Time Series Forecasting using Sparse Grids

Jochen Garcke[1], Thomas Gerstner[2] and Michael Griebel[1]

[1]Institut für Numerische Simulation
Universität Bonn
Wegelerstr. 6
D–53115 Bonn
{garcke,griebel}@ins.uni-bonn.de

[2]Institut für Mathematik
Johann Wolfgang Goethe-Universität
D–60054 Frankfurt am Main
gerstner@math.uni-frankfurt.de

Abstract

We present a machine learning approach for the forecasting of time series using the sparse grid combination technique. In this approach, the problem of analyzing a time series is first transformed into a higher-dimensional regression problem based on a delay embedding of the empirical data. Then, a grid-based approach is used to discretize the resulting high-dimensional feature space. In order to cope with the curse of dimensionality, we employ sparse grids in the form of the combination technique. Here, the regression problem is discretized and solved for a sequence of conventional grids with varying mesh widths. The sparse grid solution is then obtained by a linear combination of the solutions on these grids. We present the results of this approach for forecasting a benchmark time series and intraday foreign exchange rates using historical exchange data of several currencies.

Keywords: time series analysis, forecasting, machine learning, sparse grids, regression

1 Introduction

The forecasting of time series is an important mathematical problem with many applications. Examples include desaster prediction in natural sciences, trading in economics or health monitoring in medicine. The main approaches to tackle forecasting problems are either based on the physical or economical modelling and simulation of the underlying process or on the statistical evaluation of historical time series data from which an empirical model is derived.

In this paper we take the latter approach by transforming the forecasting problem into a machine learning regression problem [6, 9, 11, 15, 21, 26]. The idea behind this approach is that the underlying process generating the time series will behave similarly in similar situations. The machine learning algorithm now attempts to learn the behaviour of the time series just from empirical observations. The learned model can then be used to forecast in future comparable situations. To this end, the time series data is cast into a number of data points in a D-dimensional feature space together with a label. The label represents the difference between the value of the time series at the current time and a fixed time into the future. The D-dimensional data points and their corresponding labels are then used as input for a suitable regression algorithm. In our approach, the features are based on a delay embedding of the data [19, 22, 25]. In particular, we use approximations of first or second derivatives at each time

step of the series under consideration. Furthermore, we allow for the concurrent treatment of the data from several related time series to improve the quality of the prediction for one of these series.

Delay embedding is a powerful tool to analyze dynamical systems. Taken's theorem [22] gives the conditions under which a chaotic dynamical system can be reconstructed from a sequence of observations. In essence, it states that if the state space of the dynamical system is a k-dimensional manifold, then it can be embedded in $(2k+1)$-dimensional Euclidean space using the $2k+1$ delay values $f(t), f(t-\tau), f(t-2\tau), \ldots, f(t-2k\tau)$, where τ denotes a fixed time step. Heuristic computational methods, such as the Grassberger-Procaccia algorithm [12], can be used to estimate the embedding dimension k.

In this work we apply our recent approach for data mining problems [8, 9] to discretize and solve the resulting regression problem. Our approach is based on the regularization network formulation [11] and uses a grid, independent of the data positions, with associated local ansatz functions to discretize the feature space. This approach is similar to the numerical treatment of partial differential equations with finite elements. To avoid the curse of dimensionality, at least to some extent, a so-called sparse grid [2, 27] is used in the form of the combination technique [13].

The sparse grid approach is based on a hierarchical subspace splitting and a sparse tensor product decomposition of the underlying function space. To this end, the regularized regression problem is discretized and solved on a certain sequence of conventional grids. The sparse grid solution is then obtained by a linear combination of the solutions from the different grids. It turns out that this method scales only linearly with the number of data points to be treated [9]. Thus, this approach is well suited for machine learning applications where the dimension D of the feature space is moderately high, but the amount of data is very large. This is in contrast to support vector machines and related kernel-based techniques whose cost scale quadratically or even cubically with the number of data points, but on the other hand allow to deal with very high-dimensional feature spaces.

In this article we show that sparse grid regression can indeed be a useful tool for the forecasting of time series. We illustrate the performance of our approach for a model problem, the Mackey-Glass series, and for real financial data, which are intraday foreign exchange rates collected over time. For the forecasting of the Mackey-Glass data we achieve results on par with existing techniques. For the forecasting of financial data we achieve prediction accuracies of almost 60%, profits of up to 25% of the maximum attainable profit and we measure average revenues per transaction larger than typical transaction costs.

The remainder of this article is organized as follows: In Section 2 we describe how to transform the problem of forecasting a time series into a data mining problem based on delay embedding and discuss the resulting regularized regression problem. Section 3 gives the basics of the sparse grid combination technique, which is our employed numerical algorithm. Then, in Section 4, we give the results for our new approach for both, the Mackey-Glass model data and real historical foreign exchange data. Here, we consider the Euro, the US dollar, the Japanese Yen, the Swiss Franc and the British Pound from the years 2001 to 2005. Finally, Section 5 gives some concluding remarks.

2 Time series forecasting as a data mining problem

We consider time series data which are given on equidistant points in time with a fixed distance τ, which we will call ticks. If the raw time series data is not given on an equidistant grid, it has to be pre-processed accordingly. A standard approach is to employ piecewise constant upwind interpolation,

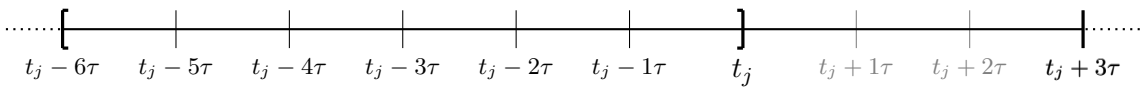

Figure 1: The transformation of an equidistant time series involves a time horizon $[t_j - (K-1)\tau, t_j]$ backward in time, here $K = 7$, from which the features to describe the behaviour of the time series at and up to time t_j are derived, and a prediction time frame, here $\hat{K} = 3$, which defines the time $t_j + \hat{K}\tau$ at which the response is predicted. This time horizon is then moved over the whole time series to convert the raw data into input data for the machine learning procedure.

which in particular avoids using data from the future. As an extension to the forecasting of one time series on its own, we also consider information from related time series for the prediction. In the following we assume that the different times series are all given on the same grid, again after suitable pre-processing if necessary.

This way, given J time steps and data from R related time series, the available input data which can be used in the forecasting process has the general form

$$\{t_j, f_r(t_j)\} \text{ for } j = 1, \ldots, J \text{ and } r = 1, \ldots, R.$$

Here, t_j denotes the j-th point in time, f_r denotes the data of the r-th time series and $t_{j+1} = t_j + \tau$. In the following we describe how the available data is transformed into input data for the actual forecasting procedure.

