OPELM and OPKNN in Long-Term Prediction of Time Series using Projected Input Data

Dušan Sovilj[a], Antti Sorjamaa[a], Qi Yu[a], Yoan Miche[a], Eric Séverin[b]

[a]*Department of Information and Computer Science, Helsinki University of Technology, 02015 TKK, Finland*
[b]*University of Lille 1 - Laboratoire Economie Management, 59653 Villeneuve d'Ascq cedex, France*

Abstract

Long-term time series prediction is a difficult task. This is due to accumulation of errors and inherent uncertainties of a long-term prediction, which leads to deteriorated estimates of the future instances. In order to make accurate predictions, this paper presents a methodology that uses input processing before building the model. Input processing is a necessary step due to the *curse of dimensionality*, where the aim is to reduce the number of input variables or features. In the paper, we consider the combination of the Delta Test and the Genetic Algorithm to obtain two aspects of reduction: scaling and projection. After input processing, two fast models are used to make the predictions: Optimally-Pruned Extreme Learning Machine and Optimally-Pruned k-Nearest Neighbors. Both models have fast training times, which makes them suitable choice for direct strategy for long-term prediction. The methodology is tested on three different data sets: two time series competition data sets and one financial data set.

Key words: Long-Term Time Series Prediction, Projection, Genetic Algorithm, Delta Test, OPELM, OPKNN, Hannan-Quinn Information Criterion

Email addresses: `dusan.sovilj@tkk.fi` (Dušan Sovilj), `antti.sorjamaa@tkk.fi` (Antti Sorjamaa), `qi.yu@tkk.fi` (Qi Yu), `yoan.miche@tkk.fi` (Yoan Miche), `e.severin@wanadoo.fr` (Eric Séverin)

1. Introduction

Time series prediction is a difficult task, especially when trying to predict several tens or hundreds time steps to the future. In order to make accurate and reliable predictions, several aspects have to be considered. Some of these aspects include selection of adequate inputs, choosing the accurate model, proper validation method and the choice of long-term prediction strategy. Moreover, one of the choices can greatly affect the other choices, making the selection process very complex and hard to optimize. In this paper, we focus on two problems: input processing and selection of the prediction methodology.

Many techniques exist for the approximation of the underlying process of a time series: linear methods such as ARX, ARMA and others [1, 2], and nonlinear ones such as artificial neural networks [3, 4] and other computational intelligence techniques [5, 6]. In general, these methods try to build a model of the process. The model is then used on the last values of the series to predict the future values. The common difficulty is the determination of sufficient and necessary information for an accurate prediction. Moreover, both linear and nonlinear techniques have problems when facing with large number of variables, a phenomenon known as the *Curse of Dimensionality* [7]. Most of the models used for time series prediction are also used for financial forecasting [8, 9, 10]. Linear models are less desirable as models in this domain, because financial time series tent to exhibit periods during which they are harder to predict (which depend on the past values of the series).

Long-term prediction requires long historical data space, causing the input to have a lot of variables or dimensions. Therefore, input processing is necessary due to the Curse of Dimensionality. It is a well known phenomenon that prevents all learning models from achieving a good performance. In order to decrease the dimensionality, one can, for example, project the high-dimensional data into a low-dimensional data space. This projection must be done carefully and preferably in a supervised way in order to retain the information contained in the original data. Another aspect of input processing is the scaling of variables, which allows the interpretability by looking at the scaling weights.

There are several techniques capable of performing the optimization of the projection. Each technique comprises of the search algorithm and the relevance criterion. The search algorithm guides the search through the solution space, while the relevance criterion measures the quality of each solution encountered during the search process. Examples of different search algorithms are Forward Search, Backward Search [11, 12] and Genetic Algorithm [13]. As for the relevance criteria, there are for example Correlation, Mutual Information, Gamma Test and Cross-validation error (for reference on these criteria and others, see [14]). In the paper, we are using the Delta Test as a relevance criterion, while the Genetic Algorithm is chosen as the search algorithm. The Delta Test has already been used as a search criterion for time series prediction, but in the problem of variable selection and a different search algorithm [15].

After input processing steps, the next step is prediction of the values using a suitable model. In the paper, we use Optimally-Pruned Extreme Learning Machine (OPELM) and Optimally-Pruned k-Nearest Neighbors (OPKNN), two nonlinear models with fast training times and accurate predictions.

