PORTFOLIO MANAGEMENT AND MARKET RISK QUANTIFICATION USING NEURAL NETWORKS

Prof. Dr. Jürgen Franke
Department of Mathematics
Universität Kaiserslautern
Erwin-Schrödinger-Str.
67663 Kaiserslautern
Germany

Abstract

We discuss how neural networks may be used to estimate conditional means, variances and quantiles of financial time series nonparametrically. These estimates may be used to forecast, to derive trading rules and to measure market risk.

1 Introduction

Neural networks are now a well-established tool in financial engineering. The main applications, considered up to now, are to classification, forecasting and portfolio management, but also to option pricing (compare, e.g., Anders[1], Bol et al.[2] and Refenes et al.[11]). In this paper, we first introduce the basic concepts, relating them to nonlinear time series models. Then, we give a short review of asymptotic theory, including a study of an appropriate resampling method. To illustrate the potential of neural network based procedures in practice, we also discuss too realistic case studies from stock and FX markets.

In the last two sections, we propose procedures which allow to estimate conditional variances and quantiles of nonlinear time series using neural networks. These nonparametric approaches may be used to quantify the risk of financial assets either by estimating the conditional volatility or the conditional value-at-risk. The kind of information conditioned upon may be rather arbitrary and of a high-dimensional structure.

2 Nonlinear time series models based on neural networks

One of the well-known stylized facts about financial time series is their serial uncorrelatedness, i.e. the univariate data appear to be white noise. Hence, we expect only nonlinear predictors to show any reasonable performance, and, additionally, we should use in forecasting not only past observations of the time series of interest, but also other economic information from the past. For forecasting the time series S_t, we therefore consider as basic model a nonlinear $AR(\tau)$ - process with exogeneous components $X_t \in R^d$

$$S_{t+1} = m(S_t, S_{t-1}, ..., S_{t-\tau}, X_t) + \varepsilon_{t+1} \qquad (1)$$

The conditional expectation of the ε_t given information up to time t is 0. More specific assumptions on these innovations will be made later on. The d-variate exogeneous component X_t consists of values of other financial and economic time series up to time t. We do not assume a particular parametric form of the predictor function m which is the conditional expectation of S_{t+1} given $S_t, S_{t-1}, ..., S_{t-\tau}, X_t$. Therefore, we have to estimate it nonparametrically if we want to use it in forecasting S_{t+1}. As we have situations in mind where the autoregressive order $\tau+1$ and the dimension d are large, familiar smoothing methods like kernel estimators, discussed e.g. by Kreiss[9], are not applicable without assuming a particular, e.g. additive, structure of the function m on $R^{\tau+1+d}$. Neural networks offer an alternative class of estimators which are flexible and computationally feasible.

To keep the notation simple, we first give a short review of neural network function estimators in the context of a heteroscedastic regression model similar to the time series model (1):

$$Z_t = m(X_t) + \varepsilon_t \qquad (2)$$

where $X_1, X_2, \ldots$ are independent identically distributed with density $p(x), x \in R^d$, and the residuals $\varepsilon_1, \varepsilon_2, \ldots$ are independent with

$$\mathcal{E}\{\varepsilon_t | X_t = x\} = 0\,,\ \mathcal{E}\{\varepsilon_t^2 | X_t = x\} = \sigma_\varepsilon^2(x) < \infty.$$

We assume that the conditional mean $m(x)$ and the conditional variance $\sigma_\varepsilon^2(x)$ of Z_t given $X_t = x$ are continuous and bounded functions.

We want to estimate the function m on R^d using feedforward neural networks with one hidden layer. As the basic building block we consider the so-called *neuron* as a nonlinear transformation of a linear combination of the inputs $x = (x_1, ..., x_d)'$:

$$x \mapsto \psi(b + u_1 x_1 + \ldots u_d x_d)$$

ψ is a fixed *activation function*; in the following we always choose the centered sigmoid function

$$\psi(s) = \frac{2}{1+e^{-s}} - 1.$$

Combining H neurons, we get the *network function*

$$f_H(x, \vartheta) = v_0 + \sum_{h=1}^{H} v_h \psi(b_h + w_h' x)$$

where $\vartheta = (b_1, \ldots, b_H, w_1', \ldots, w_H', v_0, \ldots, v_H)'$ denotes the parameter vector consisting of the *network weights* with $w_h' = (w_{1h}, \ldots, w_{dh}),\ h = 1, \ldots, H$.

$f_H(x, \vartheta)$ specifies a mapping from the input space R^d to the output space which, in our case, is one-dimensional. Such network functions are universal approximators (Hornik et al.[7]), i.e. any regression function $m(x)$ may be approximated arbitrarily well using a large enough number H of neurons and appropriate parameters ϑ. In practice, feedforward networks with more than one hidden layer of neurons may provide a more parsimonious fit to m. As the theory and numerical practice is essentially the same for this more general case, we restrict our considerations here mainly to networks with only one hidden layer.