2.1 Delay embedding into a feature space

Based on the interpolated historical input data consisting of $J \cdot R$ data points we now want to predict the value or trend of, say, the first time series f_1. Given a point in time t_j we want to forecast the trend for f_1 at some point in time $t_j + \hat{K}\tau$ in the future, where $\hat{K}$ denotes a fixed number of ticks into the future. To this end, we convert the given time series up to a specific time t_j into data in a D-dimensional feature space, also called attribute space, which is supposed to describe the situation at the specific time t_j. The D-dimensional vector in feature space, where D will be made precise in the following, is put together by delay embedding the given tick data (see, for example, [4, 18, 19]), using the same selection of D features for all time positions to be treated. Thus, we use a time horizon $[t_j - (K-1)\tau, t_j]$ of size K backward in time at t_j, see also Fig. 1, and consider for each time series f_r a fixed number K of delayed values

$$f_r(t_j), f_r(t_j - \tau), f_r(t_j - 2\tau), \ldots, f_r(t_j - (K-1)\tau), \tag{1}$$

as the raw data describing the situation at time t_j. In principle the resulting $R \cdot K$ delayed values could all directly be used to obtain a $D(= R \cdot K)$-dimensional feature space with $f_1(t_j)$ being the first coordinate, $f_1(t_j - \tau)$ the second, and so on up to $f_R(t_j - (K-1)\tau)$ being the $(R \cdot K)$-th coordinate.

But this is not the only way of delay embedding the data for a given point in time t_j. Instead of directly employing the values $f_r(\cdot)$ of the time series, discrete first derivatives in t_j

$$f'_{r,k}(t_j) = \frac{f_r(t_j) - f_r(t_j - k\tau)}{k\tau} \tag{2}$$

with $k = 1, \ldots, K-1$ can be used in the backward time horizon [19], yielding $K-1$ coordinates for each time series and $R(K-1)$ coordinates in total. Normalized first derivatives

$$\tilde{f}'_{r,k}(t_j) = \frac{f_r(t_j) - f_r(t_j - k\tau)}{k\tau f_r(t_j - k\tau)} \tag{3}$$

can be considered as well, which take into account relative changes instead of absolute ones. Alternatively, a combination of time series values, first derivatives, higher order derivatives or even statistically derived values such as variances or frequencies can be employed as attributes.

Note that, in principle, one can choose such an attribute at all possible time positions of our backward time horizon $[t_j - (K-1)\tau, t_j]$ for time t_j for the selected D dimensional feature space, e.g. use all K values of the time series $f_r(t_j - i\tau)$ for $i = 0, \ldots, K-1$ or all $K-1$ values of the normalized first derivative (3), but that is usually not necessary. A suitable selection from the possible time positions of a given attribute in the time horizon $[t_j - (K-1)\tau, t_j]$, or even only one, could be sufficient for a given situation.

In any case, the number of features obtained by the delay embedding can easily grow large. Therefore, the number K of delay values, that is the size of our backward time horizon, and the total number of derived attributes D have to be chosen properly from the large number of possible embedding strategies. A good choice of such derived attributes and their parameters is non-trivial and has to be determined by careful experiments and suitable assumptions on the behaviour of the considered time series.

Overall, the transformation into feature space, i.e. the space of the embedding, based on the backward time horizon of a given point in time t_j can be formally denoted by an operator $T : \mathbb{R}^{R \cdot K} \to \mathbb{R}^D$

$$\underline{x}(t_j) = T\big(f_1(t_j), \ldots, f_1(t_j - (K-1)\tau), f_2(t_j), \ldots, f_2(t_j - (K-1)\tau), \ldots, f_R(t_j), \ldots, f_R(t_j - (K-1)\tau)\big)$$

with the feature vector $\underline{x}(t_j) = (x_1, \ldots, x_D) \in \mathbb{R}^D$ describing the situation at t_j, where the single features $x_d, d = 1, \ldots, D$, are any of the derived quantities mentioned.

As the response variable in the machine learning process we employ the normalized difference between the time series f_1 at the current time t_j and at some time $t_j + \hat{K}\tau$ in the future, with $\hat{K}\tau$ denoting the time horizon of prediction, i.e.

$$y(t_j) = \frac{f_1(t_j + \hat{K}\tau) - f_1(t_j)}{f_1(t_j)}. \tag{4}$$

This will give a regression problem later on. If one is only interested in the trend, the sign of $y(t_j)$ can be used as the response variable which will result in a classification problem. The employed time window for a time point t_j is illustrated in Fig. 1.

This transformation of the time series data into a D-dimensional feature vector $\underline{x}$ by T and the associated response variable y can be applied at $J - (K-1) - \hat{K}$ different time points t_j over the whole given data series, since at the beginning and end of the given time series data one has to allow for the time frames of the delay embedding and the prediction, respectively. Altogether, the application of such an embedding transformation and the evaluation of the associated forecast values (4) over the whole time series results in a data set of the form

$$S = \left\{(\underline{x}_m, y_m) \in \mathbb{R}^D \times \mathbb{R}\right\}_{m=1}^{J-(K-1)-\hat{K}}, \tag{5}$$

with $\underline{x}_m = \underline{x}(t_{m+K-1})$ and $y_m = y(t_{m+K-1})$.

This dataset can now be used by any machine learning algorithm [15], such as neural networks, multivariate adaptive regression splines or support vector machines, to construct a function $u : \Omega \subset \mathbb{R}^D \to \mathbb{R}$ which describes the relationship between the features $\underline{x}$ and the response y in an approximate way.

A practical goal of time series analysis is forecasting from a "new" current situation into its immediate future, therefore the function u, learned on existing historical data, can then be evaluated at any "new" time point t_e to predict the response y_e. To this end, the same operator T is now used to transform the newly collected time series data from $t_e - (K-1)\tau$ up to the "new" current time point t_e into a D-dimensional feature vector $\underline{x}_e$ which describes the behaviour of the time series at, and up to, time point t_e. The evaluation of the reconstructed continuous function u in such a new situation $\underline{x}_e$ is supposed to yield a good prediction $u(\underline{x}_e)$. This is due to our assumption that the underlying process behaves similarly in similar situations.

