frequency space, where we expect to see more energy at higher frequencies. The traditional tool to accomplish this, the discrete Fourier transform (DFT), is not applicable; this is because the DFT assumes that the frequency components do not vary in time. This time-variance of frequency is exactly what we hope to observe. To combat this shortcoming, we use the Discrete Short-Time Fourier Transform (STDFT) (3)[2]. Being able to predict these swings would allow an interested party to either exploit the swing or allow them to exit the market until it has settled.

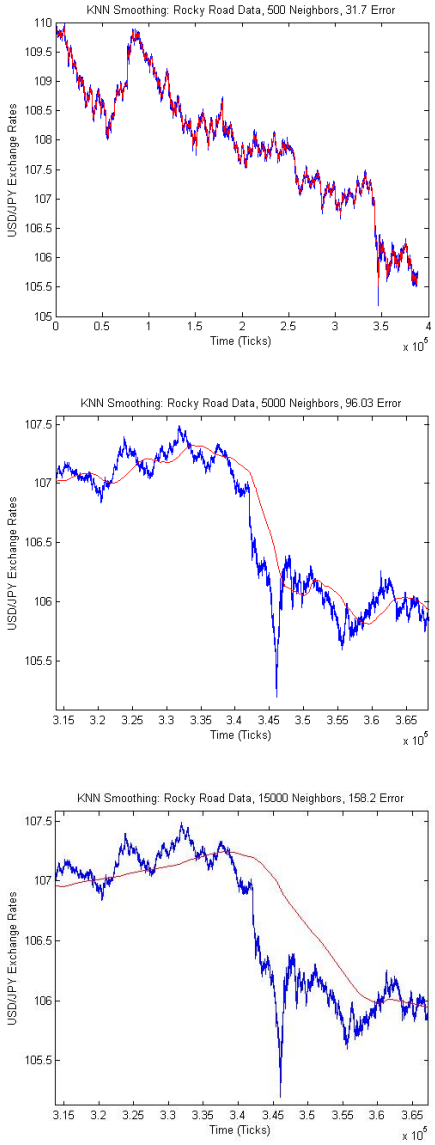

Figure 4. From top to bottom: KPN filtering with k values of 500, 5,000, and 15,000, respectively. The blue line shows the raw data, and red indicates the property values after KPN smoothing.

A. Discrete Short-time Fourier Transform

The STDFT is nearly identical to the DFT, except for the inclusion of a window function, w[n], which multiplies the signal. This window marginalizes distant data points.

$$\mathbf{STDFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-i\omega n} \tag{3}$$

This is equivalent to taking the Fourier Transform of a subset of the data, allowing the study of a signal's frequency evolves over time. We use a Hamming window for all time-frequency analysis, shown in Figure 5.

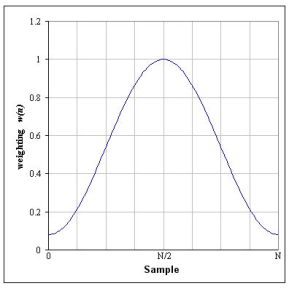

Figure 5. The tails of the Hamming window abate discontinuities in the STFT

B. Application to FOREX

We applied the STDFT to FOREX data and generated a periodogram of the result. We then attempted to discern changes in the periodogram that correlated with major swings in the exchange rate. Transformation of the signal results in a loss of some temporal resolution. Therefore, we must choose a window that is fine enough to accurately characterize the fluctuations. An example periodogram can be seen in Figure 6. There are some distinguishable spikes around major swings,