To estimate the conditional expectation $m(x) = \mathcal{E}\{Z_t | X_t = x\}$ from a sample $(X_1, Z_1), ..., (X_N, Z_N)$, we fix the number H of neurons and calculate the nonlinear least squares estimate $\widehat{\vartheta}_N$ of the parameter ϑ by solving

$$\widehat{D}_N(\vartheta) \equiv \frac{1}{N} \sum_{t=1}^{N} (Z_t - f_H(X_t, \vartheta))^2 = \min_{\vartheta \in \Theta_H} !$$

$\widehat{\vartheta}_N$ is consistent in the sense that $\widehat{\vartheta}_N \longrightarrow \vartheta_0$ for $N \to \infty$, where ϑ_0 is the parameter for which the given network provides the best approximation of m, i.e.

$$\mathcal{E}(m(X_t) - f_H(X_t, \vartheta))^2 = \min_{\vartheta \in \Theta_H} !$$

Under the above conditions, $\widehat{\vartheta}_N$ is asymptotically Gaussian:

Theorem: For $N \to \infty$,

$$\sqrt{N}(\hat{\vartheta}_N - \vartheta_0) \longrightarrow \mathcal{N}(0, \Sigma_1 + \Sigma_2).$$

with covariance matrices
$\Sigma_i = A(\vartheta_0)^{-1} B_i(\vartheta_0) A(\vartheta_0)^{-1}, i = 1, 2$, where $A(\vartheta) = \nabla^2 D_0(\vartheta)$,
$B_1(\vartheta) = 4 \cdot \int \sigma_\varepsilon^2(x) \nabla f_H(x, \vartheta) \nabla' f_H(x, \vartheta) p(x) dx$,
$B_2(\vartheta) = 4 \cdot \int (m(x) - f_H(x, \vartheta))^2 \nabla f_H(x, \vartheta) \nabla' f_H(x, \vartheta) p(x) dx$.

The second part Σ_2 of the asymptotic covariance matrix represents the effect of misspecification due to fitting a network function with given H to an arbitrary regression function m. In the correctly specified case, where $m(x) = f_H(x, \vartheta_0)$, we have $\Sigma_2 = 0$.

A simple proof of the theorem is given by Franke and Neumann[6]. A much more general result, which, under appropriate assumptions, also covers the time series model (1), has been given by White[13]. An immediate consequence of the theorem is

$$f_H(x, \widehat{\vartheta}_N) \longrightarrow f_H(x, \vartheta_0) \qquad \text{for } N \to \infty.$$

By the universal approximation property of neural networks, $f_H(x, \vartheta_0)$ converges to $m(x)$ for $H \to \infty$. Therefore, $f_H(x, \widehat{\vartheta}_N)$ should become a consistent nonparametric estimate of $m(x)$ if H increases with N with an appropriate rate. White[14] has proven a corresponding result. In practice, H is chosen by comparing the performance of the function estimators $f_H(x, \widehat{\vartheta}_N)$ for various H on a *validation set* of data, which has not been used in calculating the estimate $\widehat{\vartheta}_N$. Alternatively, one could use the *neural information criterion* of Murata et al.[10] which is a version of Akaike's AIC adapted to neural network based regression and autoregression models.

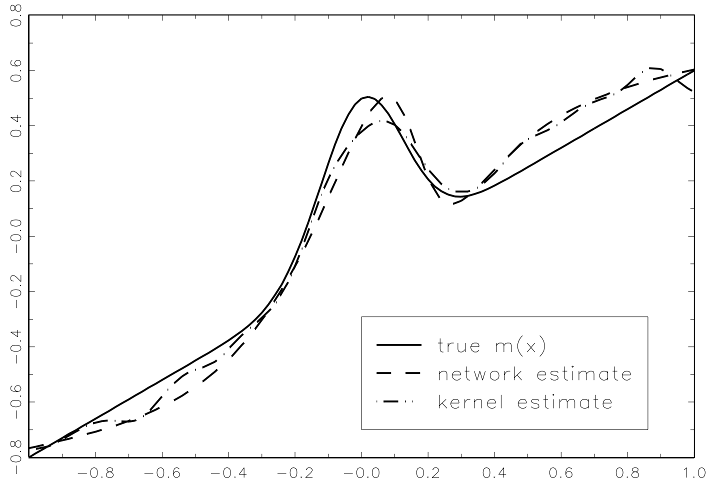


Figure 1: autoregressive function and estimates

Resampling may be used to improve the asymptotic normal approximation for the law of $f_H(x, \widehat{\vartheta}_N)$, where, for practical purposes, the covariance matrices Σ_1 and Σ_2 would have to be estimated anyhow. We present a residual-based bootstrap for the simple nonlinear autoregression of order 1 or NLAR(1)

$$S_{t+1} = m(S_t) + \varepsilon_t, \tag{3}$$

but the generalization to higher order models is straightforward. We start the procedure with some initial estimate $\widehat{m}_N$ which may be a neural network function estimate itself or some other consistent estimate for m. It allows for calculating sample versions of the innovations ε_t by

$$\widehat{\varepsilon}_t = Y_t - \widehat{m}_N(X_t)\,,\ \ t = 1, \ldots, N,$$

which have to be centered around 0:

$$\widetilde{\varepsilon}_t = \widehat{\varepsilon}_t - \frac{1}{N}\sum_{k=1}^{N} \widehat{\varepsilon}_k\,,\ t = 1, \ldots, N.$$

Let $\widetilde{F}_N$ denote the empirical distribution given by $\widetilde{\varepsilon}_1, \ldots, \widetilde{\varepsilon}_N$.

