

SEMINAR ON “HIGH FREQUENCY FINANCE”

organized by:

Winfried Pohlmeier, University of Konstanz

Richard Olsen, Olsen & Associates, Switzerland

Topic 7: Overview of forecasting models

Denis Dökümcü

University of Konstanz

Denis.Doekuemcuc@uni-konstanz.de

University of Konstanz

Department of Econometrics

Summer Semester 2002

Contents

1	Introduction	1
2	Operators and time deformation	1
2.1	Operators	1
2.1.1	Exponential Moving Average (EMA)	3
2.1.2	The Iterated EMA Operator	4
2.2	ϑ - time scale	4
2.3	Intrinsic time scale	5
3	Forecasting return	6
3.1	A simple forecasting method	6
3.2	Multi-horizon forecast of returns	7
4	Forecasting volatility	9
4.1	Overview of volatility models	10
4.1.1	Parameter Estimation of GARCH Models	11
4.1.2	The HARCH Model	12
4.2	Volatility for value-at-risk: Three simple models	13
5	Measure for the quality	16
5.1	Quality measures for volatility	16
6	Neural Net	19
6.1	Definition	19
6.2	Feedforward Multilayer Perceptron	20
6.3	Learning with Back-Propagation	22
6.4	Example	25
6.5	Limits of the Neural net	27
7	Conclusion	28

1 Introduction

In former times it was normal to achieve the latest datas from the stock exchanges a few minutes later. But now, in modern times it is possible to get the datas “immediately”. Every person from every country can have every information about stock prices after a second. We live in a world with *High Frequency Finance*.

If you have informations tick by tick, how will be the quality of your forecast? In this paper we will take a look at forecasting models and the quality of them. The variables, we want to estimate/forecast, must be observable. So we can compare the real value and the forecast value of the variable in the future for statistical quality tests. The following variables can be forecasted:

- (1) The absolute size of future returns.
- (2) The future return over the forecast period.
- (3) A full probability distribution function of future returns.
- (4) The volatility over the forecast period.

This paper is divided in two independent parts. In the first part, we will examine the standard approach for forecasting with tools from the statistics. There we want to concentrate us of forecasting the return (see Section 3) and the volatility (see Section 4). In Section 5 we will find some measures for the quality. In the second part, that is only Section 6, we want to introduce a new field in forecasting: neural net. The results in this Section will not be complete, because the section should only wake up the interest for this new field and gives some introducing words to it.

But before we can start with our forecasting, we need some definitions.

2 Operators and time deformation

2.1 Operators

The volatility forecasting models, we want to consider here, are based on time series operators. These operators are used to transform raw, inhomogeneous time series to (homogeneous or inhomogeneous) time series of the variable to be analyzed, here the volatility. We can divide the operators in two classes:

- Microscopic operators: depend on the actual sample of inhomogeneous time series. Eliminating some random ticks leads to a different result.
- Macroscopic operators: obtain average behavior of their time series argument. They are immune to adding, eliminating few ticks, and to small variations of individual ticks.

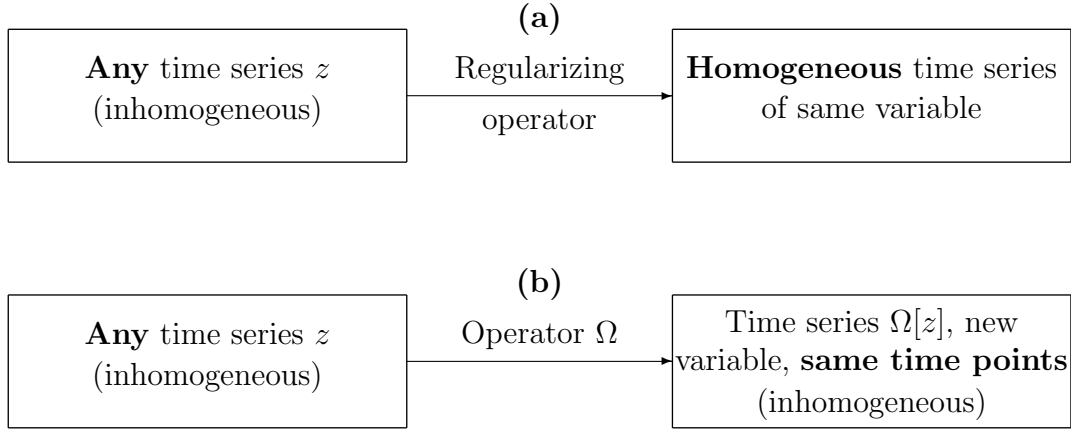


Figure 2.1: Different operator types to study time series: (Source:[3])

(a) Sampling an inhomogeneous time series at regular time intervals. The resulting homogeneous time series can be treated by standard methods of time series analysis.

(b) Computing a new variable from the initial variable while keeping the initial (inhomogeneous) time points. Example: computing a series of local volatility values from the initial price series.

In the following we focus on operators with these useful properties:

- Linear operators: $\Omega[z_1 + cz_2] = \Omega[z_1] + c\Omega[z_2]$.
- Time-translation invariant operators: $\Omega[z(t - \Delta t)](t) = \Omega[z(t)](t - \Delta t)$.
- Causal operators: $\Omega[z](t)$ depends only on information which are already know at time t ($\Omega[z](t)$ must not depend on future informations).

An operator with these three properties can be written with the kernel $\omega(t)$:

$$\begin{aligned}\Omega[z](t) &= \int_{-\infty}^t dt' \omega(t - t')z(t') \\ &= \int_0^{\infty} dt' \omega(t')z(t - t').\end{aligned}$$

The kernel $\omega(t)$ is only defined on the positive semiaxis and should decay for t large enough. For the further understanding it is necessary to define some values:

- The n -th moment of a causal kernel ω is defined as

$$\langle t^n \rangle_{\omega} := \int_0^{\infty} dt \omega(t) t^n.$$

- The range R and the width w is defined as

$$\begin{aligned} R[\Omega] &= \langle t \rangle_\omega = \int_0^\infty dt \omega(t) t \\ w^2[\Omega] &= \langle (t - R)^2 \rangle_\omega = \int_0^\infty dt \omega(t) (t - R)^2 \end{aligned}$$

- The aspect ratio $AR[\Omega]$ is a measure for the tail and is defined as

$$AR[\Omega] = \langle t^2 \rangle_\omega^{1/2} / \langle t \rangle_\omega.$$

A low AR means that the kernel of the operator has a short tail.

2.1.1 Exponential Moving Average (EMA)

The *exponential moving average* is a macroscopic operator and the simplest linear operator. Macroscopic operators do not care about small variations of the individual ticks, even adding or eliminating few ticks. So it take average behaviors of time series arguments. The EMA computes a moving average, but with an exponentially losing power of the past. It has a exponentially decay kernel

$$\text{ema}(t) = \frac{e^{-t/\tau}}{\tau}.$$

The advantage of this operator is, that its computation is very efficient and more complex operators can built with it, like moving averages, differentials, derivatives and volatilities. The form of the kernel leads to an iterative formula

$$\text{EMA}[\tau; z](t_n) = \mu \text{EMA}[\tau; z](t_{n-1}) + (\nu - \mu) z_{n-1} + (1 - \nu) z_n \quad (2.1)$$

with

$$\begin{aligned} \mu &= e^{-\alpha} \\ \alpha &= \frac{t_n - t_{n-1}}{\tau} \end{aligned}$$

and ν depends on the chosen interpolation scheme,

$$\nu = \begin{cases} 1 & \text{previous point} \\ (1 - \mu)/\alpha & \text{linear interpolation} \\ \mu & \text{next point} \end{cases} . \quad (2.2)$$

Unfortunately, this iterative formula is not computed in practice.

