Comparative Study of Artificial Neural Networks and Hidden Markov Model for Financial Time Series Prediction

Afrah R. Shaaib
Department of Computer Science
Faculty of Information Technology
Misurata University
Misurata, Libya
afrah.shaaib@ijeit.misuratau.edu.ly

Abstract - **Financial Time Series analysis and prediction is one of the interesting areas in which past data could be used to anticipate and predict data and information about future. There are many artificial intelligence approaches used in the prediction of time series, such as Artificial Neural Networks (ANN) and Hidden Markov Models (HMM). In this paper HMM and HMM approaches for predicting financial time series are presented. ANN and HMM are used to predict time series that consists of highest and lowest Forex index series as input variable. Both of ANN and HMM are trained on the past dataset of the chosen currencies (such as EURO/ USD which is used in this paper). The trained ANN and HMM are used to search for the variable of interest behavioral data pattern from the past dataset. The obtained results was compared with real values from Forex (Foreign Exchange) market database [1]. The power and predictive ability of the two models are evaluated on the basis of Mean Square Error (MSE). The Experimental results obtained are encouraging, and it demonstrate that ANN and HMM can closely predict the currency market, with a small different in predicting performance.**

Index Terms: **HMM, ANN, Time series, Currency market, Prediction, Predicting.**

I. INTRODUCTION

The analysis and prediction of financial time series is of upmost importance for professionals and academics in the field of finance. Currency exchange is very attractive for both corporate and individual traders who make money on the Forex - a special financial market assigned for the foreign exchange. The currency trading (Forex) market is cash interbank market established 1971. It is the biggest and fastest growing market on earth. According to the Bank for International Settlements (BIS), as of April 2010, average daily turnover in global foreign exchange markets is estimated at 3.98 trillion dollars [2]. It allows businesses to convert one currency to another foreign currency. The currency market as a whole is referred to as Forex.

Currency exchange has always been one of greatest challenges in business, finance and Artificial Intelligent (AI). It helps investors to hedge against potential market risks. Nevertheless, currency exchange predicting is a complex and difficult task because of many factors that influence the market. There are two types of analysis namely fundamental and technical. (1) Fundamental analysis: which factors involved in price analysis, such as: supply and demand, seasonal cycles, weather, and government policy …etc., (2) Technical analysis: which charts are based on market action involved in price, volume, and history. The currency markets in the recent past years have become an integral part of the universal economy. Any fluctuation in this market influences personal and corporate financial lives, and the economic state of a country. The currency market has always been one of the most popular investments due to its high returns [3]. So an intelligent predicting model would be desirable to develop. However, for better results, it is important to find the appropriate technique for a given predicting task.

There are numerous predicting models available. These models developed based on various techniques. Such as Statistical analysis like Autoregressive Moving Average (ARMA) model [4], Artificial Neural Networks (ANN) [5], Fuzzy Logic (FL) [6], Neuro-Fuzzy systems [7], Support Vector Machine (SVM) [8], evolutionary algorithms [9], Hidden Markov Models (HMM) [10]. In this paper, two approaches (HMM and ANN) to predict financial time Series (currency prices for Forex market) is presented. They predict currency exchange rates of Euro with US Dollar (USD).

An ANN mimics the behavior of neurons in the brain. The basic components of an ANN are its nodes or neurons and the connections between the nodes. A node is primarily a computational unit. It receives inputs, calculates a weighted sum and presents the sum to an activation function. The nodes are generally arranged in layers. The use of a neural network, however, which learns rather than analyses [11].

HMMs are statistical models of sequential data that have been successfully used in many machine learning

Received 22 December 2014; revised 20 January 2015; accepted 28 January 2015.
Available online 29 January 2015.

applications, especially for speech recognition. The theory of HMMs was developed in the late 1960s and early 1970s. In recent years, HMMs have been applied to a variety of applications, such as handwriting recognition [12, 13], pattern recognition in molecular biology [14, 15], and fault-detection [16].

The following section discuss the ANN and HMM algorithm in a detail, while the remainder of this paper describes the approach of using ANN and HMM as predictors. The experimental results are presented in Section III. Finally, the more important conclusions are summarized in Section IV.