To generate the bootstrap resamples of the original time series, we first draw independent bootstrap innovations $\varepsilon_1^*, \ldots, \varepsilon_N^*$ from $\widetilde{F}_N$, i.e.

$$\varepsilon_t^* = \widetilde{\varepsilon}_k \text{ with probability } \frac{1}{N},\ \ k = 1, \ldots, N.$$

Then, we generate the bootstrap data as

$$S_{t+1}^* = \widehat{m}_N(S_t^*) + \varepsilon_t^*,\ t = 1, \ldots, N.$$

Using standard Monte Carlo techniques, we may mimic the behaviour of any quantity of interest based on a whole family of independent bootstrap resamples $S_0^*(i), \ldots, S_N^*(i),\ i = 1, \ldots, B$. The mean-squared error of the function estimate at x ,

$$\text{mse}(x) = \mathcal{E}(m(x) - f_H(x, \widehat{\vartheta}_N))^2,$$

may, e.g., be approximated by its bootstrap analogue

$$\text{mse}^*(x) = \frac{1}{B}\sum_{i=1}^{B}(\widehat{m}_N(x) - f_H(x, \widehat{\vartheta}_{N,i}^*))^2$$

where $\widehat{\vartheta}_{N,i}^*$ is the weight vector estimated from fitting the network function to the i-th bootstrap resample. The validity of this bootstrap approach has been shown for the regression model (2) by Franke and Neumann[6]. The proof can be generalized to the autoregressive case, too. However, the innovations ε_t have to be independent and identically distributed as, otherwise, the first step of drawing independent, identically distributed bootstrap innovations would make no sense. In the heteroscedastic case, other bootstrap procedures have to be considered.

We illustrate the performance of neural network estimates for nonlinear autoregressive functions and of the bootstrap approximations for their distribution with a small Monte Carlo study. The data $S_0, \ldots, S_N$, where $N = 200$, were generated by the NLAR(1)-scheme (3) with independent Gaussian innovations ε_t with mean 0 and standard deviation $\sigma_\varepsilon = 0.3$. The autoregressive function is a bump function

$$m(x) = 0.7x - 0.1 + 1.5\phi(x), \tag{4}$$

where ϕ denotes the standard normal density. On the interval $[-1, +1]$, where the stationary law of S_t is mainly concentrated, m is quite well approximated by a neural network function $f_3(x, \vartheta_0)$ with $H = 3$ hidden neurons and, therefore, 10-dimensional parameter vector ϑ_0. Figure 1 shows $m(x)$, the network function estimate $f_3(x, \widehat{\vartheta}_N)$ and, for sake of comparison, a Nadaraya-Watson-type kernel estimate $\widehat{m}(x, b)$ with bandwidth $b = 0.7$. The latter also served as initial estimate of the bootstrap procedure.

To investigate the performance of the bootstrap, we approximated the distribution of $d(x) = f_3(x, \widehat{\vartheta}_N) - m(x)$ by the distribution of $d^*(x) = f_3(x, \widehat{\vartheta}^*_N) - \widehat{m}(x, b)$. The quantities of interest were calculated from $M = 500$ independent Monte Carlo copies of the true sample and from $B = 500$ bootstrap resamples from the original data set $S_0, ..., S_N, N = 200$ resp. Figure 2a shows the function m together with the "true" 90 % - confidence band for m based on 500 Monte Carlo runs, where the band is not a uniform one, but formed by interpolating confidence intervals for $m(x)$ for various x. The neural network provides a good estimate of the autoregression function m, in particular around the origin where most of the observations are concentrated. Figure 2b compares this "true" confidence band with the corresponding 90 % - bootstrap confidence band. Remembering that the bootstrap is based on only one medium-sized time series sample, both bands agree remarkably well. Finally, for 4 different x Figure 3a-d show kernel density estimates, each with Gaussian kernel and bandwidth $b = 0.02$, of the estimation error $d(x)$ and its bootstrap approximation $d^*(x)$. Again, the performance of the bootstrap is quite satisfactory.

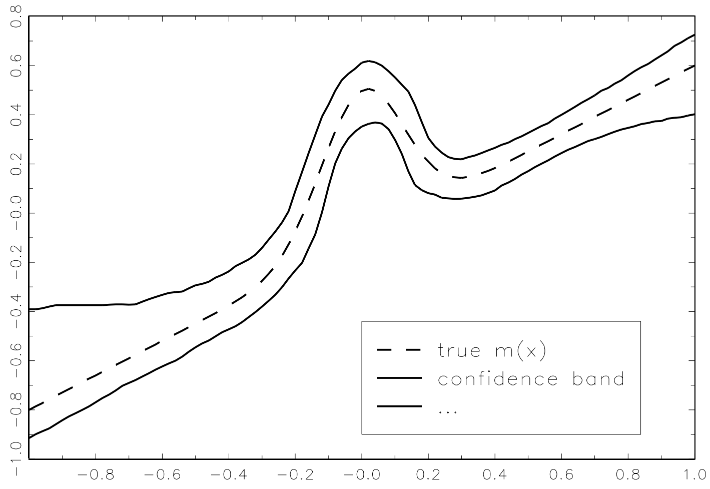


Figure 2a: Monte Carlo 90%-confidence band for m(x)

