Bayesian learning for sales rate prediction for thousands of retailers

Thomas Ragg\textsuperscript{a,}\textsuperscript{*}, Wolfram Menzel\textsuperscript{a}, Walter Baum\textsuperscript{b}, Michael Wigbers\textsuperscript{b}

\textsuperscript{a}Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe, Postfach 6980, Karlsruhe 76128, Germany
\textsuperscript{b}Zeitungsgruppe BILD Vertriebsabteilung, Axel Springer Verlag, Hamburg, Germany

Abstract

Every newspaper publisher has to solve the problem of printing a large number of copies and distributing them to the retail traders trying to keep the return quote as low as possible. To solve this task he needs to estimate as accurately as possible the sales rates for each retail trader. In this paper, we want to show how a prediction system for many thousands of retail traders can be built based on the prediction of the individual sales rates. This prediction is based on a neural network approach. We use a Bayesian learning algorithm to regularize the networks automatically. Furthermore, a top down search based on mutual information is used to optimize the input structure of the networks. The neural network approach reduces the return quota significantly. We conclude with the observation that several data sets are hard to predict and give reasons for that behaviour. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Sales rate prediction; Bayesian learning; Feature selection; Mutual information; Time series prediction; Non-stationarity

1. Introduction

In this paper, we deal with a problem consisting of several thousands of similar prediction tasks. Every newspaper publisher has to solve the problem of printing a large number of copies and distributing them to the retail traders trying to keep the return quote as low as possible. On the one hand the newspaper publisher wants to maximize his profit by selling as many copies as possible. On the other hand, he

\textsuperscript{*} Corresponding author.
E-mail address: ragg@ira.uka.de (T. Ragg).
needs to minimize the number of unsold copies to reduce his costs for production and logistic. To solve this task he needs to estimate as accurately as possible the sales rates for each retail trader.

Prediction of sales rates is a challenging task, since the underlying times series is extremely noisy and bears the danger of overfitting easily [20,13]. At the same time the large number of data sets, e.g., data from several thousands of retail traders, allows for a good evaluation of the approach. Furthermore, promising results of our project are integrated into a real time prediction system and evaluated in practice.

We first show how a solution can be optimized and the limits of the approach can be discovered by making use of stochastic methods. That is, we propose a method to select a subset consisting of a few features well suited for all retail traders to avoid high-dimensional input spaces. This is done by simultaneously exploring the mutual information of the the input–output relation of all data sets. Then a neural network model is trained for each data set with a Bayesian learning rule. The Bayesian framework has the advantage that the necessary regularization term is weighted automatically. This is important because the large number of data sets makes empirical parameter fitting difficult.

Nonetheless, we observed that several data sets are difficult to predict. By making use of a statistical test, i.e., the Kolmogorow–Smirnow test, we found that a high test error results from either one of two sources: (1) non-stationarity of the underlying time series or (2) extremely different behaviour for different days, e.g., the data corresponding to the selling rates on Monday has a different distribution compared to those on Friday.

2. Sales rate prediction

Among the newspapers and magazines published by the Axel Springer Verlag is Germany’s most highly circulated daily newspaper—BILD. More than 5 million copies arrive on time daily at 112,000 sales outlets in the whole of Germany (Fig. 1). The newspaper is printed at eight main printing plants and is distributed via 100 press wholesalers. On an average 4.3 million copies are sold, resulting in a return of 900,000 copies, i.e., 17.3% of the amount ordered. This result is achieved by the wholesalers, who have been using computer based disposition systems since the 1970s to estimate the sales. The prediction part is usually performed with a moving average model whose parameters are adapted manually. The rapidly changing behaviour of the customer makes devising a new automatic delivery system a challenging task. More mobility, changing leisure-time activities, a denser net of points of sale, and less regular customers cause the need for modern regulation systems.

In 1992, the sales department of the publishing house started to develop a new regulation system for the retailer level called ‘RAMBOS’. The main goal was to reduce the percental return quota by 1 point. The reduction of the return quota from 17.3% to 16.3% means a saving of about 3 million German marks each year. Besides controlling the return to the desired level, the system’s task is also—and
Fig. 1. The figure illustrates the problem and its solution: About 5.2 million copies of ‘BILD’ are printed each day, which have to be distributed to the wholesalers and from there to the sales outlets. About 17.3% of the total number of copies is returned each day resulting in a loss of approximately 200,000 German marks.