II. METHODOLOGY

A. ANN

Work on artificial neural network has been motivated right from its inception by the recognition that the human brain computes in an entirely different way from the conventional digital computer. The brain is a highly complex, nonlinear, and parallel computer (information-processing system). It has the capability to organize its structural constituent, known as neurons, so as to perform certain computations (e.g., pattern recognition, perception, and motor control) many times faster than the fastest digital computer in existence today [11].

ANN is a mathematical model or computational model based on biological neural networks. It consists of an interconnected group of artificial neurons, and processes information using a connectionist approach to computation. An artificial neuron is a simple unit that computes a linear, weighted sum with an additional output function. Perhaps, the greatest advantage of ANNs is their ability to be used as an arbitrary function approximation mechanism that 'learns' from observed data [17].

The most important neural network type is the Multi-Layer Perceptron (MLP) with strict feed-forward architecture of three layers. The connections between inputs and outputs are typically made via one or more hidden layers of neurons or nodes. The input layer is defined to assume the values of the input vector and does not perform any additional computation, the hidden layer of neurons or nodes is fully connected to the input and output layers and usually uses the sigmoid function as output function. Fig. 1 shows a typical ANN with three inputs, and one hidden layer of two neurons and one output [17].

Each neuron can have multiple inputs; while there can be only one output (Fig. 2.a).Inputs to a neuron could be from external stimuli or could be from output of the other neurons. Copies of the single output that comes from a neuron could be input to many other neurons in the network. It is also possible that one of the copies of the neuron's output could be input to itself as a feedback. There a connection strength, synapses, or weight associated with each connection. When the weighted sum of inputs to the neuron exceeds a certain threshold, the neuron is fired and an output signal is produced. The network can recognize input patterns once the weights are adjusted or tuned via some kind of learning process [18].

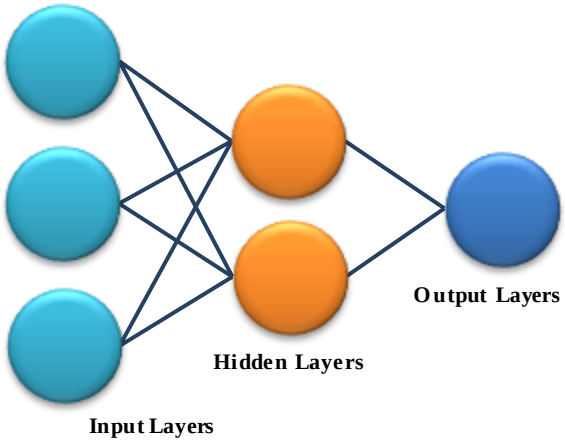
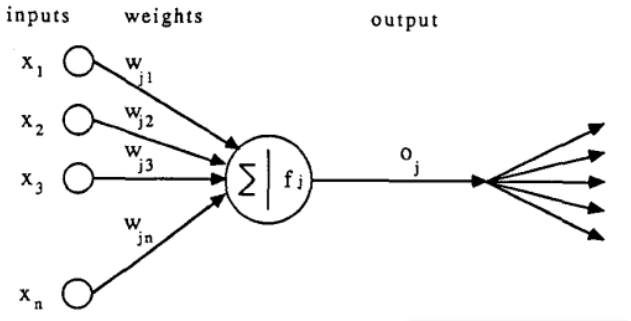
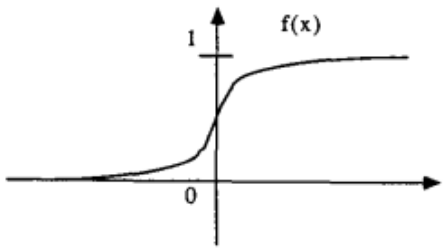


Figure 1. Architecture for a Typical ANN with Three Inputs, One Hidden Layer of Two Neurons, and One Output.



(a) Mathematical Model of Neuron



(b) Sigmoid function

Figure 2. Schematic of Artificial Neuron [18]

The "generalized delta rule" is suggested by Rumelhart, et al. and gives a recipe for adjusting the weights on the internal units based on the error at the output [19]. To be more specific, let

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \quad (1)$$

Be the measure of error in pattern p and let $E = \sum_p E_p$ be the overall measure of the Error, where t_{pj} is the target output for j-th component of the actual output pattern produced by the network representation with input pattern p.

The network is specified as

$$o_{pj} = f_j(net_{pj}), \quad (2)$$

$$net_{pj} = \sum_k w_{jk}\, o_{pk}\ , \qquad (3)$$

where f_i is a differentiable and non-decreasing function and w_{jk} is a weight to be adjusted. The function f_j is normal a sigmoid type function as shown in Fig. 2.b.

