International Research Journal of Finance and Economics
ISSN 1450-2887 Issue 49 (2010)

http://www.eurojournals.com/finance.htm

Financial Time Series Forecasting: Comparison of Neural Networks and ARCH Models

AK Dhamija
Defence Research & Development Organisation (DRDO), Delhi, India
E-mail: dhamija.ak@gmail.com; Webpage: www.akdhamija.webs.com
Tel: 91-11-2393270 (O); 91-1127351028(R); 91-9811453163(M)

VK Bhalla
Faculty of Management Studies (FMS), Delhi, India
E-mail: vkbfms@yahoo.co.uk

Abstract

This paper compares the predictive accuracy of neural networks and conditional heteroscedastic models like ARCH, GARCH, GARCH-M, TGARCH, EGARCH and IGARCH, for forecasting the exchange rate series.The Multi Layer Perceptron (MLP) and Radial Basis Function (RBF) networks with different architectures and conditional heteroscedastic models are used to forecast five exchange rate time series. The results show that both Neural network and conditionally heteroscedastic models can be effectively used for prediction. RBF networks do considerably better than MLP networks in neural networks case. IGARCH and TGARCH fare better than other conditional heteroscedastic models. Neural networks' performance is better than that of conditional heteroscedasticity models in forecasting exchange rate. It is shown that neural network can be effectively employed to estimate the conditional volatility of exchange rate series and the implied volatility of NIFTY options. Neural network is found to beat conditional heteroscedastic models in out-of-sample forecasting.

Keywords: Neural Network; Heteroscedasticity; Conditional Heteroscedastic Models; Exchange Rate; Predictive Accuracy.

1. Introduction

During last twenty five years many different nonlinear models have been proposed in the literature to model and forecast exchange rates. Some authors claimed that exchange rates are rather unpredictable and so random walk model is better predictor (Chang and Osler, 1999; Meese and Rose, 1990; Gencay, 1999). Kuan and Liu (1995) estimate and select feedforward and recurrent networks to evaluate their forecasting performance of five exchange rates against USD. The networks performed differently for different exchange rate series. Yao and Tan (2000) show that if technical indicators and time series data are fed to neural networks to capture the underlying process of the movement in currency exchange rates, then useful prediction can be made. Yang and Gradojevic (2006) construct a neural network that never performs worse than a linear model but always performs better than the random walk model when predicting Canadian dollar/dollar exchange rate. Kiani and Kastens (2008) have successfully employed Neural networks to forecast the exchange rate.

The GARCH model has been used in the past for volatility estimation in U.S. dollar foreign exchange markets (Bail-lie and Bollerslev, 1989) and in the European Monetary System (Neely, 1999). Initial studies into explanatory power of out-of-sample forecasts gave out disappointing results (West and Cho, 1995). Jorion (1995) found that volatility forecasts for several major currencies from the GARCH model were outperformed by implied volatilities generated from the Black-Scholes option-pricing model. These studies used the squared daily return as the variable to be forecast. Since, the exchange rate may move around a lot during the day, it has been established that one can significantly improve the forecasting power of the GARCH model by using sum of intraday squared returns (Andersen and Bollerslev, 1998). This measure is referred to as integrated or realized volatility. Variance forecasts thus obtained show that volatility shocks are quite persistent and the forecasts of conditional variance converge to the steady state quite slowly.

The studies experimenting on forecasting exchange rates so far, have not included the data for the period of the current financial meltdown. This study expressly uses this data and establishes that neural network and autoregressive conditional heteroscedastic models both can effectively capture the long-term non-linearities of the data and predict successfully into the tumultuous period also.

2. Neural Networks

2.1. Neuron

The basic building block of a neural network is the **neuron.** A neuron can be represented by a mapping $y: \mathbb{R}^n \to \mathbb{R}$ transforming a n dimensional input into a real number. The neuron consists of a propagation function $f: \mathbb{R}^n \to \mathbb{R}$ and an activation function $g: \mathbb{R} \to [0,1]$ where $g(x)$ takes the output of f(x) as argument.

Thus, a neuron can be represented in the general form as

$$y(x) = g(f(x; w)) \quad (1)$$

If $f(x)$ is a polynomial, its degree is called the order of the neuron and its coefficients are the parameters of the neuron. These neurons are assembled in layered structure to construct the artificial neural network (ANN). The theoretical framework of neural networks mentioned in this section has been adopted from Giacomini (2003).

2.2. Artificial Neural Networks

Artificial neural networks produce the mapping $ø_{NN}: \mathbb{R}^n \to \mathbb{R}$ and can be written as

$$g(y_1, \ldots, y_m) = ø_{NN}(X_{1,\ldots}x_n) \quad (2)$$

Where $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ is the input vector and $y = (y_1,\ldots, y_m)^T \in \mathbb{R}^m$ is output vector. A particular Neuron will fire when weighted sum $\sum_{i=1}^{n} w_i x_i > \theta$. The θ is the threshold level for neurons to fire. This threshold level can be built into the propagation function be weighting it with $w_0 = -1$. Therefore, the propagation function $f(x) = \sum_{i=1}^{n} w_i x_i - \theta$ is the weighted sum of inputs. The activation function g of a neuron may assume many forms. It can be a linear function or non-linear function. Most commonly used function is a sigmoid function. These interconnected neurons (Haykin, 1999) can be disposed according to a certain architecture.

A network $ø_{NN}$ where the threshold values (Bishop, 1995) are incorporated in the input vector $x = (x_0, \ldots, x_n)^T \in \mathbb{R}^{n+1}$, $x_0 = -1$and the output vector is $y = (y_1, \ldots, y_m)^T \in \mathbb{R}^m$ is represented on Figure 1.

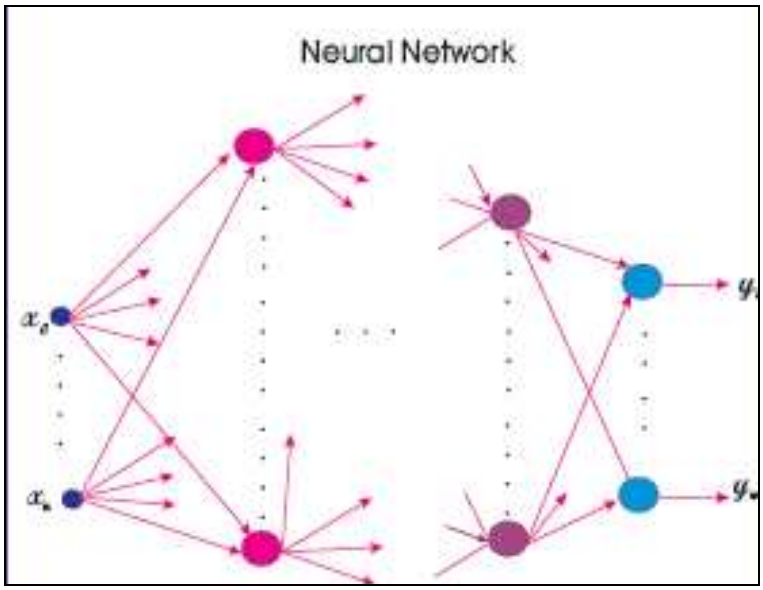
2.3. Multi Layer Perceptron Networks - MLP

Neural networks where the hidden neurons have sigmoidal activation function and the output neurons have sigmoidal or identity function are called *Multi Layer Perceptrons* (MLP) Networks $ø_{MLP}$:

$\mathbb{R}^n \to \mathbb{R}^m$. This architecture consists of an input layer, an output layer and k-hidden layers, each containing j_k neurons.