The paper is structured as follows. Section 2 gives a short introduction to time series prediction, and the notations used throughout the remainder of the paper. Section 3 presents the global methodology (input processing and prediction models). Experimental results, including preprocessing steps applied on two time series data sets, are showed in Section 4. Finally, conclusion and discussion are given in Section 5.

2. Time Series Prediction

Time series prediction is a problem of predicting the values of some phenomena, based on previously measured values of the same phenomena. The previous values are used as inputs for the model. One-step ahead prediction, usually referred to as the short-term prediction, is a task of predicting the next value of a series. For example, for a series of values $v_j, j = 1, \ldots, t$, we are interested in the next value at time step $t + 1$ as:

$$\hat{v}_{t+1} = f(v_t, v_{t-1}, \ldots, v_{t-d+1}),$$

where f is the model and d denotes the number of previous values used to build the *regressor*. Regressor is constructed by sliding a window of length d over the complete series, and registering a range of d consecutive values as a sample for the model. Given the series with t values, the j-th sample in the regressor has the following values

$$(\mathbf{x}_j; y_j) = (v_j, v_{j+1}, \ldots, v_{j+d-1}; v_{j+d}), \quad 1 \leq j \leq t - d. \tag{1}$$

Long-term prediction presents a new challenge in this field. The problem consists of not just predicting the value v_{t+1}, but also the next v_{t+l} values for $l \geq 2$. In this setting, predicting the value v_{t+l} is called the l–th step ahead prediction, or the l–th horizon of prediction. This type of problem is much harder since we are facing growing uncertainties arising from various sources. In long-term prediction, there are several strategies to build the models: Recursive and Direct [16], as well as their combination DirRec [17].

Direct strategy is adopted in the paper since it gives more accurate predictions than the recursive one [16]. With this approach, each horizon of prediction requires a separate model. To predict the $t+l$–th value, the regressor is build in a similar way as for the one-step ahead prediction given in Equation (1), with the exception of different outputs. The samples are formed in a following way:

$$(\mathbf{x}_j; y_j) = (v_j, v_{j+1}, \dots, v_{j+d-1}; v_{j+d-1+l}), \quad 1 \leq j \leq t-d+1-L. \tag{2}$$

where L is the maximum horizon of prediction. Finally, once the model is trained on these samples, the final prediction is done using last known values of the series, i.e.

$$\hat{v}_{t+l} = f_l(v_t, v_{t-1}, \dots, v_{t-d+1}), \quad 1 \leq l \leq L. \tag{3}$$

3. Methodology

The proposed methodology in this paper relies on various algorithms, for both main stages of the processing. Figure 1 illustrates the global format of the methodology.

The high-dimensional data is first projected, as detailed in Section 3.1, using a Genetic Algorithm to determine the weights of the projection matrix. The relevance criterion to be optimized is the Delta Test, presented in Section 3.1.3, which is a minimization problem.

Once the data is projected with a matrix obtained from the first step, the predictions are made using two new inspiring methodologies (Sections 3.2 and 3.3). First one of the methodologies is called Optimally-Pruned Extreme Learning Machine (OPELM), which originates from the ELM [18] principle of fast training of a Single Layer Feedforward Neural Network. Second methodology examined here is the Optimally-Pruned k-Nearest Neighbors (OPKNN). It relies on the k-Nearest Neighbors approach as the kernel

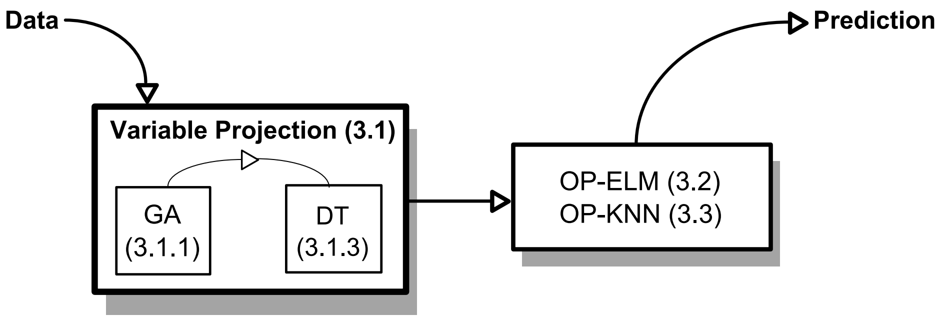

Figure 1: Global methodology summarized. The high-dimensional data is first projected using the Genetic Algorithm (GA) and the Delta Test (DT). After that, the OPELM/OPKNN algorithm is used for the actual prediction. Numbers indicate the sections where the concepts are explained.

function. The OPKNN is deterministic and even faster than the OPELM, also without any extra parameters.