2.2 Regularized least squares regression

In the following we formulate the scattered data approximation problem in D-dimensional space by means of a regularization network approach [5, 11]. As stated above, we assume that the relation between $\underline{x}$ and y in the data set (5) can be described by an unknown function $u : \Omega \subset \mathbb{R}^D \to \mathbb{R}$ which belongs to some space V of functions defined over $\mathbb{R}^D$. The aim is now to recover the function u from the given data S of some size M, with e.g. $M = J - (K-1) - \hat{K}$, as good as possible. A simple least squares fit of the data would surely result in an ill-posed problem. To obtain a well-posed, uniquely solvable problem, we use regularization theory and impose additional smoothness constraints on the solution of the approximation problem. In our approach this results in the variational problem

$$\min_{u \in V} R(u) \qquad \text{with} \qquad R(u) = \frac{1}{M} \sum_{m=1}^{M} (u(\underline{x}_m) - y_m)^2 + \lambda \|Gu\|_{L_2}^2. \tag{6}$$

Here, the mean square error enforces closeness of u to the data, the regularization term defined by the operator G enforces the smoothness of u, and the regularization parameter λ balances these two terms. Different error and smoothness measurements may also be suitable. Further details can be found in [5, 9, 23]. Note that there is a close relation to reproducing kernel Hilbert spaces and kernel methods where a kernel is associated to the regularization operator G, see also [21, 26].

3 Sparse grid discretization

In order to compute a numerical solution of (6), we restrict the problem to a finite dimensional subspace $V_N \subset V$ of dimension $\dim V_N = N$. Common data mining methods such as radial basis approaches or support vector machines work with global ansatz functions associated to data points which leads to $N = M$. These methods allow to deal with very high-dimensional feature spaces, but typically scale at least quadratically or even cubically with the number M of data points. Thus, they cannot be applied to the huge data sets prevalent in time series prediction. Instead, we use grid based local basis functions, i.e. finite elements, in the feature space, similarly to the numerical treatment of partial differential equations. With such a basis $\{\varphi_n\}_{n=1}^N$ of the function space V_N we can approximately

represent the regressor u as

$$u(\underline{x}) \approx u_N(\underline{x}) = \sum_{n=1}^{N} \alpha_n \varphi_n(\underline{x}). \tag{7}$$

If the basis functions $\varphi_n(\cdot)$ are chosen independent from the data and the data locations the resulting regression algorithm scales linearly in regard to the number of data [9]. This allows the treatment of large data sets while still representing a non-linear function. Note that the restriction to a suitably chosen finite-dimensional subspace involves some additional regularization (regularization by discretization [17]) which depends on the choice of V_N. In the following, we simply choose $G = \nabla$ as the smoothing operator. Although this does not result in a well-posed problem in an infinite-dimensional function space, its use is reasonable in a discrete function space V_N with $N < \infty$, see [9, 10].

Now we plug (7) into (6). After differentiation with respect to the coefficients α_n, the necessary condition for a minimum of $R(u_N)$ yields the linear system of equations [9]

$$(\lambda \mathcal{C} + \mathcal{B} \cdot \mathcal{B}^T)\underline{\alpha} = \mathcal{B}\underline{y}. \tag{8}$$

Here $\mathcal{C}$ is a square $N \times N$ matrix with entries $\mathcal{C}_{n,n'} = M \cdot (\nabla\varphi_n, \nabla\varphi_{n'})_{L_2}$ for $n, n' = 1, \ldots, N$, and $\mathcal{B}$ is a rectangular $N \times M$ matrix with entries $\mathcal{B}_{n,m} = \varphi_n(\underline{x}_m), m = 1, \ldots, M, n = 1, \ldots, N$. The vector $\underline{y}$ contains the response labels y_m, $m = 1, \ldots, M$. The unknown vector $\underline{\alpha}$ contains the degrees of freedom α_n and has length N. A solution of this linear system then gives the vector $\underline{\alpha}$ which spans the approximation $u_N(\underline{x})$ with (7).

3.1 Sparse grid combination technique

Up to now we have not yet been specific what finite-dimensional subspace V_N and what type of basis functions $\{\varphi_n\}_{n=1}^N$ we want to choose. If uniform grids were used here, we would immediately encounter the curse of dimensionality and could not treat higher dimensional problems, i.e. values of D greater than 3. Instead we employ sparse grid subspaces as introduced in [2, 27] to discretize and solve the regularization problem (6), see also [9]. This discretization approach is based on a sparse tensor product decomposition of the underlying function space. In the following we describe the relevant basic ideas, for details see [2, 6, 8, 9, 27].

To be precise, we apply sparse grids in form of the combination technique [13]. Thereby, we discretize and solve the problem on a suitable sequence of small and in general anisotropic grids $\Omega_{\underline{l}}$ of level $\underline{l} = (l_1, \ldots, l_D)$, which have uniform but different mesh sizes $h_d = 2^{-l_d}, d = 1, \ldots, D$, in each coordinate direction. The points of a given grid $\Omega_{\underline{l}}$ are numbered using the multi-index $\underline{i} = (i_1, \ldots, i_D)$ with $i_d \in \{0, \ldots, 2^{l_d}\}$ for $d = 1, \ldots, D$. For ease of presentation, we assume a unit domain $\Omega = [0, 1]^D$ here and in the following.

A finite element approach with piecewise multilinear functions

$$\phi_{\underline{l},\underline{i}}(\underline{x}) = \prod_{d=1}^{D} \phi_{l_d,i_d}(x_d), \quad i_d = 0, \ldots, 2^{l_d}, \tag{9}$$

on each grid $\Omega_{\underline{l}}$, where the one-dimensional basis functions $\phi_{l_d,i_d}(x_d)$ are the so-called hat functions

$$\phi_{l_d,i_d}(x_d) = \begin{cases} 1 - |\frac{x_d}{h_{l_d}} - i_d|, & x_d \in [(i_d - 1)h_{l_d}, (i_d + 1)h_{l_d}] \cap [0, 1] \\ 0, & \text{otherwise,} \end{cases}$$

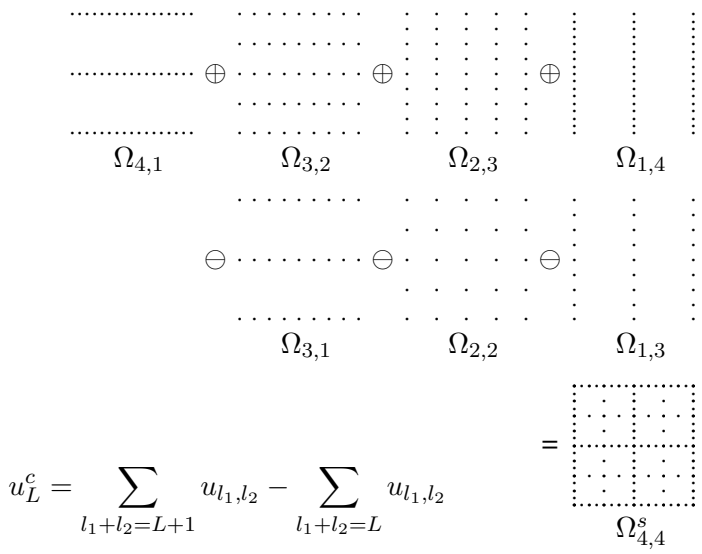

Figure 2: Grids employed by the combination technique of level $L = 4$ in two dimensions.