Each p-component of $y = (y_1, \ldots, y_m)$ is released by the m-neuron at the output layer as a function of the input $x = (x_1, \ldots, x_n)$ and of the parameters w. Writing in compact from, with weights on the input vectors and $d - 1$ as total number of hidden layers.

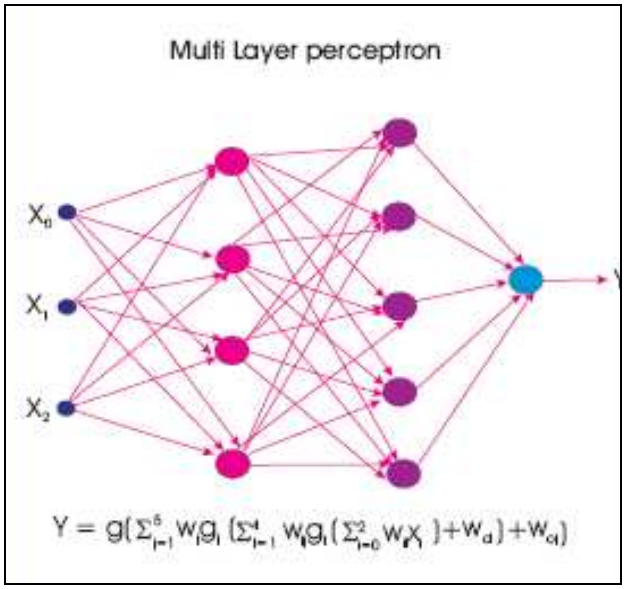
Figure 1: Feed-forward neural network $ø_{NN}$



$$y_p = g_p\left(\sum_{i=0}^{j_{d-1}} w_i\left(g_i\left(\sum_{u=0}^{j_{d-2}} w_u \ldots \sum_{t=0}^{j_1} w_t\left(g_t\left(\sum_{k=0}^{n} w_k x_k\right)\right)\ldots\right)\right)\right) \tag{3}$$

Figure 2 shows the graph of a neural network $ø_{MLP}$, where $d = 3$, $n = 2$, $j1 = 4$, $j2 = 5$ and $m = 1$ or (2 – 4 – 5 – 1) MLP.

Figure 2: (2 - 4 - 5 - 1) MLP $ø_{NN}$



2.4. Radial Basis Function Networks - RBF

Radial Basis Function (RBF) neurons are neurons in which the propagation function has the form $f(x) = \| x - w \|$, where $x = (x_1, \ldots, x_n)^T$, $w_r = (w_1, \ldots, w_n)^T \in \mathbb{R}^n$ are the inputs and weights. The activation function $h(x)$ has the form of a radial symmetric function, commonly the gaus-sian function.

Networks with one hidden layer containing r RBF neurons and output neurons with propagation function $f(x) = \sum_{i=1}^{n} w_{ij}x_i$ and identity activation function $g(x) = x$ are called RBF networks $ø_{RBF}$: $\mathbb{R}^n \rightarrow \mathbb{R}^m$ with r RBF neurons on the hidden layer. Each p-component of the output $y = (y_1, \ldots, y_m)$is given by

$$y_p(x) = \sum_{i=1}^{r} w_{ip} h_i (\| x - w_i \|) \quad (4)$$

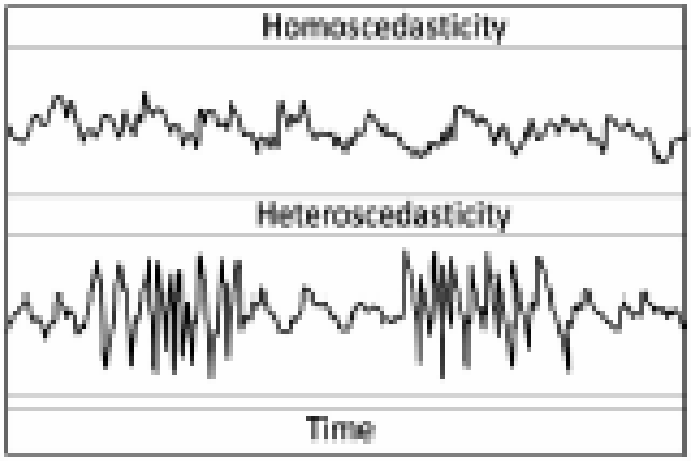
The propagation function calculates how close (using the euclidian distance) the input vector x is to the vector w_r. The gaussian activation function produces higher values for input vectors that are close to the vector w_r and smaller values for inputs that are far away from it. Thus the weights form clusters in the input space.

3. Heteroscedasticity

The works of Mandelbrot (1963) and Fama (1965) were among the first few works that examined the statistical properties of stock returns. During the 1980s, Engle (2003) worked on improving time-series analysis. Statistical techniques then in use mostly treated volatile variables, such as stock prices, as constants, even though such variables can change significantly from day to day and week to week. Engle (2003) after observing the variance of stock returns, developed a statistical technique known as ARCH (autoregressive conditional heteroscedasticity), which uses previously observed patterns of variance to predict future volatility. Refinements of ARCH models are now being used in banking and finance to help determine the prices and risk involved of investing in stocks . The theoretical framework described in this section is adopted from Tsay (2005)

A univariate stochastic process Y is said to be homoscedas-tic if the standard deviations of terms are constant for all times t. Otherwise, it is said to be heteroscedastic (see Figure 3 adopted from www.riskglossary.com). A process is unconditionally heteroscedastic if unconditional standard deviations σ_t are not constant. It is conditionally heteroscedastic if conditional standard deviations $\sigma_{t|t-1}$ are not constant. For example, stock or bond returns tend to be conditionally heteroscedastic. These prices exhibit non-constant volatility, but periods of low or high volatility are generally not known in advance. New Delhi electricity prices, on the other hand, exhibit unconditional heteroscedasticity. The prices tend to have higher volatilities during the Summer than during other seasons. This is predictable, therefore the electricity prices exhibit unconditional heteroscedasticity. If a process is unconditionally heteroscedastic, then it is necessarily conditionally heteroscedastic. The converse is not true. All these notions extend to higher dimensions. A multivariate stochastic process Y is said to be homoscedastic if its covariance matrix is constant for all times t, etc.

