EUFIT 95, Third European Congress on Intelligent Techniques and Soft Computing,
28–31 August 95, Aachen, Germany

A NEURAL NETWORK APPROACH FOR PREDICTING THE SALE OF ARTICLES IN SUPERMARKETS

Frank M. Thiesing, Ulrich Middelberg, Oliver Vornberger
University of Osnabrück
Dept. of Math./Comp. Science
D-49069 Osnabrück, Germany
Tel. +49 541 969 2558
Fax. +49 541 969 2770
frank@informatik.uni-osnabrueck.de
http://brahms.informatik.uni-osnabrueck.de/prak/prakt.html

Keywords: Feedforward Multilayer Perceptron, Time Series Prediction, Parallel Computing

1 Introduction

Time series prediction for economic processes is a topic of increasing interest. In order to reduce stock-keeping costs, an appropriate forecast of the demand in the future is necessary. In this paper we use neural networks for a short term forecast for the sales of articles in supermarkets. Multilayer perceptrons are trained on the known sale volume of the past for a certain group of related products. Additional information like changing prices and advertising campaigns are also given to the net to improve the prediction quality. The net is trained on a window of inputs describing a fixed set of recent past states by the back-propagation algorithm [1]. For enhancement the algorithm is also implemented on parallel systems.

2 Sale forecast by neural networks

For our project we use the sale information of 53 articles of the same product group in a supermarket. The information about the number of sold articles and the sales revenues in DM are given weekly starting September 1994. In addition there are advertising campaigns for articles often combined with temporary price reduction. Such a campaign lasts about two weeks and has a significant influence on the demand on this article. Sale, advertising and price for two different articles are shown in figures 1 and 2.
We use feedforward multilayer perceptron networks with one hidden layer together with the back-propagation training method [2], [3]. For prediction the past information of n recent weeks is given to the input layer. The only result in the output layer is the sale of one article for the next week. So there is a window of n weeks in the past and one week in the future (see figure 3).

3 Preprocessing the input data

An efficient preprocessing of the data is necessary to input it into the net. All information must be scaled to fit into the interval $[0, 1]$. We assume that the necessary information is given for T weeks in the past. With the following definitions

$$\begin{aligned} ADV_i^t &:= \text{number of advertising days for article } i \text{ within week } t \\ SAL_i^t &:= \text{sale of article } i \text{ within week } t \\ MAXSAL_i &:= \max_{1 \le t \le T} \{SAL_i^t\} \end{aligned}$$