2.1.2 The Iterated EMA Operator

The above basic EMA operator can be iterated to build a family of iterated exponential moving average operators $\text{EMA}[\tau, n; \cdot]$. The iterated EMA is defined as a simple recursion

$$\text{EMA}[\tau, n; z] = \text{EMA}[\tau, n - 1; z] \quad (2.3)$$

with $\text{EMA}[\tau, 1; z] = \text{EMA}[\tau; z]$. The kernel is

$$\text{ema}[\tau, n](t) = \frac{1}{(n-1)!} \left(\frac{t}{\tau}\right)^{n-1} \frac{e^{-t/\tau}}{\tau}.$$

This family of functions is related to the Laguerre polynomials, which are orthogonal with respect to the measure e^{-t} (for $\tau = 1$). Any kernel can be written as a sum of iterated EMA kernels, through an expansion of the Laguerre polynomials. High-order iterated EMA's could be necessary, because the expansion has a slow convergence. This happens if one constructs an operator with a faster decay than exponential. For the iterated EMA are

$$\begin{aligned} \text{Range} & R = n\tau, \\ 2^{\text{nd}} \text{ moment} & \langle t^2 \rangle = n(n+1)\tau^2, \\ \text{width} & w^2 = n\tau^2, \text{ and} \\ \text{aspect ratio} & AR = \sqrt{(n+1)/n}. \end{aligned}$$

The iterated $\text{EMA}[\tau, n]$ operator has a shorter, more compact kernel than a simple EMA, if we assume a large n and the same range $n\tau$. This comes from the AR, which converges toward 1 for large n . Every basic EMA, which is a part of the iterated EMA, has a smaller range than the full kernel ($\tau < n\tau$). But even the tail is exponential, it decreases faster due to the small basic EMA range τ .

2.2 ϑ - time scale

The time in time series analysis plays an important role. In this section and the following section, the definitions are only a brief overview, so many (perhaps important things) are missing. But a nearer examination would lead to a very very long paper.

Looking at the financial markets for a whole week, no data are evaluated at weekends or on holidays. There is a lag in our continuous data. So a new time scale must be found, the ϑ -scale. A simple new one would be, if the weekends are not watched, that means the time from Friday, 11 p.m. GMT to Monday 8 a.m. GMT are cut off from our time scale.

But now to the ϑ -scale. It models the seasonal, intradaily and intraweekly aspect of heteroscedasticity. ϑ can be defined as

$$\vartheta = \vartheta(t) := \int_{t_0}^t a(t') dt' = a_0(t - t_0) + \sum_{k=1}^3 \vartheta_k(t) \quad (2.4)$$

with the activity a defined as

$$a(t) = \sum_{k=1}^3 [a_{0,k} - a_{1,k}(t)] = a_0 + \sum_{k=1}^3 a_{1,k}(t) > 0, \quad (2.5)$$

where a_0 is the base level and $a_{1,k}$ is the activity during the opening hours, which is modelled with a polynomial with smooth transition to the constant behavior of the closing time. So the activity $a_{1,k}$ can be written as

$$a_{1,k}(t) = \begin{cases} 0 & T_k < o_k \text{ or } T_k > c_k \text{ or weekend,} \\ a_{open,k}(t) & o_k < T_k < c_k, \end{cases}$$

with o_k the opening and c_k the closing hours. The only thing is unknown yet, is why the sum goes from one to three. There exists three generic markets in the world: East Asia, Europe and America. The datas can received from the opening of the East Asian market until the closing of the American market, well nearly 24 hours a day.

ϑ_k is the business time scale of the k^{th} market and can be defined as

$$\vartheta_k(t) = \int_{t_0}^t a_{1,k}(t') dt'. \quad (2.6)$$

The advantage of this time scale is, that periods with high activity are expanded and periods with low activity are contracted.

2.3 Intrinsic time scale

Another time deformation is the intrinsic time. The definitions here are only a brief overview, too. The intrinsic time can be defined as

$$\tau(t_c) := \tau(t_{c-1}) + k \frac{\vartheta(t_c) - \vartheta(t_{c-1})}{\Delta\vartheta} \frac{|\Delta x|^E}{c}, \quad (2.7)$$

where t_c is the current time, the price difference Δx is taken on the same interval as $\Delta\vartheta$. The constants E and c are the scaling law inverse exponent and factor. The constant factor k is a calibration depending on the particular time series, therefore it keeps τ in line with the physical time in the long run.

The τ -scale neither use the physical time t nor need to have fundamental informations about the behavior of the series. It is only defined with the values of the time series itself. That is the reason why it is called intrinsic time. The consequence of the τ -scale is that periods of high volatility are expanded and periods of low volatility are contracted, so the relative importance of events to the market are emphasized. With this time scale a forecasting model has a dynamic memory of the price history. In contrary to the ϑ -scale τ is only known for the past. An disadvantage of this time scale is that it is necessary to combine two forecasting models, for a price forecasting model. The first model is for the intrinsic time and the second one for the price (see Section 2.2).

3 Forecasting return

3.1 A simple forecasting method

There are many ways to get the real size of returns. One popular way is, using the realized or historical volatility to achieve the returns. The historical volatility is defined as

$$v(t_i) = \left[\frac{1}{n} \sum_{j=1}^n |r(\Delta t; t_{i-n+j})|^p \right]^{1/p} \quad (3.1)$$

with the return

$$r(t_i) = r(\Delta t; t_i) = x(t_i) - x(t_i - \Delta t)$$

and n the number of return observations, the size of the total sample $n\Delta t$, p is often set to 2, so we have the v^2 as the variance of the returns about zero. The only limitation to p is $p > 0$, but fractions are also possible. The disadvantage is that the historical volatility of the last 10 minutes is hardly the same of the following 10 minutes. So the absolute size of future returns is not correct.

There exists also another kind of return, the overlapping return,

$$r_i = r(t_i) = x(t_i) - x(t_i - m\Delta t) = x_i - x_{i-m},$$

with $m \in \mathbb{N}$. On this way the number of observations can be increased.

Another a approach to get the return is the way over the Black-Scholes-Formula. Here the price of an option is a function depending of the volatility.

$$C(S, t) = SN(d_1) - Ee^{-r(T-t)}N(d_2), \quad (3.2)$$

with the price C of a call option, T is the expiry date, E the expiration price and $N(\cdot)$ is the density of the standard normal distribution. The parameters $d_{1,2}$ are defined as

$$\begin{aligned} d_1 &= \frac{\ln(S/E) + (r + \sigma^2/2)(T - t)}{\sigma\sqrt{T - t}}, \\ d_2 &= d_1 - \sigma\sqrt{T - t}. \end{aligned}$$

Every parameter, expect of the volatility is given/observable, so we can calculate the volatility with the observed market data. This volatility is the expected (implicit) volatility of the market participants. But there is a little error, so we cannot use this approach, even when some persons do it. If we take the volatility from the Black-Scholes-Formula, we must not forget the constraints in the model: a constant volatility and interest rate, log-normal density for the stock prices and a brownian motion. There is an escape from this error, using a expanded Black-Scholes-Formula, which is solved by a BSDE (Backward Stochastic Differential Equation). So the implicit volatility, we get here, can be used.

With these two kinds of volatilities we can reformulate the equations to receive our return r_i .

3.2 Multi-horizon forecast of returns

This section examines again a forecast model for return, but now for a multiple horizon. The model supports several forecast intervals. Hourly returns are forecasted as well as daily, weekly, monthly, and even quarterly returns.

For building such a model, the physical time has to be substituted through the intrinsic time (see Section 2.3). The model is based on nonlinear indicators, which are modelled with moving averages (see Section 2.1.1).

Definition 3.1 *A indicator for market prices come from simple trading systems, like the technical analysis. There the indicator (function) gives a signal to buy or to sell. The crossing of a special level (i.e. the 200 days line) upwards gives a buy signal and downwards a sell signal. So the indicators can be used as a predictor of a variable or of its changes.*

An ideal indicator would lead to a good price forecast, but on the one hand we do not know whether such an indicator exists at all, on the other hand we have no ideal indicators. So it is necessary to combine different indicators to optimize their respective influences. Here, a linear combination of the price indicators $(z_{xj})_j$ is used. The weights are estimated by a multiple linear regression. The forecast of the price \tilde{x}_f for a fixed forecast horizon $\Delta\vartheta_f$ is defined as

$$\tilde{x}_f = x_c + \sum_{j=1}^m c_{x,j}(\Delta\vartheta_f) z_{x,j}(\Delta\tilde{\vartheta}_f, \tau_c), \quad (3.3)$$

where x_c is the current price, m the number of indicators (from two to five per horizon) (see [3]). The coefficients $c_{x,j}(\Delta\vartheta_f)$ are the weights of the indicators, which are estimated in intrinsic time scale.