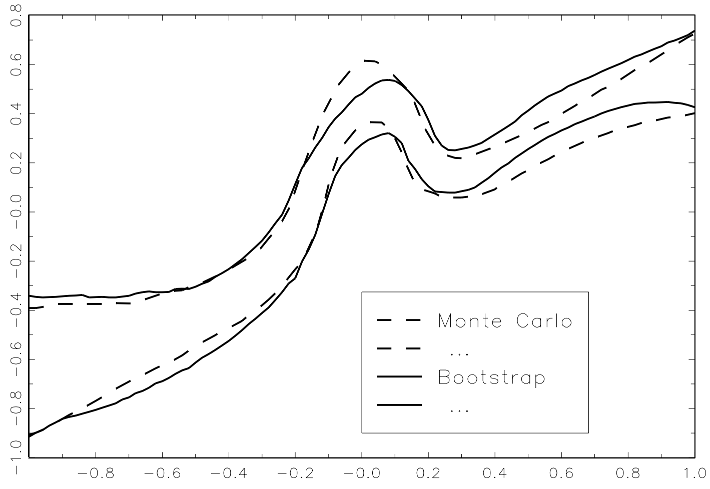


Figure 2b: Bootstrap and Monte Carlo 90%-confidence bands

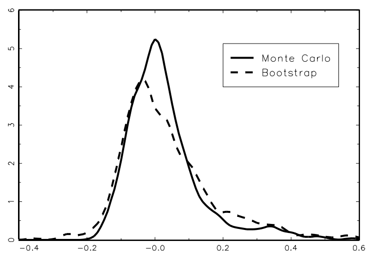


Figure 3a: error density at x = −0.8

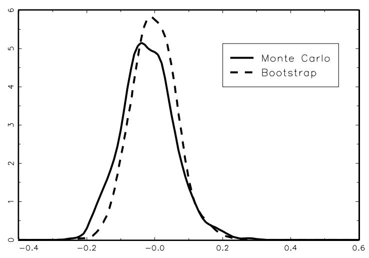


Figure 3b: error density at x = 0.0

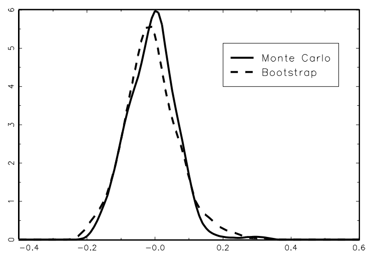


Figure 3c: error density at x = 0.2

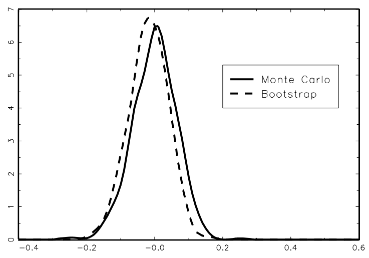


Figure 3d: error density at x = 0.8

3 Managing portfolios using neural networks

To illustrate the performance of neural networks in real applications which are of considerable complexity we give a short sketch of two case studies. In the first example, the task was to predict stock prices three months (60 trading days) ahead where the main goal was to generate trading signals for managing a portfolio of those stocks. The candidates for inclusion in the portfolio were 28 Dutch stocks dominating the CBS index. The available data were daily closing prices of all those stocks from 1993 to 1996. For model building and network parameter estimation, the data up to the end of 1995 were used. The data of 1996 were put aside for model validation.

As potential arguments for the forecasting function $f_H(x, \widehat{\vartheta}_N)$ several linear and nonlinear transformations of past stock prices $S_{t-\tau}, ..., S_t$, were considered, e.g. moving averages, envelopes, average directional movement indicators and other familiar tools of technical market analysis. Additionally, as exogeneous variables X_t in (1), the CBS index itself, foreign exchange rates, international interest rates, the MG base metal price and other intermarket data were taken into account. More than 60 candidates were investigated as potential coordinates of the input vector x. The final inputs were selected using experience of expert traders and statistical model selection procedures. More details are given by Franke[4]. The best network consisted of only $H = 3$ hidden neurons, but used 25-dimensional input vector x. The total number of parameters, therefore, was $dim(\widehat{\vartheta}_N) = 82$.

The point forecasts of stock prices varied considerably which is not surprising in view of the long forecasting period of 60 lags. However, they were condensed to a mere trend forecast, i.e. the information used in trading was solely if the stock price will
- increase significantly (by more than 5 %)
- decrease significantly (by more than 5 %)
- stay at approximately the same level.