3.1. Variable Projection

The goal of the variable projection is to decrease the input dimensionality. Then, the applied methodologies should have most of the information of the original data set contained in a smaller number of variables, and the performance of the methodologies should increase, while the training times should decrease. Projection also includes two special cases, selection and scaling, which allow the interpretability of the variables, by examining the selection subset and scaling weights, respectively. Although the projection is the general approach for variable processing, the interpretability is lost when considering this general case.

Let us consider the case where a data set $\mathbf{X}$ contains N samples and d variables, i.e. $\mathbf{X} \in \mathbb{R}^{N \times d}$. In a projection, a matrix $\mathbf{P} = [a_{ij}]_{d \times p}$ with size $d \times p$ is optimized according to the relevance criterion, and later used to obtain new data set

$$\mathbf{X}_P = \mathbf{X}\mathbf{P}. \tag{4}$$

In this setting, both selection and scaling can be represented as a $d \times d$ matrix with weights w_i on the main diagonal, i.e.

$$\mathbf{P^s} = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_d \end{bmatrix}_{\mathbf{d}\times\mathbf{d}}.$$

The values $w_i, i = 1, \dots, d$, are either binary $\{0, 1\}$ values in case of a selection or the real values in $[0, 1]$ range for the scaling.

A good property of the projection is the ability to linearly transform the data set to a lower dimensional space when the matrix $\mathbf{P}_{d\times p}$ has less columns than rows, i.e. $p < d$. However, the number of parameters in $\mathbf{P}_{d\times p}$ is dp, making the problem harder compared to the selection and scaling problems with d parameters. Furthermore, the correct value of p, the number of dimensions to project to, is an additional parameter that has to be optimized. The advantage is the manual choice of p, enabling full control of the dimensionality of the formed data set $\mathbf{X}_P$. In Section 3.1.4, we present an algorithm to automatically select the projection dimension p when using the Delta Test as the relevance criterion.

Following subsections give more details about the Genetic Algorithm, and the setup for its parameters used in the experiments, the relevance criterion Delta Test, and an algorithm to automatically select good value of the projection dimension.

3.1.1. Genetic Algorithm

Genetic Algorithm (GA) is a population based stochastic algorithm used for optimization purposes. It has been successfully applied for variable selection problem [12, 19, 20, 21, 22, 23], because of its ability to carry out optimization on a global scale. The convergence properties depend on the available resources, as well as the correct control parameters of the algorithm itself.

In this work, the GA serves as the search algorithm to optimize the projection matrix $\mathbf{P}$ in order to minimize the Delta Test. The GA has already been used with the Delta Test for variable selection [12], where the chromosomes are binary arrays of zeros and ones. In the scaling and projection problems, where parameters have real values, a natural choice is to consider Real-Coded Genetic Algorithm. This setup is reported in [24] with the combination of scaling and projection performed at the same time. In the experiments, we are not using the projection and the scaling at the same time, but instead we consider the problems separately.

3.1.2. Genetic Algorithm Setup for the Methodology

The GA population consists of individuals/chromosomes that represent the values of the matrix **P**. The initial population is created to have a lot of zero values in the majority of chromosomes. This approach improves over pure random initialization in terms of returned DT values [24]. Table 1 summarizes the approach that is adopted for the experiments.

Table 1: Summary of the custom initialization of the GA population. The random values are uniformly selected from the interval [-1,1]. Genes to be set to zero are selected randomly.

Standard	Custom Initialization, 80 % of the population		
20 %	Part 1	Part 2	Part 3
100 % random	90 % zeros 10 % random	80 % zeros 20 % random	70 % zeros 30 % random

In the experiments for the projection, 80% of the individuals have many zeros, while the remaining 20% are created in a standard way by choosing randomly from a uniform distribution over [-1,1] range. The custom individuals are further divided into three equally sized parts in which individuals have different number of zero genes, as presented in Table 1. In the case of scaling, the initial population is created in the same way as for the projection, except that the range of the random genes is set to $[0, 1]$.