results in the discrete function space $V_{\underline{l}} = \text{span}\{\phi_{\underline{l},\underline{i}}, i_d = 0, ..., 2^{l_d}, d = 1, ..., D\}$ on grid $\Omega_{\underline{l}}$. A function $u_{\underline{l}} \in V_{\underline{l}}$ is then represented as

$$u_{\underline{l}}(\underline{x}) = \sum_{i_1=0}^{2^{l_1}} ... \sum_{i_D=0}^{2^{l_D}} \alpha_{\underline{l},\underline{i}} \phi_{\underline{l},\underline{i}}(\underline{x}).$$

To obtain a solution in the sparse grid space V^s_L of level L, see [2, 9], the combination technique considers a certain sequence of grids $\Omega_{\underline{l}}$. One gets an associated system of linear equations (8) for each of the involved grids $\Omega_{\underline{l}}$, which we solve by a diagonally preconditioned conjugate gradient algorithm. To be precise, the combination technique [13] linearly combines the resulting discrete solutions $u_{\underline{l}}(\underline{x})$ of (6) from the involved grids $\Omega_{\underline{l}}$ according to the formula

$$u^c_L(\underline{x}) = \sum_{q=0}^{D-1} (-1)^q \binom{D-1}{q} \sum_{|\underline{l}|_1 = L+(D-1)-q} u_{\underline{l}}(\underline{x}), \tag{10}$$

see also Figure 2 for an example in two dimensions. Let us stress for clarity that each partial solution of (6) used in (10) is computed with the same regularization parameter λ, which in practise is determined in an outer loop using k-fold cross-validation, see [9, 15] for details. The resulting function u^c_L lives in the sparse grid space V^s_L which has dimension $N = \dim V^s_L = \mathcal{O}(h_L^{-1}(\log(h_L^{-1}))^{D-1})$, see [2]. It therefore depends on the dimension D to a much smaller degree than a function on the corresponding uniform grid $\Omega_{(L,...,L)}$ whose number of degrees of freedom is $\mathcal{O}(h_L^{-D})$. Note that for the approximation of a function u by a sparse grid function $u^c_L \in V^s_L$ the error relation $||u - u^c_L||_{L_p} = \mathcal{O}(h_L^2 \log(h_L^{-1})^{D-1})$ holds, provided that u fulfills certain smoothness requirements which involve bounded second mixed derivatives [2]. The combination technique can be further generalized [6, 16] to allow problem-dependent coefficients.

Note that we never explicitly assemble the function u^c_L but instead keep the solutions $u_{\underline{l}}$ which arise in the combination technique (10). If we now want to evaluate the solution at a newly given data

point $\tilde{\underline{x}}$ by $\tilde{y} = u_L^c(\tilde{\underline{x}})$ we just form the combination of the associated point values $u_{\underline{l}}(\tilde{\underline{x}})$ according to (10). The cost of such an evaluation is of the order $\mathcal{O}(L^{D-1})$. Overall, both the computation of a sparse grid based regression model and its evaluation scale linearly in the number of data, which allows the treatment of large data sets [8, 9].

4 Numerical results

In the following, we present results of the sparse grid forecasting approach described in the previous sections for a benchmark time series and for a collection of financial time series representing intraday foreign exchange rates.

The discretization level L of the sparse grid combination technique and the regularization parameter λ in (6) are the hyperparameters of our approach. In our experiments we perform r-fold cross-validation on the training data to determine a good choice for L and λ. To this end, we divide the training data into r equally sized disjoint subsets. For $i = 1$ to r, we pick the i-th of these subsets as further testing set and learn with the data from the remaining $r - 1$ subsets. We then evaluate the correctness rates of the current training and testing set. This way we obtain r different training and testing correctness rates. The r-fold cross validation result is then just the average of the r correctness rates. The pair of values of L and λ which performs best in the average of all r splittings over a suitable range of L, λ is used for the forecast and final evaluation on the actual test data. For further details see e.g. [9, 15].

4.1 Mackey-Glass time series

The Mackey-Glass time series [20] is a well-known model example, we use it here to compare the results of our approach with existing results of other methods. This time series is based on the Mackey–Glass differential equation defined as

$$\frac{df(t)}{dt} = \alpha \frac{f(t-\kappa)}{1 + (f(t-\kappa))^{10}} - \beta f(t),$$

where we use $\alpha = 0.2, \beta = 0.1$, and $\kappa = 17$ as in previous works, see e.g. [3, 14]. The Mackey-Glass differential equation describes a feedback system, in which a time lag between the sensing of a change in the control variable and the mounting of an appropriate response is present. It is primarily used to model the complex dynamics of physiological control systems, like for example blood cell regulation [20].

For the sake of comparison we follow exactly the setup of earlier studies [3, 14]. The time series is generated using the fourth-order Runge–Kutta method with a time step of 10^{-1} and an initial condition of $f(0) = 1.2$. Values at time points corresponding to a natural number are taken as data points for the raw time series, i.e. only every tenth value computed by the Runge-Kutta method, therefore we have $\tau = 1$ in the notation from Section 2.1. In the benchmark one employs the 1000 data points from $t = 124$ to 1123, where the first 500 data are used as training data and the last 500 data for testing.

The feature values for the regression come from a standard delay embedding with time step size 6 and three values from the past [3, 14]. In our notation from Section 2.1 we have $r = 1$ since only one time series is present. For the time horizon in (1) this setup uses $[t_j - 18, t_j]$, i.e. we have the values $\tau = 1$ and $K = 19$. The operator T goes from R^{19} to R^4 and selects in the transformation every sixth raw value $f(\cdot)$ to obtain $\underline{x}$. Therefore one uses

$$\underline{x}(t_j) = [f(t_j), f(t_j - 6), f(t_j - 12), f(t_j - 18)] \tag{11}$$

to predict, following [3, 14], the value $f(t_j + 6)$, i.e. $\hat{K} = 6$.

