Charles University in Prague

Faculty of Social Sciences

Institute of Economic Studies

MASTER THESIS

Forecasting Realized Volatility Using Neural Networks

Author: **Bc. Jindřich Jurkovič**

Supervisor: **PhDr. Jozef Baruník, Ph.D.**

Academic Year: **2012/2013**

Acknowledgements

I would like to express my gratitude to my supervisor, PhDr. Jozef Baruník, Ph.D., for his valuable advice, careful reading of the work's drafts, his active comments, and especially for encouraging me to finish the work.

Declarations

1. I hereby declare that I compiled this thesis independently, using the listed resources and literature.

2. I allow Charles University to make my thesis publicly available for study purposes

Prague, May 16, 2013 signed, Jindřich Jurkovič

Abstract

In this work, neural networks are used to forecast daily Realized Volatility of the EUR/USD, GBP/USD and USD/CHF currency pairs time series. Their performance is benchmarked against nowadays popular Heterogenous Autoregressive model of Realized Volatility (HAR) and traditional ARIMA models. As a by–product of our research, we introduce a simple yet effective enhancement to HAR model, naming the new model HARD extension. Forecasting performance tests of HARD model are conducted as well, promoting it to become a reference benchmark for neural networks and ARIMA.

Keywords: Realized volatility, forecasting volatility, neural networks, HAR model, HARD model, ARIMA models, currency pairs, financial markets.

Abstrakt

Předkládaná práce se zabývá předpovídáním časových řad denní realizované volatility vybraných měnových párů EUR/USD, GBP/USD a USD/CHF, pomocí neuronových sítí. Jejich výsledky jsou porovnány s výsledky aktuálně populárního modelu HAR (Heterogenous Autoregressive) a již tradičních modelů ARIMA. Vedlejším produktem těchto snah je zdokonalení modelu HAR, které je nazváno HARD rozšířením. Po otestování jeho predikčních schopností je tento model dále použit jako referenční model pro testování neuronových sítí a modelů ARIMA.

Klíčová slova: Realizovná volatilita, předpovídání volatility, neuronové sítě, HAR model, HARD model, ARIMA modely, měnové páry, finanční trhy.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Technological progress over the last several decades has been remarkable. Evolution of computers has been especially precipitous and revolutionized many fields in today's technological society, financial markets making no exception. It has also brought some new challenges in practical and theoretical terms.

In a globalized world, financial markets are irreplaceable as international businesses and majority of economies depend on them. High quality decision making in financial markets is therefore essential; From assessing the relevant data, through their analysis, to deducing conclusions and taking proper actions, all processes are being more and more automated – even up to the level of fully automated trading computer systems, which account for up to more than 90 % of trading volume on several futures markets! This would be impossible without the devastating computational power, state of the art modeling and most importantly, machine learning techniques and the theory behind them.

Since the early stages of computer science and theoretical informatics, scientists have been building theoretical frameworks for "automated knowl-

edge discovery". The belief that computers will be capable of independent thinking was intensified during the Space Race and Cold War times, rich in high-quality scientific research. Endeavours are traceable even in Czechoslovak research history, for example the theoretical works such as *Hájek et al. (1966)* or *Hájek & Havránek (1978)*.

With considerable amounts of data being storaged in large computer databases, new approaches to data handling and analysis have to be constantly incorporated. Article by *Fayyad et al. (1996)* nicely summarizes the idea of data mining and knowledge discovery. Many theoretical methods are currently going through renaissance, induced by the rise of cloud–based computing.

Some of the basic machine learning methods, already exploited by many professionals in financial markets, slowly work their way into academic econometric research. Relative availability of decent computational power and open source dedicated software, along with better availability of market data to individuals, renders the task more and more feasible. In this work, we use the most basic tool for pattern recognition – the neural networks – for predicting the *realized volatility* time series in forex market.

The concept of realized volatility as an estimator of intrinsically unobservable true volatility (quadratic variation) of the price process, is relatively new, promoted mainly by works of *Barndorff-Nielsen & Shephard (2001)*, *Barndorff-Nielsen & Shephard (2002)* and *Andersen et al. (2003)*. Being an observable random variable, realized volatility allows to revert to more classical methods[1] of time series analysis. A simple long memory realized volatility model based on traditional autoregressive models was proposed by

[1]Overshadowed by ARCH and GARCH models, introduced by *Engle (1982)* and *Bollerslev (1986)*, respectively

Corsi (2009) and called HAR (Heterogenous Autoregressive) model.

Due to the relative freshness of realized volatility concept and the fact that neural networks are still closer to computer scientists rather than econometricians, the attempts to model realized volatility making use of neural networks are fairly scarce among econometric research papers, with exceptions such as *McAleer & Medeiros (2011)*. Yet, neural networks have been theoretically proven to be universal approximators – see *Hornik et al. (1989)*.

The presented work – to a large extent motivated by reading the book *McNelis (2005): Neural Networks in Finance* – hopes to contribute a bit to the, for the time being, thin set of publicly available econometric research utilizing neural networks in forecasting realized volatility. The work *a priori* resigned from the excessive theoretical depth underlying the machine learning and takes a more direct approach to data analysis, characteristic for the real–world data analysis practitioners. Chapters are organized as follows:

Chapters 2–5 introduce the realized volatility and give a short overview of general methods and models applied later in the work, with the emphasis on neural networks (NNs). Along the way, in Section 4.3, an improved version of the HAR model is presented.[2] In second part, Chapters 6–8 describe the three particular data sets of EUR/USD, GBP/USD and USD/CHF time series of daily realized volatilities and test the performance of models in consideration. Results of all the tests are well arranged in tables. Concluding remarks along with several ideas on possible further advance are summarized in Chapter 9.

The software used to estimate the actual models were `GRETL` for linear models and `MEMBRAIN` for neural networks. Some additional test were conducted using `MATLAB`. See References for the Internet links.

[2]More of a by–product of our research.

Chapter 2

Volatility

The term *volatility* inherently associates with financial markets. Often used in a rather intuitive way to express the perceived dynamics in the price movements (or returns), there is no widely accepted rigorous definition of the term.

Through the econometric literature, however, the idea of volatility as one of the intrinsic features of financial time series coincides with *conditional variance*, which – while theoretically well defined – is a latent variable, unobservable in its nature.

Yet, many trading models and portfolio management theories incorporate volatility without any hesitation. Hence the study of volatility is of great importance. Assessing the latent "true" volatility is an uneasy task. Part of the problem proves to be the usual assumptions of frictionless markets accompanied by continuous price-setting process.

Quite unsurprisingly, prices on financial markets are discreet, jumping rather than shifting among price levels, and financial markets clearly display microstructure reflecting the very way trades are exercised via quotes. It would seem that the theoretical approach to volatility employs statistical

virtuosity to create its own perfect world in a way that its validity is unverifiable in the real world. To rephrase a statement in *Andersen et al. (2003)*, any practical implementation (of volatility) must confront the fact that *no financial market* provides a frictionless trading environment with continuous price recording.

Now, how do you measure, predict and validate unobservable variable? The simple answer is, you don't. In the previous statement the important word is *practical*. The only reasonable practice and way out of this trouble is to use some proxy variable instead. That's where *realized* volatility comes to play, solving at least the problem of measurability. However, it is important to keep in mind that it is just a proxy to the theoretical volatility.

To stress this point adequately, let us take a brief detour through integrated variance to hint a connection (or rather detachment) between theoretical volatility and realized volatility measure.

2.1 Integrated Variance

Examining arbitrary financial time series of sufficient length, one will find peaceful periods as well as periods of violent changes. One of the reasonable explanations to this is that volatility of price process is not constant.

Should we assume that a given financial time series behaves according to an intrinsic time-variant *actual* underlying probability distribution or density function of *a priori* known properties, then the natural candidates for volatility measurement would be its variance σ^2 or simply standard deviation σ. The latter is commonly referred to as volatility.

The basic idea wrapped in different mathematical representations is that the time-variant variance matches the data variance on corresponding time

intervals (possibly infinitely short yet dense with observations) with respective standard deviations $\sigma(\tau)$ defined at any given point in time τ.

For a continuous time interval I we then define *integrated variance* by setting

$$IV_I \equiv \int_{\tau \in I} \sigma(\tau)\, d\tau\,. \tag{2.1}$$

Clearly such definition cannot be utilized directly, as we do not have instantaneous standard deviations at our disposal. However, definition (2.1) serves as a starting point for establishing proxy measures, as integrated variance is considered to be representative "true volatility" measure over given time interval.

Some other works, including *Corsi (2009)* define integrated variance on the interval as

$$IV_I^2 \equiv \int_{\tau \in I} \sigma^2(\tau)\, d\tau\,, \tag{2.2}$$

and call the square root of (2.2) the integrated volatility. Of course, the uncomputability remains.

The way we constructed integrated variance – although by no means rigorous – suggests the need for price process to be continuous. The reason is purely mathematical, so that the sigmas are integrable on time intervals. Assumption of continuity can be dropped in a way that preserves integrability. Indeed, there are models of price process encompassing price jumps, substituting the assumption of continuity with semi-continuity and dealing with jumps separately.

Further investigation of such models wouldn't serve this work any purpose. Curious reader will find sufficient number of references along with more rigorous way of reaching the integrated variance formula (2.1) in *Andersen et al. (2003)* or in *McAleer & Medeiros (2011)*. Especially the former men-

tioned paper expands the theoretical basis substantially only to admit that in the end, these results are inapplicable.

2.2 Realized Volatility

For the transition to realized volatility, we take integrated volatility (2.2) and simply replace the integral with sum and instantaneous variances with observable realized returns over finite number of consecutive subintervals of I. It is essentially possible to take subintervals of different lengths – say, based on similar number of ticks in the market. Let us have n consecutive subintervals $I_i \subset I$ with respective returns r_i and express the sum as

$$\sum_{i=1}^{n} c_i r_i^2 \,, \tag{2.3}$$

where c_i are constants reflecting length of subintervals I_i. Another measure could be used in similar manner, replacing squared returns with their absolute values, such as

$$\sum_{i=1}^{n} c_i |r_i| \,. \tag{2.4}$$

Similarity of expressions (2.4) and (2.3) with those in (2.2) and (2.1), respectively, is obvious. Most practical applications consider subintervals of the same length based on expression (2.3). Pleasant bonus comes as computational ease (constants c_i disappear) with more available sampled datasets.

Let us therefore define *realized variance* over time interval I the usual way as

$$RV_I \equiv \sum_{i=1}^{n} r_i^2 \,, \tag{2.5}$$

where r_i represent realized returns[1] over respective subintervals I_i of the *same* length. *Realized volatility* on interval is then defined as square root of (2.5).

[1] That is, the difference between closing price and opening price within subinterval.

There are, of course, many ways to sample the data. Choosing different sampling, represented by n when utilizing definition (2.5), will result in different values of RV over the same interval! It would seem that finer sampling should yield more precise approximation of integrated variance, nevertheless, due to real-wold market microstructure, that is not the case. It is imperative to be aware of the simple truth that if we look at RV as an estimator of IV, then such estimator is biased and inconsistent.[2] Moreover, the bias increases with the sampling frequency.[3]

To avoid any confusion, let us forget the ostensible relation of realized variance to integrated variance, and view it realistically as an empirical variance measured on real-world time series. Definition (2.5) creates whole class such variances, dependent on the sampling. We shall focus on *daily* realized variance incorporating intraday returns. Typical intraday subintervals or data frequencies are minute data, 5-minute, 15-minute, 30-minute, hourly and 4-hour datasets, denoted M1, M5, M15, M30, H1 and H4, respectively.

[2] *McAleer & Medeiros (2011)* state that under regularity conditions, realized variance is a consistent estimator of integrated variance. If these assumptions are broken, RV_t becomes biased and inconsistent (*e.g.* returns are serially correlated). See the reference for a more detailed discussion and further references.

[3] *Corsi (2009)*, see the reference also for roughly estimates of biases and further references on that topic.

Chapter 3

Statistical Methods

This chapter summarizes the mix of methods and statistics used forth in the thesis. The mix defines the necessary minimum – presented as simply as possible – in order to anticipate what's going on in chapters to follow and how the results are obtained and evaluated in this work.

This rather general summary should prove useful when rationalizing expectations we lay on the results and realizing limitations of the approaches beforehand. The actual models are discussed in more detail in subsequent chapters.

3.1 Measures

Say we know a thing or two about calculating statistics on a single set of data already, *i.e.* we skip the definitions of basic descriptive statistics such as median, mean value (μ), standard deviation (SD) *etc.* and focus on the statistics used to describe and evaluate models.

Now, for the sake of concreteness, say we have have a set of data y_t, $t = 1, \ldots, T$ and a set of estimates of those data, denoted $\hat{y}_t$. Where these

estimates come from is not important at this moment. We do, however, want to measure how good these estimates are. Here are definitions of commonly used statistics;

First of all, (*residual*) *error* in the single estimate at t is a difference

$$e_t \equiv \hat{y}_t - y_t \tag{3.1}$$

which serves as a basis for practically all other statistics. *Mean error* (ME), *mean squared error* (MSE), *mean absolute error* (MAE) are straightforwardly derived from (3.1). Often times, *root mean squared error* ($RMSE$) is used instead of MSE, as it is expressed in the same units as the original data, *i.e.*

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T} e_t^2} = \sqrt{MSE} \tag{3.2}$$

Further on, *mean percentage error* ($M\%E$) and *mean absolute percentage error* ($MA\%E$) are defined as

$$M\%E = \frac{1}{T}\sum_{t=1}^{T} \frac{e_t}{y_t}[\times 100\,\%] \tag{3.3}$$

and

$$MA\%E = \frac{1}{T}\sum_{t=1}^{T} \frac{|e_t|}{y_t}[\times 100\,\%] \tag{3.4}$$

where the term $[\times 100\,\%]$ could be considered optional, reserved for situations when the values expressed in percents seem to be more adequate than common ratio.