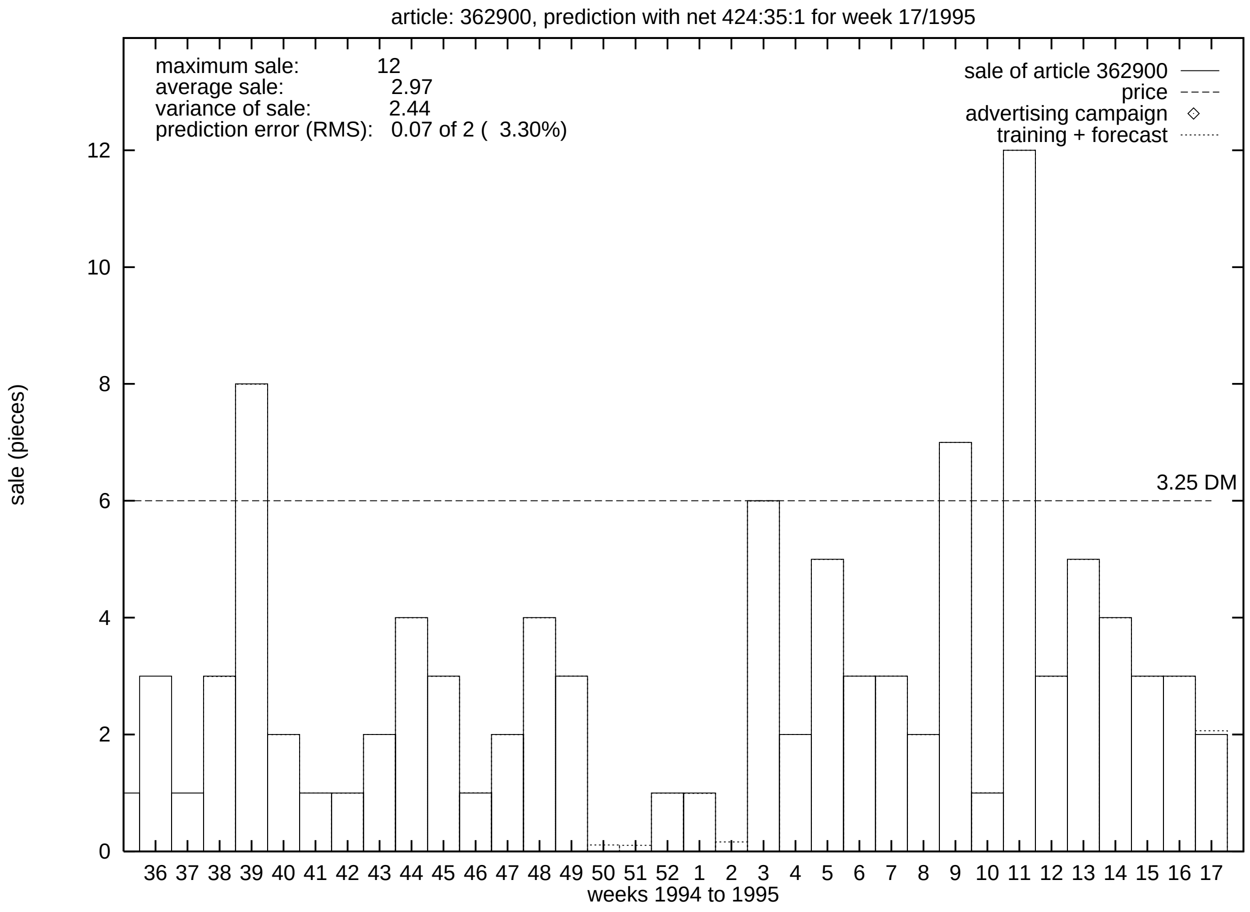

Figure 1: article 362900 (without advertising)

we have decided to use the following inputs for each article i and week t:

$$
\begin{aligned}
adv_i^t &:= \frac{ADV_i^t}{6} \text{ (campaign days of 6 opening days)} \\
pri_i^t &:= \left.\begin{cases} 1.0 & : \text{ price increases} \\ 0.5 & : \text{ price keeps equal} \\ 0.0 & : \text{ price decreases} \end{cases}\right\} \text{within week } t \\
sal_i^t &:= \frac{SAL_i^t}{MAXSAL_i} \cdot 0.8
\end{aligned}
$$

For each article i and recent week t we use a three-dimensional vector:

$$vec_i^t := \left(adv_i^t, pri_i^t, sal_i^t\right)$$

For a week t in the future the vector is reduced by the unknown sale:

$$\widehat{vec}_i^t := \left(adv_i^t, pri_i^t\right)$$

To predict the sale for one article within a week t, we use a window of the last n weeks. So we have the following input vector for each article i:

$$input_i^t := \left(vec_i^{t-n}, vec_i^{t-n+1}, ..., vec_i^{t-1}, \widehat{vec}_i^t\right)$$

