Reliability of neural network based Value-at-Risk Estimates

Ralf Prinzler

August 1999

Abstract

Value-at-Risk (VaR) has become an increasingly popular measure in financial risk measurement. To calculate VaR various models have been suggested. This paper shows briefly how neural networks can be applied to calculating VaR. The paper focusses on the reliability of the VaR estimates using a bootstrap procedure. Results on a foreign exchange portfolio suggest that connectionist VaR estimates based on mixture density models show bias that is relatively small on average but varies with time. Standard errors due to estimation error and random network initialisation are relatively small and lead to moderate model failure rates.

Keywords: Value-at-Risk, Mixture Densities, Neural Networks, Bootstrapping

1 Introduction

Over the past few years the Value-at-Risk (VaR) has become an increasingly popular measure in financial risk measurement. The VaR of a single investment or a portfolio of investments is defined as the worst expected financial loss over a given investment horizon with a given probability. In a statistical sense the VaR constitutes the upper bound of a one-sided prediction interval placed on the probability distribution of future portfolio losses.
More formally we can define the VaR as the p-quantile of the distribution of future portfolio losses

$$\begin{aligned} \text{Prob}\left(L_t \leq \text{VaR}\right) &= p & (1) \\ L_t &= W_{t-\Delta t} - W_t & (2) \end{aligned}$$

where W_t denotes the market value of a portfolio at a given time t and L_t refers to the loss cumulated over the investment horizon Δt. p refers to a given probability. Positive values of L_t therefore denote a loss; negative

values denote a gain. If we specify an investment horizon of $\Delta t = 1$ day and a probability of say 95% a VaR measure of 100.000$ simply says that the probability of loosing more than 100.000$ over the next day is less than 5%. If we wish to calculate the VaR of a portfolio the most common approach is to identify basic risk factors that determine future portfolio value. The choice of risk factors depends on the portfolio. Typical risk factors are exchange rates, prices of stocks and commodities, interest rates etc.[1] The next most common step is to express the impact of relevant risk factors $\mathbf{R} = (R_1, R_2, \ldots, R_k)'$ on the portfolio value W. The portfolio loss - the object of interest in VaR calculation - is typically modelled as a function of relative or log-changes in those risk factors. This is expressed in equation (3) where we denote relative or log-changes (risk factor returns) by $\mathbf{X} = (X_1, X_2, \ldots, X_k)'$.

$$\begin{aligned} L_t &= W(\mathbf{R}_{t-\Delta_t}) - W(\mathbf{R}_t) \\ L_t &= L(\mathbf{X}_t) \end{aligned} \tag{3}$$

Thus we can write for a very general VaR model

$$\text{Prob}\left(L(\mathbf{X}_t) \leq \text{VaR} \,|\Phi_{t-\Delta_t}\right) = p \tag{4}$$

where $\Phi_{t-\Delta_t}$ denotes the information set available at the time the prediction is made. The contents of Φ depend on the stochastic process that is used to describe the evolution of $\{\mathbf{R}\}_t$ over time. If we consider the special case of independent and identically distributed (i.i.d.) risk factor returns $\mathbf{X}$ the past history of risk factor prices is irrelevant for predicting VaR except the most recent realisations of $\mathbf{R}_{t-\Delta t} = \mathbf{r}_{t-\Delta t}$ which determine the current portfolio value and serve as scaling parameters in (4).

Equation (4) is important for some reasons. First we have to keep in mind that the assumption of i.i.d. factor returns is inadequate for many financial time series. This implies that VaR calculation has to account for intertemporal as well as for cross-sectional dependencies[2] among risk factor returns. The existence of intertemporal dependencies leads straight away to the insight that adequate VaR calculations have to be based on the conditional distribution of market risk returns, reflecting the history of the price process. Second we see from (3) that the distribution of future portfolio losses can in principle be calculated from the joint distribution of market risk returns. In practice this calculation can be done only for special cases. Even if we know the distribution $f_{\mathbf{X}}(\mathbf{x})$ of risk returns the calculation of $f_L(l)$ can be impossible, if $L(\mathbf{X}_t)$ is a nonlinear function.[3]

Neural networks can enter into a VaR model in two different ways. The first one is to use neural netwoks as density estimators. Alternatively we

[1] Sometimes the term *risk factor* refers to relative or log-changes in those prices.

[2] Cross-sectional dependencies are common and reflected for example in correlations between market risk returns.