Very important is the *sum of squared residuals* which is defined

$$SSR = \sum_{t=1}^{T} e_t^2 = T \times MSE \tag{3.5}$$

as it is a crucial statistic of interest for the *least squares* method (see forth). The closer the estimates $\hat{y}_t$ are to the original data y_t, the closer to zero is the SSR statistic.

Another frequently used statistic is the *coefficient of determination* commonly known as R–squared and denoted R^2. There are several ways to define it but, let's stick with the definition

$$R^2 = 1 - \frac{\sum (\hat{y}_i - y_i)^2}{\sum (y_i - \bar{y})^2} \tag{3.6}$$

which is often adjusted for the number of parameters p in the model generating estimates $\hat{y}_t$ as

$$adj.R^2 = 1 - \frac{(T-1)}{(T-1-p)} \frac{\sum (\hat{y}_i - y_i)^2}{\sum (y_i - \bar{y})^2} \tag{3.7}$$

and called *adjusted R–squared*. The adjusted version penalizes the model for over-parametrization. However, for "small enough" p and "big enough" T, the (3.6) and (3.7) are close to being equal.

The last important standard measure we define is so–called *Theil's U*, first introduced in *Theil (1966)*. Let's use equivalent definition

$$U \equiv \left[\sum_{t=1}^{T-1} \left(\frac{e_{t+1}}{y_t} \right)^2 \right]^{\frac{1}{2}} \left[\sum_{t=1}^{T-1} \left(\frac{\Delta y_{t+1}}{y_t} \right)^2 \right]^{-\frac{1}{2}} \tag{3.8}$$

While R–squared is used mainly for assessing how well the model fits the data (in the in–sample period), Theil's U is used almost exclusively as a forecast performance indicator – lower U (on the same set of data) represents better forecasting performance.. Since these default measures for models comparison play a crucial role in the evaluation of models later in this work, they are explained in detail in Appendix A.

Sometimes (such as in Chapter 7) a need to decide not only whether one model is better than the other but, whether it's forecasts are statistically *significantly* better, arises. We satiate this need with rather standard Diebold–Mariano test, defined in Appendix A as well.

3.2 Least Squares

Whenever we need to evaluate performance of a statistical model, we employ some kind of measure for the model. One such measure, defined by formula (3.5), is of greater importance than the others.

Many mathematical optimization problems are formulated as a minimization or maximization problem of some sort. And many statistical methods rely on formulating the model estimation problem as minimizing the SSR measure on a set of data, conditional on a predetermined functional form of the model. Hence the *least squares* method.

Although other measures are generally possible, the least squares method is well worked-out. The main reason is that the squared measure is analytically much easier to handle than, for example, an absolute value as a measure. Minimizing the sum of squares is the most common approach.

There are, of course, many other useful measures, depending on the problem at hand. Sometimes the proper measure of interest can be *e.g.* mean error on the set, or a *one way* errors only – say, mean or maximum negative deviation. However, least squares remain the basic reference method in model estimation.

3.3 Regression Analysis

Regression analysis is a basic concept for estimation of parameters of functions describing relationship among random variables. For parametric regression methods, it's essential that the functional form of relationship is known to researcher.[1] Although regression analysis encompasses large num-

[1] In non–parametric regression, the structure of the model is "guessed" from the data along the way.

ber of methods, these can be divided basically into two groups: linear and non–linear regression.

Linear regression applies whenever the relationship among variables can be expressed as a linear combination, where parameters to estimate are in position of scalars. That is, for univariate dependent variable Y and set of k explanatory variables X_i and scalar parameters β_i,

$$Y_j = \beta_0 + \sum_{i=1}^{k} \beta_i X_{i,j} + \epsilon_j \quad . \tag{3.9}$$

Many functional relationships can be written in form (3.9) through various transformations. Whenever the linear form is not feasible and the parameters of the function in question are non–linear, then non–linear regression methods come into place. The idea is always to fit the data through estimated parameters as close to the predetermined function as possible.

Linear regression is closely connected to the least squares method, as there are number of least-squared-based estimators of vector β. The main advantage of least squares estimators is their closed form, *i.e.* analytical solution. There is, however, a list of assumptions, such as linearly independent explanatory variables (no multicolinearity), white noise error term ϵ (uncorrelated, zero mean, constant variance). Some of these assumptions can be relaxed under certain conditions for one method or another. Despite its limitations, linear regression is a handy "first approximation" tool. For sufficiently large datasets the trend is to rely on their statistical approximation power while being aware of possible bias and using robust statistics for errors.

One exhaustive list of assumptions for specific methods, as well as examples of their alternations can be found in *Baltagi (2008)* or *Greene (2003)*, both complete reference books for the linear regression methods.

Whenever the observed data are naturally ordered in time, we talk of time series data. And whenever the time component is reflected in the model, *e.g.* by time shifts of a particular explanatory variable, we say the model is "time variant".

A special case of time analysis of univariate time variant models is hard to overestimate in its significance. An example of linear time variant model based on (3.9), where the random variables are in fact time-shifted values of the dependent variable:

$$y_t = \beta_0 + \sum_{i=1}^{k} \beta_i X_{t-i,t} + \epsilon_t \quad . \tag{3.10}$$

Strong (auto)correlation is a typical feature of time series data, therefore many methods commonly used with factor models need to be either adapted or abandoned. Analysis of time series also other – conceptual – problems, one of which we already encountered when introducing the realized volatility; Obviously, the value of RV_I in (2.5) depends on the specific division of I.

Let us point out several examples and connect them to RV. The one just mentioned inherently originates in discretization of what is assumed a continuous process. The choice od sampling the data itself creates uncertainty whether it captures the character of underlying process properly. Aliasing of sinusoids with different frequencies is probably the best known example.

Even the choice of daily realized volatility can be ambiguous, since the trading activity shifts around the globe accordingly to time zones and many assets are traded 24/7 almost whole week. Any choice of sampling the data is therefore arbitrary, with a possible unclear impact.

Another problem concerning trading time are holidays. There are many different countries with many holidays, during which the trading in the whole region may be affected or stopped. Other effects aside, the result is that subsequent observations within a time series are not uniformly equidistant.

Acquiring consistent data over long periods of time can turn out to be difficult; Over lengthy time horizons, the methodology of time series recording may have changed without explicit notification. In general, behavior of time series can evolve in time, placing demands on the methods, models, and of course researchers.

Time series analysis is a relatively separate, self-sufficient discipline. Unlike with factor models, where insight into the modeled problem is essential for assessing the right factors into functional relations, in time series analysis, the statistical experience of a researcher plays much greater role.

Classical approaches to time series analysis (before ARCH and GARCH models were formulated) are well covered in *Cipra (1986)*.

Chapter 4

Box–Jenkins Class of Models

During 1970, George Box and Gwilym Jenkins published the book *Time series analysis: Forecasting and control*[1], which presented a complete framework for analyzing time series. Not only correlation and autocorrelation in time series was not considered a problem, it was in fact the very core of the analysis. In this respect, their approach was revolutionary at the time.

From their analysis, the family of models arose: autoregressive process (AR), moving average process (MA), and autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) processes.

Based on the shape of a time series' (partial) autocorrelation function(s), the authors present methodology for model identification, parameter estimation and model evaluation. As *Cipra (1986)* stated already several decades ago, their methodology converges to fully automated process in the end.

The theoretical center of gravity of their approach lies in statistical properties of a general *linear process*, a stochastic process of the form

$$y_t = \epsilon_t + \beta_1 \epsilon_{t-1} + \beta_2 \epsilon_{t-2} + \dots , \tag{4.1}$$

[1] See *Box & Jenkins (1970)* in References.

with β_i being parameters and ϵ_t a white noise. If a linear stochastic process exhibits time–invariant behavior in terms of its probability distribution, it is called *stationary*, or weakly stationary, if it has constant mean, variance and $cov(y_t, y_s) = cov(y_{t+h}, y_{s+h})$.[2] Moreover, if a linear process can be rewritten as

$$y_t = \epsilon_t + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \dots , \tag{4.2}$$

then such process is called *invertible*.

Box and Jenkins focused on special cases of linear processes – those for which α_i and β_i equal to zero, except a finite number of indexes i. Parameters of such models are to preserve stationarity and invertibility of the process.

The basic model in the methodology is an autoregressive (or AR) process of order p. AR(p) process is defined

$$y_t = \epsilon_t + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \ldots + \varphi_p y_{t-p} \quad . \tag{4.3}$$

Such process is invertible by definition, from (4.2). Verifying its stationarity is not that straightforward. For example, for the elementary AR(1) process, the stationarity is ensured for $|\varphi_1| < 1$, meaning that the AR(1) process has the (4.1) form, specifically

$$y_t = \epsilon_t + \varphi_1 \epsilon_{t-1} + \varphi_2^2 \epsilon_{t-2} + \dots . \tag{4.4}$$

Another type of special linear process is a *moving average* process of order q. The MA(q) process is of form

$$y_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q} \, . \tag{4.5}$$

The question of stationarity and invertibility turns over; The process is stationary for any thetas, with zero mean, and with variance

$$var(y) = (1 + \theta_1^2 + \ldots + \theta_q^2) \, var(\epsilon) \, . \tag{4.6}$$

[2] See *Cipra (1986)*.

Verifying whether the process is invertible is more complicated. However, *e.g.* MA(1) process is invertible once again whenever $|\theta_1| < 1$.

Note that while AR processes can be easily estimated analytically (typically with least squares linear regression), the MA process requires some non–linear estimation method, usually iterative. This is due to unobservable nature of ϵ components in (4.5).

4.1 Combined AR(I)MA Models

When AR(p) and MA(q) processes are combined, one gets ARMA(p, q) process defined simply

$$y_t = \epsilon_t + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \,. \tag{4.7}$$

Mean value of such process is zero, while its stationarity and invertibility is equivalent with stationarity of AR(p) and invertibility of MA(q) alone.

Many time series are not stationary. However, often times, they can be easily transformed into stationary series and treated with Box–Jenkins methodology. There exist many generalizations and variations to ARMA models. A fundamental generalization is ARIMA – an autoregressive *integrated* moving average model.

ARIMA model is essentially an ARMA model applied to a differenced series, rather than original series. The degree of differencing is denoted d, *i.e.* the model is ARIMA(p, d, q). We define differences recursively:

$$\begin{aligned} \Delta^1 y_t &\equiv \Delta y_t = y_t - y_{t-1} \\ \Delta^d y_t &\equiv \Delta^{d-1} y_t - \Delta^{d-1} y_{t-1} \end{aligned} \tag{4.8}$$

so the ARIMA(p, d, q) is defined

$$\Delta^d y_t = \epsilon_t + \sum_{i=1}^{p} \varphi_i \Delta^d y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \,. \tag{4.9}$$

There are also other generalizations to ARMA model; For example, ARMAX includes exogenous variables into (4.7) formula[3], autoregressive models with lagged independent variables, or autoregressive moving average models built on discrete probability distributions – DARMA.

Dealing with actual time series, parameters p, d and q are of course not know in advance. Therefore model identification is a substantial part of Box–Jenkins methodology, based mainly on the analysis of autocorrelation and partial autocorrelation functions (or rather their estimates). Once again, see *Cipra (1986)* for a well-written comprehensive summary of the methods along with an exhaustive list of references.

4.2 HAR–RV

A variant of autoregressive model in context of realized volatility was proposed by *Corsi (2009)*. In his setting, an "additive cascade" of realized volatilities measures spun over three different time horizons is constructed – utilizing daily RV and its weekly and monthly averages – and the linear combination is then named *Heterogenous Autoregressive Model of Realized Volatility* or HAR-RV.

Daily realized volatility follows standard definition

$$RV_d = \sqrt{\sum_{i=1}^{n} r_i^2}\,, \tag{4.10}$$

where n depends on intraday sampling frequency. Weekly and monthly averages are simple moving averages of RV_d from past 5 and 22 days, respectively. These numbers reflect the number of usual business days within

[3]In fact, we will use ARMAX models later on, still calling it AR(I)MA. Notation will be explained properly where relevant.

a week/month. To distinguish this particular choice of spans from other potential choices, we shall denote the model also HAR(1,5,22). Definition of averages follows:

$$RV_{d,\overline{w}} = \frac{1}{5}(RV_d + RV_{d-1} + \ldots + RV_{d-4}) \tag{4.11}$$

for weekly average of daily realized volatility and

$$RV_{d,\overline{m}} = \frac{1}{22}(RV_d + RV_{d-1} + \ldots + RV_{d-21}) \tag{4.12}$$

for monthly average of daily realized volatility.

Note that the averages of aggregated daily volatilities incorporate the most recent daily realized volatility in Corsi's setting. The model itself is then of form

$$RV_d = c + \beta_1 RV_{d-1} + \beta_2 RV_{d-1,\overline{w}} + \beta_3 RV_{d-1,\overline{m}} + \omega_d\,, \tag{4.13}$$

where ω_d comprises of latent volatility measurement as well as error term – see *Corsi (2009)* for detailed description. The term ω_d is not assumed to be zero-mean necessarily.

The HAR-RV model is supposed to exhibit solid performance in modeling and forecasting daily RV. It captures some empirically observed features of returns time series such as fat tails and long memory of volatility (i.e. low but persistent autocorrelation of variances).

