Optimal portfolio management using neural networks - a case study

Jürgen Franke and Matthias Klein
Department of Mathematics
University of Kaiserslautern
D-67653 Kaiserslautern
Germany

29. Juni 1999

1 Introduction

Neural networks are now a well-established tool for solving classification and forecasting problems in financial applications (compare, e.g., Bol et al., 1996, Evans, 1997, Rehkugler and Zimmermann, 1994, Refenes 1995, and Refenes et al. 1996a) though many practioners are still suspicious against too evident success stories. One reason may be that the construction of an appropriate network which provides a reasonable solution to a complex data-analytic problem is rarely made explicit in the literature. In this paper, we try to contribute to filling this gap by discussing in detail the problem of dynamically allocating capital to various components of a currency portfolio in such a manner that the average gain will be larger than for certain benchmark portfolios. We base our solution on feedforward neural networks which are constructed employing various statistical model selection procedures described in, e.g., (Anders, 1997, or Refenes et al., 1996b).

Neural networks which are used as the basis of trading strategies in finance should be assessed differently than in technical applications. The task is not to construct a network which provides good forecasts with respect to mean-square error of some quantities of interest or to provide good approximation of some given target values, but to achieve a good performance in economic terms. For portfolio allocation, the main goal is to achieve on the average a large return combined with a small risk. Therefore, we do not consider forecasts of the foreign exchange (FX-) rate time series using neural networks, but we try to get the allocation directly as the output of a network. Furthermore, we do not minimize some estimation or prediction error, but we try to maximize an economically meaningful performance measure, the risk-adjusted return, directly (compare also Heitkamp, 1996).

In the subsequent chapter, we describe the details of the portfolio allocation problem. The following two chapters provide some technical information on how the networks were fitted to the available data and how the network inputs and outputs were selected. In chapter 5, finally, we discuss the promising results.

Acknowledgement: This paper is based on a joined project with Commerzbank AG, Frankfurt am Main. In particular, we are obliged to U. Kern of Commerzbank for his cooperation and continuing advice. More details on this project may be found in (Klein, 1998). The data we used were provided by DATASTREAM. The numerical analysis has been done using the software package ThinksPro.

2 Managing a portfolio of currencies - the problem

We consider a portfolio consisting of four major currencies: German Mark (DEM), British pound (GBP), Japanese yen (JPY) and US dollar (USD). The portfolio is managed from the viewpoint of a DEM-investor who follows a weekly buy-and-hold strategy, i.e. the whole availabe capital is allocated to the four currencies and held unchanged for one week (5 trading days). Only then, the capital may be redistributed. We neglect gains (like interest) and losses (like transaction costs) caused by managing the portfolio as we want to concentrate on the return due to variations in the foreign exchange (FX) rates alone.

There are essentially two statistical approaches to managing a currency portfolio. Using data from the past, we could try to forecast the FX rates GBP/DEM, JPY/DEM, USD/DEM separately, one week ahead and, then, allocate our capital taking these forecasts and perhaps some information about their statistical variability into account. In this paper, we adopt a different approach and try to infer good portfolio weights directly from the data without taking the detour via FX rate predictions.

Before we try to optimize the allocation we have to decide how to quantify the performance of a portfolio. For this purpose, we introduce some notation:

p = number of currencies to be considered ($p = 4$ in our particular application)

$e_i(t)$ = exchange rate of currency no. i with respect to DEM at time t, $i = 1, \ldots, p$

$r_i(t) = \dfrac{e_i(t+5)}{e_i(t)} - 1$ = return of currency no. i over a period of one week (5 trading days), $i = 1, \ldots, p$

$\boldsymbol{\pi}(t) = (\pi_1(t), \ldots, \pi_p(t))^T$ = vector of weights of currencies $1, \ldots, p$ as parts of the portfolio at time t.

The portfolio weights have to satisfy

$$\pi_i(t) \geq 0, \quad i = 1, \ldots, p \quad \text{and} \quad \sum_{i=1}^{p} \pi_i(t) = 1. \tag{1}$$

The return of the portfolio over a period of one week is given by

$$R(\boldsymbol{\pi}(t)) \equiv \sum_{i=1}^{p} \pi_i(t) r_i(t) = \boldsymbol{\pi}^T(t)\mathbf{r}(t)$$

where $\mathbf{r}(t) = (r_1(t), \ldots, r_p(t))^T$ denotes the vector of returns of single currencies. Following the common portfolio theory of Markowitz, the investor bases his decision how to allocate the capital not only on the expected return but, additionally, on the risk. As a measure of risk, we consider the volatility or standard derivation of the portfolio return. Assuming that the volatility is approximately constant in short time intervals, a sample version of the portfolio volatility is given by

$$\begin{aligned} s(\boldsymbol{\pi}(t)) &= \left\{ \frac{1}{5} \sum_{k=0}^{4} \left(\boldsymbol{\pi}^T(t)\mathbf{r}(t-k) - \frac{1}{5} \sum_{l=0}^{4} \boldsymbol{\pi}^T(t)\mathbf{r}(t-l) \right)^2 \right\}^{\frac{1}{2}} \\ &= \sqrt{\boldsymbol{\pi}^T(t) C(t) \boldsymbol{\pi}(t)} \end{aligned}$$

where $C(t) = (C_{ij}(t))_{i,j=1,\ldots,p}$ is a sample version of the $p \times p$ covariance matrix of the single FX rate returns localized around time t :

$$\begin{aligned} C_{ij}(t) &= \frac{1}{5} \sum_{k=0}^{4} (r_i(t-k) - \overline{r}_i(t)) \cdot (r_j(t-k) - \overline{r}_j(t)) \\ \text{with} \quad \overline{r}_i(t) &= \frac{1}{5} \sum_{k=0}^{4} r_i(t-k), \quad i = 1, \ldots, p. \end{aligned}$$

Combining return and risk, we consider the *risk-adjusted return*

$$R^A(\boldsymbol{\pi}(t)) = \frac{R(\boldsymbol{\pi}(t))}{s(\boldsymbol{\pi}(t))}$$

as the performance measure for which the portfolio weights $\boldsymbol{\pi}(t)$ are to be optimized.