Figure 3: Homoscedastic vs. Heteroscedastic



3.1. Structure of a Model

Let r_t be the log return of an asset at time index t. In the volatility study r_t is either taken as serially uncorrelated or with minor lower order serial correlations, but it is a dependent series. The conditional mean and variance of r_t given the information set available at time $t-1$ as I_{t-1} are specified as,

$$\mu_t = E(r_t \mid I_{t-1}), \quad \sigma_t^2 = Var(r_t \mid I_{t-1}) = E[(r_t - \mu_t)^2 \mid I_{t-1}] \tag{5}$$

Since serial dependence of r_t is weak, we can say that r_t follows a simple time series model like stationary $ARMA(p, q)$ model. The model becomes

$$r_t = \mu_t + e_t, \quad \mu_t = c + \sum_{i=1}^{p} \varphi_i r_{t-i} - \sum_{i=1}^{q} \theta_i e_{t-i}, \tag{6}$$

where p, and q are non-negative integers and e_t are innovations or error terms, $e_t \sim N(0, \sigma_t^2)$. This is the mean equation for r_t. The order (p, q) of an ARMA model may depend on the frequency of the return series. The variance can be specified as

$$\sigma_t^2 = Var(r_t \mid I_{t-1}) = Var(e_t \mid I_{t-1}) \tag{7}$$

3.2. The ARCH Model

The major assumption behind the least square regression is homoscedasticity i.e constancy of variance. If this condition is violated, the estimates will still be unbiased but they will not be minimum variance estimates. The standard error and confidence intervals calculated in this case become too narrow, giving a false sense of precision. ARCH and related models handle this by modeling volatility itself in the model and thereby correcting the deficiencies of least squares model. The ARCH models (Engle, 1982; Tsay, 2005), characterized by mean and volatility equations, are specified as

$$r_t = \mu + e_t, \quad e_t = \sigma_t \in_t, \quad \sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i e_{t-i}^2 \tag{8}$$

$$e_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i e_{t-i}^2 + \in_t, t = p+1, \ldots, T \tag{9}$$

where $\in_t$, denotes the error term and T is the sample size. This is called ARCH(p) model. The next step is to check the ARCH effects by using residuals of mean equation. We can apply the usual Ljung-Box statistics $Q(p)$ to the $\{e_t^2\}$ series (McLeod and Li, 1983) or apply the white's test of significance of $\alpha_i = 0 (i = 1, \ldots, p)$ by F-statistic under the null hypothesis $Ho: \alpha_1 = \ldots = \alpha_p = 0$.

This F-statistic is distributed as χ^2 distribution.

If ARCH effects are found significant we can use the PACF of e_t^2 to determine the ARCH order. After specifying the volatility model we perform the joint estimation of the mean and volatility models. Lastly we evaluate the fitted model and further refine it. The standardized residuals, $\tilde{e}_t = \frac{e_t}{\sigma_t}$ can be seen to check the adequacy of a fitted ARCH model. We can evaluate the QQ plots of $\tilde{e}_t$ and $\tilde{e}_t^2$ to check validity of mean and variance equations respectively. After finding the parameters of the model, prediction can be done.

3.3. The GARCH Model

Bollerslev proposed a useful extension known as the generalized ARCH (GARCH) model. The e_t follows a GARCH(p, q) model (Bollerslev, 1986; Tsay, 2005) if

$$e_t = \sigma_t \in_t, \quad \sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i e_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{10}$$

In addition to ARCH conditions, we also have $\beta_j \geq 0$, and $\sum_{i=1}^{max(p,q)} (\alpha_i + \beta_i) < 1$. The constraint on α_i+β_i implies that the unconditional variance of e_t is finite, whereas its conditional variance σ_t^2 evolves over time. The α_i and β_j are ARCH and GARCH parameters, respectively. Similar to ARCH models, GARCH models also observe volatility clustering, leverage effect and heavier tails. Specifying the order of a GARCH model is not easy and only lower order GARCH models are used in most applications.

3.4. The Integrated GARCH (IGARCH) Model

IGARCH models are unit-root GARCH models. An IGARCH(p,q) model can be written as

$$e_t = \sigma_t \in_t, \quad \sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i e_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{11}$$

where additional constraints are $\sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j = 1$ and $1 > \beta_j > 0$.

3.5. The GARCH-M Model

Often the return of a security may depend on its volatility. In these cases, we use GARCH-M or GARCH in mean model. A GARCH(p,q)-M model can be specified as

$$r_t = \mu + k\sigma_t^2 + e_t, \quad e_t = \sigma_t \in_t$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i e_{t-i}^2 + \sum_{j=i}^{q} \beta_j \sigma_{t-j}^2 \tag{12}$$

The constant parameter k is called the risk premium parameter.

3.6. The Exponential GARCH (EGARCH) Model

An EGARCH model is specified as

$$e_t = \sigma_t \in_t, \quad In(\sigma_t^2) = \alpha_0 + \sum_{i=1}^{p} \alpha_i \frac{e_{t-i}}{\sigma_{t-i}} + \sum_{j=1}^{q} \lambda_j In(\sigma_{t-j}^2) + \sum_{i=1}^{p} \gamma_i \left(\frac{|e_{t-i}|}{\sigma_{t-i}} - \sqrt{\frac{2}{\pi}} \right) \tag{13}$$

For $e_t \sim N(0,\ \sigma_t^2)$ the standardized variable $\frac{e_t}{\sigma_t}$ follows a standard normal distribution and consequently $E(\frac{|e_t|}{\sigma_t}) = \sqrt{\frac{2}{\pi}}$. The parameters α_i capture the leverage effect. For good news $(\frac{e_{t-i}}{\sigma_{t-i}} > 0)$ the impact of innovation e_{t-i} is $(\alpha_i + \gamma_i)\frac{e_{t-i}}{\sigma_{t-i}}$ and for bad news $(\frac{e_{t-i}}{\sigma_{t-i}} < 0)$ it is $(-\alpha_i + \gamma_i)\frac{e_{t-i}}{\sigma_{t-i}}$. If α_i becomes 0, $In(\sigma_t^2)$ responds symmetrically to $\frac{e_{t-i}}{\sigma_{t-i}}$. To produce a leverage effect αi must be negative. The fact that the EGARCH process is specified in terms of log-volatility implies that σ_t^2 is always positive and, consequently, there are no restrictions on the sign of the model parameters.

3.7. The Threshold GARCH (TGARCH) Model/GJR Model

A TGARCH(p, q) model as proposed by (Glosten et al., 1993) can also handle leverage affects model by assuming the following form

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i e_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 + \sum_{i=1}^{p} \gamma_i v_{t-i} e_{t-i}^2 \tag{14}$$

where

$$v_{t-i} = \begin{cases} 1, & \text{if } e_{t-i} < 0 \\ 0, & \text{if } e_{t-i} \geq 0 \end{cases} \tag{15}$$

and α_i, γ_i, and β_j are non-negative parameters satisfying conditions similar to those of GARCH models. It can be seen that a positive e_{t-i} contributes $\alpha_i e_{t-i}^2$ to σ_t^2, whereas a negative e_{t-i} has a larger impact $(\alpha_i + \gamma_i) e_{t-i}^2$ with $\gamma_i > 0$.