To obtain a rule for adjusting weights, the gradient of E_p with respect to w_{ji} is used and it is represented as follows:

$$-\frac{\theta E_p}{\theta w_{ji}} = \delta_{pj} o_{pi}\ , \qquad (4)$$

where δ_{pj} is defined in two ways. If a unit is an output unit, it is given by

$$\delta_{pj} = (t_{pj} - o_{pj}) f_j'(net_{pj}) \qquad (5)$$

and for a unit in an arbitrary hidden layer

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{kj} \qquad (6)$$

where f_j' is the derivative of f_j.

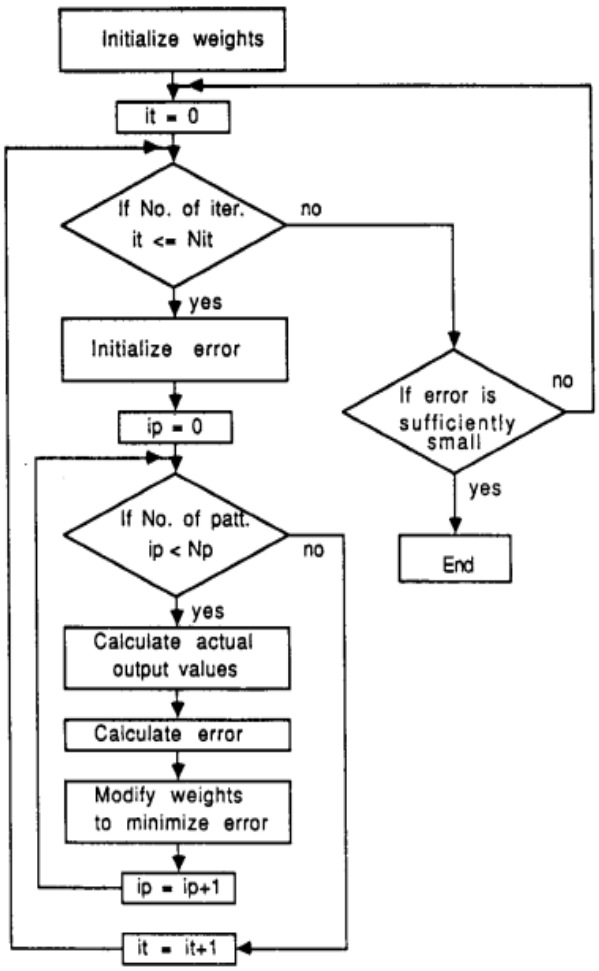


Figure 3. Flowchart for Back-Propagation Algorithm [18]

The rule of adjusting weights can be derived using Eq. (4), and given as

$$\Delta w_{ji}(n+1) = \eta \delta_{pj} o_{pi} + \alpha \Delta w_{ji}(n), \qquad (7)$$

where η is the learning rate parameter and α is the momentum constant to determine the effect of past weight changes. The flowchart the backpropagation learning algorithm following Eqs. (1) and (7) is shown in Fig. 3.

B. ANN as Predictor

The load data where analyzed. The current load is affected by the past loads and the pattern in which the currency load is included. For example, Monday loads are affected by Sunday and Saturday loads and their patterns are similar. Therefore, the following nonlinear load model is proposed for one-day ahead predicting [18].

$$y(i) = F\big(W_i, Y(i-1)\big), \qquad (8)$$

where $y(i) = \{y(i,t): t = 1,2,3,\dots,24\}$: the actual load vector at day i.
$y(i,t)$: the actual load at day i, time t.

$$Y(i-1) = [y(i-1), y(i-2), \dots, y(i-k)]^T \qquad (9)$$

k : index for data length.
W_i: the weight vector.
F(.,.): nonlinear vector function representing ANN.

In constraint to the conventional approaches, the nonlinear function is used with the weight vector to represent the load model. The weight vector W_i can be thought of as the storage that contains a certain load data, and *F(.,.)* is the general nonlinear function that can comprise all the load data. In order to predict the load *y(i)* the weight vector W_i should be estimated using previous load data.

C. HMM

A HMM is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters.