For a DEM-investor, the part of the portfolio consisting of DEM is, under the assumptions made above, a riskless investment with return 0. If we consider a portfolio consisting of a fraction γ of DEM and a fraction $(1-\gamma)$ of a risky portfolio consisting only of other currencies than DEM with weights $\boldsymbol{\pi}(t) = (\pi_1(t), \ldots, \pi_p(t))^T$, then the return and sample volatility of the combined portfolio are just given by $(1-\gamma)\ R(\boldsymbol{\pi}(t))$ and $(1-\gamma)\ s(\boldsymbol{\pi}(t))$. To determine an optimal allocation, we, therefore, first may optimize the risky part of the portfolio separately, as this problem does not depend on the investor's willingness to take risks. The investor's attitude to risk can be taken into account afterwards by choosing the DEM-fraction γ in the combined portfolio appropriately.

We are looking for a neural-network-based method which, using information available at time t, provides portfolio weights $\boldsymbol{\pi}(t)$ which on the average guarantee a good performance, the latter measured by the risk-adjusted return $R^A(\boldsymbol{\pi}(t))$. Data, which are eligible as inputs of the portfolio selection procedure, have to be readily available on a daily basis, and we need a sufficient number of observations from the past to estimate the parameters of the function which determines the portfolios weights. For sake of convenience, we therefore restricted ourselves to common financial time series provided by DATASTREAM and to appropriate transformations of these data. The potential inputs can be partitioned into two main types:

- the FX-rate series themselves and transformations of them well-established in trading as so-called *technical indicators*
- other financial data, called *fundamental indicators*.

We used daily fixings of the FX-rates GBP/DEM, JPY/DEM, USD/DEM at Frankfurter Devisenb"rse, 13:00 MEZ, and, additionally, at Barclay's Bank International (London), 16:00 MEZ and at JP Morgan (New York), 22:30 MEZ, to incorporate some information about the short-term fluctuations of the time series of interest. Technical indicators which form the basis of various trading strategies (Müller and Nietzer, 1993) are just functions of the FX-rate time series $e_i(s),\ i = 1, \ldots, p, s \leq t$, e.g.

$$\frac{1}{q}\sum_{k=0}^{q-1} e_i(t-k) \quad \text{(moving average)},$$

$$\sum_{k=0}^{q-1}(e_i(t-k) - e_i(t-k-1))^+ / \sum_{k=0}^{q-1}|e_i(t-k) - e_i(t-k+1)| \quad \text{(relative strength index)},$$

where z^+ denotes the positive part of z; the β of a FX-rate, i.e. the slope of the least-squares regression line fitted to the data pairs $(t-k, e_i(t-k)),\ k = 0, \ldots, q-1$; the historical volatility, i.e. sample standard deviation of the daily log-returns $\log(e_i(t-k)/e_i(t-k-1)),\ k = 0, \ldots, q-1$, etc (for other functions of past data, we considered as potential inputs, compare Klein, 1998). From the viewpoint of neural networks, all technical indicators may be looked at transformed inputs derived from preprocessing the raw input time series $e_i(s),\ i = 0, \ldots, p, s \leq t$.

As fundamental indicators we used the prices of FX-rate forwards, various international interest rates, stock indices for Germany, Great Britain, Japan and the USA, prices and forward prices of commodities like oil or raw metals, etc.

3 Fitting a neural network to the data

As candidates for a portfolio selection procedure, we study only completely connected feed forward neural networks which prescribe a mapping of d input variables $x_1, \ldots, x_d$ to p output variables $z_1, \ldots, z_p$. The mapping is characterized by the structure of the network, i.e. the number H of hidden layers, the number h_k of neurons in each hidden layer, $k = 1, \ldots, H$, and the activation function ψ_k of the neurons in the k-th hidden layer, $k = 1, \ldots, H$. The software, we used, did not allow for different activation functions of neurons in the same layer, but in this paper we study only networks with same activation function $\psi_k \equiv \psi,\ k = 1, \ldots, H$, in all hidden layers anyhow. For ψ, we select a sigmoid function which is antisymmetric around 0:

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

We allow for bias neurons in the input and each hidden layer such that the network function is given by

$$z_j = w_{0j}^{(H)} + \sum_{i=1}^{h_H} z_i^{(H)} w_{ij}^{(H)}\ , \quad j = 1, \ldots, p$$

where, recursively, $z_j^{(k)}$ denotes the output of the j-th neuron in the k-th hidden layer

$$z_j^{(k+1)} = \psi\left(w_{0j}^{(k)} + \sum_{i=1}^{h_k} z_i^{(k)} w_{ij}^{(k)}\right)\ , \quad j = 1, \ldots, h_{k+1}, \quad k = 1, \ldots, H-1,$$

and

$$z_j^{(1)} = \psi\left(w_{0j}^{(0)} + \sum_{i=1}^{d} x_i w_{ij}^{(0)}\right)\ , \quad j = 1, \ldots, h_1.$$

Given the network structure, the network parameters or weights $w_{ij}^{(0)},\ i = 0, \ldots, d,\ j = 1, \ldots, h_1$, and $w_{ij}^{(k)},\ i = 0, \ldots, h_k,\ j = 1, \ldots, h_{k+1}, k = 1, \ldots, H-1$, and $w_{ij}^{(H)},\ i = 0, \ldots, h_H,\ j = 1, \ldots, p$ have to be chosen by optimizing a performance measure on a given training set of data. To stress the dependence on the parameters, we write ϑ for the vector of all network weights, $\mathbf{x} = (x_1, \ldots, x_d)^T$ for the vector of inputs and

$$f_j(\mathbf{x}; \vartheta) \equiv z_j, \quad j = 1, \ldots, p,$$

for the j-th output coordinate of the network with weight vector ϑ given input $\mathbf{x}$.