It is easy to see that the HAR(1,5,22) model is a special case of linear lagged model, similar to AR(22), with some restrictions put on individual parameters. That is, the model (4.13) is equivalent to

$$RV_d = c + \alpha RV_{d-1} + \gamma \sum_{i=2}^{5} RV_{d-i} + \psi \sum_{i=6}^{22} RV_{d-i} + \omega_d\,, \tag{4.14}$$

with a constant term c and three other parameters α, γ and ψ to estimate. In light of (4.14), we can say that what the HAR model essentially does

is, by restricting AR(22) model parameters, it artificially keeps the AR(22) model to fit the given data at its full potential (possibly preventing if from overfitting) for the sake of sustaining its forecast ability.

4.3 HARD Extension

We introduce a simple, yet – how we will see later during the tests on the real data – effective enhancement to the HAR–RV model.

Time series data sets of RV are usually accompanied by a calendar date marking the day that the RV entry relates to. The problem with equidistance of entries, or rather the lack of it, has already been mentioned in Section 3.3. The usual way of dealing with such data is to incorporate dummy variables for each day of the week. We shall take a different course of action.

First, let's define a new variable *days passed*, as a number of days that have passed since the last entry for RV_d[4], and denote it D_d. Then, add it to the HAR model and call the new model HARD. It's as simple as that. Hence, we specify the HARD extension of HAR model as follows:

$$RV_d = c + \alpha D_d + \beta_1 RV_{d-1} + \beta_2 RV_{d-1,\overline{w}} + \beta_3 RV_{d-1,\overline{m}} + \omega_d \,, \tag{4.15}$$

i.e. in a way very reminiscent of the original (4.13) formula. The idea is to give the model a chance to accommodate the time gaps present in financial markets trading.

[4]So, for example, if one entry of RV is from Monday, and the following one is from Tuesday, then the value of D on Tuesday will be 1. Or, if one entry comes from Friday and the following one from the subsequent Monday, then the value of D on Monday will be 3, and so on.

Chapter 5

Neural Networks

The last class of models covered in this work originates in neuroscience. The urge to understand how brain works has led to emergence of a whole new framework for mathematical modeling known as *neural networks*. Due to its origin, the neural networks terminology is very biological. We shall, however, abstract from actual physiological processes of brain and use mathematical definitions instead.

The concept itself is, in principle, no different from many other approximation or estimation methods; Having set of explanatory (input) variables and dependent variables we attempt to find certain transformations of inputs such that the result of the process is close enough to the dependent variable(s). Neural network is, put simply, a specific type of data–transforming structure.

5.1 The Architecture

The base building block of any neural network is *neuron*. Each neuron is a functional unit with capacity to receive an input signal (or multiple signals),

then transform it and send the transformed signal forth.

Even explanatory variables can be referred to as *input neurons* which send the information forth without transformation. Neuron that does not send its input signal forth after transformation (the dead–end neuron) will be referred to as *output neuron*. Thus, once a neural network is established, we can communicate with it by filling its input neurons and then reading its response from the output neuron(s).

Connections between neurons determining the path of a signal are called *synapses*. They are neuron-to-neuron one–way data pipes with assigned weights – coefficients representing the respective intensities with which a neuron receives signals from another neurons. A neuron thus receives its input signal as a linear combination of output signals from preceding connected neurons. Not necessarily all neurons are connected.[1] Establishing concrete synapses and their weights is a substantial part of neural network estimation.

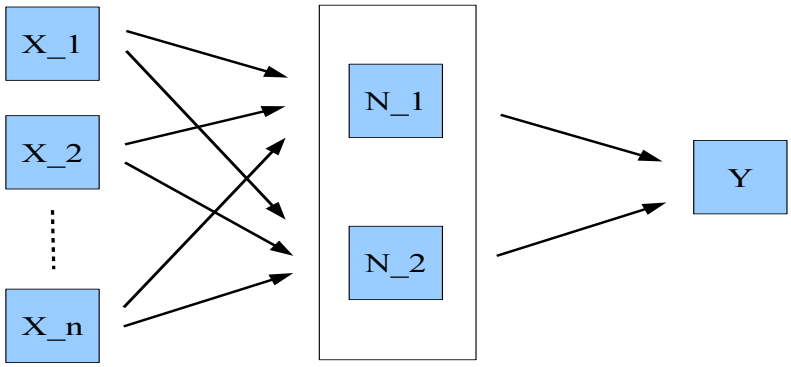
A set of neurons that transform signals at the same time, yet independently on one another within that set, is called *layer* or *hidden layer*. If we look at a neural network as if it was a circuit, layers are serially connected blocks of neurons in which neurons parallelly (independently on one another) process input signals coming from previous layer before sending them forth to the next layer. Clearly, any neural network has *at least* two layers; an *input layer*, constituted by a set of input neurons, and an *output layer*, in simple applications usually represented by one output neuron (although multiple output neurons are, of course, possible).

[1] Technically, two neurons that are not connected are equivalent to those same neurons being connected by a *zero*–weighted synapse.

5.2 Feedforward networks

Feedforward network (ffw NN) is the basic NN structure. Figure 5.1, modified from *McNellis (2005)*, depicts a simple ffw network with n inputs, one hidden layer containing two neurons, and a single output neuron.

Figure 5.1: Simple feed forward network



Arrows between neurons represent connections and the direction that the signal propagates forward. Note that the neurons within one layer are not interconnected, therefore the signals entering each neuron within one layer can be processed parallelly/independently.

Estimating such network means assigning weights to "arrows", and choosing the proper function type and its parameters for each neuron (excluding input neurons). For the depicted network, that means estimating $(2n+2)$ parameters assigned to connections and p parameters for functions of choice for the two hidden neurons and the output neuron. It is clear that the number of parameters increases substantially with every new neuron.

Let us use the network from Figure 5.1 to demonstrate mathematical representation of that network. Assuming that hidden neurons N_1 and N_2

nest functions f_1 and f_2 respectively, connections between input layer and the hidden layer are marked α_i for $i = 1, ..., n$ and connections between the hidden layer and the output neuron, which nests activation function f_O, we can define the particular network by set of equations

$$\begin{aligned} y_t &= f_O(\textstyle\sum_{i=1}^{2} \beta_i N_i) \\ N_{i,t} &= f_i(\textstyle\sum_{j=1}^{n} \alpha_j x_{j,t}), \quad i = 1, 2\,. \end{aligned} \tag{5.1}$$

A type of connection that skips some layers along the way is called *jump* connection. Imagine that in our network from Figure 5.1, we add one more set of connections γ, arrows leading from each input neuron x_i directly into the output neuron y. Then the new network would be defined

$$\begin{aligned} y_t &= f_O(\textstyle\sum_{i=1}^{2} \beta_i N_i + \sum_{k=1}^{n} \gamma_k x_{k,t}) \\ N_{i,t} &= f_i(\textstyle\sum_{j=1}^{n} \alpha_j x_{j,t}), \quad i = 1, 2\,. \end{aligned} \tag{5.2}$$

As more neurons, hidden layers, connections and jump connections are added to the network, the transparency of its mathematical representation quickly dissipates, at least for humans.

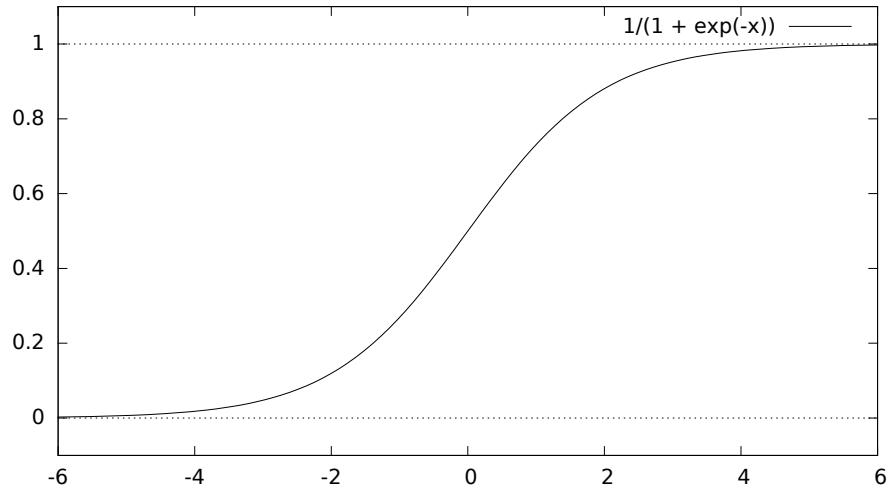
5.3 Signal Processing

An active neuron always receives a linear combination of its input signals. The resulting summation of weighted signals is then processed by neuron's activation function. In a very general specification of neural networks, there are no restrictions to what types of functions can be used as activation function. Yet there are several types of functions, used more commonly than others; Basic class of functions used are logsigmoid functions, defined

$$L(x) = \frac{1}{1 + e^{-x}} \tag{5.3}$$

The logsigmoid function is sometimes called *a squasher function*, as any value entering the function is "squashed" into interval $(0; 1)$ – see Figure 5.2. Logsigmoid function can be adapted, parametrized to squash into different interval, "shift" the x–axis, scale the input value or the output value.

Figure 5.2: Logsigmoid function



Another function with similar shape but a different squashing interval is a hyperbolic tangent function, defined by formula

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{5.4}$$

and depicted in Figure 5.3, from which the similarity with logsigmoid function is obvious. Again, hyperbolic tangent can be further parametrized to satisfy the needs of a particular problem. Both these functions can simulate a sensitivity response of a given neuron to an input signal. If an input signal is too weak, neuron doesn't recognize it. If the signal is in the "correct" range, neuron recognizes it with intensity parametrically predefined by these functions, and if a signal is too strong, the neuron reaches only its limited maximum response. Furthermore, specifically parametrized logsigmoid function can approximate cumulative Gaussian function (CGF), which

is a welcome feature.[2] A plot of a different kind of squasher activation function, hyperbolic secant, is shown by Figure 5.4 below. Another commonly used activation function is a simple identity. See *McNelis (2005)* for more examples of typical activation functions.

Figure 5.3: Hyperbolic tangent

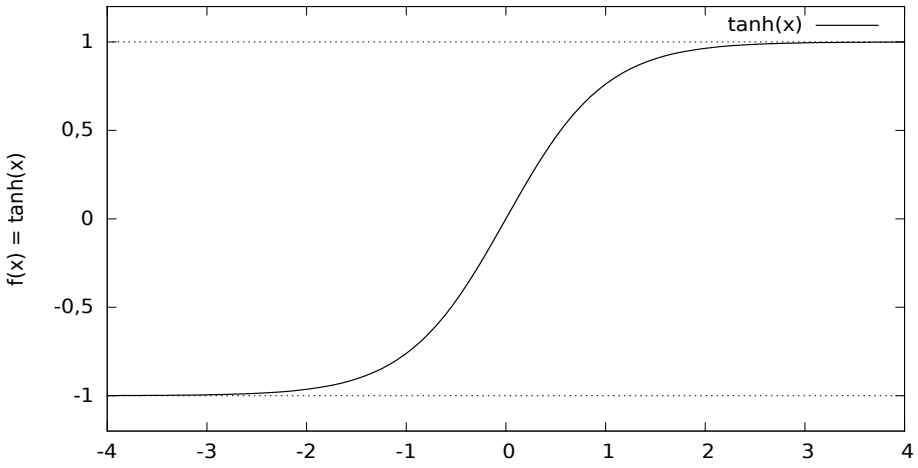
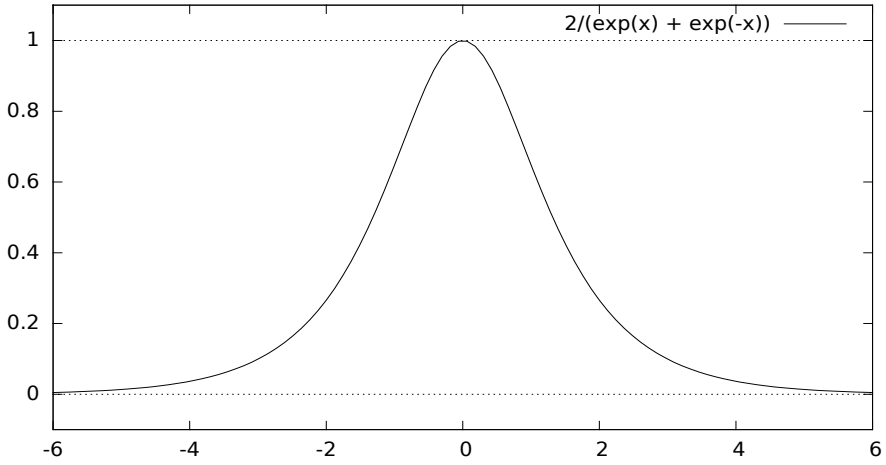


Figure 5.4: Hyperbolic secant

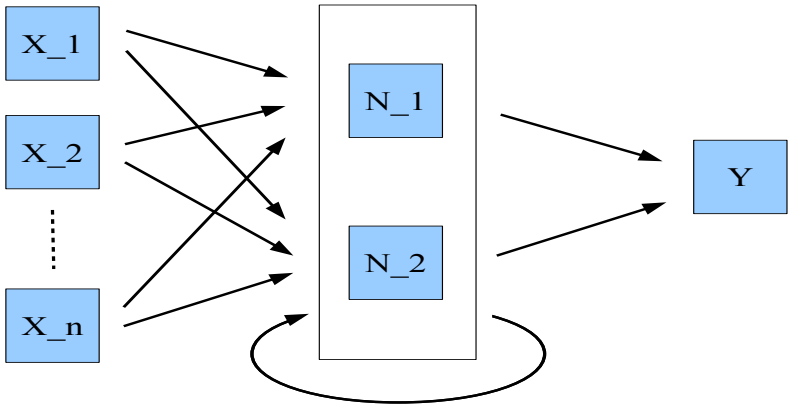


[2]Of course, CGF can also be used as activation function

5.4 Recurrent networks

Recurrent structures allow to build a kind of memory or inertia to neurons. There are several ways how to do it. First, let us schematically add a recurrent link to our previous example network, as shown on Figure 5.5. New connection transmits the signal from neuron N_2 back into itself.