4. Neural Networks in Volatility Estimation

4.1 Estimation of Conditional Volatilities

Neural networks can be used to estimate the conditional volatility (Giacomini, 2003; Eun and Resnick, 2004; Bhalla, 2008) of financial time series. Consider that a time series with stochastic volatility follows an AR(p)-ARCH(p) process with the following form

$$\begin{aligned} r_{t+1} = \ & NN\left(r_t, r_{t-1}, r_{t-2}, \ldots r_{t-p+1}, X_t^1, \ldots, X_h^t\right) + \\ & NN\left(r_t, r_{t-1}, r_{t-2}, \ldots r_{t-p+1}, X_t^1, \ldots, X_h^t\right) \in_{t+1} \end{aligned} \tag{16}$$

where $\in_t$, is i.i.d. with $E(\in_t) = 0$, $E(\in_t^2) = 1$.

Defining $z_t = (r_t, \ldots r_{t-p+1}, X_t^1, \ldots, X_h^t) \in \mathrm{R}^{p+h}, z \in \mathrm{R}^{p+h}$ the AR(p)-ARCH(p) process can be written as

$$r_{t+1} = \ (z_t) + \ (z_t) \in_{t+1} \tag{17}$$

We can write

$$\psi^2(z) = \theta(z) - \phi^2(z) \tag{18}$$

where

$$E\left[\mathrm{r}_{t+1} \mid \mathrm{z}_t = \mathrm{z}\right] = \phi(z) \tag{19}$$

$$E\left[\mathrm{s}_{t+1}^2 \mid \mathrm{z}_t = \mathrm{z}\right] = \theta(z) \tag{20}$$

$$Var\left[\mathrm{r}_{t+1} \mid \mathrm{z}_t = \mathrm{z}\right] = \psi^2(z) \tag{21}$$

Using a neural network Φ_{NN} to approximate $\phi(z)$ and Θ_{NN} to approximate $\theta(z)$, we obtain

$$\hat{\phi}(z) = \Phi_{NN}(z, \hat{w}_1) \tag{22}$$

$$\hat{\theta}(z) = \Theta_{NN}(z, \hat{w}_2) \tag{23}$$

where

$$\hat{w}_1 = argmin \frac{1}{N-p} \sum_{t=p}^{N-1} (r_{t+1} - \Phi_{NN}(z_t; w_1))^2 \tag{24}$$

$$\hat{w}_2 = argmin \frac{1}{N-p} \sum_{t=p}^{N-1} (s_{t+1}^2 - \Theta_{NN}(z_t; w_2))^2 \tag{25}$$

An estimator of $\Psi^2(Z)$ is obtained as

$$\hat{}^2(z) = \hat{\theta}(z) - \hat{}^2(z) \tag{26}$$

Hardle et al. (2002) used the approach where the residuals are substituted by the sample residuals. Approximating the residuals through the sample residuals

$$\hat{\in}_{t+1} = r_{t+1} - \hat{}(z_t) \tag{27}$$

and the squared sample residuals with a neural network Ψ_{NN} with parameters

$$\hat{w} = argmin \frac{1}{N-p} \sum_{t=p}^{N-1} (\in_{t+1}^2 - \psi_{NN}(z_t; w))^2 \tag{28}$$

the estimation of the conditional volatility can be written as

$$\hat{}^2(z) = \psi_{NN}(z, \hat{w}) \tag{29}$$

4.2. Estimation of Implied Volatilities

In their landmark paper,Black and Scholes (1973) gave model to determine the price of a call option C_t at time t, which is given by the formula

$$C_t = S_t \Phi(d_1) - Ke^{-r} \Phi(d_2) \tag{30}$$

$$d_1 = \frac{ln \frac{s_t}{K} + \left(r + \frac{1}{2}\sigma^2\right)\tau}{\sigma\sqrt{\tau}} \tag{31}$$

$$d_2 = d_1 - \sigma\sqrt{\tau} \tag{32}$$

where S_t is the spot price of the underlying asset, σ the volatility of the underlying asset price process, r the risk free interest rate, τ the time to maturity, K the strike price of the option and Φ the cumulative distribution function of the normal distribution. The Black Scholes model assumes that σ is constant over the price process of a given underlying asset.

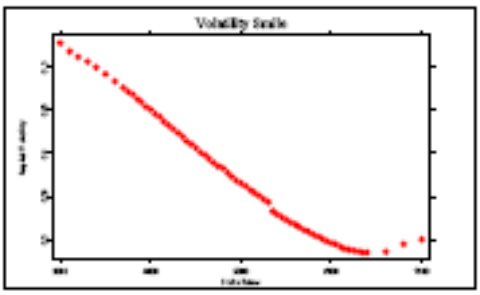
Since actual volatility of the underlier can not be observed directly, we use the volatility which is implied by option prices observed in the markets. This is called implied volatility and is defined as the parameter $\hat{\sigma}$ that yields the actually observed market price of a particular option when substituted into the Black-Scholes formula. The implied volatility of a European put with the same strike and maturity can be deduced using the **put-call parity.**

$$C_t - P_t = S_t - Ke^{-r} \tag{33}$$

In opposite to the theoretical formulation, the implied volatilities are not constant. They form a *volatility smile* when plotted against the strike prices K at time t,(Hardle et al., 2002) and change also according to the time to maturity τ.

Calculation of implied volatilities at different strikes and maturities yields a surface called implied volatility surface on a specified grid using a bi-dimensional kernel smoothing procedure. A Nadaraya-Watson estimator with a quartic kernel is usually employed.

Figure 4: Volatility Smile



We can write the dependency of the implied volatility the strike price K or the moneyness $\frac{K}{S}$ and time to maturity τ as.

$$\hat{\sigma} = f(K, \tau) \tag{34}$$

or

$$\hat{\sigma} = f\left(\frac{K}{S}, \tau\right) \tag{35}$$

This relation being non linear form can be estimated with neural networks, given that implied volatilities for a strike price or moneyness and for different maturities are available to construct the training set. The network Ψ_{NN}

$$\hat{\sigma} = \psi_{NN}\left(\frac{K}{S}, \; ; \hat{w}\right) \tag{36}$$

where

$$\hat{w} = argmin \; \frac{1}{n-1}\sum_{t=1}^{n}\left(\hat{\sigma}_i - \psi_{NN}\left(\frac{K}{S}, \tau; \hat{w}\right)\right)^2 \tag{37}$$

is used to generate *implied volatility surface.*

5. Experiment

The experiment described on this section compares one step ahead forecasts of times series produced by MLP and RBF networks with different architectures by changing the number of neurons in the hidden layer.