Given a training set $(\mathbf{x}(t), \mathbf{y}(t)),\ t = 1, \ldots, N$, of input vectors $\mathbf{x}(t)$ and target outputs $\mathbf{y}(t)$, we determine a nonlinear least-squares estimate of the optimal parameter by minimizing the root mean-square error

$$E(\vartheta) = \left\{ \sum_{t=1}^{N} \sum_{j=1}^{p} (y_j(t) - f_j(\mathbf{x}(t);\ \vartheta))^2 \right\}^{\frac{1}{2}} .$$

To get a more stable behaviour of the numerical minimization procedures, we alternatively consider *weight decay* (compare, e.g., Baun, 1994), i.e. introducing a penalty for large parameter values and minimizing

$$\tilde{E}(\vartheta) = E(\vartheta) + \lambda ||\vartheta||^2$$

instead of $E(\vartheta)$. In the applications, discussed later on, we chose $\lambda = 0.5 \cdot 10^{-6}$, the default value of the software used. In the following, $\hat{\vartheta}$ denotes a minimizer of $\tilde{E}(\vartheta)$. Weight decay is a safeguard against the tendency of the neural network training to overfit in particular extreme data sets by increasing the weights more and more to adapt to special features of the training set. Therefore, this regularization technique does not only lead to faster convergence of the numerical algorithms but also improves the ability of the fitted network to generalize to new data (compare, e.g., Bishop, 1995). Equivalently to the above formulation, weight decay can be achieved by minimizing $E(\vartheta)$ under the constraint $||\vartheta||^2 \leq \Lambda$. Assuming a nonparametric regression setting, White (1990) has shown that the network function $f(\mathbf{x}; \vartheta)$ provides a consistent estimate of the true regression function if $\Lambda \to \infty$ (i.e. $\lambda \to 0$) and the number of neurons increases, too, with an appropriate rate depending on sample size $N \to \infty$. In this sense, weight decay fits into the theory of adaptive choice of smoothness parameters in nonparametric regression which strives to achieve an automatic balance between overfit by a network function with too many significant parameters and underfit by a too simple network.

As numerical procedures for minimizing $\tilde{E}(\vartheta)$, we considered three different algorithms: a conjugate gradient procedure, Quickprop, which is a second-order method based loosely on Newton's method, and simulated annealing, all of them as implemented in ThinksPro (compare the handbook by Logical Designs, 1996). The two networks, discussed in some detail in chapter 5, were trained with Quickprop which for this type of problems provided the best compromise between speed and precision.

To reduce the computation time, we tried to use the well-known technique of *early stopping* or *stopped training*. Here, a separate data set, the test set, apart from the training set is used. The error function $E(\vartheta)$ is calculated for the test set for those parameter values ϑ determined from the training set in the course of the numerical iterations. Ideally, $E(\vartheta)$ should increase on the test set once the numerical procedure has achieved a reasonable fit to the data and proceeds into the direction of overfitting, i.e. a minimum of $E(\vartheta)$ on the test set plotted against the number of numerical iterations determines the right point to stop. This ideal situation is illustrated on the left-hand side of Figure 3.1. In our applications, however, this plot looked rather differently. The right-hand side of Figure 3.1 shows, e.g. for network NN01 of chapter 5, $E(\vartheta)$ for a test set (top line) and for the training set (bottom line) plotted against the number of quickprop iterations (in thousands). After an initial increase, $E(\vartheta)$ decreased monotonically on the test set. Perhaps this inapplicability of early stopping is due to our use of weight decay in determining the network parameters which should avoid overfitting to some extent.

Datei 2.1 here

Figure 3.1: Root mean-squared error on training and test set as a function of the number of numerical iterations for an ideal situation (left) and for network NN01 (right)

As coordinates of the input vectors $\mathbf{x}(t)$, selections from the FX-rates and from the technical and fundamental indicators introduced in chapter 2 are considered, where all the information contained in $\mathbf{x}(t)$ is available at day t. The target output $\mathbf{y}(t)$ corresponds to the optimal portfolio weights $\boldsymbol{\pi}^o(t) = (\pi_1^o(t), \ldots, \pi_p^o(t))^T$, which maximize the risk-adjusted return

$R^A(\boldsymbol{\pi}(t))$ over all $\boldsymbol{\pi}(t)$ satisfying (1). Therefore, $\boldsymbol{\pi}^o(t)$ is observable only at day $t+5$. As a *training set,* we considered daily data $(\mathbf{x}(t), \boldsymbol{\pi}^o(t)),\ t = 1, \ldots, N = 1825$, from the years 1989 - 1995. For comparing different networks fitted to the training set we studied their performance on the *validation set* $(\mathbf{x}(t), \boldsymbol{\pi}^o(t)), t = N+1, \ldots, N+M,\ M = 455$, from the period 1996 - 20.09.1997.