[3] For an overview over some popular VaR models see for example [6].

could use neural networks to model or to approximate $L(\mathbf{X})$.[4] Of course one may combine both applications. Throughout this paper we consider the case where neural netwoks are used as density estimators.
The rest of the paper is organized as follows. Section 2 introduces Mixture Density Networks as a very flexible instrument for density estimation and shows how VaR estimates can be calculated if we consider linear pricing functions. Section 3 gives a brief overview over the bootstrap which offers a conceptually simple although computationally extensive way to estimate the reliability of VaR estimates. Section 4 shows two applications.

2 The Estimation of Value-at-Risk using Mixture Density Networks

As we have seen in the previous section one major building block in VaR analysis is the estimation of the joint distribution of market risk returns $f_{\mathbf{X}}(x)$. There are various ways to estimate this distribution ranging from nonparametric to parametric techniques. Only recently hybrid models have been introduced.[5]
In this paper we assume that the unknown joint distribution of market risk returns $\mathbf{X} = (X_1, X_2, \ldots, X_k)'$ can be approximated by the following mixture model:[6]

$$f_{\mathbf{X}|\mathbf{Z}=\mathbf{z}}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i(\mathbf{z}) h_i(\mathbf{x}|\mathbf{z}) \tag{5}$$

where the component densities h_i are Gaussian with a single variance parameter

$$h_i(\mathbf{x}|\mathbf{z}) = (2\pi)^{-k/2} \sigma_i(\mathbf{z})^{-k} \exp\left(-\frac{||\mathbf{x} - \boldsymbol{\mu}_i(\mathbf{z})||^2}{2\sigma_i^2(\mathbf{z})} \right). \tag{6}$$

In order to guarantee (5) to be a probability density function the mixing coefficients have to satisfy

$$\sum_i \alpha_i = 1, \qquad \alpha_i \geq 0, \qquad i = 1, 2, \ldots m.$$

Figure 1 provides a graphical representation of a *Mixture Density Network* (MDN).[7]
To understand the inner working of a MDN note that (5) and (6) define a conditional probability distribution, where the parameters $\alpha_i, \sigma_i, \mu_i$ of each

[4] This could be interesting in cases where the portfolio under consideration includes options.

[5] See [3] for details.

[6] For a further application of mixture models in VaR estimation see [9].

[7] See [1] and [2].

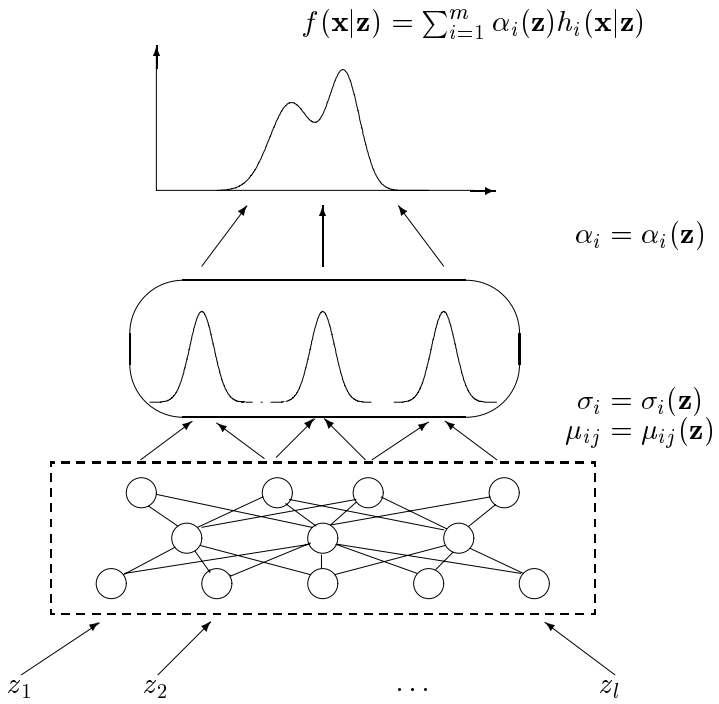

Figure 1: Mixture Density Network

component density $h_i, i = 1, 2, \ldots, m$, are modelled as functions of the neural network inputs $\mathbf{z} = (z_1, z_2, \ldots, z_l)'$. Thus the network input represents the conditioning variables. If we choose m and the number of hidden units in the neural network sufficiently large a MDN is capable of modelling a very broad class of distributions.