$\Delta\tilde{\vartheta}_f$ in (3.3) is the forecasting horizon expressed in intrinsic time. For defining this variable, a new forecasting model for $\Delta\tilde{\vartheta}_f$ has to be build. This model is similar to (3.3), so $\Delta\tilde{\vartheta}_f$ can be written as

$$\Delta\tilde{\vartheta}_f := \tilde{\vartheta}_j - \vartheta_c = \sum_{j=1}^m c_{\tau,j}(\Delta\vartheta_f) z_{\tau,j}(\Delta\vartheta_f, \vartheta_c), \quad (3.4)$$

where the model is now computed in ϑ - scale (see Section 2.2). The coefficients $c_{\tau,j}(\Delta\vartheta_f)$ are estimated through a multiple linear regression, too, and $z_{\tau,j}(\Delta\vartheta_f, \vartheta_c)$ are the intrinsic time indicators. Now $\Delta\tilde{\vartheta}_f$ in (3.3) can be substitute by (3.4).

The described model does not depend on a fixed basic time interval like the most traditional forecasting models. The time points when a price is recorded in the database is heterogeneous spaced. The use of the intrinsic time suggest that the

forecasting models have to be computed simultaneously over several fixed time horizons $\Delta\vartheta_f$.

The time horizon $\Delta\vartheta_f$ and the intrinsic time τ_c can be computed with (2.4) and (2.7), when a forecasting horizon in physical time Δt_f and the price history until x_c is given.

There are still two variables, which are not yet clearly defined, $z_{x,j}(\cdot)$, $z_{\tau,j}(\cdot)$. These two indicators are based on moving averages. Here, we work with the exponential moving average (EMA) (see Section 2.1), because they can suitable written as recursion formula. With the help of the momentum indicator, the two above indicators can be defined.

Definition 3.2 *The momentum indicator is defined as*

$$m_x(\Delta\vartheta_c, \tau_c) := x_c - \text{EMA}[\Delta\tau_r; x](\tau_c). \quad (3.5)$$

It compares the latest price with its own exponential moving average. The momentum is computed by the recursion formula (2.1) with intrinsic times as time scale. There exists also first and second momenta. The first momenta defines the difference of two EMA momenta with different ranges and can be considered as the first derivative of $x(\tau)$. The second momenta is a linear combination of three EMA momenta with different ranges, which gives information on the curvature of the series over a certain past history.

Although there are many kinds of technical indicators, momenta indicators are widely used in technical systems. Perhaps, because they are simple to compute. Here, we only use these indicators. With the above definition the indicator for returns $z_x(\Delta\tau_r, \tau_c)$ with a range $\Delta\tau_r$ can be defined as

$$z_x(\Delta\tau_r, \tau_c) = \left[\frac{m_x^{(o)}(\Delta\tau_r, \tau_c)}{\sqrt{1 + (m_x^{(o)}(\Delta\tau_r, \tau_c)/m_{max})^2}} \right]^p, \quad (3.6)$$

where $m_x^{(o)}(\Delta\tau_r, \tau_c)$ is the normalized momenta of order o of returns, m_{max}^p is the maximum value the indicator can achieve and the exponent p is the accentuator of the indicator movements. For saving the sign of the moving averages, the power p must be an odd number for price indicators.

The definition (3.6) can easily used for other indicators, like our second one, $z_\tau(\Delta\vartheta_f, \vartheta_c)$, for the ϑ -time, where the parameters are now defined as function of τ_c and are computed with (2.7) to the ϑ -time. The function $\tau(\vartheta)$ is a monotonic positive definite function, so that all of its momenta are positive. This solution is obvious, because time can never flow backward.

When this algorithm is implemented, the indicators are continuously updated. That means, every new price leads to a recomputation of all its indicators. By

this way the indicators and coefficients are continuously updated. For updating the the coefficients $(c_{x,j}, c_{\tau,j})$, the model has to be estimated for the past history, where the length of the past history depends on the forecasting interval. This length varies from a few month for hourly forecasts till a few years for 3-month forecasts. In our model there is a problem: the indicators and the returns are from the same databases, so database errors, like missing data, badly filtered data, . . . , can occur. But this disadvantage can be eliminated by using $\hat{z}_{i,j} = z_{i,j} + \varepsilon_{i,j}$ instead of only $z_{i,j}$, where $\varepsilon_{i,j}$ is a random error with variance ρ^2 times that of z_i .

In our point of view this is a big disadvantage. When you have to compute all your indicators after every tick, it is a too big effort. For every new computation the probability for an error in measuring is increasing. A small error can lead to a huge error, because the computer uses always approximated results (e.g. only 10 digits after the comma). So the user has to find an optimal way of recomputation and using the old indicators.

4 Forecasting volatility

Before we can take a look at the forecast models, we have to define what a volatility is.

Definition 4.1 *The volatility v^2 is the average change of the return, or in other words it is the risk of holding a position in a financial asset. Mathematically, the volatility of return is equal to the square of the standard deviation σ :*

$$v = \sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (r_i - \bar{r})^2}, \quad (4.1)$$

with n as the size of sample and \bar{r} as the average of the returns

$$\bar{r} = \frac{1}{n} \sum_{i=1}^n r_i.$$

The equation (4.1) is like the realized volatility in (3.1) with exponent $p = 2$.

There are three types of volatility:

- Historical or realized volatility: it is determined by the past returns.
- Model volatility: it is a virtual variable in theoretical models like GARCH or stochastic volatility (but there may be means to estimate this variable from the data). (See also Section 4.1)
- Implicit volatility can be calculated from the market prices of derivatives like options. This is based on a model of an underlying stochastic process, like a standard geometric Brownian motion (see [1]).

4.1 Overview of volatility models

In this section a few kinds of forecasting models are explained. With these models it is possible to estimate/forecast the volatility. A standard approach for modelling the volatility is the statistical process

$$r_t = \sigma_t \varepsilon_t, \quad (4.2)$$

where the return r_t is equally spaced and ε is an independent identical distributed (i.i.d.) random variable. The volatility σ_t is the square root of the return r_t . There are many kinds of modelling the volatility in (4.2), but in this section we focus only three types of volatility models.

- *ARCH-type models*: ARCH means Autoregressive Conditional Heteroscedasticity. At these types, the volatility is a function of past returns. Naturally the model can be expanded, i.e. in the *GARCH process*. There σ_t depends on its own past values, e.g. the volatility with a GARCH(1,1) process is defined as

$$\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2, \quad (4.3)$$

where ε_t is the GARCH-process with

$$\begin{aligned} E[\varepsilon_t | F_{t-1}] &= 0, \\ \text{Var}[\varepsilon_t | F_{t-1}] &= \sigma_t^2 \quad \text{and } Z_t = \varepsilon_t / \sigma_t \text{ is i.i.v.} \quad (\text{strong GARCH}), \\ \text{Var}[\varepsilon_t | F_{t-1}] &= \sigma_t^2 \quad (\text{semi-strong GARCH}), \\ \mathcal{P}(\varepsilon_t | 1, \varepsilon_{t-1}, \varepsilon_{t-2}, \dots, \varepsilon_{t-1}^2, \varepsilon_{t-2}^2, \dots) &= \sigma_t^2 \quad (\text{weak GARCH}), \end{aligned}$$

(for further informations see [4]). The volatility σ_t can not observed directly, but (4.3) can written as a function of past returns, too. Hence (4.3) can be computed if a sufficiently large series of past return values is know. The requirement of the existence of a large series is a more or less big problem. The obtaining of data with a good quality is in some cases very difficult and/or expensive. For a bank or insurance company there will be no problems, but today there are many private investors, who want to forecast, for them - we suppose- there could be some problems.