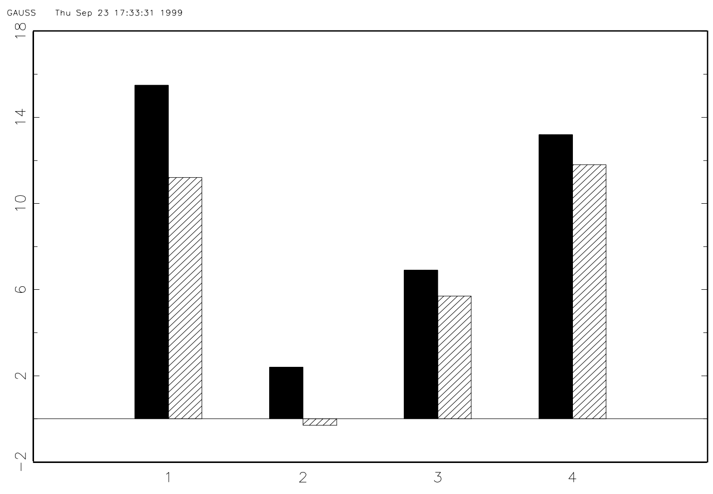


Figure 4: accumulated returns of stock portfolio

Using these forecasts, capital was allocated to the 28 stocks at the beginning of each quarter in the validation year 1996, and the resulting portfolio was held for 3 months unchanged. Only those stocks were included in the portfolio for which the prices were predicted to increase significantly up to the end of the holding period. This buy-and-hold strategy relying on neural network forecasts of stock prices was compared with the simple strategy of just buying the CBS index. Figure 4 shows the returns in percent for the network portfolio (solid bars) and the index portfolio (shaded bars). In each quarter, the network portfolio outperformed the index portfolio considerably which is even more remarkable as stock prices generally increased during the whole year of

1996, a situation in which it is not easy to beat the index.

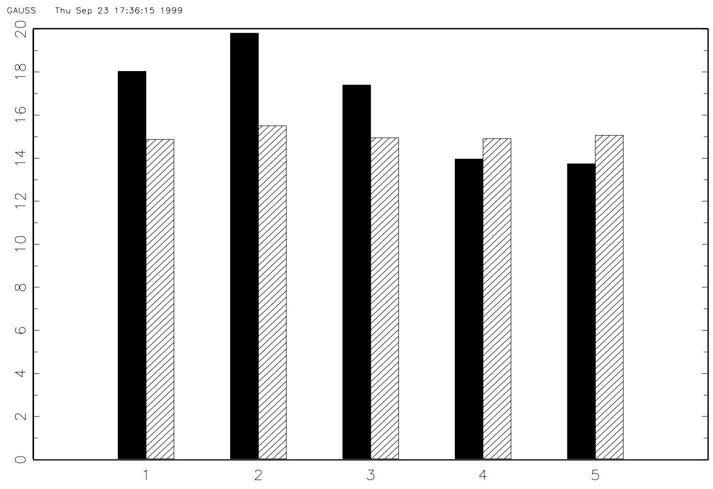


Figure 5: accumulated returns of currency portfolio

In the second example, the task was to construct a rule for allocating capital in a portfolio of three major currencies (US-Dollar, British Pound and Japanese Yen). A weekly buy-and-hold strategy was considered, i.e. at a particular day of the week, e.g. Tuesday, the portfolio composition was decided upon, based on the output of a neural network, and then the portfolio was held unchanged for one week. As inputs for the network, technical indicators calculated from past foreign exchange rates and intermarket data as in the above example were considered. Data from 1989 - 1995 were used for model building and parameter estimation, and the performance of the resulting allocation rules were evaluated using data from 1996 - September 1997. In this case, feedforward neural networks with more than one hidden layer proved to be more efficient than networks with only one layer of hidden neurons considered elsewhere in this paper. A typical network showing a good performance had two hidden layers with $H_1 = 9$ and $H_2 = 5$ neurons, respectively, and a 17-dimensional input vector x, resulting in $dim(\widehat{\vartheta}_N) = 230$ parameters to be estimated from the data. The details are given by Franke and Klein[5].

The network allocation rules were compared with various other portfolios, those consisting of one currency only, an equilibrium allocation of one third of the capital to each of the currencies and a well-established portfolio from real trading. For the validation period 1996 - September 1997, Figure 5 shows the annualized accumulated return in percent of one particular network allocation (solid bars) compared to the best of the competitors (shaded bars) which, during that period, always happened to be the portfolio containing only the, then, strong British pound. The performance is given for alle 5 possible weekly holding periods *1: Monday-Monday, 2: Tuesday-Tuesday, ... , 5: Friday-Friday.* That particular network outperformed all other allocations for the first three periods, but did not do so well for Thursdays and Fridays. This observation is not so surprising as differences in general trading behaviour between the start and the end of a week are well known. Therefore, in practice, one neural network did not suffice, but a system of networks, one for each day of the week, had to be developped.

4 Neural network estimates of volatility

The last two sections have illustrated that neural networks provide good estimates for the conditional mean of a financial time series even given a rather complex information set. In this section, we show how estimates of the conditional variance and volatility may be constructed following the same kind of approach. We now consider the following nonlinear heteroscedastic time series model:

$$S_{t+1} = m(S_t, S_{t-1}, ..., S_{t-\tau}, X_t) + \sigma_t \eta_{t+1} \tag{5}$$

where $\eta_1, \eta_2, \ldots$ are independent identically distributed with mean 0 and variance 1. We assume that the stochastic volatility σ_t is of a similar functional form as the conditional mean

$$\sigma_t = \sigma(S_t, S_{t-1}, ..., S_{t-\tau}, X_t) \tag{6}$$

Time series satisfying (5) and (6) are nonlinear AR-ARCH-processes with exogeneous components $X_t \in R^d$. The familiar parametric AR-ARCH-models are just a special case of this general type of stochastic process.