Based on the training set, we get a network parameter $\hat{\vartheta}$ by minimizing $\tilde{E}(\vartheta)$ and, for any input vector $\mathbf{x}$, a corresponding output vector $\mathbf{z} = f(\mathbf{x}; \hat{\vartheta})$. $z_1, \ldots, z_p$ are, however, usually no legitimate portfolio weights, as they do not have to satisfy the constraints (1). The right strategy to determine portfolio weights as outputs of a neural network would have been to solve the constrained optimization problem:

$$\tilde{E}(\vartheta) = \min_{\vartheta}! \quad \text{under the constraint that} \quad f_j(\mathbf{x}; \vartheta),\ \ j = 1, \ldots, p, \quad \text{satisfy (1)}$$

for all admissible inputs $\mathbf{x}$. However, the network software we used did not allow for such constraints on the outputs. Therefore, we applied the suboptimal strategy of determining the unconstrained minimizer $\hat{\vartheta}$ of $\tilde{E}(\vartheta)$ and normalizing the outputs afterwards to sum up to 1 (the nonnegativity constraint was automatically satisfied in the applications):

$$\pi_j^n = \frac{f_j(\mathbf{x}; \hat{\vartheta})}{\sum_{i=1}^p f_i(\mathbf{x}; \hat{\vartheta})}, \quad j = 1, \ldots, p.$$

Of course, the normalized outputs $\pi_1^n, \ldots, \pi_p^n$ in general do not solve the constrained optimization problem (compare Klein, 1998, for a simple counterexample), but even this suboptimal network-based portfolio weights showed a satisfactory performance. For comparison, we also studied another portfolio allocation strategy where all the capital is invested into only one currency for which the network output is maximized:

$$\pi_j^m = \begin{cases} 1 & \text{if} \quad f_j(\mathbf{x}; \hat{\vartheta}) = \max_{i=1,\ldots,p} f_i(\mathbf{x}; \hat{\vartheta}) \\ 0 & \text{else} \end{cases} .$$

This approach contradicts the idea of diversifying to reduce the risk but it also performed reasonably well in practice.

4 Selection of network inputs

In chapter 2, we introduced a large set of financial time series which may be used as inputs of the network determining a portfolio allocation: the FX-rate series themselves, lots of functions of their past values, known as technical indicators, and intermarket data like stock index or interest rate series etc. The main task in constructing a suitable network is the selection of an appropriate set of inputs which is not too large but contains enough information to allow for a satisfactory performance. This part of the problem is completely analogous to the selection of independent variables from a large set of candidates well-known from classical linear regression, and there are several procedures, already known from the linear setting, which allow for a systematic selection of input variables. Those procedures which we have used in our application are shortly described in the following. The major source of information is, of course, to draw on the knowledge of experts who manage portfolios of currencies on a regular basis. Their experience was not only used for fixing the large set of potential inputs in advance but also for constructing various promising subsets of actual inputs for the networks.

As a first step, studying the correlation between inputs and the determinants of the target outputs, in our applications the returns 5 days ahead, may uncover strong linear relations which may be exploited to forecast the currencies and to determine the portfolio weights. However, the sample correlations assumed values between 0.1 and 0.25 such that no important linear dependencies were evident.

A sensitivity analysis tries to quantify the influence which a single input has on the output of a network. Here, one has the goal in mind to identify a few important inputs and to remove the others from the system. There are lots of proposals in the literature (compare, e.g., Anders, 1997, or Refenes et al., 1996) how to do such a sensitivity analysis for neural networks in a computationally feasible manner. We used the normalized effect of an input variable in the form implemented in ThinksPro (compare Logical Designs, 1996). For each input variable, we consider its average over the training set

$$\overline{x}_i = \frac{1}{N} \sum_{t=1}^{N} x_i(t), \quad i = 1, \ldots, d.$$

Then, $\mathbf{x}^{[i]}(t) = (x_1(t), \ldots, x_{i-1}(t), \overline{x}_i, x_{i+1}(t), \ldots, x_d(t))^T$ is the input vector at time t with $x_i(t)$ replaced by $\overline{x}_i$. Let, for $i = 1, \ldots, d$,

$$z_j(t) = f_j(\mathbf{x}(t); \hat{\vartheta}) \quad \text{and} \quad z_j^{[i]}(t) = f_j(\mathbf{x}^{[i]}(t); \hat{\vartheta}), \quad j = 1, \ldots, p,$$

be the network outputs at time t using the original input vector $\mathbf{x}(t)$ and the reduced input vector $\mathbf{x}^{[i]}(t)$ with the variability of the i-the input removed. Then, the *effect* of the i-th input is

$$\text{eff}_i = \left\{ \frac{1}{Np} \sum_{t=1}^{N} \sum_{j=1}^{p} (z_j(t) - z_j^{[i]}(t))^2 \right\}^{\frac{1}{2}}$$

and the *normalized effect* is given by

$$\text{eff}_i^n = \frac{\text{eff}_i}{\frac{1}{d} \sum_{k=1}^{d} \text{eff}_k}.$$

If all inputs have the same influence on the output, then $\text{eff}_i^n = 1, \ i = 1, \ldots, d$. Therefore, inputs with $\text{eff}_i^n \ll 1$ should be removed from the network and perhaps replaced by other factors from the set of potential inputs.

The sensitivity analysis pretends that the input time series $x_1(t), \ldots, x_p(t)$ are independent which, of course, is not the case in our application. It is not feasible to take any dependencies between these time series into account but it is possible to avoid at least strong linear relationships. To detect such dependencies, at least between values of the time series at the instant t, one can look at the proportion R_i^2 of variability explained by regressing the i-th input factor linearly on the other input factors. If the largest of these values, say $R_{i^*}^2 = \max_{i=1,\ldots,d} R_i^2$, is close to 1, e.g. ≥ 0.9, then $x_{i^*}(t)$ is almost a linear function of the $x_i(t), \ i \neq i^*$, and the i^*-th input may be removed from the model.