Mixture density networks contain well known models as a special case. Consider the case where one single risk factor return X is assumed to follow an ARCH(1) process:

$$\begin{aligned} x_t &= \sigma_t u_t \\ \sigma_t^2 &= \omega + \beta x_{t-1}^2, \qquad \omega \geq 0, \beta \geq 0 \end{aligned}$$

where u_t is white noise with zero mean and unit variance. If we require u_t to be distributed normally $u_t \sim N(0,1)$ it is well known that the conditional distribution of x_t is given by

$$x_t | \Phi_{t-1} \sim N(0, \sigma_t^2).$$

This model can be implemented easily into a MDN.[8] The ARCH(1) relevant information set Φ_{t-1} just contains x_{t-1}.

[8]We need to adjust the MDN slightly by choosing one single component density ($m =$

To estimate the mixture parameters, i.e. to fit the model we need to minimize a suitable error function. In our case a suitable[9] error function is given by the negative log-likelihood

$$E = -\ln \mathcal{L} = \sum_{t=1}^{T} E^t \tag{7}$$

$$E^t = -\ln \left\{ \sum_{i=1}^{m} \alpha_i(\mathbf{z}_t) h_i(\mathbf{x}_t|\mathbf{z}_t) \right\} \tag{8}$$

which is minimized with respect to the network weights using standard techniques applicable to neural networks. In this paper we apply standard error-backpropagation.[10]

We now discuss how we can use a MDN for VaR estimation. We consider the case of linear portfolios[11] $\mathbf{w} = (w_1, w_2, \ldots, w_k)'$ where it is relatively simple to calculate an estimate of VaR. When training the mixture density network we estimate the joint distribution of market risk returns

$$\hat{f}_{\mathbf{X}_t|\mathbf{Z}_t=\mathbf{z}_t}(\mathbf{x}) = f(\widehat{\boldsymbol{\alpha}}_t, \widehat{\boldsymbol{\sigma}}_t^{2,(x)}, \widehat{\boldsymbol{\mu}}_t^{(x)}|D_{\mathbf{x},\mathbf{z}}). \tag{9}$$

conditioned on the network inputs $\mathbf{z}_t$. $D_{\mathbf{x},\mathbf{z}}$ denotes the data set used to estimate the model. The choice of $\mathbf{z}_t$ depends on the VaR model used.

The portfolio loss L_t is given by

$$L_t = L(\mathbf{X}_t) = \mathbf{w}'\mathbf{X_t} \tag{10}$$

Since $L(\mathbf{X}_t)$[12] is a simple function we can find an analytical expression for $f_{L_t}(l)$ by density transformation. It can be shown that $f_{L_t}(l)$ again is a mixture density. Our estimate of $f_{L_t}(l)$ is given by

$$\hat{f}_{L_t}(l) = \sum_{i=1}^{m} \widehat{\alpha}_{i,t} h_{i,t}(l) \tag{11}$$

where

$$h_{i,t}(l) = \frac{1}{\sqrt{2\pi}\widehat{\sigma}_{i,t}} \exp\left[-\frac{1}{2} \frac{(l - \widehat{\mu}_{i,t})^2}{\widehat{\sigma}_{i,t}^2} \right]$$

1), with the mixing coefficient fixed to unity and mean fixed to zero, and one single linear hidden unit with bias. For the input vector we choose $z_t = \ln x_{t-1}$. Furthermore we have to create a special network weight structure and make slight amendments to the error function.

[9] See [5] for a critical discussion.

[10] For more details see [1] and [2].

[11] Each element w_i represents the (negative) sensitivity of the portfolio value with respect to X_i.

[12] Throughout this paper we assume the portfolio loss function $L(\mathbf{X}_t)$ to be constant. With other words we assume the same portfolio composition for every t.

with

$$\widehat{\mu}_{i,t} = \sum_{j=1}^{k} w_j \widehat{\mu}_{ij,t}^{(x)} \quad \text{and} \quad \widehat{\sigma}_{i,t}^2 = \widehat{\sigma}_{i,t}^{2,(x)} \sum_{j=1}^{k} w_j^2 \quad i = 1, 2, \ldots, m.$$