Another variation is the *HARCH type*. Here stands the H for Heterogeneous. The parameter σ_t^2 is now defined as

$$\sigma_t^2 = c_0 + \sum_{j=1}^n c_j \left(\sum_{i=1}^j r_{t-i} \right)^2, \quad (4.4)$$

where $c_0 > 0$, $c_n > 0$, $c_j \geq 0$ for $j = 1, \dots, n-1$. I.e. the HARCH(2) would be written as

$$\sigma_t^2 = c_0 + c_1 r_{t-1}^2 + c_2 (r_{t-1} + r_{t-2})^2. \quad (4.5)$$

The parameters c_i , $i = 0, \dots, n$ and α_0 , α_1 , and β_1 can be estimated (i.e. with a maximum Likelihood approach).

- *Stochastic volatility models:* In stochastic volatility models the volatility depends on its own past values. A problem exists in this model, the volatility variable σ_t is neither observable nor directly computable from the past.
- *Models based on realized volatility:* Instead of modelling σ_t directly, this models defines σ_t as the realized volatility computed at $t - 1$. The realized volatility should be computed with high frequency to keep the stochastic error low, but then there are some disadvantages:
 - The realized volatility is biased, if it is computed with high frequency. But by a correction factor this disadvantage can be eliminated.
 - The fine volatility (high frequency) lags behind the coarse volatility (low frequency) in the lead-lag analysis. This leads to a worse forecast quality when predicting the volatility of the next step of the model.
 - The realized volatility is not the best predictor of the volatility.

For modelling external influences of the volatility σ_t the most stochastic processes can be expanded by adding some terms.

4.1.1 Parameter Estimation of GARCH Models

Like in Section 4.1 states, a GARCH(1,1) process for the volatility is defined as

$$\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2, \quad (4.6)$$

with σ_t^2 as the conditional variance and ε_t^2 as the squared innovations.

For the estimation of the parameters of the process, we have to build the log-likelihood function. If we assume that the innovations ε_t are normally distributed, we get

$$\mathcal{L}(\theta) = -\frac{n}{2} \ln(2\pi) - \frac{1}{2} \sum_{i=1}^n \left(\ln(\sigma_i^2) + \frac{\varepsilon_i^2}{\sigma_i^2} \right). \quad (4.7)$$

Another assumption for ε_t is a Student-t distribution, then we get

$$\begin{aligned} \mathcal{L}(\theta) = & \frac{n}{2} \left[\ln(v-2) + 2 \ln[\pi^{1/2} \Gamma\left(\frac{v}{2}\right)] - \ln \Gamma\left(\frac{v+1}{2}\right) \right] \\ & - \frac{1}{2} \sum_{i=1}^n \left[\ln(\sigma_i^2) + (v+1) \ln \left(1 + \frac{\varepsilon_i^2}{\sigma_i^2(v-2)} \right) \right], \end{aligned} \quad (4.8)$$

where v is the number of degrees of freedom. In both ways, we have to maximize the function to get our parameters,

$$\max_{\theta} \mathcal{L}(\theta).$$

The solutions of the above problem is obvious, so we leave to solve this the reader as an exercise.

The process (4.6) is estimated for several frequencies to test the effects of the temporal heterogeneity. The frequencies go from daily to 10-minutes intervals. At the high frequency (4.2) must be expanded by a fourth-order autoregressive term $\mu_t = \sum_{i=1}^4 \phi_i r_{t-i}$ to account for the statistically significant autocorrelation of the returns.

4.1.2 The HARCH Model

In this section we want to take a short look at HARCH models and say something about the coefficients. The HARCH(2) model is defined in (4.4) as

$$\sigma_t^2 = c_0 + \sum_{j=1}^n c_j \left(\sum_{i=1}^j r_{t-i} \right)^2, \quad (4.9)$$

where

$$c_0 > 0, \quad c_n > 0, \quad c_j > 0 \quad \text{for } j = 1, \dots, n-1.$$

Equation (4.9) can be written as

$$\begin{aligned} \sigma_t^2 &= c_0 + c_1 r_{t-1}^2 + c_2 (r_{t-1} + r_{t-2})^2 \\ &= c_0 + (c_1 + c_2) r_{t-1}^2 + c_2 r_{t-2}^2 + 2c_2 r_{t-1} r_{t-2}, \end{aligned} \quad (4.10)$$

this is nearly the form of an ARCH(2) process if we forget the last term.

When we take the expectation of r_t^2 of (4.2) and compare it to the expectation of (4.9), we get

$$E(r_t^2) = E(\sigma^2) = c_0 + \sum_{j=1}^n \left(\sum_{i=1}^j E(r_{t-i}^2) \right). \quad (4.11)$$

The cross products $r_{t-1} r_{t-2}$ in (4.10) have an expectation of zero, so

$$E(r_t^2) = E(r_{t-1}^2), \quad t \geq 1.$$

With equation (4.11) we get

$$E(r_t^2) = \frac{c_0}{1 - \sum_{j=1}^n j c_j}, \quad (4.12)$$

and so we have found a limit for the coefficients

$$\sum_{j=1}^n j c_j < 1.$$

4.2 Volatility for value-at-risk: Three simple models

In this section we regard the volatility as a variable that determines the risk. So, we insert the volatility in a simple model to compute the *value-at-risk*.

Definition 4.2 *The value-at-risk (VaR) is the expected loss of a portfolio after one business day corresponding to the 1% quantile ¹. This means the loss, which is worse than 99% of the expected cases and better than the remaining 1%.*

VaR is only one way to measure the risk. Another way is through an extreme value analysis, but here it is easier to regard the VaR. In addition, the value-at-risk is a popular view of risk assessment methods. Here, we discuss only the univariate forecasting models. The volatility, we regard here, is naturally only a forecast from “now” to “now plus one business day”.

The three volatility forecast models are

- (1) the volatility forecast of RiskmetricsTM develops by J.P. Morgan 1996 as a well-known example, (see [6])
- (2) an improved version based on tick-by-tick data, and
- (3) a further improved multi-horizon version.

Each of these models are observations of volatilities in the past. However, these volatilities should be valid for the future. The computed values of the volatility are estimations of the future volatility. We can use the past volatility as an estimation of the future, if we assume autoregressive heteroscedasticity.

- (1) The RiskMetrics uses a simple volatility forecast based on an IGARCH process with the following conditional expectation of the squared return:

$$\sigma^2(t) = \mu \sigma^2(t - \Delta t) + (1 - \mu)[x(t) - x(t - \Delta t)]^2 \quad (4.13)$$

with $\mu = 0.94$. This formula is computed only once per business day, that means that the volatility value is valid till a new one is computed one business day later. The time scale t is business time scale, extracting weekends, with $\Delta t = 1$ business day. When we take a closer look on (4.13), we see an exponential moving average (EMA) iteration (see Section 2.1 for more informations). So we can write (4.13) as

$$\sigma^2(t) = \text{EMA} [\tau; [x(t) - x(t - \Delta t)]^2]. \quad (4.14)$$

Equation (4.14) is evaluated at discrete time points separated by $\Delta t = 1$ business day, with EMA range $\tau = \mu/(1-\mu) = 15.67$ business days. Because of the discrete

¹1% is just a number. The user can take every quantile, which reflects his risk.

time points, (2.2) has to be displaced by a version for discrete, homogeneous time series

$$\mu = \nu = \frac{1}{1 + \alpha} = \frac{\tau}{\tau + 1}, \quad (4.15)$$

as explained in [7, 3]. The parameter μ has to be chosen to optimize the volatility forecasting quality in (4.13). J.P. Morgan did this in 1996 over a wide range of financial assets and test periods.