Having processed the time series data as described, we then perform 10-fold cross-validation on the training data for a suitable range of L, λ to select the level L of the sparse grid combination technique and the regularization parameter λ in equation (6). With these parameters we learn the regression model u_L^c on the full training data and obtain a root mean square error (RMSE) on the actual test data of 0.001335. This error is on par with the, to our knowledge, best recent results of 0.00132 and 0.0015 achieved with the approaches from [14] and [3], respectively, see these articles for further results by other approaches.

Let us remark that a discretization by the fourth order Runge-Kutta method with time step 10^{-1} results at $t = 500$ already in an error which is of the order 10^{-2} worse compared to that of the fourth order Runge-Kutta method with time step 10^{-4}. One observes a discrepancy when comparing the accuracy of the prediction of the discrete time series data, which is of order 10^{-3}, with the accuracy of the employed benchmark data stemming from an approximation of the underlying differential equation, which is of order 10^{-1}. However this is not a topic here.

4.2 Intraday FX forecasting

We now present results for the prediction of intraday foreign exchange rates with our sparse grid combination technique. Our aim is to forecast the EUR/USD exchange rate. First, we use just the EUR/USD exchange rate time series as input and employ a delay embedding of this single time series. We then additionally take other exchange rates into account and show the corresponding results, where we also consider trading on strong signals only in order to cope with transaction costs.

4.2.1 Data

The data we use were obtained from Olsen Data, a commercial data provider. In the following, we employ the exchange rates from 01.08.2001 to 28.07.2005 between EUR/USD (denoted by € =: f_1), GBP/USD (£ =: f_2), USD/JPY (¥ =: f_3) and USD/CHF (Fr =: f_4). To represent a specific currency pair we will use its above symbol instead of f_r in the following. For this data set the data provider mapped the recorded raw intraday tick data to values $f_r(t_j)$ at equidistant points in time which are $\tau = 3$ minutes apart in the following way: If in the time interval $[t_j - \tau, t_j]$ a raw tick is present the time series data is mapped by piecewise constant interpolation to the value for $f_r(t_j)$; if no raw tick is present in $[t_j - \tau, t_j]$ no value is recorded for that t_j. Due to this, the data set contains a multitude of gaps, i.e. where no values exist, which can be large when only sparse trading takes place, for example over weekends and holidays. The properties of this input data concerning these gaps are shown in Table 1. Here, the total number of ticks in the time frame would be 701,280 for each currency pair, but between 168,000 and 186,000 ticks are missing due to the above mentioned reasons. The number of gaps varies between about 4,000 and 6,000 while the gap length varies between one and about 900 with an average of about 30. These characteristics are similar for the four currency pairs.

For the following experiments with the sparse grid regression approach the associated input data set S is obtained from the given tick data following the procedure from Section 2.1, specific details will be given later. Note that the embedding operator T at a time point t_j depends on a certain number of delayed data positions in $[t_j - (K-1)\tau, t_j]$. Typically not all K time positions in the backward time horizon are employed for a given T. Therefore we can allow some missing data, as described above, in any given time window $[t_j - (K-1)\tau, t_j]$ for the transformation of the equidistant time series data,

exchange rate	total ticks	missing ticks	number of gaps	max. gap length	avg. gap length
EUR/USD	701,280	168,952	6,403	879	26
USD/CHF	701,280	171,015	6,192	920	27
USD/JPY	701,280	184,264	4,144	911	44
GBP/USD	701,280	185,442	5,278	912	35

Table 1: Total and missing number of ticks, number of gaps, and maximum and average gap length of the input data.

as long as the data $f_r(\cdot)$ at the positions necessary for the actual computation of T at that time t_j are present. Vice versa, if for one of the time positions employed by T at a t_j no value is recorded we cannot obtain the corresponding feature vector $\underline{x}_j$ and consequently do not use the, partly missing, information from the time window of that t_j as input for the computation of the regression model. Furthermore, we employ the common practice of restricting the values of large outliers in the attributes to a suitably chosen maximum value to avoid a too strong influence of those at all. Afterwards we linearly map the derived delay embedded features into $[0, 1]^D$.

In all our experiments we attempt to forecast the change in the EUR/USD exchange rate. The aim of our regression approach is to predict the relative rate difference $y(t_j) = (€(t_j + \hat{K}\tau) - €(t_j))/€(t_j)$ at $\hat{K}$ steps into the future in comparison to the current time. Such a forecast is often also called a trading signal.

For the experiments we separate the available data into training data (90%) and test data (10%), this split is done on the time axis. On the training data we perform 3-fold cross-validation (again splitting in time) to find good values for the level parameter L from (10) and the regularization parameter λ from (2) of our regression approach.

4.2.2 Quality assessment in foreign exchange rate prediction

To judge the quality of the predictions of the exchange rate by our sparse grid combination technique on any given set of $\tilde{M}$ data points (derived from time series data according to Section 2.1) we employ several domain specific quality measures. Note that the following measurements do not directly take the cost of trading into account, they only consider the foreign exchange rate.

We use the so-called realized potential

$$rp = cp/mcp$$

as the main measurement. Here cp is the cumulative profit

$$cp = \sum_{j=1}^{\tilde{M}} \frac{\text{sign}(u_L^c(\underline{x}_j)) \cdot (f_1(t_j + \hat{K}\tau) - f_1(t_j))}{f_1(t_j)},$$

i.e. the sum of the actual gain or loss in the exchange rate realized by trading at the employed $\tilde{M}$ time steps t_j, with corresponding $\underline{x}_j$, according to the forecast of the method, while mcp is the maximum possible cumulative profit

$$mcp = \sum_{j=1}^{\tilde{M}} \frac{|f_1(t_j + \hat{K}\tau) - f_1(t_j)|}{f_1(t_j)},$$

1 feature	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{\in}'_9$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$
Level 2	1.89	1.72	50.7	2.10	1.91	50.6	2.40	2.18	50.6	2.40	2.18	50.6
Level 3	3.02	2.76	51.5	2.44	2.22	51.0	2.26	2.05	50.8	2.40	2.18	50.6
Level 4	**3.37**	**3.07**	**51.9**	3.08	2.81	51.6	2.33	2.12	51.0	2.40	2.18	50.6
2 features	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{\in}'_9, \tilde{\in}'_4$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$
Level 2	3.81	3.48	51.9	3.53	3.22	51.6	2.42	2.2	50.6	2.41	2.19	50.6
Level 3	**4.69**	**4.29**	**52.6**	4.57	4.17	52.4	3.56	3.25	51.8	2.25	2.05	50.8
Level 4	4.58	4.18	52.5	4.61	4.21	52.5	4.29	3.91	52.4	2.39	2.17	51.0

Table 2: Training: 3-fold cross-validation results for the forecast of EUR/USD using $\hat{K} = 15$ and the features $\tilde{\in}'_9$ and $\tilde{\in}'_4$ for varying refinement level L and regularization parameter λ.

i.e. the gain when the exchange rate would have been predicted correctly for each trade. Note that these measurements also take the amplitude of the potential gain or loss into account. Since in practice one has to consider transaction costs, a forecasting tool which achieves a realized potential rp of 20% starts to become useful according to practitioners.