Figure 5.5: Simple recurrent network



In the next step, the N_2 neuron receives weighted signals from all the inputs, including a weighted value of its *previous* activation. Depending on the weight, the neuron maintains some kind of inertia, deviated by new values of inputs. These neurons are sometimes called decay neurons.

Memory can be built into the network by introducing another type of neurons – delay neurons – which serve as simple signal repeaters. A delay neuron sends previous linear combination of inputs of its original neuron. Notice that this is very different from what a decay neuron does!

Networks incorporating delay neurons are also known as *Elman networks*. Recurrent networks share some similarities with moving average model described in Chapter 4. They are also limited to time series data.

5.5 Summary

Neural networks are yet another approach in building non-linear models, best described as structures for data transformations. Although they use simple analytical functions – be it squasher or other activation functions – for these transformations, analytical form of non-trivial neural network is impractical, too often too complex to work with (depending on the complexity of a network itself).

In principle, estimating a neural network is once again a minimization problem. Due to its complexity and non-linearity, estimation of a network is substantially time-demanding[3], and then there is always the risk of converging to local minimum rather than global minimum of the particular setting.

Mainly for the complexity reasons, neural networks have acquired an unkind reputation of "black boxes", incapable of providing a real insight into the analyzed data. It is probably fair to say that this criticism, while strongly simplified, is not *entirely* unjustified; Nevertheless, neural networks *can* be analyzed inside out and, where the application demands it and complexity allows it, they are being analyzed, although answering the question "what exactly happens inside a particular complex neural network" can be considerably difficult, time-costly and discouraging in general.

[3]For instance, while the actual simple AR models used later in estimation were estimated literally in a blink of an eye, some of the presented neural networks took up to several hours of iterations to estimate on the same – admittedly mediocre – hardware.

Chapter 6

Data

Specific data used in the forthcoming estimation of models and subsequent evaluation of their forecast performance is described here along with their statistical properties.

We focus our attention on forex market, namely these three currency pairs: EUR/USD, GBP/USD and USD/CHF, *i.e.* three of so-called "major pairs" or simply "the majors".[1] These are the most traded and liquid pairs in the world markets, which renders them more than adequate for realized variance studies.

Time series we use come from the database of financial data provider *Tick Data* that filters the data using its proprietary algorithms before exporting it to end user.[2] The resulting realized variance is calculated by making use of five-minute intervals price quotes (or five-minute realized returns) throughout the day as in formula (2.5). Each daily entry is accompanied by corresponding date stamp. This is the basic simple structure of each time series.

The realized volatility itself is then obtained as a square root of calculated

[1] The fourth major pair of the pack would be USD/JPY.

[2] See www.tickdata.com for the white paper on data filtering.

realized variance. The average daily volatilities over last trading week and trading month are calculated for the sake of HAR models. Added is variable "days_passed" which – for every entry in time series – represents the number of days passed since the last entry.

Each of these three time series is then divided into two periods; *in-sample* model estimation period and *out-of-sample* forecasting period. The length of the forecasting period is set to represent a one whole trading year preceding the end of the series, which constitutes for about 12% of the full data available. A lucid table summing up basic statistical properties of the data is present for each time series. Also present are graphical plots of daily realized volatilities and their differenced versions (mainly for visual inspection of the stationarity assumed in Box-Jenkins type of models).

Due to an anomaly in USD/CHF series, this particular time series deserves extra attention. On 6th of September 2011, there has been a precipitous price movement in the series – clearly exogenously driven and therefore inherently unpredictable within the time series forecast framework in consideration – resulting in an unprecedented realized variance and volatility values on that very day. The specific value of realized volatility on that day was nearly eight times the median realized volatility of the series, well beyond limits even if accounted for the standard deviation.

Since the anomaly falls into the out-of-sample period, it has no effect on the estimation process. It could, however, substantially corrupt the forecast statistics of estimated models, so we exclude that anomaly from the data set. To manifest the point in doing so, we present *two* statistic tables for USD/CHF showing the adverse effect of that one single entry to the series. Model forecasts later on are – of course – performed without this entry.

6.1 EUR/USD

The dotted lines in Figures 6.1 and 6.2 of EUR/USD realized volatility time series and differenced series, divide the in-sample and out-of-sample periods. Note that the differenced series seems to be stationary "at a first glance". Such observation needs to be yet vindicated by statistical tests.

Figure 6.1: EUR/USD RV_d

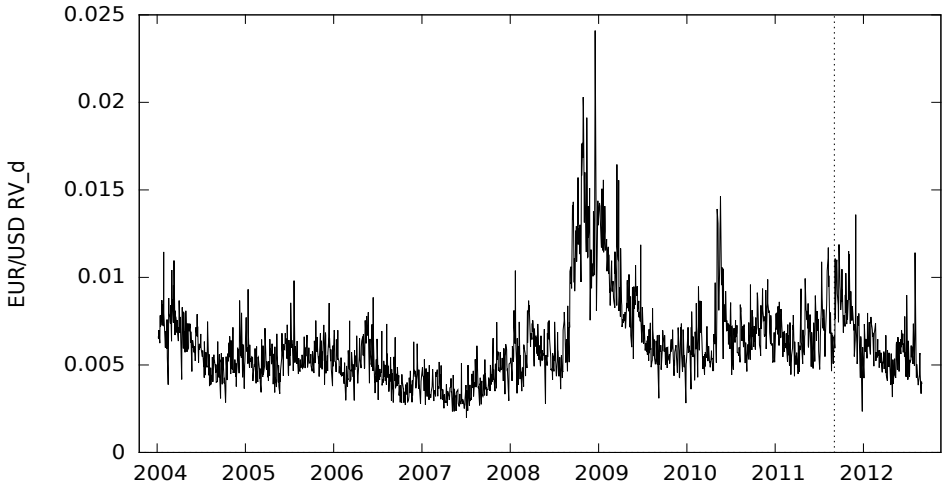
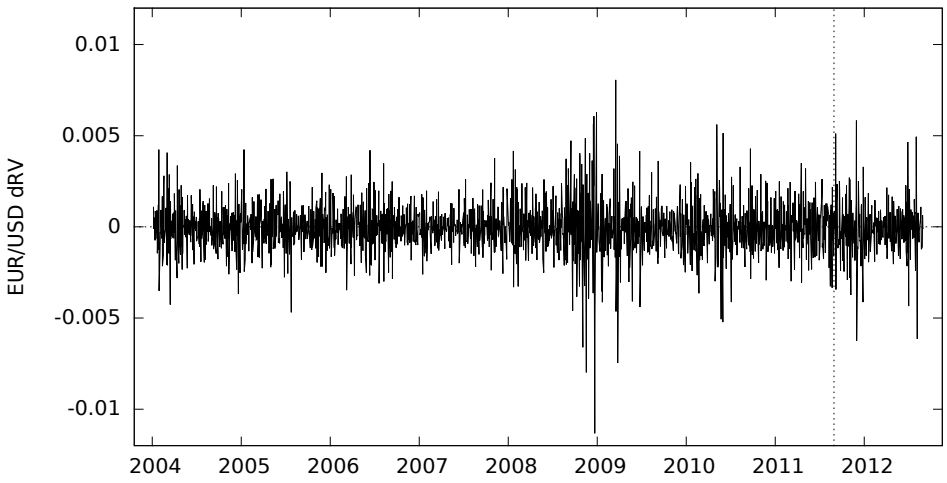


Figure 6.2: EUR/USD $\triangle RV_d$



The null hypothesis of standard KPSS test[3] is that the time series in question is stationary, alternative being unit root. We use `Gretl` to conduct KPSS test on the in–sample; Using 22 lag span, the critical value at 0.10 level of significance is 0.347, KPSS test statistic value 0.0201, *i.e.* we cannot reject stationarity of series at any reasonable significance level (0.10, 0.05 and 0.01).

Table 6.1: EUR/USD RV_d Summary Statistics

Full dataset using observations 2004/01/06–2012/08/30 (1771 valid observations for RV_d)			
Mean	0.0062200	Standard deviation	0.0024080
Median	0.0057458	C.V.	0.38714
Minimum	0.0019682	Skewness	1.8475
Maximum	0.024105	Ex. kurtosis	5.8164
Estimation sample using observations 2004/01/06–2011/08/31 (1564 valid observations for RV_d ... 88.31 % of the data)			
Mean	0.0061991	Standard deviation	0.0024734
Median	0.0057158	C.V.	0.39899
Minimum	0.0019682	Skewness	1.8882
Maximum	0.024105	Ex. kurtosis	5.8492
Forecast sample using observations 2011/09/01–2012/08/30 (207 valid observations for RV_d ... 11.69 % of the data)			
Mean	0.0063777	Standard deviation	0.0018375
Median	0.0060185	C.V.	0.28811
Minimum	0.0023446	Skewness	0.99676
Maximum	0.013583	Ex. kurtosis	1.2170

[3]See *Kwiatkowski et al. (1992).*

6.2 GBP/USD

Figures 6.3 and 6.4 depict the original and the differenced GBP/USD series.

Figure 6.3: GBP/USD RV_d

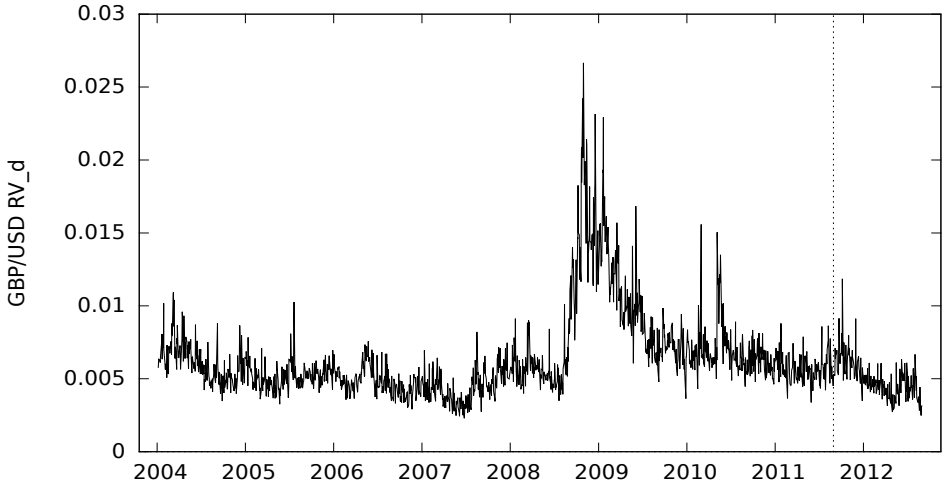
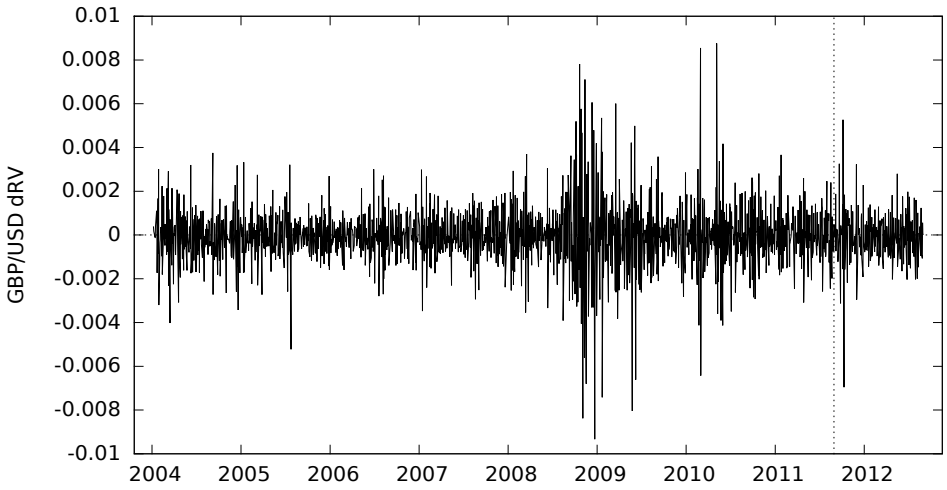


Figure 6.4: GBP/USD $\triangle RV_d$



Stationarity null is tested with the same parameters as before, critical value at 0.10 level of significance being 0.347, KPSS test statistic 0.0284 for the differenced series. Again, we cannot reject the stationarity of GBP/USD

series at any reasonable significance level. We consider the series stationary as well.

Table 6.2: GBP/USD RV_d Summary Statistics

Full dataset using observations 2004/01/06–2012/08/30 (1740 valid observations for RV_d)			
Mean	0.0062883	Standard deviation	0.0028270
Median	0.0055984	C.V.	0.44957
Minimum	0.0022950	Skewness	2.5571
Maximum	0.0266550	Ex. kurtosis	8.9350
Estimation sample using observations 2004/01/06–2011/08/31 (1533 valid observations for RV_d ... 88.10 % of the data)			
Mean	0.0064599	Standard deviation	0.0029310
Median	0.0056864	C.V.	0.45373
Minimum	0.0022950	Skewness	2.4615
Maximum	0.026655	Ex. kurtosis	8.0829
Forecast sample using observations 2011/09/01–2012/08/30 (207 valid observations for RV_d ... 11.90 % of the data)			
Mean	0.0050178	Standard deviation	0.0013173
Median	0.0048311	C.V.	0.26252
Minimum	0.0024848	Skewness	1.1066
Maximum	0.011850	Ex. kurtosis	3.0885

6.3 USD/CHF

Final set of figures and tables in this chapter; Notice the distinctive one-time jump in the RV_d plot right after the dividing line. In-sample and out-of-sample are divided a week before the jump occurs. Single jump in RV_d translates into two extreme values in the differenced series.