Fig. 2. The figure illustrates the concept of the system. Several features are computed to encode the problem. A neural network predicts the sales rate while a fuzzy controller determines an extra charge to prevent sell outs. The delivery amount for each sales outlet is the sum of these two values.

this is important—to reduce sold outs, i.e., to reduce lost sales. Further goals were the automation of the whole process by using an adaptive forecasting model, to handle special events by taking calendar effects like vacations, exhibitions and sport events into account and to handle regions with different sales characteristics. The system performs two main steps to calculate a delivery for each outlet (Fig. 2):

"Fig. 1. The figure illustrates the problem and its solution: About 5.2 million copies of ‘BILD’ are printed each day, which have to be distributed to the wholesalers and from there to the sales outlets. About 17.3% of the total number of copies is returned each day resulting in a loss of approximately 200,000 German marks.

Fig. 2. The figure illustrates the concept of the system. Several features are computed to encode the problem. A neural network predicts the sales rate while a fuzzy controller determines an extra charge to prevent sell outs. The delivery amount for each sales outlet is the sum of these two values.

this is important—to reduce sold outs, i.e., to reduce lost sales. Further goals were the automation of the whole process by using an adaptive forecasting model, to handle special events by taking calendar effects like vacations, exhibitions and sport events into account and to handle regions with different sales characteristics. The system performs two main steps to calculate a delivery for each outlet (Fig. 2):"
(1) A neural net forecasts the sales for every retailer. (2) A fuzzy logic component calculates an extra quantity as a reserve.

Over the years the neural network models and the learning algorithms have evolved in several ways, stretching from common backpropagation with a momentum term over Rprop with weight decay to Bayesian learning. In addition to the forecast, an extra quantity is delivered as a reserve for sales that are not covered by the prediction. The purpose of the extra quantity is to avoid sell outs within the limits of the desired return quota. The fuzzy controller can be regarded as a marketing tool that controls the degree of market exploitation.

Input features of the fuzzy controller are, for example, the retailer’s most recent observations of sell outs, the seasonal influence, the variance of the sales, and the type of outlet. The actual overall result in the regions working with the system today is a reduction of the return quota by 1.7 points—meaning 9.8% less return—while guaranteeing the same market exploitation measured by the sold out quota.

3. Network optimization

The sales department of the Axel Springer Verlag developed a prediction system based on neural network models for the time series of each retail trader. The system uses pre-processed input features over the time series \( (x_t) \), e.g. differences like \( (x_t - x_{t-1}) \) or higher differences, long time history measures and general movement indicators. Predictions are made for one week ahead, i.e., always from Monday to Monday, Tuesday to Tuesday, etc. The target data is the difference of the sales rate of next week sale to a short term moving average. In this sense, the neural network is an error correction model for the linear moving average model. All in all, 47 input features were used, resulting in a large input space and a long training time for the single networks. A standard weight decay regularizer was used, i.e., an additional term \( E_R \) was added to the sum of squares error function \( E_D \) [8,1]:

\[
E = E_D + \lambda E_R = \sum_n (y(x_n, w) - t_n)^2 + \lambda \sum_i w_i^2.
\]

The weighting parameter \( \lambda \) was adjusted experimentally by a series of training runs for a few data sets such that the overfitting effect on a validation set was just suppressed and then fixed for all data sets. The fast learning algorithm Rprop was used to minimize the error function [22].

Our cooperation with the Axel Springer Verlag started at this point with the following goals:

- improving the overall performance by using a Bayesian learning rule,
- reducing the number of input features under the constraint that all network models should use the same feature to ensure a simple system maintenance,
- determine reasons for a poor system behaviour.
The reduction of the feature space leads not only to a faster computation in the prediction phase—which is time critical since it has to be done for approximately 1000 retail traders per wholesaler—but also allows for a better approximation of the non-linear target function in a low-dimensional space. This is important since having a data set of finite size and large input dimensions bears the danger of the curse of dimensionality [25,1].