Furthermore, we give the prediction accuracy pa, often also called hit rate or correctness rate,

$$pa = \frac{\#\{u_L^c(\underline{x}_j) \cdot (f_1(t_j + \hat{K}\tau) - f_1(t_j)) > 0\}_{j=1}^{\tilde{M}}}{\#\{u_L^c(\underline{x}_j) \cdot (f_1(t_j + \hat{K}\tau) - f_1(t_j)) \neq 0\}_{j=1}^{\tilde{M}}}$$

which denotes the percentage of correctly predicted forecasts. Prediction accuracies of more than 55 % are often reported as worthwhile results for investors [1, 24].

4.2.3 Forecasting using a single currency pair

In a first set of experiments we aim to forecast the EUR/USD exchange rate from the EUR/USD exchange data. We begin with using two features which use the normalized discrete first derivative (3)

$$\tilde{\in}'_k(t_j) = \frac{\in(t_j) - \in(t_j - k\tau)}{k\tau \in(t_j - k\tau)},$$

with two different k. Here, the values of k are parameters to be determined as is $\hat{K}$, the time horizon for the forecast into the future. We chose $k = 9$, $k = 4$ and $\hat{K} = 15$, based on numerical experiments, details are given in [7].

This combination can be interpreted as an approximation to the normalized second derivative

$$\tilde{\in}''_k(t_j) = \frac{\in(t_j) - 2\in(t_j - k\tau) + \in(t_j - 2k\tau)}{(k\tau)^2 \in(t_j - k\tau)}$$

with $k = 4$. However, the use of two first derivatives $\tilde{\in}'_9$ and $\tilde{\in}'_4$ captures more information in the data than just the second derivative would.

	features	data points	L	λ	trades	$pa\%$	cp	mcp	$rp\%$
all ticks									
1 feature	$\tilde{€}'_9$	510,553	4	0.0001	51,056	51.5	0.741	32.37	2.29
2 features	$\tilde{€}'_9, \tilde{€}'_4$	508,616	3	0.0001	50,862	52.1	1.084	32.28	3.36
signal $> 10^{-4}$									
1 feature	$\tilde{€}'_9$	510,553	4	0.0001	460	56.7	0.070	0.650	10.8
2 features	$\tilde{€}'_9, \tilde{€}'_4$	508,616	3	0.0001	916	58.6	0.291	1.206	24.2

Table 3: Testing: Forecast of the EUR/USD for $\hat{K} = 15$ on the 10% remaining test data using first derivatives of the EUR/USD exchange rate.

The results from the 3-fold cross-validation on the training data are shown in Table 2. The best parameters are $\lambda = 0.0001$ and $L = 3$ for two features, which we accordingly use for the prediction on the 10% remaining test data, see Table 3 (second row) for the results using one and two features. We achieve $cp = 1.084$, $rp = 3.36\%$, and $pa = 52.1\%$ on 50862 trades using both $\tilde{€}'_9$ and $\tilde{€}'_4$ and see that employing two attributes results in a significant improvement of the performance in comparison to $\tilde{€}'_9$ as the only attribute. In particular we notice that the rp grows by about 50%. Furthermore, we observe that most of the profit corresponds to the strongest signals. If we only take predictions into account which indicate an absolute change[1] larger than 10^{-4}, we trade on just 916 signals and still achieve $cp = 0.291$, $rp = 24.2\%$ and $pa = 58.6\%$. Thus, trading on 1.8% of the signals generates 26.8% of the overall profit. This indicates that trading only on the stronger signals may result in a profitable strategy.

4.2.4 Forecasting using multiple currency pairs

Now we are interested in the improvement of the prediction of the EUR/USD exchange rate if we also take the other currency pairs £, ¥, and Fr into account. This results in a higher-dimensional regression problem. We employ first derivatives using the same k's as before for the different currency pairs.[2] Note that the number of input data points decreases slightly when we add further exchange rate pairs since some features cannot be computed any longer due to overlapping gaps in the input data.

First we only consider the first derivatives for $k = 9$ to observe the effect of the use of additional currency pairs. According to the best rp we select which of the three candidates $\tilde{\text{Fr}}'_9, \tilde{£}'_9, \tilde{¥}'_9$ is successively added. For example $\tilde{\text{Fr}}'_9$ in addition to $\tilde{€}'_9$ gave the best result using two currency pairs to predict EUR/USD. We then add $\tilde{£}'_9$ before using $\tilde{¥}'_9$. As before, we select the best parameters L and λ for each number of features according to the rp achieved with 3-fold cross-validation on the training data, see Table 4.

Using the combination with the best performance in the 3-fold cross-validation we then learn u^c_L on all training data and evaluate on the before unseen test data. The results on the training data are given in Table 5, both for the case of all data and again for the case of absolute values of the signals

[1] Observe that a change of 10^{-4} in our target attribute is roughly the size of a pip (the smallest unit of the quoted price) for EUR/USD.

[2] Different k for different currency pairs might result in a better performance, but we restricted our experiments to equal k for reasons of simplicity.