We construct a nonparametric estimate of the volatility function σ using neural networks as in section 2. As σ^2 is the conditional variance of S_{t+1} given the past we could fit a neural network function with inputs $S_t, S_{t-1}, ..., S_{t-\tau}, X_t$ as before and with outputs S_{t+1}^2 instead of S_{t+1} to the data. We would get an estimate of the conditional second moment and, subtracting the squared neural network estimate $f_H(x, \widehat{\vartheta}_N)$ for the conditional mean, an estimate of the conditional variance, too. For kernel estimates, however, Fan and Yao[3] have shown that it is more efficient to use $f_H(x, \widehat{\vartheta}_N)$ instead to calculate squared sample residuals and to smooth them instead of S_{t+1}^2 to get a nonparametric estimate of the conditional variance. We follow their approach in the neural network setting. To simplify notation, we describe the procedure for the nonlinear AR(1)-ARCH(1)-model

$$S_{t+1} = m(S_t) + \sigma(S_t)\eta_{t+1} \tag{7}$$

only. The generalization to time series models given by (5) and (6) is straightforward. In a first step, we calculate estimates of the innovations $\varepsilon_t = \sigma(S_t)\eta_{t+1}$ using the estimate $f_H(x, \widehat{\vartheta}_N)$ for $m(x)$ from section 2:

$$\widehat{\varepsilon}_{t+1} = S_{t+1} - f_H(S_t, \widehat{\vartheta}_N), t = 1, \ldots, N.$$

As $\sigma^2(x) = \mathcal{E}\{\varepsilon_t^2 | S_t = x\}$, we can estimate this function by fitting a neural network function $f_G(x, \gamma)$ with G neurons in the hidden layer to the data $S_0, ..., S_N$, where the parameter estimate $\widehat{\gamma}_N$ is given by

$$\frac{1}{N}\sum_{t=1}^{N}(\widehat{\varepsilon}_t{}^2 - f_G(S_{t-1}, \gamma))^2 = \min_{\gamma \in \Theta_G}!$$

The square root of $f_G(x, \widehat{\gamma}_N)$ is, the, a neural network based estimate of the volatility function $\sigma(x)$, i.e. of the conditional standard deviation of S_{t+1} given $S_t = x$. The consistency of this estimate for increasing sample size N and suitably increasing number G of hidden neurons again follows essentially from the work of White (1989, 1990) on neural network estimates for conditional expectations involving time series.

We study the performance of the neural network volatility estimates in a simulation study where we generate $M = 500$ Monte Carlo samples $S_0, ..., S_N$ with $N = 500$ from the nonlinear AR(1)-ARCH(1)-model (7). The sample size has to be larger as in section 2 as variances are harder to estimate than means in general. The η_t are standard normal random variables, the autoregressive function $m(x)$ is the same bump function (4) as in section 2, and the conditional variance function is chosen as in a common ARCH(1)-model as

$$\sigma^2(x) = 0.1 + 0.7x^2.$$

Figure 6a shows the true function m and a 90 % - confidence band based on the neural network function estimates $f_H(x, \widehat{\vartheta}_N)$ for the interval $[-2, +2]$, which contains the majority of the data. Comparing it to Figure 2a, we remark that the neural network estimates of the conditional expectations perform still reasonably well in the heteroscedastic case, in particular, if one recalls the heavy-tailedness of the stationary distribution of the S_t introduced by the ARCH(1)-innovations $\varepsilon_t = \sigma(S_t)\eta_{t+1}$.

Even the mean standard deviation $\mathcal{E}\{\sigma(S_t\}$ is about 0.95 and, therefore, more than three times as large as in the simulation study of section 2.

Figure 6b shows the true squared volatility function σ^2 together with a 90 % - confidence band from the Monte Carlo study. Considering the heavy-tailed law of the data S_t and the general difficulty of estimating variances the neural network estimates does reasonably well. Additionally, the simulation still suffers from numerical problems. In contrast to the homoscedastic model considered in section 2, the numerical procedure (a quasi-gradient method) for calculating the nonlinear least-squares parameters $\widehat{\vartheta}_N$ and $\widehat{\gamma}_N$ was prone to end up in local extrema with quite a bad performance of the corresponding function estimates. We solved this problem by starting the minimization routine with lots of different randomly selected initial values. Using an appropriate numerical algorithm like simulated annealing would be an alternative.