Principal component analysis (PCA) represents another possibility for finding near linear relationships between the input factors and, additionally, for reducing and transforming the input set to a set of $d' < d$ orthogonal factors. It is based on the singular value decomposition of the $(N \times d)$-matrix $\boldsymbol{X} = (x_i(t))_{i=1,\ldots,p;t=1,\ldots,N}$:

$$\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$$

where Σ is a $(d \times d)$-diagonal matrix with diagonal entries $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d \geq 0$, the *singular values*, and $\boldsymbol{U}, \boldsymbol{V}$ are $(N \times d)$- and $(d \times d)$-matrices satisfying $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T = I_d$ with I_d denoting the $(d \times d)$-identity matrix. If some of the entries are linearly dependent, then $\sigma_i = 0$ for some i. Near linear dependence is characterized by several small singular values. The columns $\mathbf{p}_i = (p_i(1), \ldots, p_i(N))^T$, $i = 1, \ldots, d$, of the $(N \times d)$-matrix $\boldsymbol{X}\boldsymbol{V} = \boldsymbol{U}\Sigma$ are called the *principal components*. In the literature on input selection for neural networks (compare, e.g., Anders, 1997, and Rehkugler and Zimmermann, 1994), it is sometimes recommended to apply PCA to the original inputs and to use those principal components $\mathbf{p}_1, \ldots, \mathbf{p}_d$, as new inputs which correspond to the singular values $\sigma_1 \geq \ldots \geq \sigma_{d'} \gg 0$ which are significantly larger than 0. An immediate drawback of this approach is the often observed fact that the principal components have no immediate interpretation in terms of the underlying real setting. Another problem will be discussed in the context of the applications in the subsequent chapter.

5 Managing a portfolio of currencies - the results

In the course of the project, various neural networks were studied differing with respect to network structure (number of hidden layers, number of neurons and type of activation function in each layer), learning rule applied to determine the network parameters $\hat{\vartheta}$ from the training set, period of time series data used for training, selection and preprocessing of inputs. The networks were fitted to the data, and their performance was evaluated on the validation set. In this chapter, we discuss only two representative networks illustrating what can be achieved and what problems may occur.

The first network NN01 had $H = 2$ hidden layers with $h_1 = 9$ and $h_2 = 5$ neurons respectively, $p = 3$ outputs corresponding to the portfolio weights of GBP, JPY and USD and $d = 17$ inputs consisting of the following time series: 4 interest rate spreads, 3 FX-rates, 3 FX-rate forwards, oil price, 4 stock indices and 2 technical indicators. The network is characterized by 230 parameters which have been estimated from the training set using 113 245 iterations of Quickprop.

The second network NN02 differed from NN01 by the number $h_1 = 7$ of neurons in the first hidden layer and by the number and selection of the $d = 21$ inputs: 4 interest rate spreads, 3 FX-rates, 3 FX-rate forwards, 4 stock indices and 7 technical indicators. The 212 network weights were calculated using 58 398 iterations of Quickprop.

Before presenting the results, we have to introduce some performance measures of currency portfolios. A standard performance measure for neural networks is the root mean-squared error (RMSE) on the validation set $(\mathbf{x}(t), \boldsymbol{\pi}^o(t)), \ t = N+1, \ldots, N+M$, where $\boldsymbol{\pi}^o(t)$ are the optimal portfolio weights introduced in chapter 3, i.e. for, e.g., the normalized network outputs $\boldsymbol{\pi}^n(t) = (\pi_1^n(t), \ldots, \pi_p^n(t))^T$:

$$RMSE = \left\{ \frac{1}{M} \sum_{t=N+1}^{N+M} ||\boldsymbol{\pi}^o(t) - \boldsymbol{\pi}^n(t)||^2 \right\}^{\frac{1}{2}}.$$

As the $\boldsymbol{\pi}^o(t)$ are known only in retrospect, we cannot expect that they are well approximated by even the best portfolio weights based only on information available at time t. Moreover, RMSE does not describe the economic performance of the portfolio which is of main interest for investors. Therefore, we consider the following alternatives:

$$|\mathcal{A}^+| = |\{t;\ R(\boldsymbol{\pi}^n(t)) \geq 0,\ \ N+1 \leq t \leq N+M\}|$$

$$|\mathcal{A}^-| = |\{t;\ R(\pi^n(t)) < 0,\ \ N+1 \leq t \leq N+M\}|$$

are the numbers of all time points for which the allocation $\boldsymbol{\pi}^n(t)$ led to a nonnegative return resp. a negative return.

$$\overline{R} = \frac{1}{M} \sum_{t=N+1}^{N+M} R(\boldsymbol{\pi}^n(t))$$

is the mean return over the validation set. Analogously,

$$S = \left\{ \frac{1}{M-1} \sum_{t=N+1}^{N+M} (R(\boldsymbol{\pi}^n(t)) - \overline{R})^2 \right\}^{\frac{1}{2}}$$

is the corresponding sample standard deviation or volatility of the portfolio returns. Combining both quantities to

$$SR = \frac{\overline{R}}{S},$$

we get the *Sharpe-ratio* which may be interpreted as a risk premium. Usually, the Sharpe-ratio is defined as the difference between the return of the risky portfolio and the return of a riskless asset (Steiner and Bruns, 1998), but in our setting a riskless investment would be a portfolio consisting only of DEM having return 0. Therefore, our definition of the Sharpe-ratio is conforming to the common one.

$$\text{POR} = \frac{\frac{1}{|\mathcal{A}^+|} \sum_{t \in \mathcal{A}^+} R(\boldsymbol{\pi}^n(t))}{\frac{1}{|\mathcal{A}^-|} \sum_{t \in \mathcal{A}^-} R(\boldsymbol{\pi}^n(t))}$$

is the *pay-off-ratio* provided $|\mathcal{A}^+|,\ |\mathcal{A}^-| \neq 0$. A value $\text{POR} > 1$ implies that the mean of positive returns is larger than the mean of negative returns. It does not contain information on the frequency of positive returns. Therefore, only POR and $|\mathcal{A}^+|$ together provide enough information about the performance of an allocation strategy.