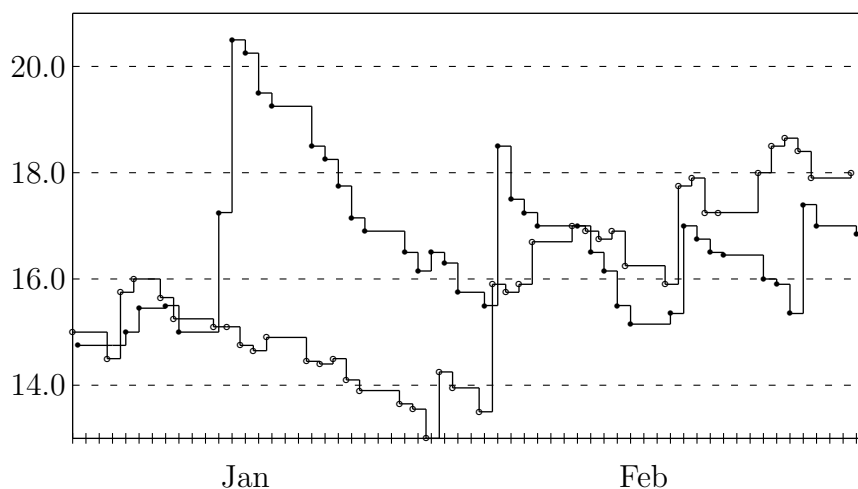


Figure 4.1: Differences in the annualized volatility. Data from [3, Figure 9.1, p. 252].

In Figure 4.1 we can see two volatilities. The differences between the two curves result from the choice of the daytime, when the price x is sampled and the volatility is computed like in (4.13) or (4.14). One curve is sampled at 7 a.m. GMT (Greenwich Mean Time), a suitable time for Asian risk managers, the other curve at 5 p.m. GMT, a suitable time for European risk managers. Against the expectations in the model the differences are very large, up to 25%, that is not only for risk manager an alarming uncertainty. From these different volatilities the two risk manager adopt different risk levels for the same financial asset, just because they live in different time zones. How we can see, the difference can exist over weeks. Figure 4.1 is just an example. But the same unexpectedly strong effects can also exist for other financial assets, sampling periods, choices of daytime, or process equations.

Both volatilities cannot be right at the same time, so there must be an error in these values. This error is of stochastic nature, because there exists no systematic bias. The difference is neither positive nor negative. Figure 4.1 shows a large stochastic error in the RiskMetric method. There are two reasons for this large error:

- The range of the kernel of about 16 business days is rather small. The number of independent observations is limited. On the other hand the choice of a short range is motivated by the goal of fast adaptivity to new market events, so we cannot essentially change this fact.
- We have only one observation per day, taken at a certain time. All the other informations on the prices during the day are eliminated. A single value per day has not the full representation for a full day. The observed price could be the bottom of a short-lived local canyon.

(2) The second model is developed by Müller (see [9]). It follows the first model as closely as possible with two innovations:

- The squared volatility is computed at every tick, not only once per business day.
- The simple returns are replaced by operator-based, smoothed returns.

Nothing else is changed; the sampling range is still 15.67 business days and the business-daily nature of (smoothed) returns are preserved. We write again the formula with help of time series operator:

$$\sigma^2 = c \text{EMA} [\tau; (x - \text{EMA}[\Delta t, 4; x])^2], \quad (4.16)$$

with $\Delta t = 1$ business day and $\tau = 15.67$ business days. (4.16) is now computed iteratively tick by tick. The iterated operator $\text{EMA}[\tau, 4; x]$ is defined by in Section 2.1.1. We use smoothed returns, $x - \text{EMA}[\Delta t, 4; x]$, instead of the simple returns, $x(t) - x(t - \Delta t)$, so we need a compensation, c . If x follows a Gaussian random walk, c has the (theoretical) correct value of $c = 128/93$. With this factor a systematic bias of the tick by tick volatility as compared to the RiskMetrics volatility is eliminated.

(4.16) is computed on a special business time scale. In our first model the weekends are omitted. Here the time from Friday 8 p.m. GMT to Sunday 9 p.m. GMT is compressed to the equivalent of only 1 hour outside the weekend. The advantage of this model is, that the volatility function is now a continuous function of time. In opposite to the first model the curve is differentiable. Taking the example of our risk managers in Europe and Asia from the RiskMetrics model, we recognize that both measure the risk of their financial assets on the same basis.

Although the range is in the both models the same, a tick-by-tick forecast has some more advantages: the smoothed instead of the simple returns and the overlapping returns lead to a reduced stochastic noise of volatility measures.

(3) The third model is a multiple-horizon version of the second one:

$$\sigma^2 = \frac{\sum_{k=0}^{n-1} f_w^k \sigma_k^2}{\sum_{k=0}^{n-1} f_w^k}, \quad (4.17)$$

with

$$\sigma_k^2 = c\text{EMA} \left[\tau_0, f_\tau^k; \left(x - \text{EMA}[\Delta t_0 f_{\Delta t}^k; 4; x] \right)^2 \right].$$

The weights f_w^k of the partial volatility forecast, their return intervals $\Delta t_0 f_{\Delta t}^k$, and their sampling ranges $\tau_0 f_\tau^k$ are in geometric sequences and can be flexibly chosen and optimized by setting the parameters n (number of partial forecast), f_w , Δt_0 , $f_{\Delta t}$, τ_0 , and f_τ .

This third model shares the same advantages as the second model and has the additional multiple-horizon property, which leads to superior volatility forecast quality.

5 Measure for the quality

Here we want to discuss the results from the sections above and find some ways to measure the quality of our forecast models. Measuring the quality of forecasts is done mostly by statistical tests. There the forecast is compared with the real or actual value of the target variable.

In Section 3 we see that the forecasting of the return brings difficulties, like

- exactness: the high frequency and the high number of computation can lead to errors,
- model constraints: the implicit volatility from the Black-Scholes Formula cannot be used. We have to solve a BSDE, which is more difficult than use known results.
- changing volatilities: if we calculate the return from the volatility, then we assume, that the volatility of the last ten minutes is the same as for the next ten minutes.

Intuitively we can compare some achieved results:

the tick-by-tick model of (4.16) has a better volatility than the model of (4.14), that is obvious, because measuring tick-by-tick is better than once a day. But it is not as good as the model of (4.17).

Measuring the quality of volatility brings two "difficulties". The first one, the bias of the realized and the forecasted volatility, if the return interval is too small. The second one, there are many quality measure to choose from.

5.1 Quality measures for volatility

Let us assume we have some forecast models, so we want to know which one is better. We have to define two variables.

As the forecasting signal we use the quantities

$$s_f = \tilde{v}_{f,t} - v_{h,t}, \quad (5.1)$$

where \tilde{v}_f is the volatility of any forecast model and v_h is the *realized hourly volatility*, defined as

$$v_{h,t} = v_h(t) = \sum_{i=1}^{a_h} r_{t-i}^2,$$

with a_h as the aggregation factor. The other variable is the *realized signal*,

$$s_r = v_{h,t+1} - v_{h,t}. \quad (5.2)$$

In contrast to the volatility, s_f and s_r can take positive and negative values, so we have to regard the direction. One measure for the direction quality is

$$Q_d = \frac{\mathcal{N}(\{\tilde{v}_f | s_f \cdot s_r > 0\})}{\mathcal{N}(\{\tilde{v}_f | s_f \cdot s_r \neq 0\})}, \quad (5.3)$$

where \mathcal{N} gives the number of elements in a set. Another measure is the realized potential, defined as

$$Q_r = \frac{\sum \text{sign}(s_f \cdot s_r) |s_r|}{\sum |s_r|}. \quad (5.4)$$

With Q_r we do not only measure the direction quality like in (5.3), we combine the direction quality with the size of movements. Q_r and Q_f are not independent and Q_r is a weighted average of $\text{sign}(s_f \cdot s_r)$, $2Q_d - 1$ is the corresponding unweighted average. So it holds, if

$$Q_r > 2Q_d - 1 \quad (5.5)$$

and $|s_r|$ is large enough, the forecast of the sign of s_r is better than the average.