$\widehat{\alpha}_{i,t}$, $\widehat{\mu}_{ij,t}^{(x)}$ and $\widehat{\sigma}_{i,t}^{2,(x)}$ denote our estimates according to (9).
Recall that VaR is defined to be the p-Quantile of $f_{L_t}(l)$ (see equation (4)). If we replace the unknown probability density function $f_{L_t}(l)$ by $\hat{f}_{L_t}(l)$ we can write using (11)

$$\alpha_{1,t} \int_{-\infty}^{\text{VaR}} h_{1,t}(l)\,\mathrm{d}l + \alpha_{2,t} \int_{-\infty}^{\text{VaR}} h_{2,t}(l)\,\mathrm{d}l + \ldots + \alpha_{m,t} \int_{-\infty}^{\text{VaR}} h_{m,t}(l)\,\mathrm{d}l = p \tag{12}$$

It is not possible to solve this equation analytically. We therefore calculate $\widehat{\text{VaR}}$, which is our estimate of the true VaR, for a given p using an iterative scheme. Furthermore it is of considerable interest to obtain information about the reliability of VaR. Since it is impossible to calculate standard errors directly we use a bootstrap procedure as described in the following section.

3 Bootstrapping Parameter Estimates

The application of the bootstrap to evaluate the performance of neural networks is not new. Tibshirani [8] compares different methods that can be applied to estimate prediction errors of neural networks and finds the bootstrap approach superior. In his paper Tibshirani focusses on the the application of multi-layer perceptrons to prediction tasks. It should be noted though that the bootstrap procedure is not limited to that particular case. LeBaron and Weigend [10] apply the bootstrap to evaluate time series predictions of financial data.
The basic idea[13] of the bootstrap procedure is to "enlarge" a given data set $(x_1, x_2, \ldots, x_n)$ from an unknown distribution F by repeated resampling with replacement. Denote the true but unknown parameter of interest by $\theta = t(F)$ and the estimate using the available data set $(x_1, x_2, \ldots, x_n)$ by $\hat{\theta} = t(\hat{F})$ where $\hat{F}$ denotes the empirical distribution.[14]
Resampling B times from the observed data sample leads to B different bootstrap samples denoted by * upon which bootstrap estimates $\hat{\theta}^*$ of θ can be calculated.

$$\begin{aligned} \hat{F} &\rightarrow (x_1^{b*}, x_2^{b*}, \ldots, x_n^{b*}) \\ \hat{\theta}^*(b) &= t(\hat{F}^*), \qquad b = 1, 2, \ldots B \end{aligned}$$

[13] See for an introduction Efron and Tibshirani [4]. In the following we use their terminology.

[14] $\hat{\theta} = t(\hat{F})$ is called the plug-in estimate of θ. We use $\hat{\theta} = t(\hat{F})$ later on since we calculate our parameter of interest VaR based on the empirical distribution $\hat{f}_L(l)$.

The information contained in the empirical distribution of $\hat{\theta}^*$ can be used to correct $\hat{\theta}$ for bias, to estimate standard errors or to construct confidence intervals. If we wish to calculate the standard error $\mathrm{se}_{\hat{\theta}}$ of the parameter estimate $\hat{\theta}$ we can use the standard error of the bootstrap estimates

$$\widehat{\mathrm{se}}_B = \sqrt{\sum_{b=1}^{B} \left(\hat{\theta}^*(b) - \bar{\theta}^*\right)^2 / (B-1)} \tag{13}$$

with $\bar{\theta}^* = \sum_{b=1}^{B} \hat{\theta}^*(b)/B$ as an estimate. Calculating standard errors using the bootstrap can be applied in situations where no analytic solutions can be found. In the case of VaR calculation our parameter of interest

$$\widehat{\theta} = \widehat{\mathrm{VaR}} = \hat{F}_L^{-1}(p)$$

is a nonlinear function of the network inputs $\mathbf{z}$, where no formula can be given to calculate $\mathrm{se}_{\hat{\theta}}$.
Bootstrap replications $\hat{\theta}^*(b)$ may also be used to correct the bias

$$\mathrm{bias}_F = \mathrm{E}_F(\hat{\theta}) - \theta$$

of an estimator. The basic idea is to estimate bias_F using the bootstrap estimate

$$\mathrm{bias}_{\hat{F}} = \mathrm{E}_{\hat{F}}(\hat{\theta}) - t(\hat{F})$$

where $t(\hat{F})$ denotes the plug-in estimate of θ. The bootstrap expectation $\mathrm{E}_{\hat{F}}(\hat{\theta})$ can be approximated by the bootstrap average $\bar{\theta}^*$ introduced above. Thus our bootstrap estimate of bias based on B replications is

$$\widehat{\mathrm{bias}}_B = \bar{\theta}^* - t(\hat{F}) = \bar{\theta}^* - \hat{\theta} \tag{14}$$