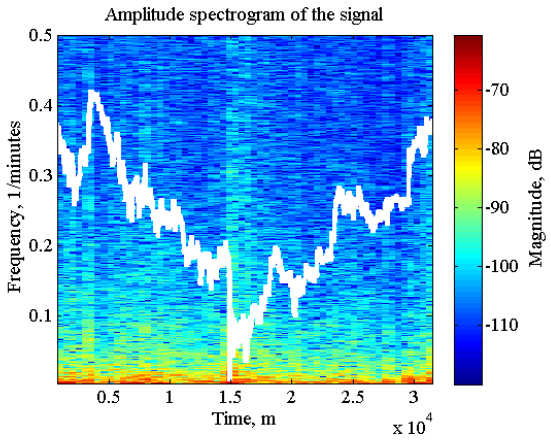

Figure 6. Exchange rate data (white) overlayed on periodogram

but filtering makes them more apparent, as seen in Figure 7. The filtering accentuates the spikes corresponding to swings,

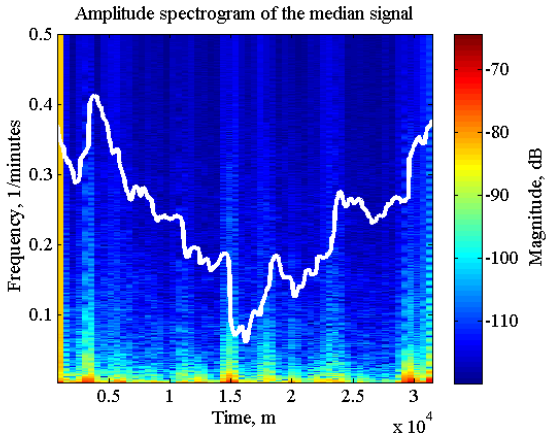

Figure 7. Median filter applied to data (white) before frequency analysis

while preserving the essential character of the time series. Although the frequency spikes indicate significant changes in exchange rate, they do not give any information as to the direction of the trend. Because of this fact, a more conservative investor might want to withdraw from the market in this time of higher volatility. Furthermore, this is a very qualitative technique which relies on human interpretion of the graph. For this to be truly useful, the information in the periodogram would need to be parsed into something an algorithm could use.

IV. Neural Networks

A. What is a Neural Network?

An artificial neural network is a computer implemented algorithm based on the architecture of a biological brain that can be used for machine learning and pattern recognition applications. A biological brain takes in external stimuli of various forms, such as sight. A particular stimulus causes a unique collection of neurons in the brain to be simultaneously activated. Different stimuli would thus activate a different collection of neurons. This type of recognition can be simulated in software and is referred to as a neural network. Neural networks are used widely for object classification in computer vision and robotics, as well as for data forecasting. The strength of using a neural network is that there does not need to be a known functional relationship between the inputs and the output. The network will simply look for nonlinear trends or patterns in the data.

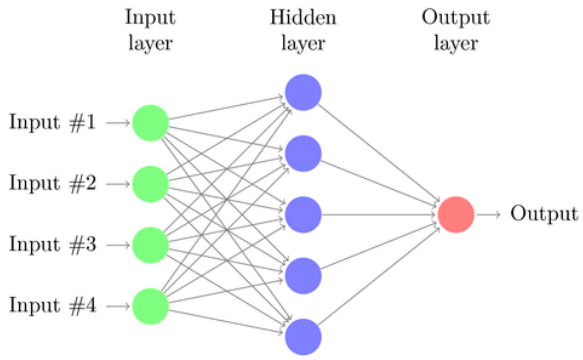

Figure 8. Basic architecture of an artificial neural network with a single hidden layer and single output.

Figure 8 shows the architecture of a simple neural network that progresses from left to right. Just as a brain receives stimulus from its various sensors, a neural network is presented inputs in the form of computer data such as an image or a time series. The next layer is the hidden layer which contains nodes called neurons. Each input is connected to all other neurons and the connections are weighted. The neuron performs a weighted sum called the potential. This potential is passed through a nonlinear, differentiable, sigmoidal function called the activation function. Without the addition of this nonlinearity, the neural network would only be able to approximate linear combinations of the inputs. The activation function takes

in values from -∞ to ∞ and maps them, typically, between 0 and 1. This activation value is used to recursively reweight the connections between inputs and neurons during training. The ouput is calculated as a weighted sum of the neuron outputs.