The data used for this research was gained in the area of Münster, a university city in north-west Germany. In total, the time series from 120 retail traders were available for training and testing. Training data ranged over 4 years from 1993 to 1996 while the generalization performance was estimated based on the sales rates of 1997 and 1998. The results were recently confirmed with data from different wholesalers in other areas and integrated into the prediction system of the Axel Springer Verlag.

4. Automatic regularization with Bayesian learning

Firstly, we used a Bayesian learning rule that computes the weighting factor \( \lambda \) automatically. Note that \( \lambda \) usually depends on both the data set used for training and the weight initialization. Thus any empirical method to determine \( \lambda \) is always suboptimal. Bayesian learning for neural networks was introduced by MacKay [10] and provides an elegant theory to prevent neural networks from overfitting by determining \( \lambda \) during the training process without the necessity of additional validation data. See [10,1,16] or [6] for a detailed discussion on Bayesian learning.

Bayes’ theorem is used for the training of neural networks in the following way: Firstly we formulate general knowledge as probability distributions, i.e., a prior distribution of the weights \( p(w) \) and a model of the noise process on the target data \( p(D|w) \) which corresponds to the likelihood function. If we can compute these quantities we can derive the posterior probability by Bayes’ rule:

\[
p(w|D) = \frac{p(D|w) \cdot p(w)}{p(D)},
\]

where \( p(D) \) is a normalization factor which can be computed by integration over the weight space. The goal is to find a weight vector \( w^* \) that maximizes the posterior distribution.

4.1. The prior probability

In the simplest case of Bayesian learning, the prior probability distribution of the weights is assumed to be Gaussian with zero mean. This restricts the complexity of the neural network by searching for small weight values. By denoting \( 1/x \) as
the variance of the Gaussian the prior is of the form
\[ p(w|\alpha) = \frac{1}{Z_w(\alpha)} \exp(-\alpha E_w) \]
\[ = \frac{1}{Z_w(\alpha)} \exp \left( -\frac{\alpha}{2} \sum_{j=1}^{W} w_j^2 \right), \]  
where \( Z_w(\alpha) = (2\pi/\alpha)^{W/2} \) is the normalization factor.

4.2. The likelihood

If a training set \( D = \{(x_1,t_1), \ldots, (x_N,t_N)\} \) of \( N \) input-target pairs is given and if we assume that the target data is generated from a function with additive Gaussian noise, the probability of observing \( t_m \) is given by
\[ p(t_m|x_m,w,\beta) = \frac{\beta}{2\pi} \exp \left( -\frac{\beta}{2} \left\{ y(x_m,w) - t_m \right\}^2 \right), \]  
where \( y(x_m,w) \) denotes the output of the neural network with weight vector \( w \) and \( \beta \) controls the noise in the data, i.e., \( 1/\beta \) is just the variance of the noise. Provided the data points are drawn independently, the probability of the data, called the likelihood is
\[ p(D|w,\beta) = \prod_{m=1}^{N} p(t_m|x_m,w,\beta) \]
\[ = \frac{1}{Z_D(\beta)} \exp \left( -\frac{\beta}{2} \sum_{m=1}^{N} \left\{ y(x_m,w) - t_m \right\}^2 \right), \]  
where \( Z_D(\beta) = (2\pi/\beta)^{N/2} \) is the normalizing constant. It follows that the sum-of-squares error of the data \( E_D(w) = 1/2 \sum\{y(x_m,w) - t_m\}^2 \) just expresses the likelihood function.

4.3. The posterior distribution of the weights

Due to Bayes’ rule (2) we get from (5) and (3) the posterior weight distribution
\[ p(w|D,\alpha,\beta) = \frac{1}{Z_s(\alpha,\beta)} \exp\{-\beta E_D - \alpha E_w\}. \]  
Instead of maximizing the posterior, it is equivalent but simpler to minimize the (negative) exponent.

Making these simple assumptions for the prior and the likelihood, i.e., a Gaussian prior for the weights and a Gaussian noise model, leads to the same error function as in Eq. (1) which is to be minimized to find a maximum of the posterior:
\[ E = \beta E_D + \alpha E_R = \beta \sum (y(x,w) - t)^2 + \alpha \sum w_i^2. \]
Usually fast learning algorithms like Scaled Conjugate Gradient [14] or Rprop [22] are used to find a local minimum of the error function. In this work Rprop is favoured for reasons given below.