The bias corrected estimate of θ is given by

$$\bar{\theta} = \hat{\theta} - \widehat{\mathrm{bias}}_B$$

Bias correction can be dangerous in practice since variability in $\widehat{\mathrm{bias}}_B$ may lead to a higher standard error in $\bar{\theta}$. This again can be checked using bootstrapping. Efron and Tibshirani[15] recommend to do without bias correction if $\widehat{\mathrm{bias}}_B$ is small compared to the standard error estimate $\widehat{\mathrm{se}}_B$. If the difference between bias and standard error is large this may indicate that $\hat{\theta} = t(\hat{F})$ is not a suitable estimate of θ.

[15] See [4] p. 138f.

4 Two examples

Data from a mixture distribution

To show the reliability of MDN-based quantile estimates we consider two examples. The first one is particularly simple. 10000 realisations of a univariate random variable was drawn from the (unconditional) mixture distribution

$$f_X(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x)$$

where $h_i(x), i = 1, 2$ denotes the density function of the normal distribution with parameters μ_i and σ_i.
In this example we set $\alpha_1 = 0.6, \alpha_2 = 0.4, \sigma_1 = 0.8, \sigma_2 = 1.2, \mu_1 = 0$ and $\mu_2 = 0.5$. Let the 95% quantile $\theta = F_X^{-1}(0.95)$ be the parameter of interest. θ can be calculated according to (12) and yields $\theta = 1.947$.[16] The 95% quantile of the empirical distribution yields $\hat{\theta} = 1.977$.
We start our analysis by training a very simple mixture density network with a single network input fixed to unity, one linear hidden unit and two component densities.[17]
First tests showed that choosing the right learning rate[18] λ is of considerable importance for training success. Too large a learning rate leads to rapid swings in the mixture parameters. We tested variable as well as constant learning rates and conducted several simulation runs with varying length. It turned out that a small learning rate combined with parameter updating after each data pattern lead to sufficiently fast training. After 100 training cycles only very small changes in network weights could be observed.
In this investigation we did not employ generalization techniques since the number of network parameters is very small compared to the number of training patterns and thus the danger of overfitting is small. Table 1 shows results for the 95%-quantile obtained with training runs ranging from 100 to 1000 cycles and constant or variable learning rates. We note that our neural network-based estimates $\hat{\theta}^{NN}$ tend to be larger than both θ and $\hat{\theta}$.
Table 2 shows some bootstrap results. From the first experiment ($\lambda = 0.01$) we see that a relatively large learning rate leads to a larger standard error. This confirms findings of earlier studies that very small learning rates should be used when training MDN. From Table 1 we see that θ lies within a one standard error interval around $\hat{\theta}^{NN}$ for both experiments. In the second experiment ($\lambda = 0.001$) we also note a very small bootstrap estimate of bias.

[16] This value has been calculated using an iterative scheme.

[17] The neural network part of the mixture model does not need to be complex since no relationship to conditional variables has to be modelled. Here in fact we would not need the neural network part at all.

[18] A learning rate is a small number multiplied with the gradient of the error function.

variable learning rate								
$\lambda = 0.1, \ldots, 0.001$			$\lambda = 0.05, \ldots, 0.001$			$\lambda = 0.01, \ldots, 0.0001$		
100	500	1000	100	500	1000	100	500	1000
1.983	1.983	1.983	1.983	1.983	1.983	1.986	1.986	1.986
fixed learning rate								
$\lambda = 0.01$			$\lambda = 0.001$			$\lambda = 0.0001$		
100	500	1000	100	500	1000	100	500	1000
1.935	1.935	1.935	1.983	1.983	1.983	1.994	1.988	1.986

Table 1: Values for 95%-quantile estimates $\hat{\theta}^{NN}$ for different learning rates λ and training cycles $(100, 500, 1000)$.

	$\hat{\theta}^{NN}$	$\bar{\theta}^*$	$\widehat{\text{se}}_B$	$\hat{\theta}^{NN} \pm \widehat{\text{se}}_B$	$\widehat{\text{bias}}_B$
$\lambda = 0.01$	1.935	2.001	0.092	1.843... 2.027	0.066
$\lambda = 0.001$	1.983	1.986	0.036	1.947... 2.019	0.003

Table 2: Bootstrap results (B=100) for two MDN with one hidden unit. The network was trained over 100 training cycles.