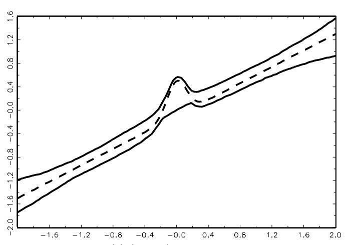


Figure 6a: m(x) (– – –) and 90%–confidence band

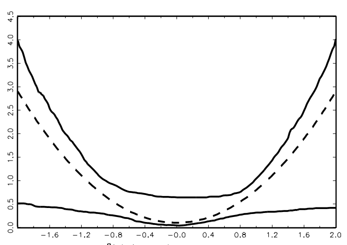


Figure 6b: σ^2(x) (– – –) and 90%–confidence band

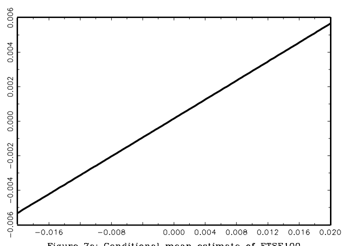
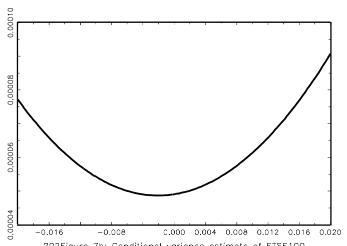


Figure 7a: Conditional mean estimate of FTSE100


202Figure 7b: Conditional variance estimate of FTSE100

We conclude this section by applying the estimators to a real data set. We selected the British FTSE100 index from January 4, 1993 to November 4, 1994, totalling 480 observations Z_t. Then, we fitted the model (7) to the daily returns $S_t = (Z_t - Z_{t-1})/Z_{t-1}$ estimating the conditional mean m and the conditional variance σ^2 by neural networks with $H = G = 3$ hidden neurons, corresponding to 10 parameters each. We also tried networks with up to 7 hidden neurons, but the estimates essentially did not change. Figure 7a and 7b show the estimates of conditional mean and variance of S_t given S_{t-1}. The mean is almost, but not exactly linear whereas the variance resembles an ARCH(1)-term apart from the asymmetry.

5 Estimating conditional value-at-risk with neural networks

Apart from volatility, another popular measure for financial hazards is the *value at risk* (VaR) as a bound which is exceeded by losses with small probability α only. There are various definitions of VaR (compare, e.g., Jorion[8]), but the crucial quantity is always the α-quantile of the return distribution of the financial asset. We consider here conditional quantiles given the information up to the present time t, and we discuss how to estimate them using neural networks. For our exposition, we concentrate on the simple nonlinear autoregression of order 1 given by (3). Generalizations to more complicated models are again straightforward. The conditional α-quantile function $q_\alpha(x)$ is given as solution of $F(q_\alpha(x)/x) = \alpha$, where $F(s/x)$ denotes the conditional distribution function of S_{t+1} given $S_t = x$

$$F(s/x) = pr\{S_{t+1} \le s | S_t = x\}$$

Nonparametric conditional quantile estimates based on common smoothing methods are closely related to kernel density estimates. Following, e.g., Samanta[12], we could estimate the joint density of S_{t+1} and S_t and the marginal density of S_t by kernel smoothing, getting the conditional density as a ratio. By integration, we get an estimate $F_N(s/x)$ for $F(s/x)$. Then, an estimate $q_{\alpha,N}(x)$ for the conditional quantile function $q_\alpha(x)$ is derived by solving $F_N(q_{\alpha,N}(x)/x) = \alpha$.

We could mimick this approach using neural networks. $F(s/x)$ is a conditional expectation of the indicator function $1_{(-\infty,s]}$ and could be approximated by neural networks as the conditional mean and variance in previous sections. However, for solving $F_N(q_{\alpha,N}(x)/x) = \alpha$ numerically, we would have to train neural networks frequently to get $F_N(s/x)$ for various values of s. If we are interested in estimating $q_\alpha(x)$ for only a few α, this approach is too cumbersome from a numerical point of view. We, therefore, follow a different approach which is based on the observation that the conditional quantile function $q_\alpha(x)$ solves

$$\mathcal{E}\left\{ |S_t - q| \left(\alpha 1_{[0,\infty]}(S_t - q) + (1-\alpha) 1_{(-\infty,0]}(S_t - q)\right) \Big| S_{t-1} = x \right\} = \min_{q \in R}!$$

We get a neural network estimate $f_Q(S_{t-1}, \widehat{\chi}_N)$ for $q_\alpha(x)$ by minimizing a sample version of this conditional expectation:

$$\frac{1}{N}\sum_{t=1}^{N} \big|S_t - f_Q(S_{t-1},\chi)\big|\big(\alpha 1_{[0,\infty]}(S_t - f_Q(S_{t-1},\chi)) \\ +(1-\alpha)1_{(-\infty,0]}(S_t - f_Q(S_{t-1},\chi))\big) = \min_{\chi\in\Theta_G} !$$

$f_Q(x,\chi)$ denotes a network function as in section 2 with Q hidden neurons. This approach has been studied by White[15] who proved the consistency of the conditional quantile estimate f_Q if N and Q increase with appropriate rates to ∞.