The rest of the operators of the GA (selection, crossover, mutation) and their parameters have been set up according to [12], a setup which returns good solutions in terms of small DT values, while the stopping condition is set to 100 completed generations.

3.1.3. Delta Test

In this paper, the Delta Test (DT) is used as a relevance criterion to optimize the projection matrix. It is a nonparametric noise estimator based on the nearest neighbor principle. The nearest neighbor of a point is defined as the (unique) point, which minimizes a distance metric to that point.

In function approximation, the main goal is to design a function that represents given input points and their associated scalar outputs. That is, given N samples of input-output pairs $(\mathbf{x}_j; y_j) \in \mathbb{R}^d \times \mathbb{R}$, we wish to find a functional dependence between $\mathbf{x}$ and y with the following equation:

$$y_j = f(\mathbf{x}_j) + r_j, \quad 1 \leq j \leq N$$

where f is the unknown function and r_j is additive noise term. The function f is assumed to be smooth, and the noise terms r_j are independent and identically distributed with zero mean. Noise variance estimation is the study of how to give *a priori* estimate for Var$[r]$ given some data, *without* considering any specific shape of f.

Let us denote the nearest neighbor of a point $\mathbf{x}_j \in \mathbb{R}^d$ as $\mathbf{x}_{NN(j)}$. The nearest neighbor formulation of the DT estimates Var$[r]$ by

$$\mathrm{Var}[r] \approx \delta = \frac{1}{2N}\sum_{j=1}^{N}(y_j - y_{NN(j)})^2\,, \tag{5}$$

where $y_{NN(j)}$ is the output of $\mathbf{x}_{NN(j)}$. This is a well-known and widely used estimator, and it has been shown [25] that this estimate converges to the true value of the noise variance when $N \to \infty$.

3.1.4. Automatic Selection of the Projection Dimension

Before optimizing the projection matrix $\mathbf{P}_{d\times p}$, the question that remains is what is the correct value of p when using the DT. Let us denote with dt_k the best value returned by the DT when optimizing matrix $\mathbf{P}_{d\times p}$. Now, suppose that we have two projection matrices $\mathbf{P}_{d\times p}$ and $\mathbf{P}_{d\times p+1}$. Since the computation of the DT is based on nearest neighbors, the matrix $\mathbf{P}_{d\times p+1}$ also includes the problems with less dimensions, that is, it generalizes the optimization of the matrix $\mathbf{P}_{d\times p}$. This is done by setting the last column of $\mathbf{P}_{d\times p+1}$ to zero column. The zero column does not have any influence in the search for nearest neighbors, and this effectively becomes the problem with one less dimension in the projection matrix, i.e. it becomes $\mathbf{P}_{d\times p}$.

Since the last column of $\mathbf{P}_{d\times p+1}$ contains real values, the optimization of $\mathbf{P}_{d\times p+1}$ should be able to reach the value dt_{p+1} that is at least as small as dt_p. However, adding new d parameters to $\mathbf{P}_{d\times p}$ increases the complexity of the problem, adds new local minima and the optimization of $\mathbf{P}_{d\times p+1}$ becomes more challenging. This complexity can prevent the optimization process to reach good DT estimates, and it is possible that the value dt_{p+1} is larger than dt_p.

Thus, as p is increasing, the DT estimates dt_p should always decrease. In practice, large number of parameters prevents $\mathbf{P}$ matrices with large p to reach the same results of those cases with lower p. When optimizing the DT as a projection problem, there will be a value of $p = p_t$ after which the search procedure, in our case the GA, is unable to return lower dt values. Thus, we can conclude that matrix $\mathbf{P}_{d\times p_t}$ is our best projection and considering $p > p_t$ values is a waste of resources.

Previous discussion stems the strategy for automatic selection of p and the projection matrix $\mathbf{P}$. Start with $p = 1$ and optimize $\mathbf{P}_{d\times 1}$ to obtain DT estimate dt_1. Then, increase p by 1, optimize $\mathbf{P}_{d\times p}$ acquiring dt_p and compare dt_p with dt_{p-1}. If it holds that $dt_p < dt_{p-1}$ then continue increasing p, otherwise stop the process and return the pair $(dt_{p-1}, \mathbf{P}_{d\times p-1})$ as the final solution. This strategy is presented in Algorithm 1.