Five different exchange rate time series and ten different architectures are used. A non linear time dependency of size (lag) p, is considered for all the series. The experiment uses a network ϕ_{NN} with one hidden layer containing h neurons to forecast the realizations of the time series at $t + 1$, as given in Equation 38.

$$r_{t+1} = \; _{NN}(r_t, \; r_{t-1}, r_{t-2}, \ldots r_{t-p+1}) \tag{38}$$

Afterwards, the performance of the forecasts are evaluated.

5.1. Time Series

Five exchange rate time series used are daily observations of:British Pound vs US-Dollar (BPUSD) from 28/12/1993 to 28/12/2008, German Mark vs US-Dollar (DEMUSD) from 28/12/1993 to 28/12/2006, Japanese Yen vs US-Dollar (JPYUSD) from 28/12/1993 to 28/12/2008, Indian Rupees vs US-Dollar (IRUSD) from 28/12/1993 to 28/12/2008 and Euro vs US-Dollar (EURUSD) from 15/11/1998 to 28/12/2008

5.2. Transformation

To eliminate trend and seasonality the time series are transformed by first differences of logarithms. After this operation, the time series elements r_t represent the logarithm of the financial return of holding a unit of the currency for one period:

$$r_t = log(p_t) - log(p_{t-1}) = log\left(\frac{p_t}{p_{t-1}}\right) \tag{39}$$

The time series $r_{t_{t=1}}^{N}$ are split into two sets, the training set and the test set:the training set contains roughly 95% of the observations, i.e., $t = (1, \ldots, t_0)$, $t_0 = mod(0.95N)$ and the test set contains roughly 5% of the observations, i.e., $t = (t_0 + 1, \ldots, N)$ The table 1 shows the information about the time series and size of subsets used.

Table 1: Time series and sample size

Time Series	From	To	t_0	N
BPUSD	28/12/1993	28/12/2008	5206	5480
DEMUSD	28/12/1993	28/12/2006	4512	4749
JPYUSD	28/12/1993	28/12/2008	5206	5480
INRUSD	28/12/1993	28/12/2008	5206	5480
EURUSD	15/11/1998	28/12/2008	3484	3667

5.3. Time Dependency

The process is modeled with lag 5, the realization at $t + 1$ is dependent on the realizations of the last 5-trading days.

5.4. Networks

Various kinds of parameters which can be adjusted in Neural Network architecture are number of units, number of hidden layers, type of neurons, learning rates for supervised and unsupervised training and initial weights etc. In our ex-periment,the RBF and MLP networks are built in XploRe Software with one hidden layer of h neurons forming the architecture $5 - h - 1$. The number h of units on the hidden layer is increased from 5 to 50 in steps from 5 units.

For each architecture, the networks are trained on the training sets until a MSE from 5.10^{-5} or less is reached. The other parameters are the defaults for RBF and MLP training quantlets from the XploRe neural networks library.

5.5. Performance Measures

The forecasts are made on the test set $t = (t_0 + 1, \ldots, N)$. The $k = N - (t_0 + 1 + lag)$ forecasts are compared with the true realizations. Following performance measures are used.

Normalized Mean Squared Error (NMSE)

$$NMSE = \frac{1}{k}\sum_{t=t_f}^{N}\frac{(r_t - \hat{r}_t)^2}{\hat{\sigma}^2} \tag{40}$$

where $\hat{\sigma}^2$ is the variance of the training set (in sample unconditional volatility)

Mean Absolute Error (MAE)

$$MAE = \frac{1}{k}\sum_{t=t_f}^{N} \mid r_t - \hat{r}_t \mid \tag{41}$$

NMSE and MAE are the metrics used to estimate the error of prediction.

Function (SIGN)

$$SIGN = \frac{1}{k}\sum_{t=t_f}^{N} \delta_t \tag{42}$$

where

$$\delta_t = \begin{cases} 1, & if\ r_{t+1}\,\hat{r}_{t+1} \geq 0 \\ 0, & if\ otherwise \end{cases} \tag{43}$$

To evaluate whether the result of the network can be used as a trading strategy, the fraction of predictions with same sign as the true realizations is calculated by the metric SIGN.

5.6. Results and Discussion

5.6.1. Neural Networks

For each time series, the result is summarized below.

BPUSD: The RBF networks performed better than the MLP for all architectures, concerning NMSE and MAE. The best network is a RBF with 20 hidden units.

DEMUSD: The MLP networks performed better than the RBF for all architectures (except for 5 hidden units), concerning NMSE and MAE. The best network is a RBF with 5 hidden units, the second best a MLP with 5 hidden units.

JPYUSD: The RBF networks performed better than the MLP for all architectures, concerning NMSE and MAE. The best network is a RBF with 10 hidden units.

INRUSD: The MLP networks performed better than the RBF for all architectures , concerning NMSE and MAE. The best network is a MLP with 15 hidden units.

EURUSD: The MLP networks performed better than the RBF for 6 architectures while RBF networks performed better than the MLP for 4 architectures, concerning NMSE and MAE. The best network is a RBF with 25 hidden units.

In all the cases the result of the network can be considered as a trading strategy, since the value of the the function **SIGN** is greater than 0.5.

5.6.2. Conditional Heteroscedastic Models

USD/GBP Series

$$r_t = -0.000065 + a_t;\ a_t = \sigma_t \in_t;\ \sigma_t^2 = 0.001795 + 0.952942a_{t-1}^2 + 0.036520\sigma_{t-1}^2 \tag{44}$$

USD/DM Series

$$r_t = -0.000043 + a_t;\ a_t = \sigma_t \in_t;\ \sigma_t^2 = 0.000888 + 0.975946a_{t-1}^2 + 0.020566\sigma_{t-1}^2 \tag{45}$$

USD/JPY Series

$$r_t = 0.000045 + a_t;\ a_t = \sigma_t \in_t;\ \sigma_t^2 = 0.002829 + 0.949508a_{t-1}^2 + 0.041940\sigma_{t-1}^2 \tag{46}$$

USD/INR Series

$$r_t = 0.000029 + a_t;\ a_t = \sigma_t \in_t;\ \sigma_t^2 = 0.000000 + 0.9829711a_{t-1}^2 + 0.169468\sigma_{t-1}^2 \tag{47}$$

USD/EUR Series

$$r_t = -0.000124 + a_t;\ a_t = \sigma_t \in_t;\ \sigma_t^2 = 0.000545 + 0.979012a_{t-1}^2 + 0.019795\sigma_{t-1}^2 \tag{48}$$

5.6.3. Comparison of Neural Network and Conditional Heteroscedastic Models

Comparisons of Neural Network models and conditional heteroscedastic models (GARCH, GARCH-M, EGARCH, TGARCH/GJR and IGARCH) for all the five exchange rates are shows in following tables (Tables 2, 3, 4, 5 and 6)