2 features	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{€}'_9, \tilde{\text{Fr}}'_9$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$
Level 2	4.96	4.56	52.2	4.63	4.25	51.9	2.76	2.53	50.7	2.47	2.27	50.7
Level 3	**4.98**	**4.58**	**52.3**	4.76	4.38	52.2	3.40	3.12	51.2	2.47	2.27	50.7
Level 4	4.57	4.21	52.0	4.89	4.49	52.1	4.02	3.69	51.6	2.51	2.30	50.7
3 features	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{€}'_9, \tilde{\text{Fr}}'_9, \tilde{£}'_9$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$
Level 2	4.84	4.49	52.2	4.63	4.29	52.0	3.23	2.98	50.9	2.56	2.36	50.7
Level 3	4.71	4.38	52.3	**4.90**	**4.56**	**52.3**	4.21	3.91	51.8	2.68	2.47	50.7
Level 4	4.28	3.97	52.2	4.84	4.49	52.1	4.60	4.27	51.9	2.74	2.52	50.9
4 features	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{€}'_9, \tilde{\text{Fr}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$
Level 2	4.26	3.96	52.1	4.40	4.08	51.9	3.24	3.01	51.1	2.61	2.41	50.6
Level 3	4.42	4.11	52.2	**4.61**	**4.29**	**52.2**	4.21	3.92	51.9	2.50	2.32	50.9
Level 4	3.69	3.43	51.9	4.30	4.00	52.1	4.50	4.18	52.0	3.18	2.95	51.3

Table 4: Training: 3-fold cross-validation results for the forecast of EUR/USD using $\hat{K} = 15$ and features derived from different exchange rates for varying refinement level L and regularization parameter λ.

	currencies	data points	L	λ	trades	$pa\%$	cp	mcp	$rp\%$
all ticks									
1 feature	$\tilde{€}'_9$	510,553	4	0.0001	51,056	51.5	0.741	32.37	2.29
2 features	$\tilde{€}'_9, \tilde{\text{Fr}}'_9$	503,939	3	0.0001	50,394	51.8	1.380	32.08	4.30
3 features	$\tilde{€}'_9, \tilde{\text{Fr}}'_9, \tilde{£}'_9$	495,654	3	0.001	49,566	51.6	1.469	31.76	4.62
4 features	$\tilde{€}'_9, \tilde{\text{Fr}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	494,238	3	0.001	49,424	51.7	1.478	31.70	4.66
signal $> 10^{-4}$									
1 feature	$\tilde{€}'_9$	510,553	4	0.0001	460	56.7	0.070	0.650	10.8
2 features	$\tilde{€}'_9, \tilde{\text{Fr}}'_9$	503,939	3	0.0001	2,318	54.0	0.469	2.504	18.8
3 features	$\tilde{€}'_9, \tilde{\text{Fr}}'_9, \tilde{£}'_9$	495,654	3	0.001	2,379	54.3	0.516	2.566	20.1
4 features	$\tilde{€}'_9, \tilde{\text{Fr}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	494,238	3	0.001	3,559	53.8	0.614	3.484	17.6

Table 5: Testing: Forecast of EUR/USD for $\hat{K} = 15$ on the 10% remaining test data using one first derivative from multiple currency pairs. Results are for trading on all signals and on signals $> 10^{-4}$.

	currencies	data points	L	λ	trades	$pa\%$	cp	mcp	$rp\%$
all ticks									
1 feature	$\tilde{€}'_9$	510,553	4	0.0001	51,056	51.5	0.741	32.37	2.29
2 features	$\tilde{€}'_9, \tilde{€}'_4$	508,616	3	0.0001	50,862	52.1	1.084	32.28	3.36
3 features	$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9$	503,017	2	0.0001	50,300	52.1	1.315	32.03	4.10
4 features	$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9, \tilde{\text{Fr}}'_4$	502,243	2	0.001	50,220	52.4	1.536	31.98	4.80
5 features	$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9, \tilde{\text{Fr}}'_4, \tilde{£}'_9$	494,975	2	0.001	49,497	52.1	1.556	31.73	4.90
6 features	$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9, \tilde{\text{Fr}}'_4, \tilde{£}'_9, \tilde{£}'_4$	492,965	2	0.001	49,296	52.1	1.538	31.60	4.87
signal $> 10^{-4}$									
1 feature	$\tilde{€}'_9$	510,553	4	0.0001	460	56.7	0.070	0.650	10.8
2 features	$\tilde{€}'_9, \tilde{€}'_4$	508,616	3	0.0001	916	58.6	0.291	1.206	24.2
3 features	$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9$	503,017	2	0.0001	1,811	58.9	0.467	2.048	22.8
4 features	$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9, \tilde{\text{Fr}}'_4$	502,243	2	0.001	1,557	59.6	0.447	1.785	25.0
5 features	$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9, \tilde{\text{Fr}}'_4, \tilde{£}'_9$	494,975	2	0.001	2,178	58.7	0.523	2.392	21.9
6 features	$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9, \tilde{\text{Fr}}'_4, \tilde{£}'_9, \tilde{£}'_4$	492,965	2	0.001	2,711	56.8	0.508	2.796	18.2

Table 6: Testing: Forecast of EUR/USD 15 ticks into the future using multiple currency pairs and derivatives on the 10% remaining test data. Results are for trading on all signals and on signals $> 10^{-4}$.

larger than 10^{-4}. Note that the performance on the training data in Table 4 suggests to employ the first two or three attributes. In any case, the use of information from multiple currencies results in a significant improvement of the performance in comparison to just using the attributes derived from the to be predicted exchange rate EUR/USD. The results on the test data given in Table 5 confirm that the fourth attribute $\tilde{¥}'_9$ does not achieve much of an improvement, whereas the additional features $\tilde{\text{Fr}}'_9, \tilde{£}'_9$ significantly improve both cp and rp. Trading on signals larger than 10^{-4} now obtains a pa of up to 56.7% and, more importantly, $rp = 20.1\%$ using 3 attributes. This clearly shows the potential of our approach. Altogether, we see the gain in performance which can be attained by a delay embedding of tick data of several currencies into a higher dimensional regression problem while using a first derivative for each exchange rate.

In a second round of experiments we now use two first derivatives with $k = 9$ and $k = 4$ for each exchange rate. We add step-by-step the different currencies in the same order as in Table 5. To be precise, we use $\tilde{\text{Fr}}'_9$ before $\tilde{\text{Fr}}'_4$, but both before $\tilde{£}'_9, \tilde{£}'_4$, etc.[3] Again we look for good values for λ and L via 3-fold cross-validation on the training data and use them in our prediction model. In Table 6 we give the results which we achieved on the test data. Note that the numbers obtained on the training data suggest to use the four features $\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9, \tilde{\text{Fr}}'_4$ only; nevertheless we show the test results with the two additional features $\tilde{£}'_9, \tilde{£}'_4$ as well. Again, the use of information from multiple currencies gives an improvement of the performance in comparison to the use of just the attributes which were derived from the EUR/USD exchange rate. In particular cp grows while going from one to several currencies. With four features based on two first derivatives for each currency pair we now achieve a somewhat better performance for all trading signals than before using several first derivatives, compare Table 5

[3]Note that a different order might result in a different performance.