Foreign exchange data

In a second example we consider VaR estimates for a foreign exchange portfolio. We take the perspective of an investor whose home currency is the German Mark (DM) and who invests his money at every instance equally into US-Dollar (USD), Swiss Franc (CHF) and Japanese Yen (JPY) cash positions. Obviously the currencies constitute the only risk factors in this portfolio. We define the daily returns by

$$X_{\text{C},t} = \ln(K_{\text{DM/C},t}/K_{\text{DM/C},t-1})$$

where K denotes the price of a currency C=USD,CHF,JPY expressed in German Mark.

In the next step we decided about an appropriate input vector $\mathbf{z}$ which is assumed to be useful in predicting the joint conditional distribution of exchange rate returns. We experimented with two different inputs. In the first case we used 5-day historical moving averages and 5-day historical moving standard deviations for each currency, in the second case we used the same inputs but calculated over a 30 day window. Both choices are arbitrary. The motivation for our approach was to find inputs that are easy to calculate and that may be able to catch the time varying properties of the return process. Results reported here were obtained for the case with inputs calculated over the 30 day window.

In our analysis we used foreign exchange data from 5th September 1985

	Experiment I	Experiment II
Component Densities	3	5
Hidden Neurons	5	5
Learning Rate	0.0001	0.0001
Training Cycles	1000	1000
Model Failures in %		
Training ($p = 0.95$)	5.5	4.7
Test ($p = 0.95$)	6.8	5.6
Training ($p = 0.99$)	1.1	1.0
Test ($p = 0.99$)	1.6	0.4

Table 3: Model specification and results for two experiments with foreign exchange data.

till 31st August 1994 (2250 data patterns). The first 2000 patterns were used for estimation, the remaining 250 pattern were used for testing. We experimented with various learning parameters and networks of different complexity. Since bootstrapping is a very time consuming procedure only two models were bootstrapped ($B = 100$). Results and model configuration are reported only for these two simulations (see Table 3) were the neural network part of the mixture model was implemented as a fully connected multi-layer perceptron with one hidden layer. The network was trained using standard error-backpropagation.

The experiments were conducted as follows.

First the complete non-bootstrapped training data set was used to estimate the mixture parameters. Based on the estimated mixture parameters and the given portfolio composition we calculated estimates $\widehat{\mathrm{VaR}}_t$ for the training data set ($t = 1, 2, \ldots, 2000$, ex post prediction) and the test data set ($t = 2001, \ldots, 2250$, ex ante prediction). VaR estimates were calculated for the 95% and 99% level. Additionally model failures were calculated. A VaR model is regarded to have failed if the actual portfolio loss L_t exceeds the prediction $\widehat{\mathrm{VaR}}_t$. See Table 3 for results.

In the next step both models were bootstrapped. In each bootstrap run a new bootstrap sample was generated, the network weights randomly initialized and the mixture parameter estimated. After estimation the bootstrap sample was presented to the network again and bootstrap VaR estimates $\widehat{\mathrm{VaR}}^*_{b,t}, b = 1, 2, \ldots, 100, \;\; t = 1, 2, \ldots, 2000$ generated. This procedure yields approximately 100 bootstrap VaR estimates for each training pattern. Note that only the training data set was bootstrapped.

The average of the bootstrapped VaR estimates $\overline{\mathrm{VaR}}^*_t$ and the standard