Figure 6.5: USD/CHF RV_d

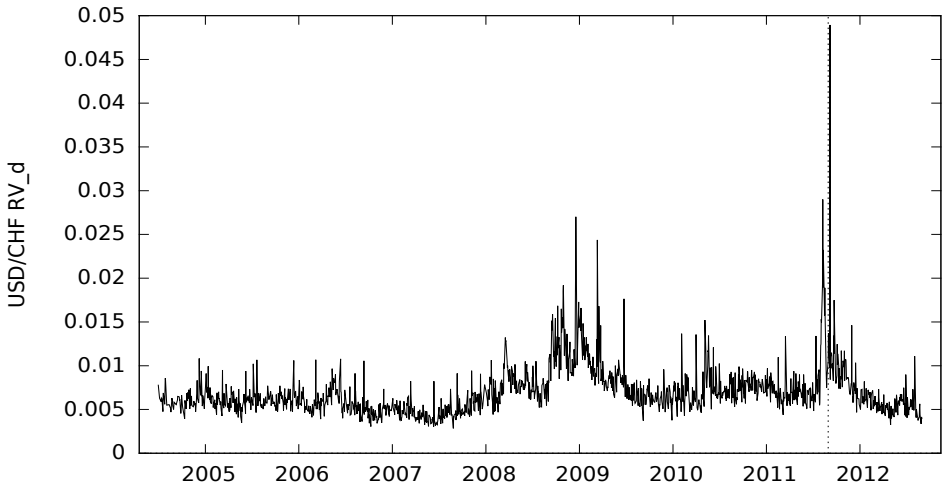
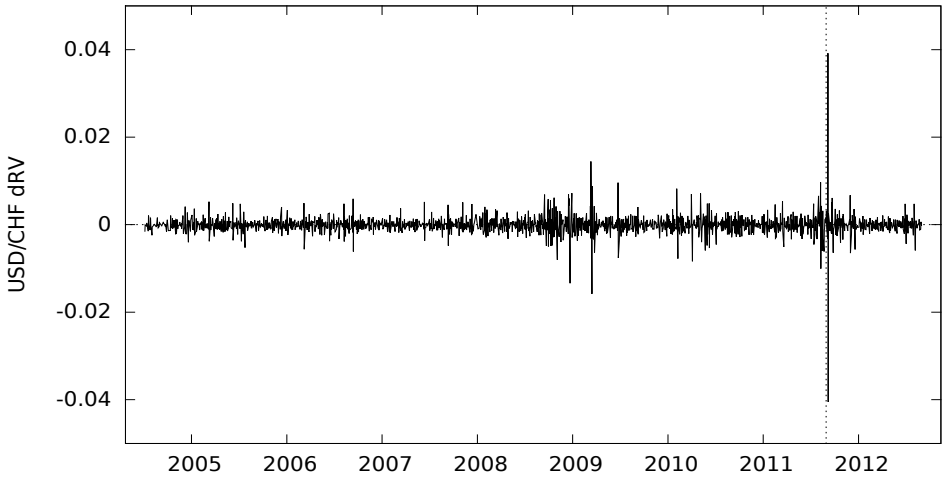


Figure 6.6: USD/CHF $\triangle RV_d$



To show the impact of that single jump on the data set properties, we present two tables – one including the extreme value (Table 6.3) and one without it (Table 6.4).

Table 6.3: USD/CHF RV_d Summary statistics

Full dataset using observations 2004/07/01–2012/08/30 (1608 valid observations for RV_d)			
Mean	0.0070024	Standard deviation	0.0028093
Median	0.0064231	C.V.	0.40119
Minimum	0.0028415	Skewness	3.9125
Maximum	0.0489440	Ex. kurtosis	37.531
Estimation sample using observations 2004/07/01–2011/08/31 (1402 valid observations for RV_d ... 87.2 % of data)			
Mean	0.0070189	Standard deviation	0.0026510
Median	0.0064597	C.V.	0.37769
Minimum	0.0028415	Skewness	2.4036
Maximum	0.0290090	Ex. kurtosis	10.434
Forecast sample using observations 2011/09/01–2012/08/30 (206 valid observations for RV_d ... 12.8 % of data)			
Mean	0.0068899	Standard deviation	0.0037180
Median	0.0061561	C.V.	0.53964
Minimum	0.0032425	Skewness	7.3363
Maximum	0.0489440	Ex. kurtosis	77.949

The jump occurs in the out-of-sample period, hence the central part of both Tables remains unchanged. In the other parts do notice the significant change in the statistical properties of the data, especially the adverse effect the jump

has on standard deviation, coefficient of variation, skewness and kurtosis of data. We use it as a justification for dropping the 2011/IX/6 RV_d entry in order to increase the upcoming forecast statistics information value.

Table 6.4: USD/CHF RV_d Statistics of the restricted data

Restricted dataset using observations 2004/07/01–2012/08/30 (1607 valid observations for RV_d)			
Mean	0.0069763	Standard deviation	0.0026079
Median	0.0064228	C.V.	0.37382
Minimum	0.0028415	Skewness	2.3329
Maximum	0.0290090	Ex. kurtosis	10.022
Estimation sample using observations 2004/07/01–2011/08/31 (1402 valid observations for RV_d ... 87.2 % of data)			
Mean	0.0070189	Standard deviation	0.0026510
Median	0.0064597	C.V.	0.37769
Minimum	0.0028415	Skewness	2.4036
Maximum	0.0290090	Ex. kurtosis	10.434
Restricted forecast sample; observations 2011/09/01–2012/08/30 (205 valid observations for RV_d ... 12.8 % of data)			
Mean	0.0066848	Standard deviation	0.0022759
Median	0.0061552	C.V.	0.34047
Minimum	0.0032425	Skewness	1.4118
Maximum	0.0174980	Ex. kurtosis	2.5944

The KPSS test statistic for the USD/CHF series yields 0.045. Remote from the 0.347 critical value at 0.10 confidence level, we do not reject the stationarity null hypothesis at any reasonable level.

Chapter 7

HARD Performance

Prior to testing neural networks' performance, let us give those networks a solid benchmark to measure up to. We already introduced a simple enhancement to the HAR model in Section 4.3. The new variable D representing "days passed" has been mentioned already too. So let's see how it contributes to efficiency when added to HAR model.

First, we re–index the (4.15) formula of the HARD extension for forecasting as follows:

$$RV_{t+1} = const. + \alpha D_{t+1} + \beta_d RV_t + \beta_w RV_{t,\overline{w}} + \beta_m RV_{t,\overline{m}} + \epsilon_{t+1}\,, \qquad (7.1)$$

i.e. shift all the variables in the (4.15) one time step forward.

The following Tables 7.1, 7.2 and 7.3 compare HARD version with the standard HAR by Corsi in detail for all three time series. The last table of this chapter, Table 7.4, summarizes the Diebold-Mariano test statistics used to assess the predictive ability of the model (detailed description follows later in corresponding section).

Notice the similarity in RV coefficients not only within the single time series but, across all time series as well. In fact, every RV coefficient of HARD

model lies within the standard deviation boundaries of its HAR counterpart. What changes substantially though, are the values and significance of constants, to adjust for the new variable D which is decidedly significant throughout all three HARD models.

Each comparison is accompanied by a set of standard statistics for estimated models and their forecast performance. It is worth emphasizing that – except a slight shift in mean error when forecasting – the HARD variant performs better than the standard HAR model, literally in *every* measurable aspect, both in terms of explaining the in-sample data and forecasting the out-of-sample data. It is often the case that better fitted model performs worse with forecasts but, not here.

7.1 Diebold–Mariano tests results

The Diebold-Mariano test is discussed in Appendix A. For practical finite-sample statistics calculations, we will use the formula

$$DM_h = \frac{\sqrt{T}\bar{d}}{\left(var(d_t) + 2 \sum_{i=1}^{h-1} cov(d_t, d_{t-i}) \right)^{1/2}} \tag{7.2}$$

in which the denominator contains the mean loss differential variance estimator, specifically, a simplified discrete version of a consistent estimator used in *Diebold-Mariano (1995)*. Moreover, as a bonus, the exact same formula (7.2) is implemented and readily available in `MATLAB` software. Index h represents the horizon or the span through which we calculate autocovariance of the loss differential d. Higher-order covariances are assumed to be zero.

Table 7.4 summarizes the values of DM_h statistics for all three time series over twenty covariance horizons. For each time series we apply two types of

loss functions into loss differential d, squared and absolute functions.

To assess the results of Diebold-Mariano test, we compare the test statistic values to critical values of Student's t-distribution. To keep things simple; as we have out-of-sample sets of 205 and 207 observations, let us use only critical values for t-distribution with 205 degrees of freedom as average for critical values corresponding to 204 and 206 degrees of freedom. Respective (right-tail) critical values are 1.65232 and 2.34468 for 5 % and 1 % levels, null and alternative hypotheses being

$$H_0 \quad ... \quad \text{HARD forecast equally or worse than HAR} \tag{7.3}$$

$$H_A \quad ... \quad \text{HARD is better.} \tag{7.4}$$

Examining Table 7.4, several simple observations can be made. First of all, on the 5 % level, we reject the null hypothesis regardless of the loss function choice for all twenty covariance horizons and conclude that the HARD model is better in forecasts than HAR.

On the 1 % level, the null is always rejected only when square loss function is used in the test. Absolute loss function employed, HARD model is still better in almost all considered covariance horizons for EUR/USD and USD/CHF series, and misses the critical value (although only by a small margin) roughly for half of the covariance horizons with GBP/USD series.

7.2 Summary

Overall, the HARD model performs better than the original HAR model in all aspects – at least for the time series considered. In practical terms, that tells us that the information on the numbers of non-trading days *is* statistically relevant for the realized volatility forecast and therefore should be employed in one way or another whenever present in out dataset.

Table 7.1: EUR/USD HAR vs. HARD comparison

variable	coefficient	S.E.	t-ratio	p-value	
HAR regressors					
const.	0.000254401	0.000135729	1.874	0.0611	*
RV_d	0.310825	0.0457455	6.795	1.54e-11	***
$RV_{d,\overline{w}}$	0.445834	0.0613669	7.265	5.87e-13	***
$RV_{d,\overline{m}}$	0.202727	0.0586549	3.456	0.0006	***
HARD regressors					
const.	0.000711199	0.000129418	5.495	4.55e-08	***
D	-0.000263657	2.31379e-05	-11.40	5.98e-29	***
RV_d	0.341444	0.0447394	7.632	4.01e-14	***
$RV_{d,\overline{w}}$	0.424903	0.0577373	7.359	2.98e-13	***
$RV_{d,\overline{m}}$	0.195361	0.0554783	3.521	0.0004	***

Statistic	HAR	HARD
Sum squared resid.	0.002416	0.002219
S.E. of regression	0.001245	0.001193
R-squared	0.747330	0.767966
R-squared (adjusted)	0.746843	0.767370
Akaike criterion	-16469.34	-16600.51
Mean Error	-3.9014e-05	-4.4368e-05
Root Mean Sq. Err.	0.0012865	0.0012083
Mean Absolute Err	0.00091384	0.00086032
Mean % Error	-4.3205	-3.9557
Mean Abs. % Error	15.114	14.06
Theil's U	0.81687	0.76977

Table 7.2: GBP/USD HAR vs. HARD comparison

variable	coefficient	S.E.	t-ratio	p-value	
HAR regressors					
const.	0.000186618	0.000133312	1.400	0.1618	
RV_d	0.348422	0.0412388	8.449	6.75e-17	***
$RV_{d,\overline{w}}$	0.445457	0.0715836	6.223	6.29e-10	***
$RV_{d,\overline{m}}$	0.177177	0.0652615	2.715	0.0067	***
HARD regressors					
const.	0.000521386	0.000134902	3.865	0.0001	***
D	-0.000185484	2.27075e-05	-8.168	6.48e-16	***
RV_d	0.363405	0.0389744	9.324	3.79e-20	***
$RV_{d,\overline{w}}$	0.433063	0.0684492	6.327	3.28e-10	***
$RV_{d,\overline{m}}$	0.175131	0.0636041	2.753	0.0060	***

Statistic	HAR	HARD
Sum squared resid.	0.002370	0.002267
S.E. of regression	0.001245	0.001218
R-squared	0.819944	0.827747
R-squared (adjusted)	0.819591	0.827296
Akaike criterion	-16141.48	-16207.36
Mean Error	-8.0167e-05	-9.0146e-05
Root Mean Sq. Err.	0.00095765	0.00089628
Mean Absolute Err.	0.00067989	0.00064891
Mean % Error	-4.5107	-4.3937
Mean Abs. % Error	13.863	13.122
Theil's U	0.84393	0.78562

Table 7.3: USD/CHF HAR vs. HARD comparison

variable	coefficient	S.E.	t-ratio	p-value	
HAR regressors					
const.	0.000442539	0.000193235	2.290	0.0222	**
RV_d	0.308791	0.0745616	4.141	3.66e-05	***
$RV_{d,\overline{w}}$	0.426482	0.0807748	5.280	1.50e-07	***
$RV_{d,\overline{m}}$	0.203709	0.0809833	2.515	0.0120	**
HARD regressors					
const.	0.000900145	0.000193972	4.641	3.80e-06	***
D	-0.000241019	3.07661e-05	-7.834	9.30e-15	***
RV_d	0.331700	0.0722360	4.592	4.79e-06	***
$RV_{d,\overline{w}}$	0.409098	0.0795643	5.142	3.11e-07	***
$RV_{d,\overline{m}}$	0.197080	0.0781015	2.523	0.0117	**