\( \alpha \) and \( \beta \) are the so-called hyperparameters since they control distributions of other parameters. \( 1/\alpha \) is just the variance of the distribution of the weights and \( 1/\beta \) is the variance of the Gaussian noise model.

The stochastic framework allows us to use Bayes’ theorem on a second level to treat the hyperparameters \( \alpha \) and \( \beta \), resulting in a computation rule to find their most probable values.

### 4.4. The evidence for \( \alpha \) and \( \beta \)

On the first level of the Bayesian framework a method was developed to optimize the weights where the hyperparameters \( \alpha \) and \( \beta \) were assumed to be known. Due to the Bayesian approach, we are able to determine the hyperparameters automatically during the training process. This can be done by maximizing the probability of \( \alpha \) and \( \beta \) given the data:

\[
p(\alpha, \beta|D) = \frac{p(D|\alpha, \beta)p(\alpha, \beta)}{p(D)}.
\]

The priors for \( \alpha \) and \( \beta \) are usually assumed to be constant in order to give equal importance to all possible values [1]. Such priors are called non-informative. The big disadvantage of uniform priors is that they are not invariant under reparametrization. A widely used method to determine non-informative priors that are invariant under reparametrization is presented by Jeffreys [9]. Gutjahr [5] computed Jeffreys prior for the hyperparameters \( \alpha \) and \( \beta \) were calculated to \( p(\alpha) = 1/\alpha \) resp. \( p(\beta) = 1/\beta \) (cf. [5,6] for an extensive discussion).

In order to optimize the hyperparameters during the training process, we have to maximize the probability of \( \alpha \) and \( \beta \) given the data according to (8). Note that it is assumed that the weight vector is optimized, i.e., the network was trained to a local minimum of the error function. This is denoted in the following with \( E_{\text{MP}}^W \), \( E_{\text{MP}}^D \), etc.\(^1\) Based on the results of the first level and a Taylor-approximation of the posterior distribution of weights as described in [1] we get together with Jeffreys priors for \( \alpha \) and \( \beta \) the logarithm of the posterior on the second level:

\[
\log p(\alpha, \beta|D) \propto \log p(D|\alpha, \beta) + \log p(\alpha, \beta)
\]

\[
= -\alpha E_{\text{MP}}^W - \beta E_{\text{MP}}^D - \frac{1}{2} \log ||A||
\]

\[
+ \frac{W}{2} \log \alpha + \frac{N}{2} \log \beta - \frac{N}{2} \log(2\pi)
\]

\[
- \log \alpha - \log \beta,
\]

\(^{1}\) MP means most probable.
where we assumed the hyperparameters to be independent. The formula is basically the same as in [1] except for the last two summands (cf. [1] or [6,18] for details of the computation). We calculate the partial derivatives of (9) with respect to $\alpha$ and $\beta$ and determine the optimal values of the hyperparameters by setting the results to zero. This provides the following update rules for $\alpha$ and $\beta$

$$\alpha_{\text{new}} = \frac{1}{2E_{\text{MP}}^W} \sum_{i=1}^{W} \frac{\hat{\lambda}_i}{\hat{\lambda}_i + \alpha} - \frac{1}{E_{\text{MP}}^W}$$

(10)

and

$$\beta_{\text{new}} = \frac{1}{2E_{\text{MP}}^D} \left( N - \sum_{i=1}^{W} \frac{\hat{\lambda}_i}{\hat{\lambda}_i + \alpha} \right) - \frac{1}{E_{\text{MP}}^D}.$$ 

(11)

In comparison to the formulas given in [1] we see that these update rules produce smaller values for the hyperparameters. This does not automatically mean that $\lambda$ will also get smaller because the latter depends on the relative values of $\alpha$ and $\beta$.

In a practical implementation we have to find optimal values for $\alpha$, $\beta$ as well as $w$ at the same time. This is solved by an iterative algorithm that periodically re-estimates $\alpha$ and $\beta$, after a minimum of the current error function was reached.

**Iterative algorithm for $\alpha$, $\beta$ and $w$**

1. **Initialize**: Set $(\alpha, \beta)$ to their initial values according to their prior. Choose the components of $w$ from the prior distribution.