Algorithm 1 Automatic selection of projection dimension

```
1: $bestDT = \infty$
2: $p = 1$
3: while true do
4:     $(dt, \mathbf{P}_{d\times p}) = optimize(\mathbf{X}, \mathbf{Y}, p)$
5:     if $dt \geq bestDT$ then
6:         break
7:     end if
8:     $bestDT = dt$
9:     $bestP = \mathbf{P}_{d\times p}$
10:    $p = p + 1$
11: end while
12: return $(bestDT, bestP)$
```

In the Algorithm 1, function *optimize* at line 4 uses the GA. The input parameters are the data set $(\mathbf{X}, \mathbf{Y})$ and target projection dimension p. Since the result of the GA will depend on the initial population, several calls of *optimize* function are necessary for a reliable DT estimate. In the experiments we have chosen to run the optimization function 10 times for each value of p.

At the end of these steps, a suitable projection dimensionality p has been found, after which the data is projected. One can then use this projected data for the actual time series prediction using the OPELM/OPKNN models presented in the following subsections.

3.2. Optimally-Pruned Extreme Learning Machine

The Optimally-Pruned Extreme Learning Machine (OPELM) methodology is based on the original Extreme Learning Machine (ELM) [18] algorithm from which it borrows the original Single Layer Feedforward Neural Network (SLFN) construction. In the following, the main concepts and theory of the ELM algorithm are shortly reviewed, with an example on the possible problems encountered by the ELM on data sets with irrelevant variables.

The OPELM algorithm is introduced as a robust methodology regarding irrelevant variables situation. The steps of the algorithm are detailed in the following subsections and include: the original ELM idea; Multiresponse Sparse Regression, a network pruning algorithm; and two model structure selection methods: Leave-One-Out error and the Hannan-Quinn information criterion.

3.2.1. Extreme Learning Machine

The ELM algorithm was originally proposed by Guang-Bin Huang *et al.* in [18] and it makes use of the Single Layer Feedforward Neural Network. The main concept behind the ELM lies in the random initialization of the SLFN weights and biases. The theorem given in [18] states that the input weights and biases do not need to be adjusted, and it is possible to calculate implicitly the hidden layer output matrix and hence the output weights. The network is obtained with very few steps and, thus, a very low computational cost.

Consider a set of N distinct samples $(\mathbf{x}_j; y_j)$ with $\mathbf{x}_j \in \mathbb{R}^d$ and $y_j \in \mathbb{R}$; then, a SLFN with C hidden neurons is modeled as the following sum

$$\sum_{i=1}^{C} \beta_i f(\mathbf{w}_i \mathbf{x}_j + b_i), j \in [\![1, N]\!], \tag{6}$$

with f being the activation function, $\mathbf{w}_i$ the hidden layer weights, b_i the biases and β_i the output weights.

In the case where the SLFN perfectly approximates the data, the errors between the estimated outputs $\hat{y}_j$ and the actual outputs y_j are zero and the relation is

$$\sum_{i=1}^{C} \beta_i f(\mathbf{w}_i \mathbf{x}_j + b_i) = y_j, j \in [\![1, N]\!], \tag{7}$$

which is written in matrix form as $\mathbf{H}\beta = \mathbf{y}$.

Huang *et al.* state that with randomly initialized input weights and biases for the SLFN, and under the condition that the activation function is infinitely differentiable, then the hidden layer output matrix can be determined and provides an approximation of the target values as good as wished (non-zero).

The way to calculate the output weights β from the knowledge of the hidden layer output matrix $\mathbf{H}$ and target values, is proposed with the use of a Moore-Penrose generalized inverse of the matrix $\mathbf{H}$, denoted as $\mathbf{H}^{\dagger}$ [26].

Theoretical proofs and more details about the ELM algorithm are presented in the original paper [18].

3.2.2. Pruning out irrelevant variables

As already mentioned, the ELM models tend to have problems when irrelevant or correlated variables are present in the training data set. This drawback has been shown in [27] on a toy example and a real world data set. For this reason, OPELM proposes a pruning of irrelevant variables, via pruning of the related neurons of the SLFN built by the ELM. The procedure consists of three main steps that are summarized in Figure 2.