There exists another measure for comparing the absolute error of a model to a benchmark model.

$$Q_f = 1 - \frac{\sum |s_r - s_f^{\text{model}}|}{\sum |s_r - s_f^{\text{benchmark}}|} \quad (5.6)$$

increases when the performance of the model increases. So the model outperform the benchmark, if $Q_f > 0$ and if $Q_f < 0$ the benchmark outperform the model.

The above measures can written in the sense of Section 3.2, too. Equation (5.3) is now written as

$$D(\Delta t_f) = \frac{\mathcal{N}(\{\tilde{x}_f | (\tilde{x}_f - x_c)(x_f - x_c) > 0\})}{\mathcal{N}(\{\tilde{x}_f | (\tilde{x}_f - x_c)(x_f - x_c) \neq 0\})}, \quad (5.7)$$

where x_f , \tilde{x}_f and x_c are the same as in Section 3.2 defined. A similar measure as the definition of Q_r in (5.4) is the signal correlation

$$C(\Delta t_f) = \frac{\sum_{i=1}^{n'} (x_{f,i} - x_{c,i})(\tilde{x}_{f,i} - x_{c,i})}{\sqrt{\sum_{i=1}^{n'} (x_{f,i} - x_{c,i})^2 \sum_{i=i'}^n (\tilde{x}_{f,i} - x_{c,i})^2}}, \quad (5.8)$$

where n' is the number of possible measures in the full sample and n is the number of full forecasting horizons in the full sample and $i' = n - n'$.

The equations (5.7) and (5.8) are measure in the physical time Δt . It makes no sense to measure something in another time scale as the time the people look at the forecasts. Like in (5.3) and (5.4) the forecasts \tilde{x} must not be the same as the real value x . For this case, the definitions are not valid.

6 Neural Net

In this section we want to take a short look on a more or less new field in forecasting. Although *neural net* exists for several years (research began over 50 years ago) and nearly everybody know the expression, the neural net are still a mystery. In many areas of the scientific life neural nets are used, but everyone use it in his special way and no one says: Yes we use neural nets in this special way. Do you know that neural nets are nearly everywhere? Here are only a few examples:

- washing machines: control instrument,
- bombs detector: e.g. at the J-F-K Airport New York,
- medicine: estimating the probability of SID (Sudden Infant Death),
- economy: predicting stock prices,
- ...

In Section 6.1 we want to explain neural nets, and use this knowledge in Section 6.4 for an example. In Section 6.5 we will describe what a neural net can or cannot.

6.1 Definition

A synonym for (artificial) neural nets could be: a learning computer. In the last years the medicine and biology becomes better and better, so the human brain is explored and the neural net is nearly full understood.

An artificial neural net is the *copy* of the human brain with its neural net. This means, a artificial neural net should use the incoming informations and learn out of this informations like a human brain.

Now, we want to take a very short view on the human neural net to have a better base to compare the human and the artificial one. The human neural net is a net build of neurons. These neurons are divided in cell body and links with different lengths and forms. The short links with many branches are called dendrits, one link could be very long (human: over 1 meter) and is called axon. The income information (electric current) runs through the cell and the axon to the next denrit and cell. In the cells the information is used for a chemical process. That means: the information in the cell is transported to the next cell if and only if the electric potential is higher than a given value. The differences and similarities of a human neural net and an artificial neural net are shown in Figure 6.1. The human neural net grows by learning, that means after experiences new links are build. So the copy, the artificial neural net, should learn, too. Like the natural neural net, the artificial could have knots with many other neurons.

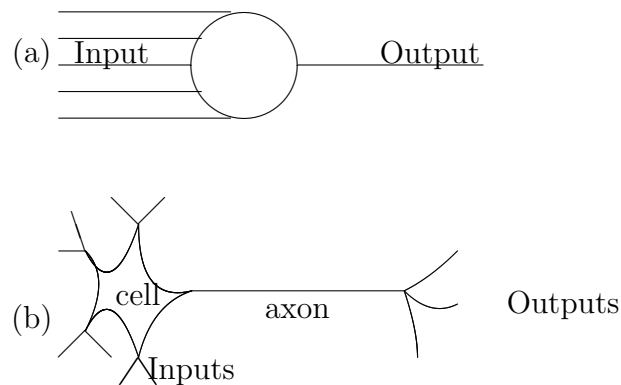


Figure 6.1: (a) The artificial neuron and (b) the natural neuron

A short example should make some things clearer. If a child see a pumpkin for the first time in its life, it do not know what this big, round and orange vegetable is. But after this experience, it knows. Now, a artificial neural net should act at the same way. If the user gives the net the informations: big, round, orange, vegetable, the artificial neural net should give the output: pumpkin.

In the following, we will concentrate on the artificial neural net and will use the notation neural net for the artificial one. The neural net is often build in many layers. That means, in the first layer(s) the information comes in and in the next (hidden) layer(s) the information is used, the last layer(s) is/are the output layer(s).

There are many different kinds of architectures for neural nets, but here we will only describe the *feedforward multilayer perceptron*, because this neural net is the ideal one for problems in the economy.

We assume that we fixed a special type of neural net architecture. Does the neural net know, what a pumpkin looks like? No, of course not, it does not yet learn. Like a child has to learn several things, the neural net must learn. This learning process is described in Section 6.3.

6.2 Feedforward Multilayer Perceptron

The multilayer perceptron is the most used neural net for economic problems. First we have to define the notation. *Perceptron* is developed by Rosenblatt in 1958. It has no hidden layer and only one output layer. That is the problem, the (one step) perceptron can only represent a few functions. With represent we mean the ability of a neural net to realize a given function. Many functions/problems are non-linear, that is the reason, why the *multilayer perceptron* was constructed. The multilayer perceptron is a multi-step perceptron. The topology of the per-

ceptron is like a multi-linear regression. In Figure 6.2 a multilayer perceptron is shown, where ξ_i and the constant 1 are the input informations o_{input} , $f(\cdot)$ the

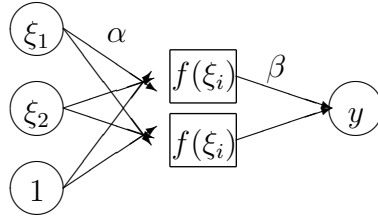


Figure 6.2: Multilayer Perceptron

activating function and y is the result in the output layer. α and β are just weights. The 1 in the lowest circle is just for comparing with the other inputs. The activating functions differ for the different kinds of nets and are in the most cases not linear. An assumption for activating function is that it is monotone increasing. The following table shows some popular activating functions: The

Heaviside Function	$\left \begin{array}{l} f(\xi_i) = \begin{cases} 1 & \text{if } \xi_i > 0 \\ 0 & \text{if } \xi_i \leq 0 \end{cases} \\ f(\xi_i) = \frac{1}{1+e^{-\xi_i}} \\ f(\xi_i) = \frac{e^{\xi_i} - e^{-\xi_i}}{e^{\xi_i} + e^{-\xi_i}} \\ f(\xi_i) = a \cdot \xi_i + b \end{array} \right.$
Logistic Function	
Tangens Hyperbolicus Function	
Linear Function	

Table 6.1: Table of activating functions.

activating function should decide, whether a signal is sent to the next layer or not. In terms of biology: the incoming signal (e.g. pain) is sent to the next cell, if it has a higher potential than a given limit (Heaviside Function). The logistic function is used for the multilayer perceptron in the most cases. This function function is called often "the" *sigmoid function*.

Now, the total process for a better understanding. The values of the input layer go multiplied by the weights α and added into the hidden layer. In the hidden layer, the activating function calculate out of the modified input values the output values ($o_{\text{output}} = f(\sum_i \alpha \xi_i)$). These values are multiplied by the weight β and go now into the output layer.