Statistic	HAR	HARD
Sum squared resid.	0.003789	0.003617
S.E. of regression	0.001647	0.001610
R-squared	0.615111	0.632560
R-squared (adjusted)	0.614285	0.631507
Akaike criterion	-13977.69	-14040.69
Mean Error	-0.00029151	-0.00030494
Root Mean Sq. Err.	0.0015259	0.0014562
Mean Absolute Err	0.0010661	0.0010189
Mean % Error	-7.1225	-7.1192
Mean Abs. % Error	15.965	15.188
Theil's U	0.88318	0.84455

Table 7.4: Diebold–Mariano test, HAR vs. HARD

	EUR/USD		GBP/USD		USD/CHF	
h	$\mathrm{dm}\|_{L_{sqr}}$	$\mathrm{dm}\|_{L_{abs}}$	$\mathrm{dm}\|_{L_{sqr}}$	$\mathrm{dm}\|_{L_{abs}}$	$\mathrm{dm}\|_{L_{sqr}}$	$\mathrm{dm}\|_{L_{abs}}$
1	3.2470	2.3699	3.7453	1.9261	3.5671	2.2873
2	3.6726	2.5125	4.1139	2.5325	3.8149	2.4940
3	3.7141	2.3496	3.7357	2.1060	4.0792	2.3278
4	3.6999	2.2903	3.9664	2.2301	4.1412	2.4897
5	3.4080	2.9055	3.8791	2.2398	3.5941	2.3893
6	3.2041	2.5659	4.1984	2.6132	3.7376	2.4383
7	3.4023	2.7115	4.5528	2.3163	4.2623	2.6638
8	3.6141	2.8236	5.2636	3.7317	4.0940	2.6573
9	3.4179	2.6356	3.7460	2.1902	3.2129	2.3740
10	3.6716	3.1866	3.6806	2.8620	3.4332	2.4364
11	3.3655	3.1566	3.4137	2.4414	3.5654	2.7099
12	3.7628	3.8392	3.8884	3.2831	3.7188	2.7653
13	3.3651	3.4646	3.4154	2.1792	3.7279	2.5802
14	4.1323	3.3986	4.0151	2.6465	3.8209	2.5045
15	3.6649	3.9161	3.7260	2.6254	4.4978	3.1369
16	3.5953	3.7239	4.3484	3.4607	4.3012	2.8397
17	3.2992	3.1479	3.5001	2.3721	3.3097	2.3508
18	3.2989	3.1265	3.8807	3.2154	3.3693	2.3819
19	3.8259	3.0072	3.9051	2.5500	3.8080	2.5082
20	3.6819	3.5299	4.0120	2.9025	3.8817	3.0191

h ... covariance horizon

$\mathrm{dm}|_{L_{sqr}}$ ... D-M statistic using square loss function

$\mathrm{dm}|_{L_{abs}}$ ... D-M statistic using absolute loss function

Chapter 8

Neural Networks' Performance

Finally we get to testing the performance of neural networks against other models, namely HAR(D) and Box–Jenkins' ARIMA models and their variations. Prior to showing the results, it needs to be said that number of neural networks had been specified and tested, before few of them were chosen into the final showdown. It turned out – rather unsurprisingly – that more complex networks tend to excel at fitting the in–sample data, however at the cost of losing drive when forecasting the out–of–sample. So only the balanced[1] NNs made it through into comparison tables placed at the end of this chapter.

Another remark about neural networks concerns *adjusted* R–squared figures in Tables 8.4, 8.6 and 8.8. These figures follow the definition 3.7, which has its roots in linear models[2] and thus its interpretation in context of non–linear models should be rather restrained. On that note, due to relatively small number of parameters estimated in HAR and ARIMA models, it is easy to see why the difference in their R–squared and adjusted R–squared

[1] That is, those NNs that didn't "overfit" the in–sample data.

[2] See Appendix for discussion over R–squared.

figures is almost negligible and doesn't have any real significance to the final comparison.

The chosen models are Diebold–Mariano tested within each time series, followed by detailed comparison tables, each segmented into four groups of models (Models' notations are explained in the section to follow.);

First part shows figures for Corsi's HAR model and our HARD extension. In an attempt to fully exploit the HAR(D) model idea, the second part of tables shows HAR(D) models that were treated for autocorrelation in residuals, leading to new "moving average" (MA) terms being added to models. Suffice to say that this further enhancement brought only subtle non–significant improvements to statistics.

Third part of the final tables shows several ARIMA models, both in its canonical form defined in Chapter 4 to compare performance with Corsi's HAR, and in its "c D" enhanced versions to compare with HARD and NNs models. Specifications of both types of ARIMA were based on the autocorrelation and partial autocorrelation functions following recommendations from *Cipra (1986)*. It deserves to be stressed that "c D" enhanced ARIMA models did really well in the final showdown.

The last, fourth part of results is dedicated to neural networks and their performance. Some of their nuances have already been discussed above. All the NNs shown in tables use the "D" variable, so they can be fairly compared only to HARD and "ARIMA c D" models. As for the notation, it's rather difficult, if not impossible, to present a simple notation to complex NN structures. So while HAR(D) and ARIMA (c D) models notation is able to define the model for reconstruction, the NN notation are more or less *only labels* to give a reader at least some sense of what inputs enter the network.

Neural networks used in the end differ in their complexity substantially.

While the simplest network – labeled "AR1MA1 1L(7N)" – employed to fit and forecast USD/CHF series has 28 parameters to estimate, the most complex network in our set – "AR14MA8 2L" for EUR/USD forecasting – had close to 600 (!) parameters. All the other networks ranged from around 50 to 80 parameters to estimate, with one more having 126 parameters. We can confront these numbers with the rest – none of HAR(D) and ARIMA (c D) models exceeded 14 parameters for estimation, with the usual number of parameters ranging from 5 to 10.

8.1 Models – notation

All the models used build on already presented definitions. Original HAR and HARD models exactly follow definitions (4.13) and (4.15), respectively, re–indexed for forecasting such as in (7.1). Use of the HARD extension model is distinguished by the presence of variable "D" in the tables. Letter "c" then signalizes the constant term being incorporated into estimation (however, explicitly writing "c" with HAR(D) models is almost redundant, as we employ the constant in all but one variations of HAR(D) models).

Both types of these models are also present with MA variant, as discussed above. Here we divert from standard MA(q) notation, as we sometimes use individual indexes of past errors, rather then the whole set *1,...,q*. These individual error terms are indexed with letter i preceding the actual index number, abbreviating the word *individual* – such as MA(i2,i4), meaning that only the error terms e_{t-2} and e_{t-4} are employed in the model. Sometimes, the notation is combined, *e.g.* MA(8,i14) meaning that error terms from past eight fitted values along with individual error term e_{t-14} enter the estimation. Several examples from which everything should be crystal clear:

Notation "HAR MA(3,i6) c" represents the model

$$\begin{aligned} RV_{d,t} &= const. + \beta_1 RV_{d,t-1} + \beta_2 RV_{w,t-1} + \beta_3 RV_{m,t-1} + u_t \\ u_t &= \epsilon_t + \left(\sum_{i=1}^{3} \theta_i \epsilon_{t-i} \right) + \theta_6 \epsilon_{t-6} \quad , \end{aligned}$$

notation "HAR MA(i2,i9) c D" is the HARD model defined as

$$\begin{aligned} RV_{d,t} &= const. + \beta_0 D_t + \beta_1 RV_{d,t-1} + \beta_2 RV_{w,t-1} + \beta_3 RV_{m,t-1} + u_t \\ u_t &= \epsilon_t + \theta_2 \epsilon_{t-2} + \theta_9 \epsilon_{t-9} \quad , \end{aligned}$$

and notation "HAR MA(i5,i8,i12)", used only once in the set, stands for

$$\begin{aligned} RV_{d,t} &= \beta_1 RV_{d,t-1} + \beta_2 RV_{w,t-1} + \beta_3 RV_{m,t-1} + u_t \\ u_t &= \epsilon_t + \theta_5 \epsilon_{t-5} + \theta_8 \epsilon_{t-8} + \theta_{12} \epsilon_{t-12} \quad , \end{aligned}$$

i.e. even without the constant term. Every other HAR(D) type model is a variation on these notation examples.

Similar situation with indexes arises with ARIMA models, although standard ARIMA(p, d, q) models are employed as well (or rather ARIMA$(p, 1, q)$ in most of our cases). Since the usual practice is to state the p, d, q parameters of ARIMA model divided by comma, using the same notation for individual indexes such as previously with HAR(D) models would be confusing. Therefore we use square brackets to incorporate the list of relevant indexes. In addition to canonical ARIMA definition from Chapter 4, constant term and variable "D" are added elsewhere. Again, a few examples will clarify everything.

Notation "ARIMA(7,1,4)" is the standard

$$\begin{aligned} RV_{d,t} &= RV_{d,t-1} + \Delta RV_{d,t} \\ \Delta RV_{d,t} &= \sum_{i=1}^{7} \varphi_i \Delta RV_{d,t-i} + \epsilon_t + \sum_{j=1}^{4} \theta_j \epsilon_{t-j} \quad , \end{aligned}$$

notation "AR(7) c D" or alternatively "ARIMA(7,0,0) c D", stands for the model

$$RV_{d,t} = const. + \beta_0 D_t + \sum_{i=1}^{7} \beta_i RV_{d,t-i} + u_t \quad ,$$

Notation "ARIMA([3,6,9-11],1,4)" would represent the model

$$\begin{aligned} RV_{d,t} &= RV_{d,t-1} + \Delta RV_{d,t} \\ \Delta RV_{d,t} &= \varphi_3 \Delta RV_{d,t-3} + \varphi_6 \Delta RV_{d,t-6} + \sum_{i=9}^{11} \varphi_i \Delta RV_{d,t-i} + u_t \\ u_t &= \epsilon_t + \sum_{j=1}^{4} \theta_j \epsilon_{t-j} \quad , \end{aligned}$$

and the notation "ARIMA(2,1,[3]) c D" stands for the model defined as

$$\begin{aligned} RV_{d,t} &= RV_{d,t-1} + \Delta RV_{d,t} \\ \Delta RV_{d,t} &= const. + \beta D_t + \sum_{i=1}^{2} \varphi_i \Delta RV_{d,t-i} + \epsilon_t + \theta_3 \epsilon_{t-3} \quad . \end{aligned}$$

These examples cover all the variants of ARIMA models used forth.

Now the hard part – notation of NN models. As was mentioned at the beginning of this chapter, some of the neural networks have up to hundreds of parameters. Obviously, this number of parameters cannot have a simple text representation. Nor can simpler NNs with 50 parameters. Therefore we resort ourselves to *just* labeling neural networks.

Nonetheless, the text sequence labeling the NN will always contain the number of lagged values of time series used as inputs, sort of autoregressive part of neural network, therefore labeled "AR". For example, NN labeled "AR3 ..." has the last three lagged values of the series on its inputs.

Second part of label will be the "MA" part, sort of moving average inputs portion. So a neural network labeled "... MA2 ..." will have the last two errors (residuals) of the network entering the network as its new inputs – implying a recurrent network.

Remaining part of NN label represents the number of layers and neurons within them. We only use networks with up to two hidden layers. The number of neurons in each layer is added in brackets, *e.g.* "1L(9N)" representing a hidden layer of nine neurons, or "2L(3,2N)" representing two hidden layers, first with three neurons, second with two neurons.

The default setting is that all the inputs connect with all the neurons of the first hidden layer, all the neurons of the hidden layer connect to all the neurons of the subsequent hidden layer and all the neurons of the last hidden layer connect to the output neuron (in fact, hadn't it been for the MA part, most of the neural networks we use would have had simple feed forward structure). In several cases, not all inputs are connected to all the neurons in the following hidden layer. This information is represented by the division of the neurons in the label, *e.g.* instead of writing "AR5 MA5 1L(5N)" we write "AR5 MA5 1L(3+2N)" to distinguish the latter network in which the "AR5" inputs go into three hidden neurons and "MA5" inputs connect to other two hidden neurons – total number of neurons in that layer being five, as well as in the former NN. More complicated combinations of connections among input neurons, hidden neurons and output neuron, are *not* captured by the network's label!

Another feature of neural networks not captured by its label in the above specification is presence of loop back links on hidden neurons, substituting the decay neurons. Also not evident form notation is the presence of cascade structures (or shortcut links). Obviously such notation is not good enough to uniquely identify the NN structure it's supposed to represent. However, the point of this exercise is to test these structures, not to spend too much effort on developing notation system for them. Let's test.

8.2 EUR/USD

Overall results on EUR/USD time series are not too impressive; Among the three neural networks that made it to the final comparison tables, only "NN AR3 MA3 1L" performed better than HARD model in all but one indicator (MAE).

However, Diebold–Mariano test shows that the supremacy over HARD model is only marginal, short of statistical significance. Interestingly, the simplest AR7 model performed on par with HARD model (and with neural networks as well) in terms of forecasting accuracy. Diebold–Mariano test statistics are shown in the Table 8.1 below.

Table 8.1: EUR/USD Diebold–Mariano statistics

h	AR HARD	NN1 HARD	NN2 HARD	NN3 HARD	NN1 AR
1	-0.5503	0.0633	-0.5878	-0.6313	0.6218
2	-0.5994	0.0649	-0.5649	-0.7338	0.5021
3	-0.6065	0.0771	-0.6530	-0.7260	0.4908
4	-0.6008	0.1011	-0.7847	-0.8528	0.4866
5	-0.5900	0.1131	-1.0891	-0.8713	0.4629

Individual statistics are in Tables 8.4 and 8.5. Neural networks fit in–sample data considerably better through SSR statistic and therefore show noticeably higher R–squared than both HARD and ARIMA (AR) models, while maintaining their forecast accuracy. The third NN achieved best MAE and MA%E statistics, along with best SSR, of all models tested.[3]

[3]Do notice how close ARIMA(2,1,2) and original HAR(1,5,22) performed!