2. **Repeat**
   - minimize the error function
   - re-estimate $(\alpha, \beta)$ according to Eqs. (10) and (11)
   - until $(\alpha, \beta)$ converge

Note that when re-estimating the hyperparameters the error function changes. Fig. 3 illustrates this effect for growing $\lambda$ for a simple error function.

In several experiments we observed that after re-estimating the hyperparameters Scaled Conjugate Gradient drives its scaling factor rapidly to its minimal value and thus stays in the same local minimum of the error function. In contrast, the simple but robust learning algorithm Rprop does not show this behaviour, since it firstly does not assume the error function to be (locally) quadratic and secondly has a typically speed up and slow down phase because of the adaption of its parameters. See [22] for details of the algorithm. This prevents it from getting stuck in a local minima after smoothing the error function as in Fig. 3.

A potential conflict might arise from the fact, that Conjugate Gradient methods are already a regularization method by themselves [7], which is not taken into consideration by the Bayesian framework. This is not the case for Rprop, making it the algorithm of choice.
Fig. 3. Illustration of the change of the surface of the error function when re-estimating the hyper-
parameters $\alpha$ and $\beta$. The dashed line corresponds to the error function before the re-estimation of $\alpha$
and $\beta$, i.e., $\lambda = \alpha/\beta = 0.05$. The solid line corresponds to the error function after the re-estimation of
$\alpha$ and $\beta$, i.e., $\lambda = \alpha/\beta = 0.25$. Note that a local minima that is in the outer region will be smoothed
strongly or even disappear as indicated by the arrow.

4.5. Weight groups

A great advantage of the Bayesian approach is that a large number of regularization coefficients can be used since they are determined automatically. Weights can be grouped together where each group gets its own hyperparameter $\alpha_k$. That
is, the error function is changed to

$$E = \beta \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 + \sum_{k=1}^{G} \left( \alpha_k \frac{1}{2} \sum_{w \in W_k} w^2 \right).$$

(12)

The extension of the Bayesian framework to several weight groups is straightforward [11,26,15]. It is common to put all the weights between two layers in one group [1,6]. In case of small data sets or in case of networks with few hidden
units, it is sometimes of advantage to use only one weight group as in the standard weight decay approach, since only one hyperparameter has to be estimated. A special approach is the automatic relevance determination where all weights from each input unit form a separate group [26,12,1]. Note that the biases should also form a separate group which should usually be excluded from regularization, since they compensate for the difference between the sample mean value of the target and the
network output [1]. In the experiments presented below all weights between two layers formed a weight group.

4.6. The evidence of a network

On the third level we can compare different models, for example, networks with
different topologies. Using Bayes’ rule we can write the posterior probability of a
model $H$ as
\[ P(H|D) = \frac{p(D|H)P(H)}{p(D)}. \]

If we assign the same prior to every model, then it is sufficient to evaluate the quantity $p(D|H)$, which is called the evidence for $H$ [10,1]. Marginalizing over $\alpha$ and $\beta$ provides
\[ p(D|H) = \int \int p(D|\alpha, \beta, H) p(\alpha, \beta|H) \, d\alpha \, d\beta. \]

The first factor on the right-hand side is just the evidence for $\alpha$ and $\beta$ from the previous level. Integration over $\alpha$ and $\beta$ is usually done by approximating their posterior distributions on the second level by Gaussians. See [1] and especially [6] for details. The logarithm of $p(D|H)$ is then given by
\[
\ln p(D|H) = -\beta_{\text{opt}} E_D - \alpha_{\text{opt}} E_W - \frac{1}{2} \ln \det(A)
- \frac{N}{2} \ln(2\pi) + \frac{N}{2} \ln(\beta_{\text{opt}}) + \frac{W}{2} \ln(\alpha_{\text{opt}})
+ 2 \ln \sqrt{2\pi} + \frac{1}{2} \ln \left(\frac{2}{\gamma}\right) + \frac{1}{2} \ln \left(\frac{2}{N-\gamma}\right). \tag{13}
\]

Note that the evidence depends on the determinant of the Hessian, $\det(A)$; i.e., the product of its eigenvalues. This makes the result more sensitive to little deviations caused by approximations. Nonetheless, the evidence is usually negatively correlated with the generalization error [11,26,21] giving us a hint which networks to prefer.