In the perceptron the output is the sum of the regressors and the constant multiplied with their parameters ($o_{\text{output}} = 1 \cdot \alpha_0 + \xi_1 \cdot \alpha_1 + \dots = \sum_{i=0}^n \alpha_i o_{\text{input}}$) and gives so the estimation of the endogen variable. The perceptron is nothing

new for economists. Every function (even not linear) can be written as

$$y = \alpha_0 + \sum_{i=1}^H \alpha_i f \left(\beta_{0i} + \sum_{j=1}^p \beta_{ji} x_j \right), \quad (6.1)$$

with H as the number of neurons in the hidden layer, p the number of input neurons and f the activating function.

6.3 Learning with Back-Propagation

We see in Section 6.1 that a neural net does not know a pumpkin at once. It has to learn. The learning is a kind of estimating the weights. To estimate we need three samples:

- training sample,
- validation sample,
- generalization sample.

The training sample is the data sample (In-Sample), we have to train with the net. This means, we give the net some pairs of information. A pair contain exogen and endogen variables, and the exogen try to declare the endogen. We give the net some input values x_i and the corresponding output values y_i , so the net can fixed its weights. The generalization sample is the sample to test the forecast quality of the net (Out-Of-Sample). If the training sample is forecasted much better than the generalization sample, then there could be two possibilities: there could exist a break in the structure of the data, or the more frequent case, the net has not only learnt, it has learnt by heart. This is called overlearning or overfitting. To interact against an overfitting, we have to use the validation sample. This sample contains data, which are not think for training or quality reasons. Overfitting can be seen, if the error in the training sample decrease, but in the validation sample increase. With a too small validation sample the overfitting can not recognize good enough, or too late. The using of the validation sample instead of the generalization sample means more work, but the generalization sample may only use for the quality aspects. To reduce the overfitting, we have to reduce the complexity of the topology.

Overfitting: If the number of hidden layers and the containing units in them can understand a more complex non-linearity than the data have, then we have an overfitting. In this case, the noise is learnt by the net, too. So not only the structure of the data relations is learnt, even the stochastic influence. Another reason for overfitting are, if the number of weights are larger than the number of observation, or if the training sample was used to long for the same net.

Like overfitting, there exists also an *underfitting*. In analogue to the overfitting case, an underfitting exists, if the number of hidden layers and the containing units in them are too small to explain the non-linearity of the time series.

A way out of this problem would be a splitting the full training sample in several smaller samples and a training in e.g. 200 nets (see [10]).

But now, we explain the *Back-Propagation*. As we see above, learning means estimating the weights. We send a training information x in the net and compare the output y with the wished output $L(x)$. The net has learnt enough, if the error $\delta = L(x) - y$ is small enough. This means the training sample is used more than one time till the error is small enough. In the back-propagation, the error δ is sent back through all layers (*error back-propagation*). In Figure 6.3 we see a scheme for the error back-propagation.

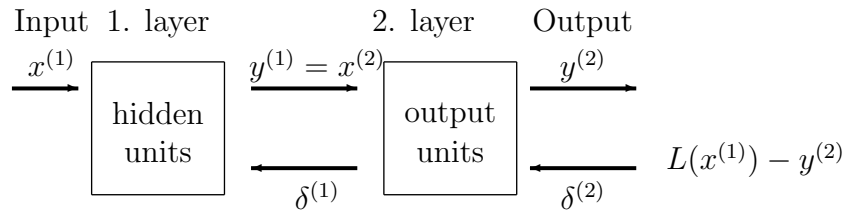


Figure 6.3: Scheme for the error back-propagation δ .

For learning we use an algorithm, that minimize the square of the error of all training patterns (see [2]). If we assume the function R has a minima depending by the weights w_i ², then the difference between $w(i-1)$ for the $(t-1)$ -th step to $w(t)$ for the t -th is proportional to the negative gradient,

$$(w(t) - (w(t-1))) \sim -\nabla_w R(w(t-1)). \quad (6.2)$$

so (6.2) can be written as

$$w(t) = w(t-1) - \gamma(t) \nabla_w R(w(t-1)). \quad (6.3)$$

The function R with the proportion factor $\gamma = \frac{1}{2}$ can now be written as

$$R = \sum_x R_x = \frac{1}{2} \sum_x (y(x) - L(x))^2. \quad (6.4)$$

The proportion factor is also called *learning rate*.

²We write w_i instead of β_i for a better understanding w like weights.

The gradient-algorithm in (6.3) for the weight w_{ij} from the neuron j to the neuron i is now

$$w_{ij}(t) = w_{ij}(t-1) - \gamma \sum_x \frac{\partial R_x}{\partial w_{ij}}. \quad (6.5)$$

For the difference of the single patterns x with $\Delta w_{ij} = w_{ij}(t) - w_{ij}(t-1)$ we write

$$\Delta w_{ij}(x) = -\gamma \frac{\partial R_x}{\partial w_{ij}} = -\gamma \frac{\partial R_x}{\partial z_i} \cdot \frac{\partial z_i}{\partial w_{ij}}, \quad (6.6)$$

with z_i as the i -th neuron. So the error δ is now

$$\delta_i = -\frac{\partial R_x}{\partial z_i} = -\frac{\partial R_x}{\partial y_i} \frac{\partial y_i}{\partial z_i} = -\frac{\partial R_x}{\partial y_i} S'(z_i), \quad (6.7)$$

with $S(z_i) = y_i$, and for third term in (6.6) we write

$$\frac{\partial z_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_k w_{ik} x_k = x_j. \quad (6.8)$$

So the total weight difference Δw_{ij} can be written as

$$\Delta w_{ij}(x) = \gamma \delta_i x_j. \quad (\text{Delta-Rule}) \quad (6.9)$$

We can see the output, and so the error for the neurons in the second step,

$$\frac{\partial R_x}{\partial y_i} = \frac{\partial}{\partial y_i} \frac{1}{2} (y_i - L_i)^2 = y_i - L_i.$$

But the errors of the other steps we do not see. For these we need a more complex delta. The error in i -th neuron in the first step, we can calculate recursive out of the error in the second step:

$$\frac{\partial R_x}{\partial y_i^{(1)}} = \sum_k \frac{\partial R_x}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial y_i^{(1)}} = -\sum_k \delta_k^{(2)} \frac{\partial z_k^{(2)}}{\partial y_i^{(1)}}. \quad (6.10)$$

With

$$\frac{\partial z_k^{(2)}}{\partial y_i^{(1)}} = \frac{\partial}{\partial y_i^{(1)}} \sum_j w_{kj}^{(2)} x_j^{(2)} = w_{ki}^{(2)}$$

and the equations (6.7) and (6.10) we can write the delta for the first step as

$$\delta_i^{(1)} = -\left(\frac{\partial R_x}{\partial y_i^{(1)}} \right) \left(\frac{\partial y_i^{(1)}}{\partial z_i^{(1)}} \right) = \left(\sum_{k=1}^m \delta_k^{(2)} w_{ki}^{(2)} \right) S'(z_i^{(1)}), \quad (6.11)$$

where m is the number of neurons in the first step. Like above, we can calculate every delta for the layers, the index (1) is replaced by $(n-1)$ and (2) by (n) :

$$\delta_i^{(n-1)} = \left(\sum_k \delta_k^{(n)} w_{ki}^{(n)} \right) S'(z_i^{(n-1)}). \quad (6.12)$$

With these equations the weights are calculated for every training pattern. After the last calculation the correct weights are put in the net with the equation (6.5). This way is called *Off-Line* version, because the weights are calculated without using the weights which are still calculated.

The new weights can be used immediately after calculating, if the equation (6.5) is replaced by

$$w_{ij}(t) = w_{ij}(t-1) - \gamma \frac{\partial R_x}{\partial w_{ij}}, \quad (6.13)$$

$$\begin{aligned} &= w_{ij}(t-1) - \gamma \delta x_j \\ &= w_{ij}(t-1) - \gamma (L(x) - y(x)) x_j \end{aligned} \quad (6.14)$$

In addition, the proportion rate γ could depend by t , e.g. $\gamma(t) = 1/t$. This version is called *On-Line* version. With a proportion factor that goes against 0, the learning algorithm converges.