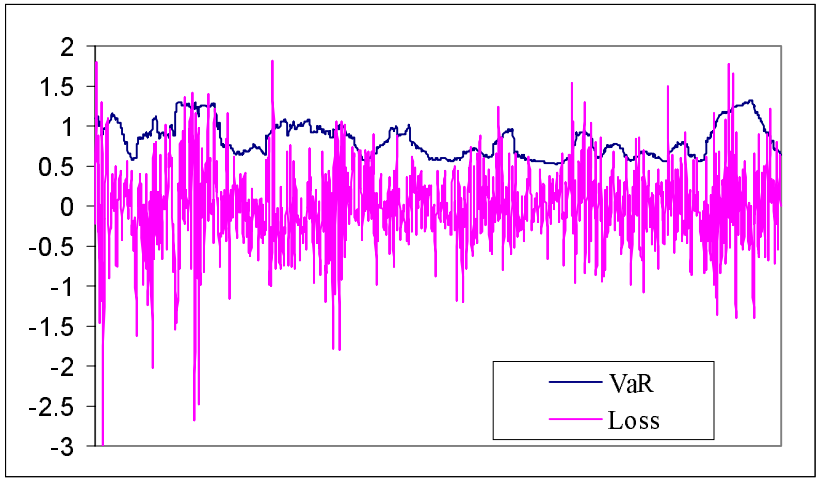

Figure 2: Value-at-Risk estimates $\widehat{\mathrm{VaR}}_t$ (VaR) for the 95% level and actual portlolio losses L_t (L). The values shown have been calculated for the first 1000 data patterns of the training data set.

error estimate $\widehat{\mathrm{se}}_{B,t}, t = 1, 2, \ldots, 2000$ were used to evaluate standard error and bias of $\widehat{\mathrm{VaR}}_t$ (See equations (13) and (14)). Figure 2 and 3 show results obtained from experiment II for the first 1000 training patterns. The results shows more variation in the bias estimate than in the standard error estimate. Furthermore we note the standard errors change in accordance with $\widehat{\mathrm{VaR}}_t$. They are higher in times with higher volatility and vice versa. The bootstrap bias estimate $\widehat{\mathrm{bias}}_{B,t} = \overline{\mathrm{VaR}}^*_t - \widehat{\mathrm{VaR}}_t$ changes over time and can be quite large compared to the standard error. Due to the variability in bias there is no obvious indication wheter or not to correct $\widehat{\mathrm{VaR}}_t$ for bias. Further insight into the source of bias is therefore required.

In a final step we investigate the possible impact of errors in $\widehat{\mathrm{VaR}}_t$. Variability in $\widehat{\mathrm{VaR}}_t$ may have different sources. A first source is the estimation error due to finite sample size. Other error components can be traced back to the random start conditions in the estimation procedure or to model misspecification. The procedure described above estimates the first two types of error simultaneously. To show the impact of errors on VaR predictions over the test set we construct a two standard error interval around $\widehat{\mathrm{VaR}}_t$. We do this by adding and subtracting the double average bootstrap standard error[19]

[19]This is a simple approach. It can be improved by calculating VaR predictions over the test set after each bootstrap simulation. This leads to a set of bootstrap preditions with time varying standard error intervals.

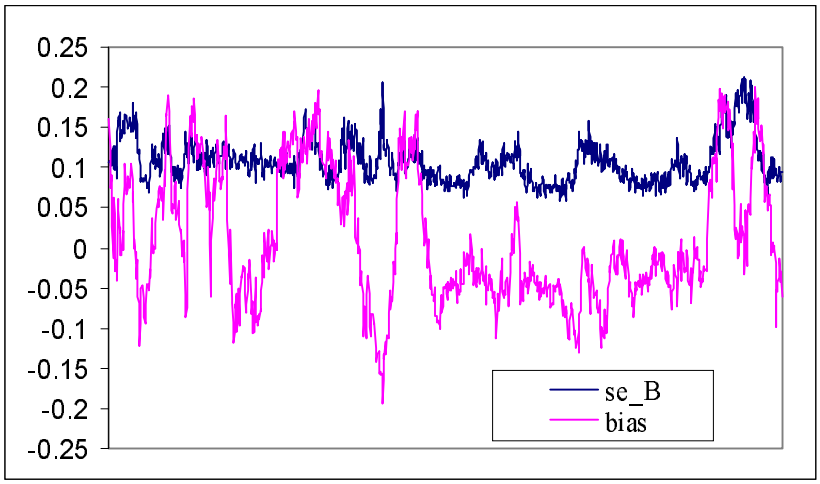

Figure 3: Bias $\widehat{\text{bias}}_{B,t}$ (bias) and standard error $\widehat{\text{se}}_{B,t}$ (se_B) estimates calculated for the first 1000 data patterns of the training data set.