5. Feature selection based on mutual information

There exists a variety of feature selection procedures for the combinatorial problem of selecting the optimal subset of $k$ features out of a total of $n$ features [4,1]. They can be distinguished by the used optimization criterion, i.e., a method that assigns each subset of features a quality value, and the search procedure that is applied to find a path in the complete search tree (cf. Fig. 4). Since the feature selection has to be done for a large number of time series, we chose a simple search algorithm, i.e., forward selection, instead of more sophisticated techniques like branch and bound or evolutionary algorithms [1,19]. There exists a variety of optimization criteria like Maximum Linear Independence or Mahalanobis distance, which take only linear dependencies into account [23,1]. To reduce the input vector as far as possible we have to take non-linear dependencies into account. In this case, it is straightforward to use mutual information as optimization criterion. The mutual information $I(X;Y)$ of two random variables $X,Y$ reflects the reduction in the uncertainty of $Y$ due to the knowledge of $X$, e.g., mutual information measures the degree of stochastic independence of $X$ and $Y$ [3]. To reduce the feature
vector as far as possible we applied this bottom-up optimization procedure based on mutual information.

5.1. Estimating the stochastical independence

In this section, we describe how the stochastical independence between two random variables can be estimated based on the data. This procedure is used to compute the stochastic independence between the input and target vector.

The entropy as a measure of information (cf. [24,3]) for a given random variable $X$ with probability function $p$ is defined as

$$H(X) = - \int p(x) \log p(x) \, dx.$$  

Similarly for a pair of random variables $X$ and $Y$ with joint probability $p(x, y)$ and conditional probability $p(y|x)$, the conditional entropy can be defined as

$$H(Y|X) = - \int \int p(x, y) \log p(y|x) \, dx \, dy,$$

where $H(Y|X)$ is the uncertainty of the outcome of the $Y$ due to the knowledge of $X$. The mutual information $I(X; Y)$ of two random variables $X, Y$ is defined as the Kullback–Leibler distance between the joint distribution $p(x, y)$ and the product $p(x)p(y)$.

$$I(X; Y) = \int \int p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$  

(14)
and measures the degree of stochastic independence of $X$ and $Y$. This can be transformed into

$$I(X; Y) = H(Y) - H(Y|X)$$

and describes the reduction in the uncertainty of $Y$ due to the knowledge of $X$ [3].

In order to calculate (14), we approximate the three probability distributions $p(x, y)$, $p(x)$ and $p(y)$, where we use a non-parametric density function approximation with an Epanechnikov kernel function $K$ [25]. If $z$ is a $d$-dimensional vector, then

$$K(z) = (3/4)^d(1 - (z_1^2 + z_2^2 + \cdots + z_d^2)),$$

if $\|z\|_2 < 1$ and 0 otherwise. The normalizing constant $(3/4)^d$ guarantees that $\int K(z) \, dz = 1$. Finally, the probability density for a given data point $k$ is estimated as

$$p(z(k)) = \frac{1}{Nh^d} \sum_{j=1}^{N} K\left( \frac{1}{h}(z(k) - z(j)) \right),$$

where $N$ is the number of training patterns and $h$ is the spread of the kernel function. In case of the Epanechnikov kernel a sensible value for $h$ can be determined (see [25]) by

$$h_{opt} = A(K) \frac{1}{N^{1/d+4}},$$

where $N$ is the number of data points and $d$ the dimension of the vector. The constant $A(K)$ depends on the kernel used and takes for the Epanechnikov kernel the value

$$A(K) = \left( \frac{8}{c_d} (d + 4)(2\sqrt{\pi})^d \right)^{-(d+4)},$$

where $c_d$ is the volume of the unit $d$-dimensional sphere. Silverman notes that the data should be pre-scaled to avoid extreme differences of spread in various directions, either be normalizing or whitening [25,4]. By summarizing Eq. (17) over all data points, we get the desired density estimations.