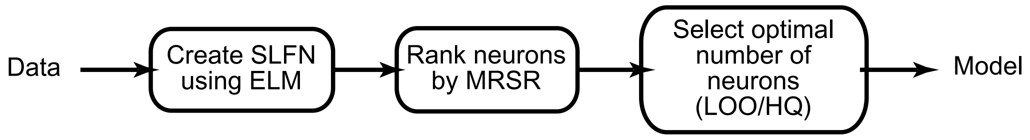

Figure 2: Three steps of the OPELM algorithm.

The very first step of the OPELM methodology is the actual construction of the SLFN using the original ELM algorithm with a large number of neurons. Second and third step are meant for effective pruning of possibly useless neurons of the SLFN. The second step, Mutliresponse Sparse Regression (MRSR), ranks the neurons according to their usefulness, while the actual pruning is performed using the results of the model structure selection criterion for the model: Leave-One-Out error or the Hannan-Quin information criterion.

The OPELM algorithm uses a combination of three different types of kernels: linear, sigmoid and gaussian, for robustness and more generality, where the original ELM uses only sigmoid and sine kernels. Having linear kernels in the network helps when the problem is linear or nearly linear.

Initialization of the network kernels is done in the following way. The gaussian kernels have their centers taken randomly from the data points, similarly as in [28], and widths randomly drawn between percentile 20 percent and percentile 80 percent of the distance distribution of the input space, as suggested in [29]. The sigmoid weights are drawn randomly from a uniform distribution in the interval $[-5, 5]$ in order to cover the whole zero mean and unit variance data range.

3.2.3. Multiresponse Sparse Regression

In order to remove useless neurons of the hidden layer, the Multiresponse Sparse Regression (MRSR), proposed by Timo Similä and Jarkko Tikka in [30], is used. It is mainly an extension of the Least Angle Regression (LARS) algorithm [31] and hence, it is actually a variable ranking technique, rather than a selection one.

An important property of the MRSR is that the obtained ranking is exact, if the problem is linear. In fact, this is the case with the OPELM, since the neural network built in the previous step is linear between the hidden layer and the output. Therefore, the MRSR provides an exact ranking of the neurons for our problem.

Because of the exact ranking provided by the MRSR, it is used to rank the neurons of the neural network model. The target is the actual output y_i, while the "variables" considered by the MRSR are the outputs of the hidden layer kernel functions $\mathbf{h}_i = \mathrm{Ker}(\mathbf{x}_i^T)$, the columns of $\mathbf{H}$.

3.2.4. Model Selection Criteria

Since the MRSR only provides a ranking of the neurons, the decision over the actual best number of neurons for the model is needed. Two possible model structure selection criteria are used to obtain sufficient number of neurons for the model, a classical Leave-One-Out computed using the fast PRESS Statistics, and the Hannan-Quinn information criterion.

Leave-One-Out (LOO). Computing the standard LOO error can be very time consuming, if the data set has a high number of samples. Fortunately, the PRESS (PREdiction Sum of Squares) statistics provides a direct and exact formula for the calculation of the LOO error for linear models. The basic idea of the PRESS is that it iteratively increases the number of samples of the linear model, without actually calculating the model, but only the LOO error. See [32] and [33] for details of this formula and its implementations.

The PRESS formula can be written as

$$\varepsilon^{\text{PRESS}} = \frac{y_i - \mathbf{h}_i \mathbf{b}_i}{1 - \mathbf{h}_i \mathbf{Q} \mathbf{h}_i^T}, \tag{8}$$

where $\mathbf{Q}$ is defined as $\mathbf{Q} = (\mathbf{H}^T\mathbf{H})^{-1}$, $\mathbf{H}$ is the hidden layer output matrix and $\mathbf{h}_i$ is the output of the added neuron.

The final decision over the appropriate number of neurons for the model can then be taken by the LOO error versus the number of neurons used. Here, the neurons are already ranked by the MRSR.

Hannan-Quinn criterion (HQ). Another criteria for complexity selection in machine learning are the ones based on information theory. Typical examples are the Akaike's Information Criterion (AIC) [34] and the Bayesian Information Criterion (BIC) [35]. Their expressions are based on the residual sum of squares (Res) of the considered model (first term of each criterion) plus a penalty (second term). Differences between criteria mostly occur on the penalty term.