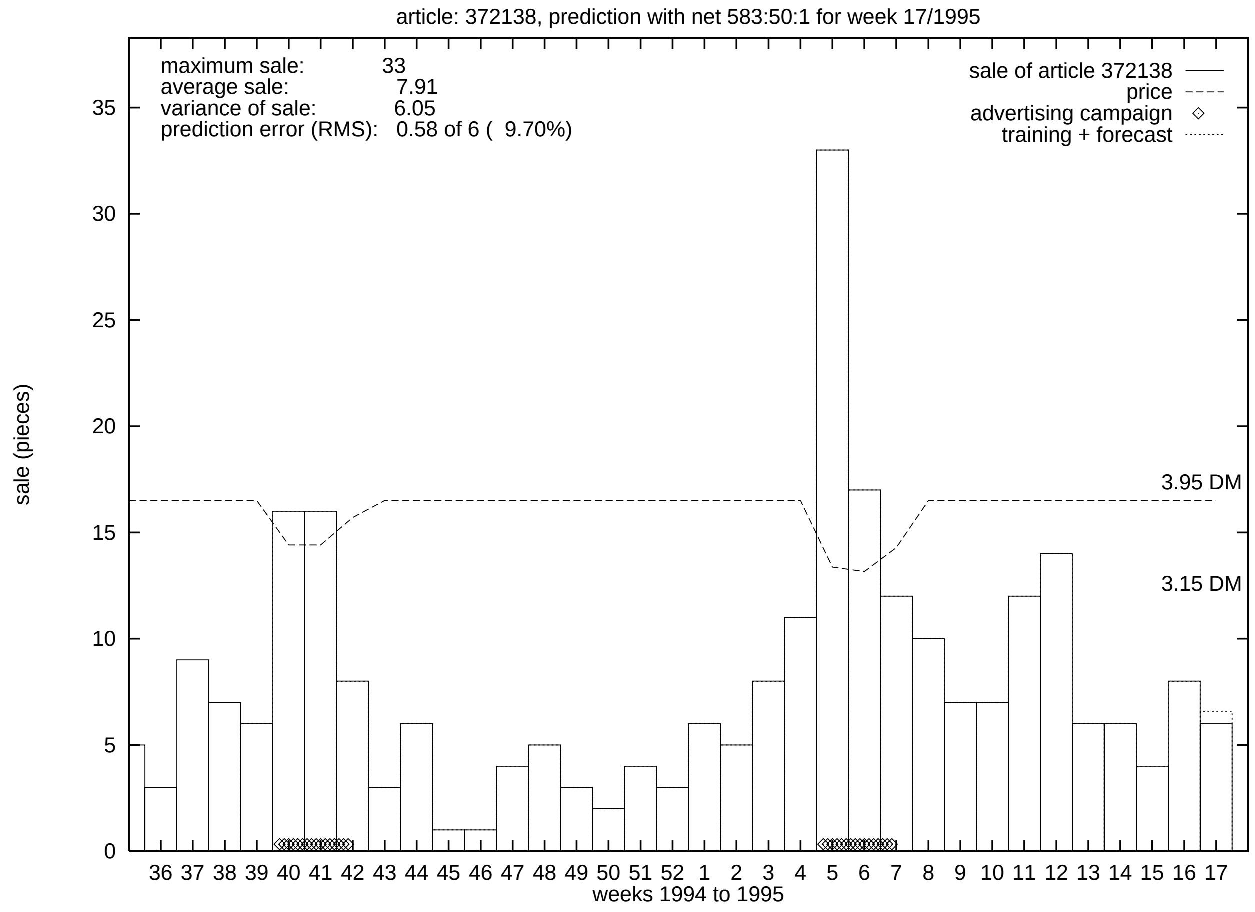

Figure 2: article 372138 (with advertising)

Because all the considered articles belong to one product group, we have quite a constant sales volume of all products. An increasing sale of one article in general leads to a decrease of the other sales. Due to this, we concatenate the *input* vectors of all p articles to get the vector given to the input layer:

$$INPUT^t := \left(input_1^t, input_2^t, ..., input_p^t\right)$$

The sale of article i within week t (sal_i^t) is the requested nominal value in the output layer that has to be learned by one net for this $INPUT^t$ vector. So we have p nets and the i-th net adapts the sale behaviour of article i. Therefor we have a training set with the following pairs (see figure 3):

$$(INPUT^t, sal_i^t) \text{ with } n \leq t \leq T$$

To forecast the unkown sale sal_i^{T+1} for any article i within a future week $T+1$ we give the following input vector to the trained i-th net:

$$INPUT^{T+1}$$

The output value of this net is expected to be the value sal_i^{T+1}, which has to be re-scaled to the value for the sale of article i within week $T+1$:

$$SAL_i^{T+1} = \frac{sal_i^{T+1} \cdot MAXSAL_i}{0.8}$$

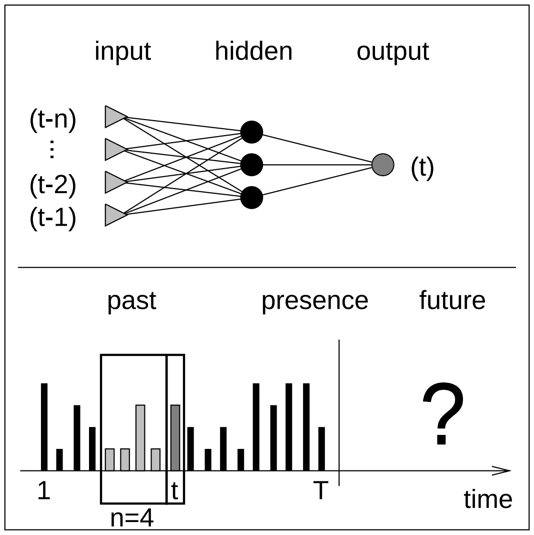

Figure 3: Feedforward multilayer perceptron for time series prediction

4 Parallelization

For enhancement the back-propagation algorithm has been parallelized in different manners: First the training set can be partitioned for the batch learning implementation. The neural network is duplicated on every processor of the parallel machine, and each processor works with a subset of the training set. After each epoch the calculated weight corrections are broadcasted and merged.
The second approach is a parallel calculation of the matrix products that are used in the learning algorithm. The neurons on each layer are partitioned into p disjoint sets and each set is mapped on one of the p processors. After the calculation of the new activations of the neurons in one layer they are broadcasted. We have implemented this on-line training in two variants: For the first parallelization of Morgan et al.[5] one matrix product is not determined on one processor, but it is calculated while the partial sums are sent around on the processor cycle. The second method of Yoon et al.[4] tries to reduce the communication time. This leads to an overhead in both storage and number of computational operations.
The parallel algorithms are implemented with both the message passing environments Parix and PVM on Parsytec's Multicluster and PowerXplorer, based on Transputers resp. PowerPC processors.