8.3 GBP/USD

With GBP/USD time series, only two neural networks made it to final comparison tables. The first neural network (AR5–MA8–1L) has been tested in two different stages of learning, its structure being untouched. Results of Diebold–Mariano tests are summarized in Table 8.2.

Both NNs performed better than the HARD model. The best among Box–Jenkins models – ARIMA(7,1,4) c D – displayed better forecasts than HARD and even the second NN. Only the first NN was able to outperform both HARD and ARIMA model but, while its Diebold–Mariano statistics are decisively better than in case of EUR/USD series, they still lack the needed statistical significance to call it a winner.

Table 8.2: GBP/USD Diebold–Mariano statistics

h	ARIMA HARD	NN1 HARD	NN2 HARD	NN1 ARIMA	NN2 ARIMA
1	1.2970	0.8613	0.6473	0.3939	-0.0390
2	1.2260	0.6622	0.8174	0.3003	-0.0663
3	1.4009	0.7099	0.8085	0.3081	-0.0552
4	1.2230	0.7058	0.6448	0.3364	-0.0539
5	1.1088	0.7093	0.6597	0.3533	-0.0512

Detailed individual estimation and forecast statistics for these and more models are summed up in Tables 8.6 and 8.7. Again, when fitting the data, NNs show supreme SSR and R–squared, the difference being even more dominant than in previous case – SSR of NNs is at least 20 % better (!) than SSRs of competing models.

In forecasts, the first NN was able to achieve best RMSE. The second

NN produced respectable MAE, however, the best MAE was achieved by ARIMA model. Overall, NNs performed on par with ARIMA, both being marginally better than HARD model.

8.4 USD/CHF

With USD/CHF series, neural networks draw much more one–sided story. They prevail over ARIMA model on 5 % level and over HARD model even on 1 % level of significance. Two NNs with lowest MAE are Diebold–Mariano tested against "ARIMA(1,1,2) c D" and HARD in Table 8.3.[4]

Table 8.3: USD/CHF Diebold–Mariano statistics

h	ARIMA HARD	NN1 HARD	NN2 HARD	NN1 ARIMA	NN2 ARIMA
1	1.3264	2.3528	2.3460	1.8548	1.8841
2	0.9424	1.7455	1.7789	1.6797	1.5839
3	0.8139	1.5891	1.5891	1.6704	1.4682
4	0.7617	1.5017	1.5140	1.5039	1.3891
5	0.7667	1.4511	1.4934	1.4236	1.3592

Individual thorough model statistics are shown in Tables 8.8 and 8.9. Neural netoworks rule yet again substantially when fitting the in–sample data, this time accompanied with by far best RMSE, MAE and MA%E which help to achieve also the best Theil's U. From the rest of results it is worth noticing that basic ARIMA(1,1,2) and its variant with "c D" both outperformed their HAR and HARD counterparts respectively in forecasts.

[4]NN1 stands for "AR1 MA1 1L(9N)" and NN2 for "AR5 MA5 1L(3+2N)" network.

Table 8.4: EUR/USD RV_d Models Estimation Summary

model method, vars	#obs.	R-squared (adj.)	SSR (adj.)
HAR(1,5,22)			
c (OLS)	1563	0.747330 (0.746843)	0.002416 (0.002418)
c D (OLS)	1563	0.767966 (0.767370)	0.002219 (0.002220)
HAR(1,5,22)			
MA(1) c	1562	0.748035 (0.747550)	0.002409 (0.002412)
MA(i4,i5,i8,i12) c	1551	0.753746 (0.753268)	0.002342 (0.002361)
MA(i2) c D	1561	0.769215 (0.768622)	0.002207 (0.002211)
MA(i2,i14) c D	1549	0.772809 (0.772221)	0.002160 (0.002181)
MA(8,i14) c D	1549	0.773746 (0.773160)	0.002151 (0.002172)
Box-Jenkins class of models (CML estimation)			
ARIMA(2,1,2)	1561	0.747783 (0.747298)	0.002412 (0.002416)
ARIMA(6,1,4)	1557	0.752300 (0.750865)	0.002368 (0.002378)
ARIMA(1,1,1) c D	1562	0.758487 (0.758022)	0.002309 (0.002312)
ARIMA(2,1,[3-4]) c D	1561	0.767087 (0.766339)	0.002227 (0.002231)
AR(7) c D	1557	0.768315 (0.767117)	0.002214 (0.002224)
Neural Networks			
AR3 MA3 1L(10N)	1560	0.785215 (0.778698)	0.002054 (0.002058)
AR5 MA5 1L(3+2N)	1558	0.787776 (0.780758)	0.002028 (0.002034)
AR14 MA8 2L(9,4N)	1550	0.792995 (0.667685)	0.001979 (0.001997)

Table 8.5: EUR/USD RV_d Models Forecasting Performance

model method, vars	RMSE	MAE	M%E, MA%E, U
HAR(1,5,22)			
c (OLS)	0.0012865	0.00091384	-4.32, 15.11, 0.81687
c D (OLS)	0.0012083	0.00086032	-3.96, 14.06, 0.76977
HAR(1,5,22)			
MA(1) c	0.0012896	0.00091568	-4.32, 15.16, 0.81902
MA(i4,i5,i8,i12) c	0.0012816	0.00091711	-3.95, 15.07, 0.81745
MA(i2) c D	0.0012111	0.00086517	-3.97, 14.16, 0.77140
MA(i2,i14) c D	0.0012295	0.00087095	-3.85, 14.28, 0.78693
MA(8,i14) c D	0.0012249	0.00085923	-3.77, 14.08, 0.78395
Box-Jenkins class of models (CML estimation)			
ARIMA(2,1,2)	0.0012958	0.00091856	-4.37, 15.16, 0.82128
ARIMA(6,1,4)	0.0012921	0.00091862	-4.34, 15.15, 0.82058
ARIMA(1,1,1) c D	0.0012504	0.00089636	-3.92, 14.64, 0.80225
ARIMA(2,1,[3-4]) c D	0.0012258	0.00086992	-3.74, 14.14, 0.77955
AR(7) c D	0.0012158	0.00085772	-3.43, 13.95, 0.77887
Neural Networks			
AR3 MA3 1L(10N)	0.0012073	0.00086188	3.76, 14.00, 0.76825
AR5 MA5 1L(3+2N)	0.0012202	0.00085319	3.12, 13.89, 0.77071
AR14 MA8 2L(9,4N)	0.0012247	0.00083443	3.95, 13.72, 0.78257

Table 8.6: GBP/USD RV_d Models Estimation Summary

model method, vars	#obs.	R-squared (adj.)	SSR (adj.)
HAR(1,5,22)			
c (OLS)	1532	0.819944 (0.819591)	0.002370 (0.002372)
c D (OLS)	1532	0.827747 (0.827296)	0.002267 (0.002268)
HAR(1,5,22)			
MA(1) c D	1531	0.828626 (0.828176)	0.002255 (0.002258)
MA(5) c D	1527	0.829804 (0.829357)	0.002240 (0.002249)
MA(i8) c	1524	0.821741 (0.821390)	0.002345 (0.002358)
MA(i8) c D	1524	0.828684 (0.828233)	0.002254 (0.002263)
MA(i5,i8,i12)	1520	0.823465 (0.823232)	0.002327 (0.002347)
MA(i5,i8,i12) c D	1520	0.829798 (0.829349)	0.002236 (0.002255)
MA(i4) c D	1528	0.828770 (0.828320)	0.002254 (0.002261)
Box-Jenkins class of models (CML estimation)			
ARIMA(7,1,2)	1525	0.822067 (0.821132)	0.002342 (0.002354)
ARIMA(0,1,2)	1532	0.819895 (0.819778)	0.002370 (0.002372)
ARIMA(0,1,2) c D	1532	0.825439 (0.825096)	0.002297 (0.002298)
ARIMA(7,1,4) c D	1525	0.829919 (0.828575)	0.002239 (0.002251)
Neural Networks			
AR5 MA8 1L(5N)	1527	0.864828 (0.857153)	0.001779 (0.001784)
AR5 MA8 1L(5N)	1527	0.862852 (0.855120)	0.001805 (0.001810)
AR7 MA4 1L(9N)	1525	0.856939 (0.843615)	0.001883 (0.001892)

Table 8.7: GBP/USD RV_d Models Forecasting Performance

model method, vars	RMSE	MAE	M%E, MA%E, U
HAR(1,5,22)			
c (OLS)	0.00095765	0.00067989	-4.51, 13.86, 0.84393
c D (OLS)	0.00089628	0.00064891	-4.39, 13.12, 0.78562
HAR(1,5,22)			
MA(1) c D	0.00089689	0.00065375	-4.46, 13.24, 0.78938
MA(5) c D	0.00089249	0.00065299	-4.37, 13.24, 0.79111
MA(i8) c	0.00093431	0.00066208	-4.36, 13.48, 0.82485
MA(i8) c D	0.00088675	0.00064046	-4.29, 12.94, 0.77864
MA(i5,i8,i12)	0.00092769	0.00064710	-2.86, 12.93, 0.81664
MA(i5,i8,i12) c D	0.00088653	0.00063690	-4.22, 12.85, 0.77592
MA(i4) c D	0.00089325	0.00065073	-4.37, 13.19, 0.78952
Box-Jenkins class of models (CML estimation)			
ARIMA(7,1,2)	0.00095243	0.00067514	-3.29, 13.53, 0.84502
ARIMA(0,1,2)	0.00095406	0.00067213	-3.42, 13.53, 0.84553
ARIMA(0,1,2) c D	0.00090719	0.00065099	-3.80, 13.08, 0.79682
ARIMA(7,1,4) c D	0.00088205	0.00063557	-3.23, 12.68, 0.77827
Neural Networks			
AR5 MA8 1L(5N)	0.00087657	0.00066326	4.78, 13.66, 0.80090
AR5 MA8 1L(5N)	0.00087313	0.00066031	5.25, 13.66, 0.79898
AR7 MA4 1L(9N)	0.00088278	0.00064137	3.82, 13.09, 0.79977

Table 8.8: USD/CHF RV_d Models Estimation Summary

model method, vars	#obs.	R-squared (adj.)	SSR (adj.)
HAR(1,5,22)			
c (OLS)	1401	0.615111 (0.614285)	0.003789 (0.003792)
c D (OLS)	1401	0.632560 (0.631507)	0.003617 (0.003620)
HAR(1,5,22)			
MA(i2,i9) c	1392	0.621917 (0.621099)	0.003718 (0.003745)
MA(4) c	1397	0.618146 (0.617324)	0.003758 (0.003771)
MA(i2,i9,i14) c D	1387	0.640621 (0.639581)	0.003531 (0.003569)
MA(i2) c D	1399	0.634479 (0.633430)	0.003598 (0.003606)
MA(3) c D	1398	0.635644 (0.634598)	0.003586 (0.003596)
Box-Jenkins class of models			
ARIMA(1,1,2)	1400	0.614239 (0.613687)	0.003798 (0.003803)
ARIMA(2,1,2)	1399	0.616988 (0.616165)	0.003771 (0.003780)
ARIMA(1,1,1) c D	1400	0.617972 (0.617151)	0.003761 (0.003766)
ARIMA(1,1,2) c D	1400	0.622079 (0.620996)	0.003721 (0.003726)
ARIMA([9-10],1,2) c D	1391	0.630082 (0.628756)	0.003642 (0.003671)
ARIMA([1,9-10],1,2) c D	1391	0.630624 (0.629034)	0.003637 (0.003665)
Neural Networks			
AR1 MA1 1L(7N)	1401	0.661881 (0.656041)	0.003321 (0.003323)
AR1 MA1 1L(9N)	1401	0.672295 (0.661442)	0.003226 (0.003229)
AR5 MA5 1L(3+2N)	1397	0.693526 (0.682754)	0.003014 (0.003025)
AR7 MA4 1L(6N)	1395	0.706728 (0.690159)	0.002887 (0.002902)

Table 8.9: USD/CHF RV_d Models Forecasting Performance

model method, vars	RMSE	MAE	M%E, MA%E, U
HAR(1,5,22)			
c (OLS)	0.0015259	0.0010661	-7.12, 15.97, 0.88318
c D (OLS)	0.0014562	0.0010189	-7.12, 15.19, 0.84455
HAR(1,5,22)			
MA(i2,i9) c	0.0015838	0.0011074	-7.63, 16.59, 0.91391
MA(4) c	0.0015014	0.0010600	-7.08, 15.98, 0.88299
MA(i2,i9,i14) c D	0.0015053	0.0010523	-7.27, 15.79, 0.87821
MA(i2) c D	0.0014423	0.0010176	-7.10, 15.23, 0.84329
MA(3) c D	0.0014351	0.0010123	-7.05, 15.17, 0.84605
Box-Jenkins class of models			
ARIMA(1,1,2)	0.0014415	0.0010222	-6.21, 15.59, 0.86891
ARIMA(2,1,2)	0.0014428	0.0010262	-6.69, 15.70, 0.87220
ARIMA(1,1,1) c D	0.0013937	0.0009815	-5.51, 14.87, 0.83897
ARIMA(1,1,2) c D	0.0013844	0.0009679	-5.17, 14.57, 0.83538
ARIMA([9-10],1,2) c D	0.0014184	0.0010005	-6.33, 15.29, 0.86282
ARIMA([1,9-10],1,2) c D	0.0014129	0.0009955	-6.03, 15.17, 0.86213
Neural Networks			
AR1 MA1 1L(7N)	0.0013385	0.00094899	5.04, 14.24, 0.81950
AR1 MA1 1L(9N)	0.0013197	0.00093650	5.01, 14.06, 0.81126
AR5 MA5 1L(3+2N)	0.0013085	0.00092650	5.24, 14.17, 0.81455
AR7 MA4 1L(6N)	0.0013144	0.00094568	5.48, 14.37, 0.82405

Chapter 9

Conclusions

Neural networks have remarkable ability to fit given data. However, this feature can often harm their ability to forecast new data, should their nature shift in one way or another. So NNs are more prone to fall into the overfitting trap than linear models are. On the other hand, they are capable of handling more complex tasks.