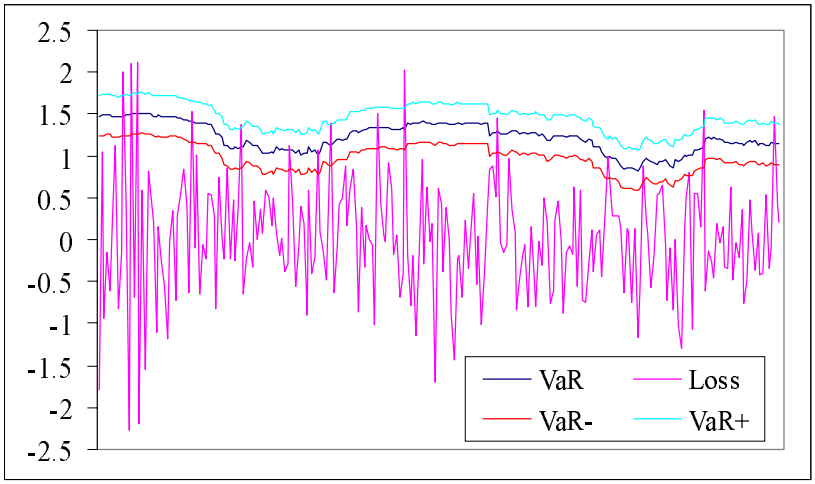

Figure 4: Value-at-Risk estimates on a 95% level and their two standard error intervals. Values are reported for a held out data set with 250 test patterns. $\lambda = 0.0001$. Corresponding model failures in %: 6.8-5.6-3.2.

over the training set

$$\begin{aligned}
\bar{\mathrm{se}}_B &= \frac{1}{2000}\sum_{t=1}^{2000}\hat{\mathrm{se}}_{B,t} \\
\widehat{\mathrm{VaR}}_t^{+} &= \widehat{\mathrm{VaR}}_t + 2\bar{\mathrm{se}}_B \\
\widehat{\mathrm{VaR}}_t^{-} &= \widehat{\mathrm{VaR}}_t - 2\bar{\mathrm{se}}_B, \qquad t = 2001, 2002, \ldots, 2250
\end{aligned}$$

to our VaR predictions.[20] Since the distribution of bootstrapped parameter estimates is asymptotically normal we can expect to find the true parameter within the two standard error interval in 95% of all cases. With other words we can assert with 97.5% confidence that the upper bound of the tolerance interval will be exceeded by portfolio losses only in 5% of all cases. Figure 4 shows $\widehat{\mathrm{VaR}}_t$ together with the tolerance interval for experiment II (test data). In experiment I the results obtained were not as good. The lower/upper border of the two standard error interval lead to 10.4-6.8-3.6 model failures in 100 days.

5 Conclusions

The results presented in this paper confirm earlier findings [7] that neural networks can successfully be applied to VaR estimation. We showed that connectionist VaR estimates based on Mixture Density Networks do not seem to be subject to a systematic bias. Standard errors calculated using a bootstrap procedure are not as large as to prevent the practical application of a connectionist VaR model. In this paper a simple and slow estimation technique was used. The use of more sophisticated model estimation and generalization techniques may speed up network learning and improve the quality of VaR estimates. Furthermore it is important to show the impact of random network initialization and model misspecification on the error of VaR estimates to explain possible sources of time-varying bias and standard errors.

References

[1] Christopher M. Bishop. Mixture density networks. Technical Report NCRG/94/004, Neural Computing Research Group, Aston University, Birmingham, February 1994.

[2] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, UK, 1995.

[20] In statistical terminology we construct a tolerance interval.

[3] Jacob Boudoukh, Matthew Richardson, and Robert Whitelaw. The best of both worlds. *Risk*, pages 64–76, May 1998.

[4] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

[5] James Hamilton. A quasi-bayesian approach to estimating parameters for mixtures of normal distributions. *Journal of Business & Economic Statistics*, 9(1):27–39, 1991.

[6] Philippe Jorion. *Value at Risk: The new benchmark for controlling market risk*. Irwin, Chicago, London, Singapore, 1997.

[7] Hermann Locarek-Junge and Ralf Prinzler. Estimating value-at-risk using neural networks. In H.-U. Buhl and C. Weinhardt, editors, *Informationssysteme in der Finanzwirtschaft IF98*, 1998.

[8] R. Tibshirani. A comparison of some error estimates for neural network models. Working Paper, University of Toronto, 1995.

[9] Subu Venkataraman. Value-at-risk for a mixture of normal distributions: The use of quasi-bayesian estimation techniques. *Economic Perspectives*, 21(2), 1997.

[10] Andreas S. Weigend and Blake LeBaron. Evaluating neural network predictors using bootstrapping. *ICONIP'94-Soul*, 1994.