Table 2: Comparative analysis of USD/GBP Series

Model	NMSE	MAE.10^2	SIGN
Neural Network	**0.56853**	**0.2274**	**0.71429**
GARCH(1,1)	1.00011	0.27177	0.63689
GARCH(1,1)-M	0.86999	0.26832	0.67199
EGARCH(1,1)	0.89286	0.32988	0.68674
TGARCH/GJR(1,1)	0.87995	0.26164	0.68790
IGARCH(1,1)	0.80463	0.27175	0.65564

Table 3: Comparative analysis of USD/DM Series

Model	NMSE	MAE.10^2	SIGN
Neural Network	**0.73085**	**0.35033**	**0.53571**
GARCH(1,1)	0.83113	0.46988	0.49777
GARCH(1,1)-M	0.88089	0.44983	0.48563
EGARCH(1,1)	0.74093	0.36979	0.52728
TGARCH/GJR(1,1)	0.76097	0.38988	0.50787
IGARCH(1,1)	0.80471	0.41044	0.49768

Table 4: Comparative analysis of USD/JPY Series

Model	NMSE	MAE.10^2	SIGN
Neural Network	**0.74021**	0.37706	**0.57143**
GARCH(1,1)	0.89899	0.36216	0.52421
GARCH(1,1)-M	0.84957	0.36010	0.52319
EGARCH(1,1)	0.79654	0.32844	0.56324
TGARCH/GJR(1,1)	0.80971	**0.32207**	0.55919
IGARCH(1,1)	0.82374	0.36236	0.53760

Comparative analysis clearly show that Neural network has an edge over the conditional heteroscedastic models like GARCH, GARCH-M, GJR (TGARCH), IGARCH, EGARCH etc for exchange rate forecasting. Neural Network clearly outperform other models in terms of the metric "SIGN" consistently. This shows that upward/downward movement of exchange rate is more accurately predicted by Neural Networks as compared to conditional heteroscedastic models. However in some isolated cases conditional heteroscedastic models did fare better than Neural Networks. These cases are mentioned below

Table 5: Comparative analysis of USD/INR Series

Model	NMSE	MAE.10^2	SIGN
Neural Network	2.0925	**0.24842**	**0.69643**
GARCH(1,1)	1.87319	0.30284	0.57721
GARCH(1,1)-M	1.83982	0.30519	0.55010
EGARCH(1,1)	1.69866	0.28722	0.61625
TGARCH/GJR(1,1)	**1.64332**	0.27517	0.62360
IGARCH(1,1)	1.68163	0.29829	0.59089

Table 6: Comparative analysis of USD/EUR Series

Model	NMSE	MAE.10^2	SIGN
Neural Network	**0.88442**	0.38644	**0.58929**
GARCH(1,1)	0.99998	0.34930	0.48804
GARCH(1,1)-M	0.93987	0.34918	0.49592
EGARCH(1,1)	0.89833	**0.30927**	0.53994
TGARCH/GJR(1,1)	0.89986	0.31414	0.52785
IGARCH(1,1)	0.92064	0.34999	0.46256

1. "MAE" metric of GJR is better than that of Neural Networks in case of USD/JPY (exchange rate of Japanese Yen) series.
2. "NMSE" metric of GJR is better than that of Neural Networks in case of USD/INR (exchange rate of Indian Rupee) series.
3. "MAE" metric of EGARCH is better than that of Neural Networks in case of USD/EUR (exchange rate of Euro) series.

Barring these three cases, Neural Networks performed significantly better than conditional heteroscedastic models. Within the conditional heteroscedastic models, the performance of IGARCH and TGARCH was better than other heteroscedastic models.

6. Estimation of Conditional Volatilities

Both Neural Networks and GARCH(1,1) models are used to estimate conditional volatilities of exchange rate series. The results are found to be comparable. For US/GBP series the estimation of conditional volatility using both Neural Networks and the GARCH(1,1) models are shown below.

Since Volatility can't be observed directly, so comparison between the charts can't be made. However since the edge of Neural Networks over the conditional heteroscedasticity models has already been shown in case of Exchange rate forecasting, it can be said that Neural Networks will fare better in this case too and more profit can be made if volatility estimates based on Neural Networks are used for volatility trading strategies.

Figure 5: Log returns and conditional volatilities from the exchange rate British Pound / US Dollar from 28/12/1993 to 28/12/2008. Estimated with RBF network, 25 hidden units, lag 5

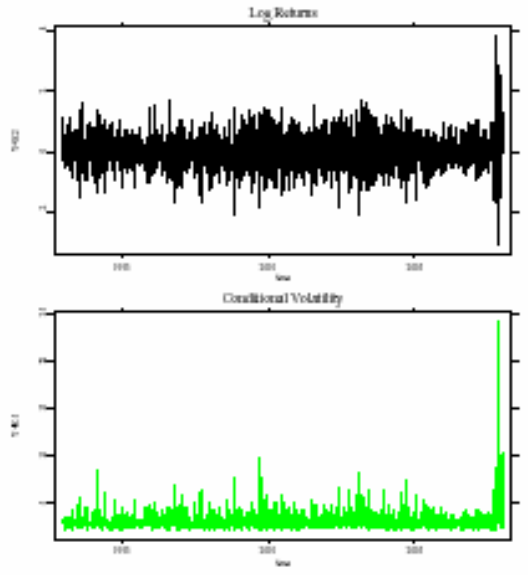
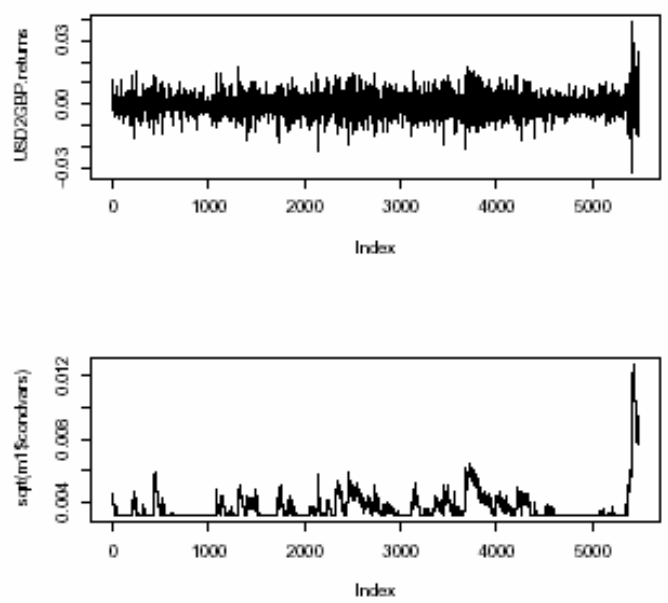


Figure 6: GARCH (1,1) Fitted model for British Pound / US Dollar Series



7. Estimation of Implied Volatilities

Finally the implied volatility surface are estimated from the data set *31-MAR-2008, 29-SEP-2008 and 31-DEC-2008.dat* using neural networks. The data set contains settlement price of the NIFTY Option (underlying asset), strike prices, interest rates, times to maturity and prices from puts and calls traded at the National Stock Exchange (NSE), India on 31/03/2008,29/08/2008 and 31/12/2008.