We present results of several neural networks' performance tests, compared to two other types of models, commonly used HAR(D) and ARIMA, on three FX time series EUR/USD, GPP/USD and USD/CHF. The networks used vary in complexity substantially – from a simple network with three input neurons and several hidden neurons, to a network with up to 600 parameters to estimate. Tests we conducted confirm that complex networks are not always better in forecasts, quite contrary, in our application, there exists a clear trade-off for every NN, between better fit of in-sample data and better forecast performance on out-of-sample data. This trade-off comes as no surprise, as this problem is attended to elsewhere in literature, especially on machine learning, where neural networks dominate.

Nevertheless, we were able to present several neural networks that surpass

other tested models in both disciplines. Some of them do so statistically significantly, as shown by Diebold–Mariano tests.

Along the way, as a by-product of our research, we managed to introduce a solid enhancement to original HAR model defined by *Corsi (2009)* and named the new model "HARD". Improvement in both fitting the data and forecasting the data consistently proved to be statistically significant – based on Diebold–Mariano tests – in all three time series. Yet its implementation is extremely simple, while the needed extra information is in most cases already present in the data.

What came as a bit of surprise was how well the ARIMA models did in comparison with HARD and NN models; Except the EUR/USD series, simple ARIMA (c D) models outperformed HARD model in terms of forecasting, and except USD/CHF series, their performance wasn't significantly worse than that of neural networks.

9.1 Possible future directions

Application of neural networks on realized volatility (variance) forecasting remains an open field. Particular neural networks used in this work have by no means the best possible structures for the task – they only evolved from a long line of trials and errors.

Natural next step would be qualitative analysis of number of neural networks, searching for regularities in their structures after they have been estimated. Decomposition of these networks into substructures could lead to better understanding of how the past path of realized volatility (variance) translates into its present or future values.

Neural networks are ideal for pattern recognitions. However, complex

networks tend to overfit the data and as a consequence show poor forecast performance. It would be undoubtedly beneficial to introduce a framework allowing to switch between relatively simple networks on the go – each of them specialized in recognizing one simple pattern. This would definitely require some extensive programming.

Another interesting question during the estimation of neural network is when to stop the network from further learning. This is again closely related to overfitting problem. Once a network passes certain level of accuracy in fitting data, its forecast performance gets permanently worse. It is also unclear how to proceed systematically when building a neural network – how to determine the optimal structure and complexity of the network for a particular application. Some answers probably rest much deeper in neural networks theory.

Appendix A

Selected statistics in detail

A.1 R–squared deconstructed

In Chapter 3, the coefficient of determination, or R–squared and adjusted R–squared, is defined by (3.6) and (3.7), respectively. Just to recall,

$$R^2 = 1 - \frac{\sum (\hat{y}_i - y_i)^2}{\sum (y_i - \bar{y})^2} \tag{3.6}$$

and

$$adj.R^2 = 1 - \frac{(T-1)}{(T-1-p)} \frac{\sum (\hat{y}_i - y_i)^2}{\sum (y_i - \bar{y})^2} \quad . \tag{3.7}$$

Let us explain those formulas;

The interpretation I tend to favor is to view the coefficient of determination as a complement measure of how big or small is the *mean squared error* (MSE) of the model, relative to the *variance* (Var) of the estimated original series. That is, the value of real interest is

$$\frac{MSE_{model}}{Var_{series}} \tag{A.1}$$

this makes sense as variation is *de facto* mean squared "error" or deviation from the mean value of a dataset.

Say we have a dataset with n observations, and a model that estimates $(n-k)$ of those observations, where $0 \leq k < n$. Rewriting the ratio (A.1) with definitions of MSE and Var we get

$$\begin{aligned} \frac{MSE_{model}}{Var_{series}} &= \frac{\sum_{i=1}^{n-k}(\hat{y}_i - y_i)^2}{n-k}\left[\frac{\sum_{i=1}^{n}(y_i-\bar{y})^2}{n}\right]^{-1} = \\ &= \left(\frac{\sum_{i=1}^{n-k}(\hat{y}_i-y_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}\right)\left(\frac{n}{n-k}\right) \quad , \end{aligned} \tag{A.2}$$

thus, substracting (A.1) from 1 we get

$$1 - \frac{MSE_{model}}{Var_{series}} = 1 - \left(\frac{\sum(\hat{y}_i - y_i)^2}{\sum(y_i-\bar{y})^2}\right)\left(\frac{n}{n-k}\right) \quad . \tag{A.3}$$

Now, compare (A.3) with (3.7): should $n = T - 1$ and $k = p$, then those two would seem to be equal. For *some* models, such as AR(k), they are *almost* identical. However, generally they are *not* equal; the problem lies in the indexed sums.

Formula (A.3) is derived assuming the number of summands are $(n-k)$ and n from definitions of MSE and Var, respectively. Formula (3.7), on the other hand, relates p to number of parameters of the model without any real link to number of summands in the sum of squared residuals of the model.

In fact, (A.3) shows – and to some extent possibly fills – the gap between both definitions (3.6) and (3.7); The first one, the R–squared definition, does not address the potential difference in number of estimates and the number of observations in the dataset and therefore makes it difficult to find proper interpretation when the two numbers differ substantially for a particular model. The second definition, the adjusted R–squared, while preserving the flaw of the former one, inquires into complexity of the model above all, rendering it useful for comparing one type of model with different number of parameters, rather than comparing different types of models. That is to say that adjusted R–squared information value is only relevant when related

to original R–squared. Without it, comparison of two adjusted R–squared values of two different models, is vague at best.

It would seem that the ideal definition of the coefficient of determination is (A.3). Luckily whenever the number of parameters is reasonably low relative to the reasonably large number of observations (say up to few per cent), all three definitions result in very close values. Except that only (A.3) has clear interpretation.

A.2 Theil's U

A Dutch statistician/econometrician Henri Theil proposed several measures of forecasting accuracy, out of which only the one presented in *Theil (1966)* survived. The idea behind this measure, known as Theil's U today, is to compare forecast performance of a given model to a forecast performance of a naive model, *i.e.* model which assumes that the next value in series will be the same as the value preceding. That is, the naive model predicts

$$\hat{y}_{t+1}^{naive} = y_t \quad , \tag{A.4}$$

so the prediction error that the naive model makes at time $(t+1)$ is therefore

$$e_{t+1}^{naive} = \hat{y}_{t+1}^{naive} - y_{t+1} = y_t - y_{t+1} = -\Delta y_{t+1} \quad . \tag{A.5}$$

By now, the definition (3.8) is already clear. Theil simply compares what could be called the "root mean squared shifted[1] percentage error" of the tested model to the same indicator of the naive model. So, we have

$$U \equiv \frac{\left[\frac{1}{T-1}\sum_{t=1}^{T-1}\left(\frac{e_{t+1}}{y_t}\right)^2\right]^{\frac{1}{2}}}{\left[\frac{1}{T-1}\sum_{t=1}^{T-1}\left(\frac{-\Delta y_{t+1}}{y_t}\right)^2\right]^{\frac{1}{2}}} \tag{A.6}$$

[1]The percentage error is shifted, relating $(t+1)$ error to t base value.

which is obviously the same expression as definition (3.8).

A.3 Diebold–Mariano test

Taken directly from *Diebold & Mariano (1995)* paper; the authors propose a test for comparing forecast performance of two different models – let's index them model A and B. They define a general loss function of the forecast errors, $g(e_{i,t})$ where $i = A, B$. Loss function is typically either simple quadratic function or the absolute value of its input.

Loss differential d is then defined as

$$d_t \equiv g(e_{A,t}) - g(e_{B,t}) \tag{A.7}$$

with its mean

$$\bar{d} = \frac{1}{T}\sum_{t=1}^{T}\left[g(e_{A,t}) - g(e_{B,t})\right] \quad . \tag{A.8}$$

Now, the asymptotic test is based on the convergence in distribution of the sample mean loss differential:

$$\sqrt{T}(\bar{d} - \mu) \xrightarrow{dist.} N(0, 2\pi f_d(0))\,, \tag{A.9}$$

where $f_d(0)$ is the spectral density of the loss differential at frequency 0.[2] The large–sample $N(0,1)$ statistic for the null hypothesis of equal forecast accuracy is then defined

$$S_1 \equiv \frac{\sqrt{T}\bar{d}}{\sqrt{2\pi\hat{f}_d(0)}}, \tag{A.10}$$

where $(2\pi)\hat{f}_d(0)$ is a consistent estimator of $(2\pi)f_d(0)$. One such estimator would be used in (7.2), except for $h \to \infty$. For finite samples however, h is of course chosen finite.

[2] See the referenced paper for details.

Bibliography

[1] Andersen, Torben G. & Bollerslev, Tim & Diebold, Francis X. & Labys, Paul (2003): "Modeling and Forecasting Realized Volatility", *Econometrica, Vol. 71, pp. 529–626*

[2] Artzner, Philippe & Delbaen, Freddy & Eber, Jean-Marc & Heath, David (1999): "Coherent Measures of Risk", *Mathematical Finance, Vol. 9, No. 3, pp. 203–228*

[3] Baltagi, Badi H. (2008): "Econometrics", *Springer-Verlag Berlin Heidelberg, ISBN 978-3-540-76515-8*

[4] Barndorff-Nielsen, O.E. & Shephard, N. (2001): "Estimating Quadratic Variation Using Realized Volatility", *Nuffield College, Oxford*

[5] Barndorff-Nielsen, O.E. & Shephard, N. (2002): "Econometric Analysis of Realised Volatility and its Use in Estimating Stochastic Volatility Models", *Journal of the Royal Statistical Society, ser. B 64*

[6] Bollerslev, Tim (1986): "Generalized Autoregressive Conditional Heteroskedasticity", *Journal of Econometrics, Vol. 31, pp. 307–327*

[7] Box, George E. P. & Jenkins, Gwilym M. (1970): "Time series analysis: Forecasting and control", *Holden Day, San Francisco*

[8] Breiman, Leo (1996): "Bagging predictors", *Machine Learning, Vol. 36, pp. 105–139*

[9] Cipra, Tomáš (1986): "Analýza časových řad s aplikacemi v ekonomii", *SNTL Praha, DT 519.338.2*

[10] Corsi, Fulvio F. (2009): "A Simple Approximate Long Memory Model of Realized Volatility", *Journal of Financial Econometrics, Vol. 7, pp. 174–196*

[11] Diebold, F. X. & Mariano, R. S. (1995): "Comparing predictive accuracy", *Journal of Business & Economic Statistics, Vol. 13, pp. 253–263*

[12] Engle, Robert F. (1982): "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation", *Econometrica, Vol. 50, No. 4, pp. 987–1008*

[13] Fayyad, U. & Piatetsky-Shapiro, G. & Smyth, P. (1996): "From Data Mining to Knowledge Discovery in Databases", *AI Magazine, Vol. 17–3, pp. 37–54*

[14] Greene, William H. (2003): "Econometric Analysis", *Prentice Hall, Pearson Education, New Jersey, ISBN 0-13-110849-2*

[15] Gujarati, Damodar N. (2004): "Basic Econometrics, 4th Ed.", *McGraw-Hill Higher Education, ISBN 0-07-233542-4*

[16] Hájek, P. & Havel, I. & Chytil, M. (1966): "The method of automatic hypotheses determination", *Computing, Vol. 1, pp. 293–308*

[17] Hájek, P. & Havránek, T. (1978): "Mechanizing Hypothesis Formation: Mathematical Foundations for a General Theory", *Springer-Verlag Berlin Heidelberg, New York, ISBN 3-540-08738-9*

[18] Hornik, K. & Stinchcomb, A. & White, H. (1989): "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks, Vol. 2, pp. 359–366*

[19] Kwiatkowski D., Phillips P. C. B., Schmidt P., Shin Y. (1992): "Testing the Null Hypothesis of Stationarity against the Alternative of a Unit Root", *Journal of Econometrics, Vol. 54, pp. 159–178*

[20] McAleer, Michael & Medeiros, Marcelo C. (2011): "Forecasting Realized Volatility with Linear and Nonlinear Univariate Models", *Journal of Economic Surveys, Vol. 25, No. 1, pp. 6–18*

[21] McNelis, Paul D. (2005): "Neural Networks in Finance: Gaining Predictive Edge", *Elsevier Academic Press, ISBN 0-12-485967-4*

[22] Theil, Henri (1966): "Applied Economic Forecasting", *North-Holland Publishing Company, Amsterdam*

[23] Walczak, Steven (2001): "An Empirical Analysis of Data Requirements for Financial Forecasting with Neural Networks", *Journal of Management Information Systems, Vol. 17, No. 4, pp. 203–222*

Internet Links

Gretl Software	gretl.sourceforge.net
MatLab	www.mathworks.com
MemBrain	www.membrain-nn.de
Tick Data	tickdata.com