5 Empirical results

Here we present the behaviour of eight nets. We have trained nets for

1. article 362900 respectively article 372138 (figures 1 and 2)
2. with two respectively three weeks past information ($n = 2$ respectively $n = 3$)
3. where the number of hidden neurons is $\frac{1}{6}$ respectively $\frac{1}{12}$ of the number of input neurons.

We are using the information of 53 articles in the input layer. The topology of the net is described by the syntax: `(input neurons:hidden neurons:output neurons)`.
The given data is split into a training set (week 36/1994 to week 16/1995) and a test set (week 17/1995). The test set is not trained and only considered to check whether the net has generalized the behaviour of the time series.

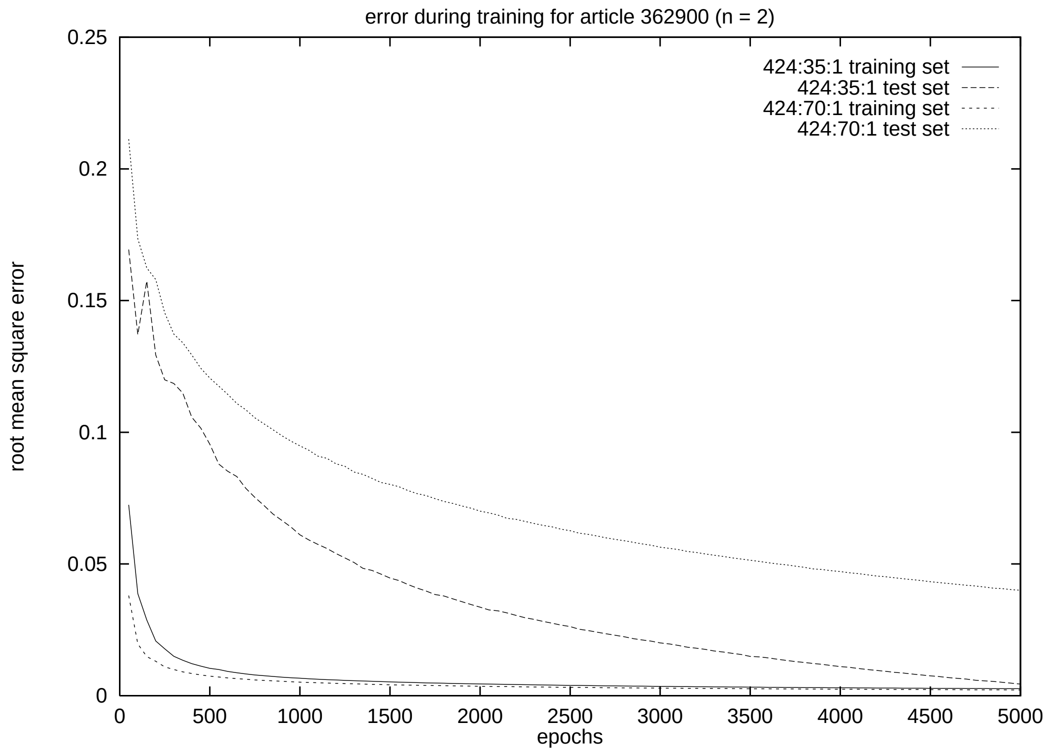

Figure 4: root mean square error of two nets for article 362900 with $n = 2$

With $n = 2$ resp. $n = 3$ we have 31 resp. 30 pairs in the training set and one in the test set. The figures 4 and 5 show the root mean square error on the training and test set, while the nets are learning 5000 epochs. This error is going down immediately on the training set, especially for the larger nets.
More important is the error on the test set — the prediction error. This is better for the smaller nets. They need more epochs to learn the rule of the time series, but because of this they can generalize their behaviour better.
The prediction error of the smaller nets in means of sales can be seen from figures 1 and 2. For the week 17/1995 the forecasted sales are drawn dotted. For both articles the error is smaller than one piece.
The time for training the nets on a sequential SUN SPARC 20-50 can be seen in table 1.

net sizes	424:35:1	424:70:1	583:50:1	583:100:1
5000 epochs	1955 sec	3896 sec	3670 sec	7345 sec

Table 1: Training times of different nets on SPARC 20-50

6 Conclusions and future research

It has been shown that feedforward multilayer perceptron networks can learn to approximate the time series of sales in supermarkets. For a special group of articles neural networks can be trained to forecast the future demand on the basis of the past data together with external information like changing prices and advertising.