HMM is characterized by the following elements:

$S = \{s_1, s_2, \dots, s_N\}$	a set of N hidden states,
$Q = \{q_1, q_2, \dots, q_T\}$	a state sequence of length T taking values from S,
$O = \{o_1, o_2, \dots, o_T\}$	an observation sequence consisting of T observations, taking values from the discrete alphabet $V = \{v_1, v_2, \dots, q_M\}$,
$A = \{a_{11}, a_{12}, \dots, a_{NN}\}$	a transition probability matrix A, where each a_{ij} represents the probability of moving from state s_i to state s_j with $\sum_{j=1}^{N} a_{ij}$,

$B = b_i(o_t)$	a sequence of observation likelihoods, also called emission probabilities, expressing the probability of an observation o_t being generated from a state s_i at time t,
$\Pi = \{\pi_1, \pi_2, \dots, \pi_T\}$	an initial probability distribution, where π_i indicates the probability of starting in state s_i. Also, $\sum_{j=1}^{N} \pi_i = 1$.

HMM should be characterized by three fundamental problems [20]:

1. Computing likelihood: Given the complete parameter set λ and an observation sequence O, determine the likelihood $P(O/\lambda)$. This problem can be solved by forward-backward algorithms.
2. Decoding: Given the complete parameter set λ and an observation sequence O, determine the best hidden sequence Q. This problem can be solved by Viterbi algorithms.
3. Learning: Given an observation sequence O and the set of states in the HMM, learn the HMM λ. This problem can be solved by Baum-Welch algorithms.

Forward-Backward algorithms:

The Forward value $(\alpha_t(i))$ is the probability of the partial observation sequence, $o_1, o_2, \dots, o_t$ until time t and given state s_i at time t. One can solve for $\alpha_t(i)$ inductively as follows:

1. Initialization: $\alpha_1(i) = \pi_i b_i(o_1),\ 1 \le i \le N$.
2. Induction: $\alpha_{t+1} = \left[\sum_{j=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1})$, $1 \le i \le T+1,\ 1 \le j \le N$
3. Termination: $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$.

The backward value $(\beta_t(i))$ is the probability of the partial observation sequence from $t+1$ to the last time, T, given the state s_i at time t and the HMM λ. By using induction, $\beta_t(i)$ is found as follows:

1. Initialization: $\beta_T(i) = 1,\ 1 \le i \le N$.
2. Induction: $\beta_t = \sum_{j=1}^{N} a_{ij} b_j(o_t)\, \beta_{t+1}(j)$, $t = T-1, T-2, \dots, 1,\ 1 \le i \le N$

The backward variable is not used to find the probability $P(O|\lambda)$. Later, it will be shown how the backward as well as the forward calculation are used extensively to help one solve the second as well as the third fundamental problem of HMMs.

Viterbi Algorithm:

The complete procedure for finding the best state sequence, which is done via the array $\psi_t(j)$, can now be stated as follows:

1. Initialization: $\delta_1(i) = \pi_i b_i(o_1)$, $1 \le i \le N$, $\psi_1 = 0$.
2. Recursion: $\delta_t(j) = \max_{1 \le i \le N}\left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t)$, $2 \le t \le T,\ 1 \le j \le N$.
   $\psi_t(j) = \operatorname{argmax}_{1 \le i \le N}\left[\delta_{t-1}(i) a_{ij}\right],\ 2 \le t \le T,\ 1 \le j \le N$
3. Termination: $P^* = \max_{1 \le i \le N}[\delta_T(i)]$.
   $P_T^* = \operatorname{argmax}_{1 \le i \le N}[\delta_T(i)]$
4. Path (state sequence) backtracking:
   $P_T^* = \psi_{t+1}(P_{t+1}^*), \quad t = T-1, T-2, \dots, 1$

Baum-Welch algorithm:

To be able to re-estimate the model parameters, using the Baum-Welch method, one should start with defining $\xi_t(i,j)$, the probability of being in state s_i at time t, and state s_j at time $t+1$, given the model and the observations sequence. In other words the variable can be defined using the forward and backward variables as follows:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$$

By using these interpretations, a method for the re-estimation of the model parameters Π, A, B for the HMM is as follows:

$$\bar{\pi}_i = \gamma_1(i),$$

One should see that last equation can be interpreted as the frequency in state s_i at time $t = 1$. The next equation should be interpreted as the expected number of transitions from state s_i to s_j divided by the number of transitions from state s_i.