This three layer design is the simplest architecture. More sophisticated neural networks may feature additional hidden layers, with each neuron output serving as input to a neuron in another hidden layer. This has the effect of allowing more abstraction in the network's predictive or classification ability. Sophistication may also be added by recursively feeding the network output into the hidden layer, a method known as backpropogation.

B. Training

To optimize the weights between inputs and neurons, the neural network needs to be trained on some input data. There are two main regimes regarding training: supervised and unsupervised. Supervised training takes input data into the neural network and attempts to approximate the nonlinear regression function of the data. This typically works well when there are trends in the data and is the preferred training method for statistical modeling, such as time series prediction.

Unsupervised training takes input data and looks for patterns or characteristics in the inputs by clustering the data. This approach is generally preferred for pattern recognition or classification in computer vision when there is not necessarily a defined trend in the data. We will focus on supervised learning because FOREX data shows definite trends.

Supervised learning requires the data be split into three subsets: training, validation, and testing. The training set comprises about 70% of the supervised learning data. One point of this data serves as input to the neural network, with initial weights randomly selected at all neuron connections. The activation values are computed and the weights are recalculated according to the optimization process discussed below in Section IV-C. The network is now tested against the validation set which comprises about 15% of the supervised learning data, and errors are computed. The process is repeated until the validation error stops decreasing. This indicates that the neural network is becoming overtrained, and is essentially memorizing the training set and fitting to its noise. At this point, the neural network has been fully trained. The remaining 15% of the data is the test data and is simply used to benchmark the performance of the algorithm.

C. Gradient Optimization

The reweighting procedure employs a method called stochastic gradient descent. This is simply a higher-dimensional form of finding a minimum via calculus. Equation (8) is the objective functio,n or the function to be minimized. For the case of optimizing the neural network, the objective function is the sum of the squared deviations of our neural network output from the known training set output values, or squared error. The following equations are generalized for multiple inputs and hidden layers.

$$\vec{\mathbf{v}}_{l,j}(n) = \sum_{l=0}^{m_{l-1}} \vec{\mathbf{w}}_{l,j,i}\vec{\mathbf{y}}_{l-1,i}(n) \tag{4}$$

$$\vec{\mathbf{y}}_{l,j} = \phi(\vec{\mathbf{v}}_{l,j}(n)) \tag{5}$$

$$\vec{\mathbf{e}}_{l,j}(n) = \vec{\mathbf{d}}_{l,j}(n) - \vec{\mathbf{y}}_{l,j}(n) \tag{6}$$

$$\vec{\epsilon}_L(n) = \frac{1}{2}\sum_{j=1}^{m_L}(\vec{\mathbf{d}}_{L,j}(n) - \vec{\mathbf{y}}_{L,j}(n))^2 \tag{7}$$

$$\vec{\epsilon}_{av} = \frac{1}{N}\sum_{n=1}^{N}\vec{\epsilon}_L(n) \tag{8}$$

$$\vec{\mathbf{w}}_{l,j,i}^{[k+1]} = \vec{\mathbf{w}}_{l,j,i}^{[k]} - \mu\frac{\partial\vec{\epsilon}(n)}{\partial\vec{\mathbf{w}}_{l,j,i}^{[k]}} \tag{9}$$

where n is the training set index, k is the training iteration, l is the layer number, L is the output layer, i is the input index, j is the neuron index, m_l is the number of neurons in layer l, $w_{l,j,i}$ is the weight between the jth neuron of the lth layer and the ith neuron of the previous layer, v is the potential, ϕ is the activation function, $y_{l,i}$ is the ith output of the lth layer, x is the training set input, d is the training set output, and μ is the learning rate, as given by Moon [1, pp. 648–652].