The implied volatility surface estimated through a MLP network with 15 hidden units is shown on Figures 7, 9 and 11. The implied volatility surface estimated through a RBF network with 25 hidden units is shown on Figures 8, 10 and 12. All pictures also show the implied volatility curves (red), used on the estimation of the surface. The results shows volatility surfaces for pre meltdown, meltdown and post meltdown periods.

The extreme volatilities due to financial meltdown are clearly visible in the figures.

Conclusion

1 The phenomenons of volatility clustering, autocorrelation, heteroscedasticity are observed in Nifty returns.
2 Neural Networks do a fairly good job in **forecasting Exchange rate** (NMSE 0.6, Sign 0.6).RBF networks do considerably better than MLP networks in this case.
3 **Conditional heteroscedastic models** can be effectively used to predict mean and volatility of NIFTY daily returns.
4 Within the conditional heteroscedastic models, the performance of **IGARCH and TGARCH** was better than other heteroscedastic models.
5 Neural Networks performance is better (around 10-15% improvement) than conditional heteroscedasticity models like GARCH, GARCH-M, EGARCH, GJR, IGARCH etc in **forecasting Exchange rate.**
6 Neural network can be effectively employed to estimate the **conditional volatility** (besides existing methods of conditional heteroscedasticity models like GARCH, GARCH-M, EGARCH, TGARCH, IGARCH etc.)
7 Neural network can be effectively employed to estimate the **implied volatility** of the options.

Figure 7: Implied volatility surface estimated using a (2-15-1) MLP. Parameters: strike prices and maturities. Data: NIFTY Option at NSE, India on 31/03/2008

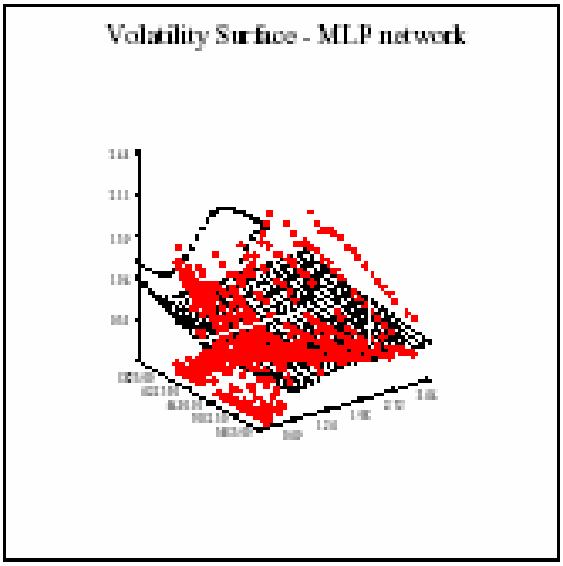
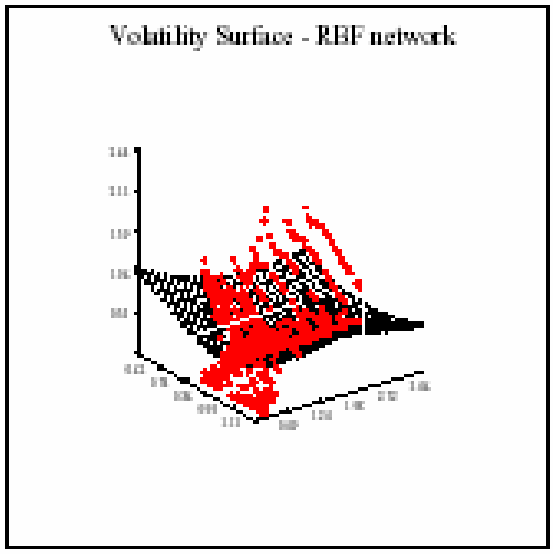


Figure 8: Implied volatility surface estimated using RBF network with 25 hidden units. Parameters: moneyness and maturities. Data: NIFTY Option at NSE, India on 31/03/2008



8 **Volatility surface** can be effectively generated after getting the forecast of implied volatility of options and plotting this along with exercise price and time to maturity.

9 RBF networks do considerably better than MLP networks in extracting the information necessary to perform a good generalization from the training set. The MLP may learn information specific to the training set that has no use for generalization. Besides that, we need to consider the possibility that MLPs with more than one hidden layer may generalize better, maybe better than RBFs.

10 The number of hidden units used does not seem to have a straight relation with the forecast performance. Networks with few hidden units performed better than networks with many hidden units and the way around.

Figure 9: Implied volatility surface estimated using a (2-15-1) MLP. Parameters: strike prices and maturities. Data: NIFTY Option at NSE, India on 29/08/2008

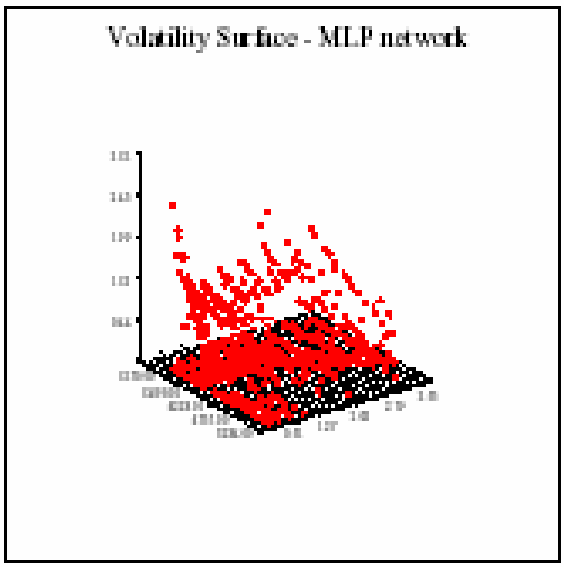


Figure 10: Implied volatility surface estimated using RBF network with 25 hidden units. Parameters: moneyness and maturities. Data: NIFTY Option at NSE, India on 29/08/2008

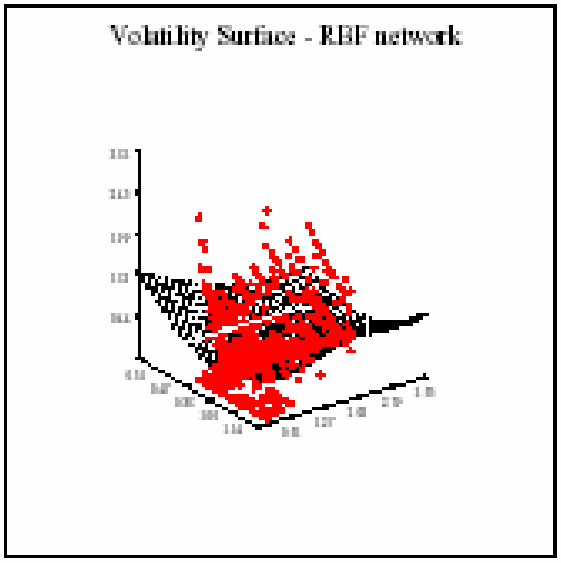


Figure 11: Implied volatility surface estimated using a (2-15-1) MLP. Parameters: strike prices and maturities. Data: NIFTY Option at NSE, India on 31/12/2008