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

And finally, next can be seen as the expected number of times in state s_j and observing the symbol v_k, divided by the expected number of times in state s_j.

$$\bar{b}_j(v_k) = \frac{\sum_{\substack{t=1 \\ o_t = v_k}}^{T-1} \gamma_t(j)}{\sum_{t=1}^{T-1} \gamma_t(j)}$$

If the current HMM is defined as $\lambda = \{A, B, \Pi\}$ and used to compute the right hand side of last three equations, and at the same time define the re-estimation HMM as $\bar{\lambda} = \{\bar{A}, \bar{B}, \bar{\Pi}\}$ as determined from the left hand side of last three equations it has been proven that either:

1. The initial model λ defines a critical point of the likelihood function, in which case $\lambda = \bar{\lambda}$, or
2. Model $\bar{\lambda}$ is more likely than model λ in the sense that $P(O|\bar{\lambda}) > P(O|\lambda)$, which means that one have found a new model $\bar{\lambda}$ from which the observation sequence is more likely to have been produced.

An iterative re-estimation process, replacing λ with $\bar{\lambda}$ can be done to a certain extent, until some limiting point is reached. The final result of this re-estimation procedure is called a maximum likelihood estimation of the HMM.

An important aspect of the re-estimation procedure is that the stochastic constraints of the HMM parameters, namely

$$\sum_{i=1}^{N} \bar{\pi}_i = 1,$$

$$\sum_{j=1}^{N} \bar{a}_{ij} = 1, \qquad 1 \le i \le N$$

$$\sum_{k=1}^{M} \bar{b}_j(v_k) = 1, \qquad 1 \le j \le N$$

are automatically satisfied at each iteration [20].

D. HMM as Predictor:

The objective of predicting is to estimate the probability of hidden state $s_{i,t}$ at time t, given the condition that observable state $o_{k,t}$ is obtained at the same time, i.e., $Pr(s_{i,t}|o_{k,t})$.

The challenge is to determine the hidden parameters from the observable parameters. There are three parameters, which represent the overall HMM model λ ($\lambda = \{A, B, \Pi\}$):

- Transition matrix (A),
- Observation emission matrix (B),
- Initial probability matrix (π) of a HMM.

First, an initial HMM structure consists of 3 states (increasing state, decreasing state and no change state). Initially the parameter values were chosen randomly. The HMM was then trained using the training dataset so that the values of A, B and π are re-estimated to be best suited with the training dataset using the popular Baum–Welch algorithm.

Also, the forward-backward algorithm is used to compute the probability $P(O|\lambda)$ of observation sequence $O = O_1, O_2, \ldots, O_T$, for the given model λ, (Logarithm of this probability is called log-likelihood value or likelihood value). Finally, the predicting result is located at the state with the highest probability. Fig. 4 illustrates this approach in a brief.

To predict the next day's prices, suppose the likelihood value for the day x is lcp_i. Now, from the historical data set observation sequences are located that would produce the same or nearly same value of lcp_i (locate past days where the Forex behavior is matched to the current day). HMM found many observations that produced the same likelihood value lcp_i. Then for each of the matched day, difference between match day and it next day is calculated using

$$wd_k = \frac{\sum_m W_m diff_m}{\sum_m W_m} \text{g}$$

where W_m is weight assigned to day m, wd_k is weighted average of price difference for current day k, $diff_m$ is price difference between day m and $m+1$. Then predict the value for day $k+1$, by

$$fp_{k+1} = p_k + wd_k$$

where, fp_{k+1} is the predicted prices and p_k is current day price [17].

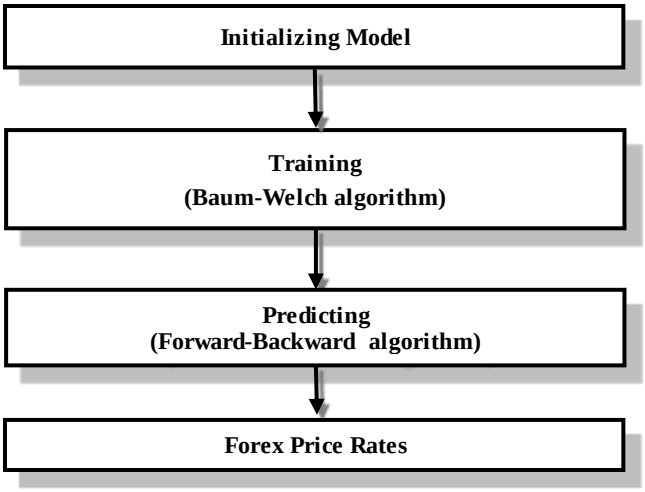


Figure 4. Predicting Algorithm Predicting Using HMM [17]