Let $R_{(1)} \geq R_{(2)} \geq \ldots \geq R_{(M)}$ denote the ordered returns $R(\boldsymbol{\pi}^n(t)),\ t = N+1, \ldots, N+M$.

$$L_\beta = \frac{\sum_{t=1}^{[\beta M]} R_{(t)}}{\sum_{t \in \mathcal{A}^+} R(\boldsymbol{\pi}^n(t))}$$

is called the *luck coefficient* depending on a small parameter $0 \leq \beta < 1$, the fraction of days with particularly large returns. We use only $\beta = 0.05$ and wide $L \equiv L_{0.05}$. L_β provides information how far the return of a portfolio over a period of time is determined by a small number $[\beta M]$ of lucky days which, presumably, will not show up in the near future again. An allocation strategy which outperforms other portfolios but has a much higher luck coefficient is of doubtful value as its good performance in the validation set may be due to sheer luck.

$$\overline{\text{RAR}} = \frac{1}{M} \sum_{t=N+1}^{N+M} R^A(\boldsymbol{\pi}^n(t))$$

is the mean of adjusted returns. Finally, K denotes the sample correlation between the adjusted returns $R^A(\boldsymbol{\pi}^n(t)),\ t = N+1, \ldots, N+M$, and the adjusted returns $R^A(\boldsymbol{\pi}^o(t)),\ t = N+1, \ldots, N+M$, of the optimal portfolio weights.

The economically most interesting performance measure is the *accumulated return* which provides information about the total gains and losses during the period of investment. As we want to use the allocation procedure as basis of a buy-and-hold strategy where the portfolio may be changed only every week (after 5 trading days) we consider the accumulated return of such a stategy. Let $\mathcal{A}_1, \ldots, \mathcal{A}_5$ be the set of all Mondays, Tuesdays, ..., Fridays in the validation period $t = N+1, \ldots, N+M$, and let $|\mathcal{A}_\mu|, \mu = 1, \ldots, 5$, denote the number of such days. Then, the accumulated return for portfolios reallocated every Monday,...,Friday is given by

$$\text{KR}_\mu = \Pi_{t \in \mathcal{A}_\mu} (1 + R(\boldsymbol{\pi}^n(t)) - 1,\ \ \mu = 1, \ldots, 5.$$

The longer the total period of investment the higher the accumulated return will be. Therefore, often the *annualized accumulated return* is considered, i.e.

$$\text{KR}_\mu^{ann} = (1 + \text{KR}_\mu)^{\frac{1}{T}} - 1\ , \quad \mu = 1, \ldots, 5$$

where T is the length of the period of investments measured in years. In our applications, the validation set corresponds to $T = 1.75$ a, and the training set to $T = 7a$. KR_μ^{ann} describes the average annual interest on the invested capital.

As the accumulated return up to a fixed instant of time provides only information about the average return and not about the risk, it is useful to consider the accumulated return $\mathrm{KR}_\mu(\tau)$ up to time τ as a function of τ and to plot the corresponding gain-/loss-curve. Its fluctuations are related to the risks taken.

In spite of a large positive accumulated return, the gain-/loss-curve will not increase monotonically. The investor has to endure periods of losses where the portfolio value decreases. The *maximum drawdown* is given as

$$MD_\mu = \min_{\tau' \in A_\mu} \frac{\min\{\mathrm{KR}_\mu(\tau);\ \tau \in A_\mu,\ \ \tau > \tau'\} - \mathrm{KR}_\mu(\tau')}{1 + \mathrm{KR}_\mu(\tau')},\ \ \mu = 1, \ldots, 5.$$

MD_μ assumes a large negative value; it provides information about the maximum relative loss in accumulated return between a time τ' and a later time τ which occured during the whole validation period. Among two allocations with the same average return, the investor would prefer that one with the smaller (in absolute value) maximum drawdown. In particular, the accumulated return will not depend so much on the time when the investor decides to liquidate the whole portfolio and to cash in his profit. MD_μ provides a quantification of the risk of an allocation strategy, however an incomplete one, as it does not provide information about the distance in time between that pair $\tau' < \tau$ for which the extremum is assumed.

The performance measure introduced above have been defined in terms of the normalized portfolio weights $\boldsymbol{\pi}^n(t)$ and of the data in the validation set. Analogously, they may be evaluated for the other portfolio allocation rules, e.g. for the maximized network outputs $\boldsymbol{\pi}^m(t)$, or for the data of the training set.

As benchmarks for assessing the relative performance of the network-based allocations, we consider four simple deterministic portfolios with a time-invariant allocation: three consist of only one of the currencies GBP, JPY and USD, and the fourth one called the equilibrium portfolio consists of 1/3 of GBP, JPY and USD each or, if we are taking DEM into account too, of 1/4 of each of the then four currencies. We have also compared $\boldsymbol{\pi}^n(t)$ and $\boldsymbol{\pi}^m(t)$ with some real currency portfolios used in practice, and those are also outperformed by the network-based allocations.