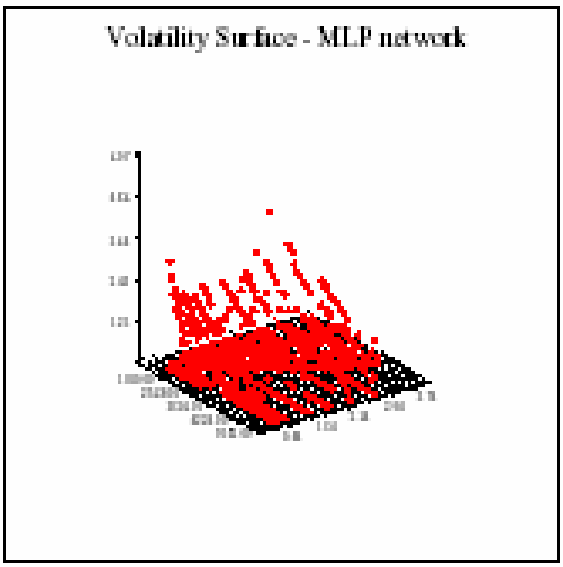
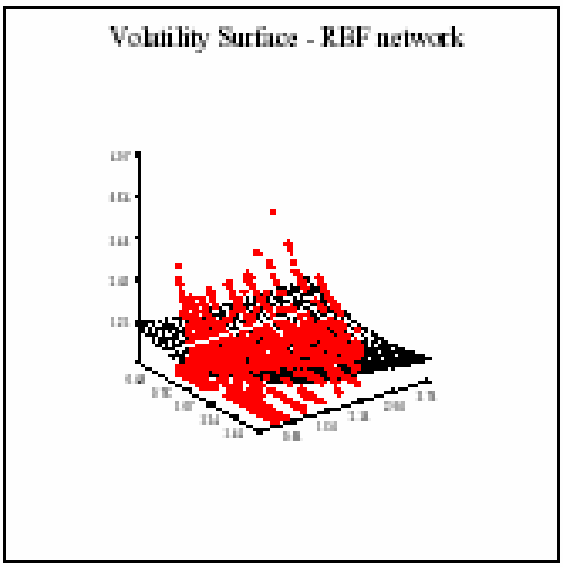


Figure 12: Implied volatility surface estimated using RBF network with 25 hidden units. Parameters: moneyness and maturities. Data: NIFTY Option at NSE, India on 31/12/2008



11 Neural network can simultaneously and effectively extract the non-linear functional form as well as model parameters (as opposed to conditional heteroscedasticity models where the functional form needs to be specified for estimation of parameters). Neural networks provide quantitative finance with strong support in problems related to non-parametric regression. Also remarkable are the heuristic considerations involved on the set up of neural networks: sometimes parameters and architectures are chosen only by trial and error.

References

[1] Andersen, T. and T. Bollerslev (1998). Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International Economic Review 39*(4), 885-905.

[2] Baillie, R. and T. Bollerslev (1989). The message in daily exchange rates: A conditional-variance tale. *Journal of Business and Economic Statistics 7*(3), 297-305.

[3] Bhalla, V. K. (2008). *International Financial management.* Delhi: Anmol Publications Pvt. Ltd.

[4] Bishop, C. M. (1995). *Neural Networks for Pattern recognition.* Oxford: Oxford University Press.

[5] Black, F. and M. S. Scholes (1973). The pricing of options and corporate liabilities. *Journal of Political Economy 81*, 637-654.

[6] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics 31*, 307-327.

[7] Chang, P. H. K. and C. L. Osler (1999). Methodical madness: Technical analysis and the irrationality of exchange-rate forecasts. *Economic Journal 10*, 636-661.

[8] Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflations. *Econometrica 50*, 987-1007.

[9] Engle, R. F. (2003). Risk and volatility: Econometric models and financial practice. Nobel Lecture.

[10] Eun, C. S. and B. G. Resnick (2004). *International Financial management*. New York: The McGraw Hill Companies.

[11] Fama, F. (1965). Random walks in stock market prices. *Financial Analysts Journal 21*, 55-59.

[12] Gencay, R. (1999). Linear, non-linear and essential foreign exchange rate prediction with simple technical trading rules. *Journal of International Economics 47*, 91-107.

[13] Giacomini, E. (2003). Neural networks in quantitative finance. Master's thesis,, University of Berlin.

[14] Glosten, L. R., R. Jagannathan, and D. E. Runkle (1993). On the relation between the expected value and the volatility of nominal excess return on stocks. *Journal of Finance 48,1779-1801.*

[15] Hardle, W., T. Kleinow, and Stahl (2002). *Applied Quantitative Finance*. Heildelberg: Springer Verlag.

[16] Haykin, S. (1999). *Neural Networks*. Upper Saddle River: Prentice Hall.

[17] Jorion, P. (1995). Predicting volatility in the foreign exchange market. *Journal of Finance 50*(2), 507-528.

[18] Kiani, K. M. and T. L. Kastens (2008). Testing forecast accuracy of foreign exchange rates: Predictions from feed forward and various recurrent neural network architectures. *Comput. Econ. 32*(4), 383-06.

[19] Kuan, C.-M. and T. Liu (1995). Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of Applied Econometrics 10*(4), 347-364.

[20] Mandelbrot, B. B. (1963). The variation of certain speculative prices. *Journal of Business 36*, 394-419.

[21] McLeod, A. I. and W. K. Li (1983). Diagnostic checking arma time series models using squared-residual autocorrelations. *Journal of Time Series Analysis 4,* 269-273.

[22] Meese, R. A. and A. K. Rose (1990). Nonlinear, nonpara-metric, nonessential exchange rate estimation. *American Economic Review Papers and Proceedings 80*, 192-196.

[23] Neely, C. J. (1999). Target zones and conditional volatility:the role of realignments. *Journal of Empirical Finance 6*(2), 177-192.

[24] Tsay, R. S. (2005). *Analysis of Financial Time Series*. New Jersey: Wiley Interscience.

[25] West, K. D. and D. Cho (1995). The predictive ability of several models of exchange rate volatility. *Journal of Econometrics 69*(2), 367-391.

[26] Yang, J. and N. Gradojevic (2006). Non-linear, non-parametric, non-fundamental exchange rate forecasting. *Journal of Forecasting 25*(4), 227-245.

[27] Yao, J. T. and C. L. Tan (2000). A case study on using neural networks to perform technical forecasting of forex. *Neurocomputing 34*(1-4), 79-98.