The AIC is known to have consistency problems [36]. The other criterion, BIC, does not give a proper complexity selection for the OPELM, most likely due to too rapid increase of the penalty term with the number of samples. The Hannan-Quinn Information Criterion [37] is close to the two previously mentioned criteria and it is shown in Equation (9).

$$HQ = N \times \log\left(\frac{Res}{N}\right) + 2 \times B \times \log\log N \tag{9}$$

The value B is the number of free parameters of the model. The idea behind the design of this criterion is to provide a consistent criterion, in which the penalty term $2 \times B \times \log\log N$ grows, but at a very slow rate with respect to the number of samples. In this way, the HQ is in the middle of the two other criteria, the AIC and the BIC.

3.3. Optimally-Pruned k-Nearest Neighbours

The Optimally-Pruned k-Nearest Neighbors (OPKNN) algorithm from [38] shares many similarities with the OPELM algorithm. The ranking of neurons and the model structure selection are *identical* to the OPELM case. The only difference between the two algorithms is the type of kernels they use. The OPELM employs gaussian, linear and sigmoid kernels, while the OPKNN uses k-Nearest-Neighbors approach. This means that each neuron of the SLFN represents a certain nearest neighbor for each input point. The process can be presented as

$$\sum_{i=1}^{C} \beta_i \mathbf{y}_{NN_i(j)}, j \in [\![1, N]\!], \tag{10}$$

where $\mathbf{y}_{NN_i(j)}$ denotes the outputs of the i^{th} nearest neighbor of the j^{th} sample.

The OPKNN has proven to be somewhat more robust than the OPELM in some applications, and it has the interesting advantage of being deterministic, since it does not use any random parts, but only the k-Nearest-Neighbors approach.

4. Experiments

4.1. Data sets

We used three different data sets in the experiments, one financial data set and two time series prediction competition data sets. Each data set[1] and the associated preprocessing steps are briefly introduced in the following section.

4.1.1. Financial Data Set

In this experiment, we use the data [39] related to 200 French companies during a period of 5 years. We have 36 input variables and 535 samples without any missing values. The input variables are financial indicators that are measured every year (for example debt, number of employees, amount of dividends) and the last variable is the Return on asset (ROA) value of the same year. The target variable is the ROA of the next year for each sample.

Return on assets (ROA) is an important indicator to explain corporate performance, showing how profitable a company is before leverage, and is frequently compared with companies in the same industry [40]. However, since it is not easy to analyze which characteristics of the companies mainly affect the ROA value, especially when trying to predict it, the problem becomes more difficult. Table 2 shows the description of all the variables contained in the database.

[1]The data sets can be downloaded from
http://www.cis.hut.fi/projects/tsp/index.php?page=timeseries.

Table 2: The meaning of the variables in the Finance data set

Index	Variable	Meaning
1	Sector	Industry
2	Transaction	Number of shares exchanged during the year
3	Rotation	Security turnover rate
4	Vrif Rotation	Not useful
5	Net dividend	Amount of dividend for one share during the year
6	Effectifs	Number of employees
7	CA	Sales
8	II	Other assets
9	AMORII	Dotations on other assets
10	IC	Property, plant and equipment
11	AMORIC	Dotations on property, plant and equipment
12	IF	Not useful
13	AI	Fixed assets
14	S	Stocks or inventories
15	CCR	Accounts receivables
16	CD	Not useful
17	L	Cash in hands and at banks
18	AC	Total of current assets
19	CPPG	Total of capital of group (in book value)1
20	PRC	Not useful
21	FR	Accounts payables
22	DD	Not useful
23	DEFI	Financial debt
24	Debt-1AN	Debt whose maturity is inferior to 1 year
25	Debt+1AN	Debt whose maturity is superior to 1 year
26	TD	Total Debt
27	CPER	Cost of workers
28	CPO	Not useful
29	DA	Dotations on amortizations
30	REXPLOI	Operating income before tax
31	CFI	Interests taxes
32	RFI	Financial income
33	RCAI	Operating income before tax + Financial income
34	REXCEP	Extraordinary item
35	IS	Taxes from State
36	ROA	net income / total assets
Output1	ROA	the value of next year

4.1.2. ESTSP 2007 Competition Benchmark

This data set is from a prediction competition organized in the European Symposium of Time Series Prediction conference (ESTSP) in 2007. The data set has 875 samples and it is shown in Figure 3.