currencies	strategy	cp	trades	cp per trade
$\tilde{€}'_9$	all ticks	0.741	51,056	1.4_{-5}
$\tilde{€}'_9, \tilde{€}'_4$	all ticks	1.084	50,862	2.1_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9$	all ticks	1.315	50,300	2.6_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9, \tilde{\text{Fr}}'_4$	all ticks	1.536	50,220	3.1_{-5}
$\tilde{€}'_9$	signal $> 10_{-4}$	0.070	460	1.5_{-4}
$\tilde{€}'_9, \tilde{€}'_4$	signal $> 10_{-4}$	0.291	916	3.0_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9$	signal $> 10_{-4}$	0.467	1,811	2.6_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr}}'_9, \tilde{\text{Fr}}'_4$	signal $> 10_{-4}$	0.447	1,557	2.9_{-4}

Table 7: Testing: cp per trade for the forecast of EUR/USD for $\hat{K} = 15$, based on Table 6, using different attribute selections on the 10% remaining test data.

and Table 6. We obtain $rp = 4.80$ for four attributes in comparison to $rp = 4.62$ with three attributes. The results on the stronger signals are also improved, we now achieve $rp = 25.0\%$ in comparison to $rp = 20.1\%$, although the cp is reduced from 0.516 to 0.447 due to less trades.

For this last setting we also show the average profit per trade in Table 7, this value needs to be above the average spread to result in a profitable strategy. The spread for EUR/USD can nowadays go down to one pip during high trading with some brokers, in our case one pip is roughly equivalent to a change of 8.5_{-5} of our normalized target attribute. We see that trading on all signals results in values which are below this threshold. However, trading on the strongest signals results in a profitable strategy. Note that in the companion paper [7] we developed a practical trading strategy using an opening and closing threshold which obtained an average profit per trade larger than three pips. If the spread is on average below three pips this results indeed in profitable trading.

5 Conclusions

In this paper, we presented a machine learning approach based on delay embedding and regression with the sparse grid combination technique for time series forecasting. We applied the approach to the model Mackey-Glass time series and to intraday foreign exchange rates. On the benchmark Mackey-Glass series we attained results which are on par with current alternative approaches. For the financial time series a realized potential of more than 20% of the maximum possible cumulative profit was achieved. Here the results were improved if not only attributes derived from one rate but from further exchange rates were taken into account. Overall, our approach was able to learn forecasting models for FX data series which show a predictive performance larger than current average transaction costs. It also indicates that FX rates have an underlying process which is not purely Markovian, but seems to have additional structure and memory which we believe is caused by technical trading in the intraday market.

To be able to handle the large amount of data of the FX example, and to learn the empirical behavior from the market data, it is essential to have a regression approach which scales linearly in the number of data, but still gives a non-linear regression function. The approach based on the sparse grid combination technique provides both.

Acknowledgements

We thank Bastian Bohn and Alexander Hullmann for their assistance with the numerical experiments.

References

[1] D. J. E. Baestaens, W. M. van den Bergh, and H. Vaudrey. Market inefficiencies, technical trading and neural networks. In C. Dunis, editor, *Forecasting Financial Markets*, pages 245–260. Wiley, 1996.

[2] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[3] H. Du and N. Zhang. Time series prediction using evolving radial basis function networks with new encoding scheme. *Neurocomputing*, 71(7-9):1388–1400, Mar. 2008.

[4] M. Engel. Time series analysis. Part III Essay, University of Cambridge, 1991.

[5] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[6] J. Garcke. Regression with the optimised combination technique. In W. Cohen and A. Moore, editors, *Proceedings of the 23rd ICML '06*, pages 321–328, 2006.

[7] J. Garcke, T. Gerstner, and M. Griebel. Intraday foreign exchange rate forecasting using sparse grids. In J. Garcke and M. Griebel, editors, *Sparse grids and applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*, pages 81–105. Springer, 2013.

[8] J. Garcke and M. Griebel. Classification with sparse grids using simplicial basis functions. *Intelligent Data Analysis*, 6(6):483–502, 2002. (shorter version appeared in KDD 2001, Proc. of the Seventh ACM SIGKDD, F. Provost and R. Srikant (eds.), pages 87-96, ACM, 2001).

[9] J. Garcke, M. Griebel, and M. Thess. Data mining with sparse grids. *Computing*, 67(3):225–253, 2001.

[10] J. Garcke and M. Hegland. Fitting multidimensional data using gradient penalties and the sparse grid combination technique. *Computing*, 84(1-2):1–25, April 2009.

[11] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–265, 1995.

[12] P. Grassberger and I. Procaccia. Characterization of strange attractors. *Phys. Rev. Lett*, 50:346–349, 1983.

[13] M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens, editors, *Iterative Methods in Linear Algebra*, pages 263–281. IMACS, Elsevier, North Holland, 1992.

[14] C. Harpham and C. Dawson. The effect of different basis functions on a radial basis function network for time series prediction: A comparative study. *Neurocomputing*, 69(16-18):2161–2170, 2006.

[15] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[16] M. Hegland, J. Garcke, and V. Challis. The combination technique and some generalisations. *Linear Algebra and its Applications*, 420(2–3):249–275, 2007.

[17] M. Hegland, O. M. Nielsen, and Z. Shen. Multidimensional smoothing using hyperbolic interpolatory wavelets. *Electronic Transactions on Numerical Analysis*, 17:168–180, 2004.

[18] I. Horenko. Finite element approach to clustering of multidimensional time series. *SIAM Journal on Scientific Computing*, 32:62–83, 2010.

[19] H. Kantz and T. Schreiber. *Nonlinear time series analysis*. Cambridge University Press, 1997.

[20] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.

[21] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[22] F. Takens. Detecting strange attractors in turbulence. In D. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer, 1981.

[23] A. N. Tikhonov and V. A. Arsenin. *Solutions of ill-posed problems*. W. H. Winston, Washington D.C., 1977.

[24] G. Tsibouris and M. Zeidenberg. Testing the efficient markets hypothesis with gradient descent algorithms. In A.-P. Refenes, editor, *Neural Networks in the Capital Markets*, chapter 8, pages 127–136. Wiley, 1995.

[25] M. Verleysen, E. de Bodt, and A. Lendasse. Forecasting financial time series through intrinsic dimension estimation and non-linear data projection. In J. Mira and J. V. Sánchez-Andrés, editors, *Engineering Applications of Bio-Inspired Artificial Neural Networks, Volume II*, volume 1607 of *Lecture Notes in Computer Science*, pages 596–605. Springer, 1999.

[26] G. Wahba. *Spline models for observational data*, volume 59 of *Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

[27] C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990*, volume 31 of *Notes on Num. Fluid Mech.* Vieweg-Verlag, 1991.