5.2. Feature selection

In the speaking of neural networks we identify $X$ to be the current feature vector while $Y$ is the target. The task is to search for an input variable subset of low dimension with high mutual information estimate. This is done by applying a bottom-up search algorithm as illustrated in Fig. 4. On every level we compute the mutual information of all input vectors which can be created from the current vector by adding a feature and select that input vector as successor with the highest mutual information content. This is the step with the largest gain of information about the target. Note that this strategy is only locally optimal. If the set of features is large then it is sensible to use a bottom-up approach because the computing time
Fig. 5. The figure illustrates two steps of the algorithm: (a) After computing the local optimal step for all data sets we choose the feature, which was selected most frequently on this level. Note that for better representation only 20 of the 47 features are shown in the diagram. (b) Repeating the locally optimal step results in diagram that shows the mutual information $I(Y;Y)$ on each level averaged over all data sets. As a stop criterion, a threshold value for the loss of information can be defined.

is much shorter and the quality of the density estimation is better compared to the top-down procedure of removing features.

If we would compute the mutual information for all possible input vectors (which is intractable for most practical applications), we still have no clear decision criterion which input vector to use, since mutual information is a monotonic function, i.e., adding any input unit to the neural network increases the mutual information. If only one data set is given, this optimization can be intertwined with the training algorithm, as we have proposed in [19,20]. Since we have 120 data sets for this task, we choose a slightly different procedure. Starting with no input feature we compute on every level the locally optimal step for every data set and count for each feature, how often it was selected. Fig. 5a shows this for the first step (for 20 features).

The feature with the maximal number is added to all data sets and the step is repeated until the gain of information drops below a threshold. This provides a diagram that shows the average mutual information on each level (Fig. 5b). One advantage of this procedure is that the averaging reduces the influence of noise in the target data on the selection process. The algorithm summarizes as follows:

**Algorithm:** Bottom-Up-Selection based on Mutual Information

**Initialize** the current input vector with no features.

**Repeat:**

1. For each data set: Compute the mutual information $I(X_{i(+)};Y)$ for all possible input vectors $X_{i(+)}$ which can be created from the current input vector by adding a feature.
2. For each data set: Select the index $i$ of the feature that leads to a vector with maximal $I(X_{i(+)};Y)$. 


Fig. 6. Average error of different sizes of input vectors (47 and 10 dimensions). ‘WD’ stands for training with standard weight decay, ‘Bayes’ stands for training with a Bayesian learning rule while ‘Evidence’ denotes the strategy to always select the network with the highest evidence. The bars represent the range between the model with the lowest test error and the model with the largest test error. For each data set 50 networks were trained. Note that the test error shown is always an average value over all 120 data sets, i.e., the lowest test error means the lowest error of 50 initializations averaged over all data sets. The horizontal line connects the average errors of the different model types. The figure shows that both feature selection and Bayesian learning improve the generalization performance.

(3) Determine that feature with index $i$ that was selected most frequently and add it to all data sets until the gain of information (averaged over all data sets) is below a predefined threshold.

Setting the threshold to 1% the algorithm aborted for the given problem with a ten-dimensional feature vector. In the experiments we compared the original network using all 47 features and a model with a subset of 10 selected features. In detail, these 10 features are: (I+II) the difference between the last original sales value and the moving average over the last 11 sales values, as well as the moving average of features over six values, (III+IV) first order difference with and without time lag, (V–VII) differences of two exponential moving averages with different length, (VIII) second order differences (IX+X) short term trend indicators. Selecting combination of features is a non-trivial task, since it is for humans hard to predict which features carry sensible information and contribute additional information to the current input structure.

Both models were trained with standard weight decay and with a Bayesian learning rule. For each data set, 50 networks with different weight initializations were trained. Fig. 6 shows the test performance of the models. The vertical bar represents the range between the best and the worst models, while the horizontal line connects the average test errors of the different model types. Applying Bayesian learning and
feature selection reduced the average error by almost 8%, which should result in a further improvement of the return quota. The error was averaged 50 initializations for each data set and then over the 120 data sets. It is also worth noting that the variance of the generalization error is smaller in this case, i.e., the risk of selecting a model with poor performance is much lower.

Until now the test error is used as a model selection criterion. The Bayesian framework also allows to compute a quality measure, the evidence for the model, solely on the training data. The higher the evidence the better the model. In our experiments, the evidence was negatively correlated with the test error ($\rho = -0.39$), which results in a further improvement, when the average errors of the models with the highest evidence are compared to the overall average as in Fig. 6.