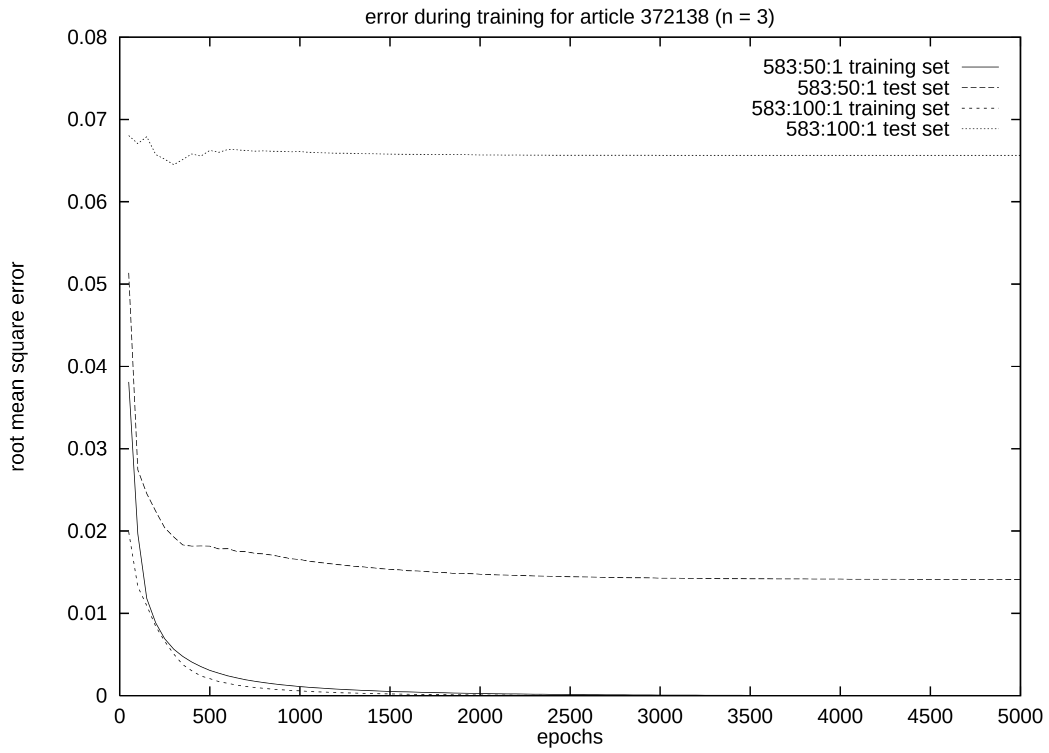

Figure 5: root mean square error of two nets for article 372138 with $n = 3$

For the future the input vectors should be improved: especially season and holiday information have to be given to the net; the value of changing prices can be modelled quantitatively.
One important aim will be the reduction of input neurons. By correlation analysis some of the hundreds of single time series should be merged or denied. This will lead to smaller nets with shorter training times.

References

[1] D.E. Rumelhart, G.E. Hinton, R.J. Williams. *Learning internal representations by error propagation.* In D.E. Rumelhart and J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, pp. 318-362, MIT 1987.

[2] Z. Tang, P.A. Fishwick. *Feed-forward Neural Nets as Model for Time Series Forecasting.* TR91-008, University of Florida, 1991.

[3] V.R. Vemuri, R.D. Rogers. *Artificial Neural Networks – Forecasting Time Series.* IEEE Computer Society Press 5120-05, 1994.

[4] H. Yoon, J.H. Nang, S.R. Maeng. *A distributed backpropagation algorithm of neural networks on distributed-memory multiprocessors.* Proceedings of the 3rd symposium on the Frontiers of Massively Parallel Computation, pp. 358-363, IEEE 1990.

[5] N. Morgan, J. Beck, P. Kohn, J. Bilmes, E. Allman, J. Beer. *The Ring Array Processor: A Multiprocessor Peripheral for Connectionist Applications.* Journal of Parallel and Distributed Computing 14, pp. 248-259, Academic Press Inc., 1992.