III. RESULTS

In this paper, the ANN and HMM algorithms experimented with foreign currency exchange daily rate of Euro against US dollar (USD) from April 2007 to February 2011. The system built using a MATLAB software application

Fig. 5 shows the actual Euro/USD highest exchange price rates and the predicted highest rate using ANN and HMM and lowest rate for dataset. While Fig. 6 shows the actual Euro/USD lowest exchange price rates and predicted lowest rates for each time using ANN and HMM.

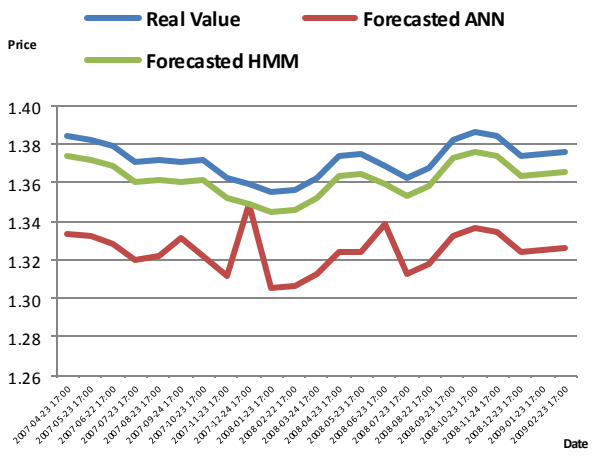


Figure 5. Euro/USD Actual and Predicted Highest Price Rate Using ANN and HMM

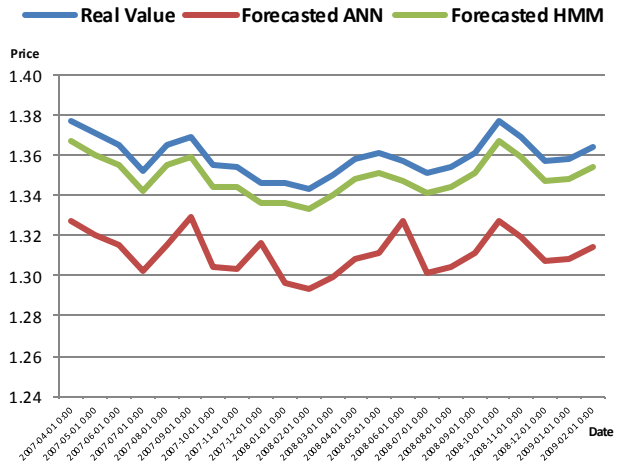


Figure 6. Euro/USD Actual and Predicted Lowest Price Rate Using ANN and HMM

Predicting performance of the above models is evaluated using Mean Square Error (MSE), it is the simplest and most widely used measure of predict accuracy. MSE measure the deviation between actual and predicted value.

$$MSE = (X_{N+h} - \hat{X}_N)^2$$

Table 1 and 2 show the MSE values on using ANN and HMM in predicting highest and lowest price rate for five days as a sample from dataset. The low values of MSE indicate that model was able to generate accurate predictions of the one step forward exchange rates for each of the time periods.

Table 1. EURO/USD Actual and Predicted Highest Price Rate Using ANN and HMM with Their MSE for Five Days as a Sample from Dataset

	EURO/USD highest price rate				
	Actual Price	Predicted price		MSE	
Date		ANN	HMM	ANN	HMM
02/11/2011	1.36223	1.31223	1.34223	0.05	0.02
02/10/2011	1.37374	1.34374	1.35374	0.03	0.02
02/09/2011	1.37457	1.32457	1.34457	0.05	0.03
02/08/2011	1.36898	1.31898	1.34898	0.05	0.02
02/07/2011	1.36278	1.32278	1.33278	0.04	0.03

Table 2: EURO/USD Actual and Predicted Lowest Price Rate using ANN and HMM with Their MSE for Five Days as a Sample from Dataset