6. Discussion and further work

The machine learning approach based on neural networks proved to be a useful method to predict sales rates for retail traders. The huge number of data sets makes it impossible to fit parameters by cross-validation. By making use of the Bayesian learning rule the prediction performance could be improved essentially, while at the same time all parameters are computed automatically.

A top-down search algorithm based on mutual information allowed us to find a subset of features which is well suited for all data sets. By averaging over the data set, the influence of noise in the target data on the selection process is reduced. Thus, instead of optimizing the input structure for each retail trader individually, we found a solution that also minimizes the system maintenance, since it is identical for all data sets.

Nonetheless, we observed that several time series are hard to predict, i.e., the test error was much larger than the training error. This could in principle be the case if the underlying time series is non-stationary. We used the Kolmogorov–Smirnow test (KS-test) procedure to determine if the target data of the training and test set belong to the same distribution. See [17,2] for a detailed description of the test procedure. If this is not the case, our time series is non-stationary. The KS-test also computes a ‘probability’ that the two samples are drawn from the same distribution. We observed a strong negative correlation ($\rho = -0.48$) between the probability value of the KS-test and the test error (Fig. 7a). Thus, stationarity in fact turns out to be crucial for the success of neural network approach.

Furthermore, each pair of training and test data with a probability value below 0.1 and a growing variance on the test set had a large test error (Fig. 7b). That is, we can determine some difficult data sets by combining these two test procedures.

For the next test we split each data set into six subsets, where each subset corresponded to a week day (from Monday to Saturday). We applied the KS-test again to all pairs of training and test data formed from the subsets and observed that for some retail traders, the probability values of the subsets differ extremely. That is we found a strong positive correlation ($\rho = 0.56$) between the variance of
Fig. 7. The figure shows the results of two statistical tests applied to the time series. The circles mark networks with a high test error compared to the training error. (a) The $p$-value of the KS-test is shown over the difference between the training and test error. Low $p$-value and high test error are strongly correlated. (b) The $F$-test measures if the variance decreases (below the lower line), stays the same (between the lines), or increases (above the upper line). Note that the scale is logarithmic. If the $p$-value of a data set was critical and the variance of the data grew, the test error was large.

the probability value over the six subsets and the test error of the corresponding data set. Every time series which was hard to predict was recognized by at least one of the two test procedures described above, i.e., either it was non-stationary with growing variance or it varied extremely from week day to week day.

A possibly strategy to deal with that problem might be to decouple the prediction for the single week days; for example, by considering them as similar but different tasks and integrate them into a multi-tasking model. The problem with this is that less data is available to fit the parameters, i.e., it will be harder to learn a non-linear relationship. This and other strategies, which take similarities between the time series of different retail traders into account, will be investigated in the near future. At the moment, the statistical tests can be used as indicators to decide for every data set whether a neural network prediction should be generated or rather a simple moving average estimation.

References


Thomas Ragg works as a researcher at the Informatics department of the Universität Karlsruhe (TH) where he received his diploma (1993) and Ph.D. (2000) in Informatics. His research interests include machine learning algorithms, Bayesian learning, evolutionary optimization of neural networks, feature selection, committee techniques, and reinforcement learning. He applied and improved these algorithms in several projects with industrial partners, particularly in time series prediction problems.
Wolfram Menzel has been full professor at the ‘Institut für Logik, Komplexität und Deduktionssysteme’ of the Informatics department of the Universität Karlsruhe (TH) since 1972. His main fields of interest are Computability Theory, Logic, and Adaptive Systems. During the last 12 years, he established a research group in the field of neural networks, which had great success in developing theoretical concepts and applying them to real world applications often with industrial partners: ranging from control problems over time series prediction in finance and industry to outlier detection in radioactivity measurements or detection of structure in music.

Michael Wigbers works in the market development department for the newspaper BILD at Axel Springer Verlag. His main tasks include developing modern regulation system for the BILD distribution. He works out new statistics in order to elaborate and improve the regulation results of our wholesalers.

Walter Baum is the head of the market development department for the newspaper BILD at Axel Springer Verlag. He is responsible for 4 teams which elaborate distribution projects: e.g. RAMBOS (regulation system), BASISregio (analyse tool) and data warehouse projects. His main occupation is to evolve new methods to get more market information and more market transparency. The development of new ideas to improve the current distribution is his main interest.