————————- Table 5.1-5.3 somewhere here

Table 5.1 presents various performance measures, introduced above, evaluated for the data of the training period. As those data have been used for estimating the parameters, they only serve as a reference for the corresponding quantities of the validation set. The column *optimal* corresponds to the optimal portfolio weights $\boldsymbol{\pi}^o(t)$ which require perfect knowledge of the FX-rates one week into the future. Mark that even the optimal allocation could not always achieve a positive return as for 491 data in the trainig period, all three FX-rates GBP/DEM, JPY/DEM and USD/DEM decreased over the corresponding week (and for 372 days, all of them increased). The columns *USD, JPY, GBP* show the performance of the three homogeneous benchmark portfolios consisting of one currency only, and the column *EQ* represents the benchmark equilibrium portfolio. The two right-most columns show the performance of the neural network-based allocations $\boldsymbol{\pi}^n(t)$ (column *NN-norm*) and $\boldsymbol{\pi}^m(t)$ (column *NN-max*).

Table 5.2 and 5.3 contain the annualized accumulated return and the corresponding maximum drawdown, again for the training period. Here, we distinguish between weekly holding periods starting from Monday, ..., Friday resp. The average of these 5 values is given in the last line of the table.

————————————- Table 5.4 somewhere here

The real test for the neural network-based allocations is given by the data of the validation period. The results are contained in Table 5.4. For the common measures used for quantifying the performance of neural networks, $\boldsymbol{\pi}^n(t)$ and $\boldsymbol{\pi}^m(t)$ are well within the range of benchmark values. The root mean-squared error of portfolio weights is considerably smaller for the equilibrium portfolio whereas the number of allocations with positive returns, corresponding roughly to the number of correct upwards/downwards-predictions for forecasting networks, is largest for the pure GBP-portfolio. Mark that in the validation period for 133 (70) days all three FX-rates increased (decreased) over the corresponding week. The portfolio return volatility S is, as expected, smaller for the two diversified strategies, i.e. for the equilibrium portfolio and for the normalized network allocations.

For two of the economically important measures, the mean return and the Sharpe-ratio, both network strategies outperform all of the benchmarks considerably, and, looking at the luck coefficient, this good performance is not due to a few extreme returns but to a good long-term behaviour. The normalized portfolio weights $\boldsymbol{\pi}^n(t)$ provide a slightly smaller mean return than $\boldsymbol{\pi}^m(t)$, but as they prescribe a less risky allocation, their Sharpe-ratio is better. Looking at the mean of risk-adjusted returns, the pure GBP-portfolio assumes the largest value due to an extremely strong pound during the whole year 1996, i.e. during the major part of the validation period. The networks are not adapted to such an extreme situation; their task is to provide allocation rules which perform well in the long run. In the later part

of the validation period (01.01.1997 - 29.09.1997) the pound was replaced as the strongest currency by the US-dollar, and now the network allocations in most cases provided a higher return than the GBP-portfolio (compare Tables 5.7 and 5.8).

————————————- Table 5.5-5.6 somewhere here

As discussed above, the most important economic criterion for comparing investments is the accumulated return. Table 5.5 contains the annualized accumulated returns for the validation period, and Table 5.6 provides the maximum drawdown as a measure of risk. It is obvious from these results that the performance of the network crucially depends on the particular day of the week where the portfolios may be reallocated. The network strategies are best for the holding period Tuesday-Tuesday, where they provide a much higher accumulated return than any of the benchmarks and still show the smallest maximum drawdowns. Network NN01 is als quite good for Mondays and Wednesday, whereas, even compared to the quite simple benchmarks, it is not performing well for Thursdays and Fridays. These observations are confirmed by the corresponding gain-/loss-curves which are shown here for the best day of the week (Tuesday, Figure 5.1) and for the worst day of the week (Thursday, Figure 5.2). The varying performance of the neural network for different days of the week are not so surprising if we recall the well-known fact that patterns of trading in stocks or currencies at the start and at the end of a week typically differ too. Therefore, we cannot expect one neural network to provide good allocations for all possible placements of the start of holding period during the week. Therefore, the final system which has been implemented as a consequence of this case study, consists of six different networks which provide different portfolio allocations together with information for which situations (due to past experience) these results may be trusted (compare Klein, 1998, for the details).

Figure 5.1 here

Figure 5.1: Accumulated return as a function of time during the validation period 02.01.1996 - 30.09.1997 (start of holding period: Tuesday - neural network NN01).

Figure 5.2 here

Figure 5.2: Accumulated return as a function of time during the validation period 04.01.1996 - 02.10.1997 (start of holding period: Thursday - neural network NN01).

To investigate the uniformity of neural network performance, we divided the validation period into two parts: 1996 and the first three quarters of 1997. Tables 5.7 and 5.8 present the corresponding annualized accumulated returns. The absolute values of the latter table usually are larger, but that is due to the economic situation during that period (compare the theoretically possible optimal performance). The performance of network NN01 was particularly bad for the Friday-Friday-period in 1997. On those days of the week, to which this network seems to be adapted, the performance is uniformly good if it is judged relative to the performance of the benchmarks.

———————————— Tables 3.7-3.8 somewhere here

In the following, we consider network NN02 to illustrate some of the problems occuring in input selection. This neural network uses 21 input variables where economic reasoning implies the presence of some strong dependencies. A FX-rate forward price should, e.g., ideally be a function of the corresponding FX-rate and of certain interest rates in the two countries involved. The singular value decomposition of the design matrix $\boldsymbol{X}$ (compare chapter 4 - following Anders, 1997, the columns of the matrix have been normed beforehand due to numerical reasons) results in singular values between 2.35 and 0.02, where the smallest ones are rather close to 0. Therefore, there seem to be some near multicollinearities among the inputs, which is confirmed by a correlation analysis. As an alternative to NN02, we therefore considered a network with the same structure but with original 21 inputs replaced by the 18 linearly independent principal components corresponding to the 18 largest singular values. We call this network NN02_{PC}.