	EURO/USD highest price rate				
	Actual Price	Predicted price		MSE	
Date		ANN	HMM	ANN	HMM
02/11/2011	1.34986	1.31986	1.31986	0.03	0.03
02/10/2011	1.35787	1.31787	1.33787	0.04	0.02
02/09/2011	1.36125	1.32125	1.33125	0.04	0.03
02/08/2011	1.35738	1.31738	1.31738	0.04	0.04
02/07/2011	1.35096	1.30096	1.32096	0.05	0.03

IV. CONCLUSIONS

This paper investigates the performance of ANN and HMM for financial time series predicting. The ANN and HMM algorithms used to builds the system and to calculate the similarity between predicted prices and the real prices. ANN and HMM were found to be adept at predicting the financial time series. While low values of MSE were obtained that show good results, the profits on trading were quite small. The accuracy predictions must be still higher in order to achieve consistent profits.

REFERENCES

[1] "FXCM Trading Station Program," 01.08.040910 ed: FXCM, 2010.

[2] "Bank of International Settlement," Triennial Central Bank Survey - Foreign exchange and derivatives market activity in 2007.

[3] R. J. Kuo, L. C. Lee, and C. F. Lee, "Integration of Artificial Neural Networks and Fuzzy Delphi for Stock Market Forecasting," in *IEEE International Conference on Systems, Man, and Cybernetics*, 1996, pp. 1073-1078.

[4] E. H. K. Fung and A. P. L. Chung, "Using ARMA Models to Forecast Workpiece Roundness Error in a Turning Operation," *Applied Mathematical Modelling*, Vol. 23, pp. 567-585, 1999.

[5] M. H. Eng, Y. Li, Q. G. Wang, and T. H. Lee, "Forecast Forex with Artificial Neural Network Using Fundamental Data," 2009, pp. 279-282.

[6] A. K. Lohani, N. K. Goel, and K. K. S. Bhatia, "Development of Fuzzy Logic Based Real Time Flood Forecasting System for River Narmada in Central India," 2005.

[7] E. Abbasi and A. Abouec, "Stock Price Forecast by Using Neuro-Fuzzy Inference System," 2008, pp. 320-323.

[8] K. Kim, "Financial Time Series Forecasting Using Support Vector Machines," *Neurocomputing*, vol. 55, pp. 307-319, 2003.

[9] K. Slany, "Towards the Automatic Evolutionary Prediction of the FOREX Market Behaviour," 2009, pp. 141-145.

[10] M. R. Hassan and B. Nath, "Stock Market Forecasting using Hidden Markov Model: A New Approach," in *Proceedings 5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, 2005, pp. 192-196.

[11] S. Haykin. Neural Networks, A Comprehensive Foundation, New York, Macmillan Publishing., 1994.

[12] R. Nag, K. Wong, and F. Fallside, "Script Recognition using Hidden Markov Models," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'86)*, 1986, pp. 2071-2074.

[13] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges, "A NN/HMM Hybrid for On-Line Handwriting Recognition," *Neural Computation*, vol. 7, pp. 1289-1303, 1995.

[14] A. Krogh, M. Brown, I. S. Mian, K. Sj olander, and D. Haussler, "Hidden Markov models in Computational Biology: Applications to Protein Modelling," *J. Mol. Biol*, Vol. 235, pp. 1501-1531, 1994.

[15] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*: The MIT Press, 2001.

[16] P. Smyth, "Hidden Markov Models for Fault Detection in Dynamic Systems," *Pattern Recognition*, vol. 27, pp. 149-164, 1994.

[17] A. Shaaib, Foreign Exchange Forecasting using Hidden Markov Model, Master Thesis, The Libyan Academy, 2014.

[18] K. Y. Lee, Y. T. Cha and J. H. Park, "Short-Term Load Forecasting using an Artificial Neural Network," Transactions on power systems, Vol. 7, No. 1, February 1992.

[19] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Internal representation by error propagation," Parallel Distributed Processing, vol. 1, pp. 318-362, Cambridge, MA: MIT Press, 1986.

[20] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, pp. 257-286, 1989.

BIOGRAPHIES

Afrah Ramadan Shaaib was born in Misurata / Libya, in June 1986. She received a BSc degree in computer science in 2006 from Misurata University. She also received an MSc degree in computer science from Libyan Academy of Higher Studies at Misurata, Libya in 2014. Currently she is a lecturer at the Faculty of Information Technology, Misurata University, Misurata, Libya. Her main research interests are artificial intelligent and machine learning.