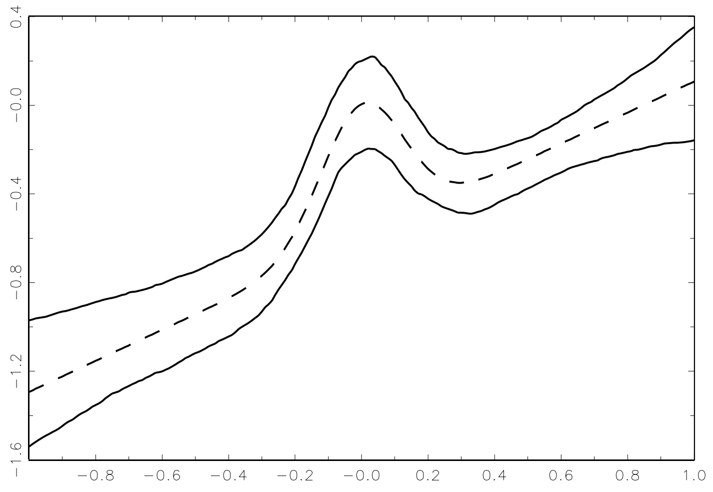


Figure 8: q(x) (– – –) and 90%–confidence band

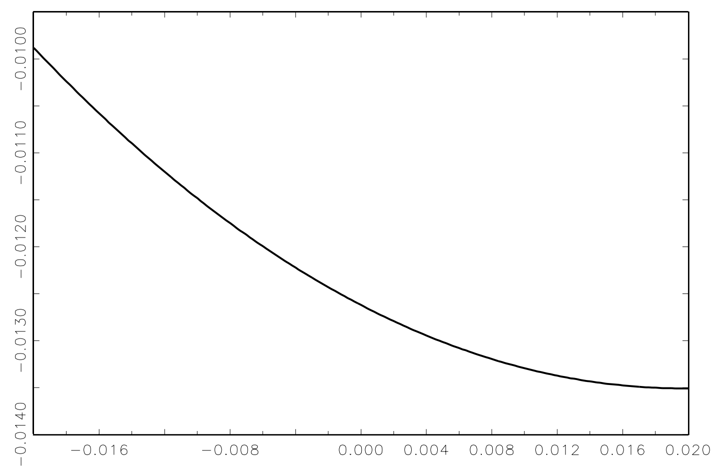


Figure 9: Conditional 5%−quantile estimate of FTSE100

We illustrate the performance of this quantile estimator with a simulation study where the generated data follow exactly the same nonlinear autoregression and specifications as in the Monte Carlo study of section 2. In particular, the sample size is $N = 201$ and the number of Monte Carlo runs is $M = 500$. Figure 8 shows the true conditional 5 % - quantile function $q_{.05}(x)$ for this time series together with a 90 % - confidence band based on the neural network quantile estimates $f_Q(x, \widehat{\chi}_N)$ with $Q = 10$. As for estimating the conditional mean, the performance is quite good in this homoscedastic situation.

Finally, we estimate the conditional 5 % - quantile function for the next return of the FTSE100-index series given the present return, where we used the same data as in section 4. Figure 9 shows the resulting estimate.

Acknowledgement:

section 3 is based on joint work with Commerzbank AG, Frankfurt, in particular with D. Oppermann and U. Kern. The data, we used, were provided by DATASTREAM.

References

1. U. Anders, *Statistische neuronale Netze.* (Vahlen, München, 1997)
2. G. Bol, G. Nakhaeizadeh and K.-H. Vollmer eds., *Finanzmarktanalyse und -prognose mit innovativen quantitativen Verfahren.* (Physica-Verlag, Heidelberg, 1996)
3. J. Fan and Q. Yao, *Efficient estimation of conditional variance functions in stochastic regression. Biometrika.* **85**, 645-660 (1998).
4. J. Franke, *Nonlinear and Nonparametric Methods for Analyzing Financial Time Series.* In: *Operation Research Proceedings 98,* P. Kall und H.-J. Luethi eds. (Springer-Verlag, Berlin, 1999).
5. J. Franke and M. Klein, *Optimal portfolio management using neural networks - a case study. Report in Wirtschaftsmathematik* (University of Kaiserslautern, 1999).
6. J. Franke and M. Neumann, *Bootstrapping neural networks.* Tentatively accepted for publication in *Neural Computation.*
7. K. Hornik, M. Stinchcombe and H. White. *Multilayer feedforward networks are universal approximators, Neural Networks* **2**, 359-366 (1989)
8. Ph. Jorion *Value at Risk: The New Benchmark for Controlling Market Risk.* (Irwin, Chicago, 1996).
9. J. P. Kreiss, *Nonparametric estimation and bootstrap for financial time series.* In: this volume.
10. N. Murata, S. Yoshizawa and S. Amari, *Network information criterion - Determining the number of hidden units for an artificial neural network model. IEEE Trans. Neural Networks* **5**, 865-872 (1994).
11. A.-P.N. Refenes, A.D. Zapranis and J. Utans, *Neural model identification, variable selection and model adequacy.* In: *Neural Networks in Financial Engineering,* A. Weigend et al. eds. (World Scientific, Singapore, 1996)
12. M. Samanta, *Nonparametric estimation of conditional quantiles. Statistics & Probability Letters* **7**, 407-412 (1989).
13. H. White, *Some asymptotic results for learning in single hidden-layer feedforward network models. J. Amer. Statist. Assoc.* **84**, 1008-1013 (1989).
14. H. White *Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings. Neural Networks* **3**, 535-550 (1990).
15. H. White, *Nonparametric estimation of conditional quantiles using neural networks.* In: *Computing Science and Statistics,* C. Page and R. Le Page eds. (Springer-Verlag, Berlin, 1992).