In our case with only one hidden layer and two $y^{(1)}$, we get

$$y_1 = \alpha_1 f(w_{01} + w_{11}x_1 + w_{21}x_2) + \alpha_2 f(w_{02} + w_{12}x_1 + w_{22}x_2), \quad (6.15)$$

where α_i is for $y^{(i)}$, hence y_1 can be presented as a linear regression.

6.4 Example

In this section we want to explain the working list for a neural net. Here we take the example of forecasting the mean of the Dow-Jones-Index. The way is divided in 5 steps.

- (1) Defining the aim
- (2) Representation of the Informations
- (3) Defining the neural net
- (4) Training the net
- (5) Testing the net

(1) We want to forecast the Dow-Jones-Index. We use informations, we get from newspapers, like Consumer Price Index (CPI), price of oil, inflation rate and interest rate. In addition we use some informations, we do not know, whether they are relevant, like unemployment rate. If this information is useless, the net will learn to ignore it.

(2) The most informations we use, we can get from newspapers or from the internet. In Table 6.2 we see the informations and typical values. All values are

neuron	input	typical values
1	CPI of month of this month	200 - 300
2	oil price in USD of this month	15 - 30
3	inflation rate in % of this month	4 - 12
4	Dow changes of this month	-100 - 100
5	employment rate of this month	1 - 8
6	interest rate of this month	7 - 12
7	CPI of the last month	200 - 300
8	oil price of the last month	15 - 30
9	inflation rate of the last month	4 - 12
10	Dow changes of the last month	-100 - 100
11	employment rate of the last month	1 - 8
12	interest rate of the last month	7 - 12
13	CPI two month ago	200 - 300
14	oil price two month ago	15-30
15	inflation rate two month ago	4 - 12
16	Dow changes two month ago	-100 - 100
17	employment rate two month ago	1 - 8
18	interest rate two month ago	7 - 12
19	political climate of this month	1 - 10
20-31	current month	January, . . . , December

Table 6.2: Representation of Information

fictive. For the output we have only one neuron, the mean Dow changes of the next month. The current month we use not only one neuron with 12 different values, we use 12 neurons with the values 1 or 0. If we use 1 neuron, then the net would follow July (7) must be greater than June (6). Instead of regarding the mean Dow, we take the changes, because the mean is different for different times (today value is not the same as the value 1 year ago), on this way we can take the net for several years, too. The data must now be sorted for the months and for the repeating items (CPI this month, CPI last month, . . .).

(3) Our net use numeric input and output values. We define for every item of the input (total 31) one neuron, for the output one neuron, too (see Table 6.1). For the hidden unit we start with 15 neurons, this is half of the sum of input and output neurons. Every value is continuous, except the values for the months.

(4) Now we train our net with 120 facts (10 years times 12 months). We should give the informations to the net in a random order. If we assume we give the data in a chronological order, then the net learn e.g. the Dow goes up and up and up in a special time area, so the Dow goes always up (the net is thinking).

For the next few data the Dow goes always down, so the net renew his option to: always down. But if we give the data in a random order, the net do not make this mistake. On this way the net learn to generalize for the total data sample.

Another problem could happen: the net has problems with some training data (the net does not gives the more or less right output). We can try to add some more neurons, or take a closer look at these data. Perhaps we find a extraordinary fact with these few data, so we can forget them and train the net again. A third way to solve the problem is to accept that the net does not learn every fact, or to accept a lower exactness.

(5) Now we have to test the net with some other data. With these data we can measure the quality of our neural net. So we know how far the real values are away from our calculated values (bias).

In our case we have high-frequency data, or in other words: we have more than enough data. There would be no problem to split the data in three classes: training data, generalization and testing data.

6.5 Limits of the Neural net

Neural nets are proud for their abilities in identifying patterns. They are even better in identifying or classifying than a man. A neural net can recognize yet something, when the data are only partly known. So they can use for finding cancer cells in a picture analysis or identifying sex and kind of insects. In Table 6.3 some examples are shown, what a neural can ([5]).

ability	example
identify patterns	identifying submarines with sonar
generalization	valuating of houses
forecasting trends	stock prices
filter	cleaning optical signals
fast work	control of robotic arms
understand relations	medical expert systems
analyze big data amounts	correlation of insurance duties
extrapolation	diagnosis of production errors

Table 6.3: Some example for ability of neural nets.

An advantage in neural nets is that training for a new example is easier and faster than write a new program or algorithm for a control process. But one big disadvantage neural nets have. They are not proud for their exactness. If you ask a neural net, what is 2.01 plus 2.02, it will answer: about 4. So if the exactness is not very important, a neural net is the right choice.

7 Conclusion

In this paper we wanted to explain different models for forecasting with high frequency data. After the 6 sections, there is still a question: What is better the standard approach in the statistics (see Sections 3-4), or the new field neural nets?

The advantages and disadvantages of the different approaches we have still discussed. Hence we can compare the two models.

We know that stock prices depend not only of their past values, there are also other indicators, like gold price, CPI, . . . , and last on least the private informations. In addition, the different persons with their (sometimes irrational) behavior/trading strategies change the prices. If we have all the informations and a mechanism that explain a stock price, then the neural net is a easy and fast tool to predict stock prices. But then we can do it in the traditional way, too. ³ A perhaps funny fact is, our neural net has a degree of intelligence that is comparable with a cock-roach (see [2]).

Hence, we must decide whether the worse results of the neural nets are sufficient for our aim, or whether a standard approach with exacter results would be better! A neural net is faster, but the statistic is exacter. The effort in working with the two approaches is nearly the same, because we see above that a neural net can be written as a autoregressive model, hence the working with it has to be similar to the "normal" statistic approach. A neural net is perhaps a little bit easier, because the only thing to do is training the net. This could be the final point for neural net, it can be bought very cheap and the user is more or less easy.

The final question for the user stay: Do you want a fast and easy system, but a worse exactness or a system with more exactness?

³We said above, that a neural net is like a multi-linear regression.

References

- [1] Fisher Black, Myron Scholes: The pricing of option and corporate liabilities. *Journal of Political Economy*, 81, 637-659, 1973
- [2] Rüdinger Brause: Neuronale Netze, *B.G. Teubner, Stuttgart* 1991
- [3] Michel M. Dacorogna, Ramazan Gencay, Ulrich Müller, Richard B. Olsen, Olivier V. Pictet: An Introduction to High-Frequency Finance. *Academic Press, San Diego* 2001.
- [4] Jürgen Franke, Wolfgang Härle, Christian Hafner: Einführung in die Statistik der Finanzmärkte. *Universität Kaiserslautern; Humboldt-Universität Berlin; R&D Energy Markets, B-Louvain-la-Neuve* Mai 2001
- [5] Jeanette Lawrence: Neuronale Netze: Computersimulation biologischer Daten. *SYS-THEMA Verlag GmbH, München* 1992. Übersetzt von Marie-Cécile Bertau.
- [6] Morgan Guaranty: RiskMetricsTM-Technical Document, 4th edition, *Morgan Guaranty Trust Company of New York, New York* 1996
- [7] U.A. Müller: Specially weighted moving averages with repeated application of the EMA operator, *Internal document UAM. 1991-10-14, Olsen & Associates, Switzerland*, 1991
- [8] U.A. Müller: Generating a time series of fixed-period spot interest rates from interest rate futures. *Internal document UAM.1996-04-19, Olsen & Associates, Switzerland*, 1996
- [9] U.A. Müller, W.S. Chang, W.K. Li and Howell Tong, Eds.: Volatility Computed by Time Series Operators at High Frequency. In: *Statistics and Finance: An Interface, Imperial College Press, London*, 2000
- [10] Carsten Ochsen: Neuronale Netze in der volkswirtschaftlichen Prognose, Teil I: Zur Theorie Neuronaler Netze. In: *Volkswirtschaftliche Reihe, Institut für Volkswirtschaftslehre I, Nr. V-202-00, Universität Oldenburg*, 2000