As computationally simpler way to reduce the number of inputs is looking at the proportions R_i^2 of variability of the i-th input factor explained by the others, $i = 1, \ldots, d$. Removing successively the three inputs with the largest R^2- values, which happened to be two FX-rates and one FX-rate-forward, resulted in R^2- values which were all less than 0.9. Therefore, we did not remove further input variables. The reduced network with 18 inputs only is called NN02_{red}.

————————————- Table 3.9 somewhere here

For the three networks NN02, NN02PC and NN02_{red}, Table 5.9 contains the error function (RMSE) and the annualized accumulated return averaged over the five days of the week. As usual, we distinguish between training and validation period and between the two network-based allocation NN-norm and NN-max. As expected, the removal of near linear dependencies among the inputs improves the RMSE, but, surprisingly, it considerably deteriorates the economic performance measured by the accumulated returns. An explanation may be that linear model reduction techniques are only of restricted value in an highly nonlinear situation. This argument is assisted by looking at the sensitivity analysis based on the normalized effects. For the original network NN02, $\text{eff}_i^n, i = 1, \ldots, 21$ assume values between 0.776 and 1.251, therefore providing no reason to remove an input variable due to a too small influence on the output given the other inputs.

One of the main criticisms against the application of neural networks as described above is the implicit dependence of the final results on the validation set. The network parameters are estimated from the training set, usually for lots of different

networks. These are compared using the validation set, and the network performing best on the validation set is presented as the result of the data analysis. To provide a fair assessment of the network performance, a third data set is needed which is neither used for parameter estimation nor for model building, i.e. for selecting the network structure. To meet those objections, we have tested the networks of the final implementation also on data which have become available only after the end of the validation period. Figure 5.3 shows the gain-/loss-curves for portfolios held from Tuesday to Tuesday during the test period 06.01.1998 - 15.09.1998. The neural network NN03 is that out of six networks in the final implementation which has proved to be the best one for that particular day of the week. The parameters of the network originally have been estimated from the training set, and, then, the network has been selected due to its performance on the validation set ending on 29.09.1997. As the allocation strategies should be used in 1998, we retrained the network, ie. we updated the parameter estimates, using all the data up to end of 1997. Again, both network-based allocations economically outperform all of the benchmarks in the long run on a really (practically) independent set of data.

Figure 5.3 here

Figure 5.3: Accumulated return as a function of time during the final test period 06.01.1998 -15.09.1998 (start of holding period: Tuesday - neural network NN03).

6 Conclusion

The case study presented in detail above illustrates the chances and problems of applying neural networks to nonlinear financial time series where the high-dimensional inputs allow for incorporating the large amount of information in a realistic manner. We have only used numerical information and intentionally avoided to take qualitative information like the expectation of traders into account. The network output should present a proposal for financial investments which is independent of subjective opinions. Then, it may serve as an additional and different source of information about future developments which the practitioner can compare with his personal expectations based on intuition and economic insight.

The case study shows that finding a good network in terms of economic performance is a complex task. The application of standard statistical model selection techniques is yet in its infancy as there is still a considerable gap between theory and the options presented by standard neural network software. Moreover, traditional purely linear statistical techniques have to be used with caution. On the other hand, once a suitable network structure has been found it may be used for some time, perhaps with occasional retraining of the network parameters which, in contrast to selecting inputs and network layout, is more or less an automatic task. We have found in the above context and also in a different study of long-term neural forecasting of stock prices (Franke, 1999, chapter 3) that a well-chosen network performs well for quite some time.

A particular feature of the allocation problem discussed in this paper was the dependence on the particular day of the week at which the weekly holding period of the portfolio started. This effect is compatible with economic intuition. It has to be taken into account in constructing a neural network based allocation rule. No single network is able to perform well in all situations, i.e. for all starting days. In our final implementation, we therefore combined different networks in a more or less ad-hoc manner, each of them providing a proposal for allocation and some information in which situations this particular network usually performs well. A more systematic manner of combining different networks, each tuned to a particular situation, would be to use a system of competing networks (Pawelzik et al., 1996). The (technical) applications presented up to now are however not exactly fitting our particular problem and, again, cannot be used relying mainly on standard software.

Finally, we remark once more that the portfolio weights which we used are suboptimal anyhow as they do not coincide with correct constrained least-squares solutions. This may be the reason that the maximizing strategy $\boldsymbol{\pi}^m(t)$ looks better than the normalized on $\boldsymbol{\pi}^n(t)$ where the latter is in principle economically more attractive from the point of view of risk diversification. Nevertheless, the suboptimal network allocations already perform reasonably well such that a correct approach should even do better once the necessary neural software is available.

Literatur

[1] Refenes a, b

[2] Bol, G., Nakhaeizadeh, G. and Vollmer, K.-H. eds.: Finanzmarktanalyse und -prognose mit innovativen quantitativen Verfahren - Ergebnisse des 5. Karlsruher Íkonometrie-Workshops. Physica-Verlag, 1996.

[3] Evans, O.: Short-term currency forecasting using neural networks. ICL Systems Journal, 279-302, 1997.

[4] Franke, J.: Nonlinear and nonparametric methods for analyzing financial time series. In: Operations Research Proceedings 98, eds. P. Kall and H.-J. Luethi, Springer, Berlin-Heidelberg-New York, 1999.

[5] Heitkamp, D.: Methodische Aspekte bei der Entwicklung von Tradingmodellen auf der Basis Neuronaler netze. Wirtschaftsinformatik 38, 238-292, 1996.

[6] Pawelzik, K., J. Kohlmorgen und K.-R. Müller: Annealed Competition of Experts for a Segmentation and Classification of Switching Dynamics. Neural Computation, 1996.

[7] Refenes, A.-P., A.N. Burgess und Y. Bentz: Neural Networks in Financial Engineering: a Study in Methodology. IEEE Trans. on Neural Networks, 1996.