Equation (9) shows that to update the connection weights, we subtract the derivative of the average error with respect to each weight from the previous weight. This derivative term is the gradient in our higher dimensional space, which points towards a maximum. This term is multiplied by μ, the learning rate, which is the step size that we take along the gradient path toward the minimum. The term is subtracted because the gradient actually points in the direction of steepest ascent and we want to descend towards the minimum. When the difference between errrors over consecutive iterations converges, this method has found a minimum. However, as the equations are written, we are not assured a global minimum. Typically, an additional term proportional to the previous gradient term is also subtracted from the right side of Equation (9). This term is referred to as the momentum, and serves to bump the computed minimum from a local minimum so that the error can continue to progress towards a global minimum.

D. Implementing Neural Networks on FOREX Data

Neural Network models have been applied to FOREX forecasting for the past 20 years. As such, there have been many academic publications describing different approaches and their success when compared to traditional economic predictive models, such as Autoregressive Integrated Moving Average (ARIMA). The neural network models can take a variety of economic data and output the next currency pair exhange rate.

Yao et al.[3] trained a neural network on 11 years of daily rates with 2,910 exchange rates per day. They found that the predictive ability of their algorithm consistently outperformed the ARIMA model. Their best monthly return was for USD/AUD at 28.49%. Dunis[4] experimented with utilizing other economic indicators as input to a neural network. While only predicting a single currency pair, Dunis input several currency pair exchange rates, price indices for various countries, and bond yields. The neural network mostly outperformed ARIMA. Simon[5] took this one step further and added simple

boolean indicators as input along with five different currency exchange rate pairs. Whether or not USA had a Republican or Democrat president was a boolean indicator and whether or not there were any major wars was another. Simon used a decade worth of data for 5 different currency pairs, and experimented with different sampling rates. These rates were daily, weekly, every six months, and every two years. Upon testing, Simon found better profits with shorter time period sampling rates, a consistent, albeit small improvement when boolean indicators are used, and best results when all five currency pairs are input. These sources unanimously show better results when compared to ARIMA and better results when there are several different types of input data.

V. Conclusion

Several analytical methods have been applied to FOREX currency pair data in an attempt to qualify and quantify predictive capability. Limited success was observed from isolated processes. However, the general consensus among our group is that the predictive power of each individual algorithm could only be enhanced by the synthesis of market indicators gleaned from each.

Linear regressive methods, ranging in complexity, were investigated. While long-term predictive methods were generally found incapable of producing robust and reliable forecasts, some localized techniques were found more suitable. In particular, a hyper-local regressive algorithm was able to produce monthly interest rates of up to 4%. Time-frequency analysis revealed indications of market instability, aiding in classification of high-risk investment periods. Both traditional and novel filtering techniques were explored, with the latter pointing to an interesting binary opportunity classification method. Capitalization attempts on opportunity classes were shown to yield monthly return rates of up to 6%. Research of current literature shed light on the potential for artificial neural networks to serve as powerful, adaptive tools for market prediction.

References

[1] T. Moon, W. Stirling, "Basic Concepts and Methods of Iterative Algorithms," in *Mathematical Methods and Algorithms for Signal Processing*, Prentice Hall, 2000.

[2] Wikipedia contributers. (December 11, 2014). Short-time Fourier Transform. [Online]. Available: http://en.wikipedia.org/wiki/Short-time_Fourier_transform

[3] J. Yao, H. Poh, and T. Jasic, "Foreign Exchange Rates Forecasting with Neural Networks," *International Conference on Neural Information Processing, Hong Kong,* September, 1996.

[4] C. Dunis, M. Williams, "Modelling and Trading the EUR/USD Exchange Rate: Do Neural Network Models Perform Better?,"*Liverpool Business School,* February, 2002.

[5] E. Simon, "Forecasting Foreign Exchange Rates with Neural Networks," *Swiss Federal Institute of Technology,* February, 2002.