There is a clear seasonality present throughout the data, except around time point 400. Since the data seems to be corrupted or otherwise completely

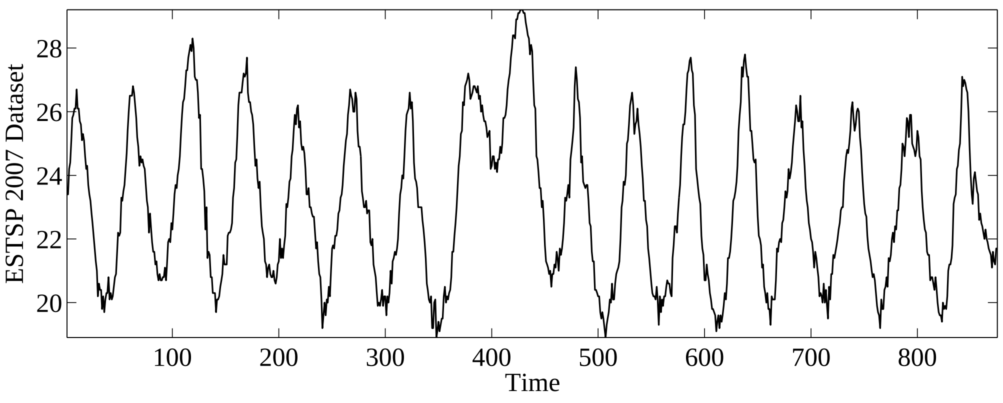

Figure 3: ESTSP 2007 competition data.

different from the rest of the data, we decided to remove the portion from the data set prior to any other preprocessing. In order to keep the phase correct, two full sequences of 52 were removed.

After that, the data set was separated into learning and test sets. First two thirds of the data was used in the learning and the remaining third as a test set.

Then, a sawtooth wave was fitted into the learning data and then removed from both data sets, in order to get rid of the seasonality. In other words, the preprocessing for the test set is done only based on the information available in the learning set.

After these steps, we have the preprocessed learning data shown in Figure 4.

4.1.3. ESTSP 2008 Competition Data 2 (ESTSP 2008-2)

Like the previous data set, this one is also from a prediction competition, except now it is taken from the ESTSP conference organized in 2008. In the prediction competition there were three data sets, out of which one was chosen for the experiments in this paper.

The competition data set 2 has 1300 samples and it is shown in Figure 5. Before any preprocessing steps, the data set was divided into learning and test sets containing the first two thirds and the last third, respectively.

For this data set, two preprocessing steps were used. First one takes care of the clear upward jump around time point 600 and the second one deals with the seasonality of the series. Both steps are done using only the

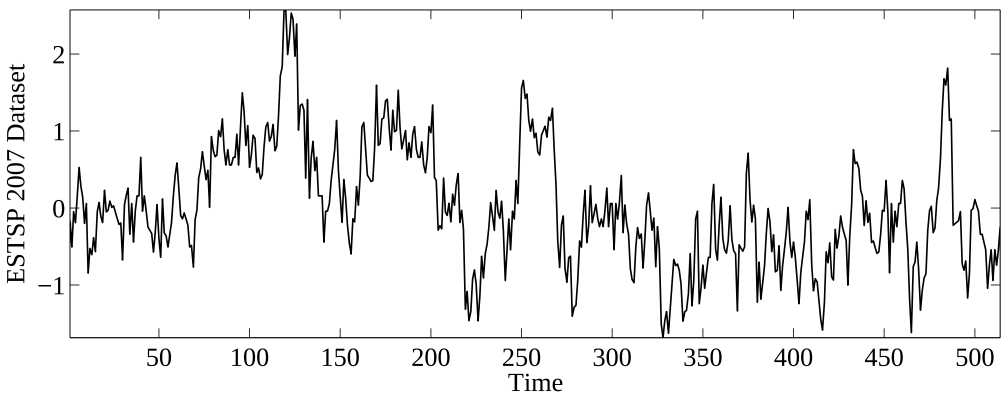

Figure 4: ESTSP 2007 competition data after preprocessing.

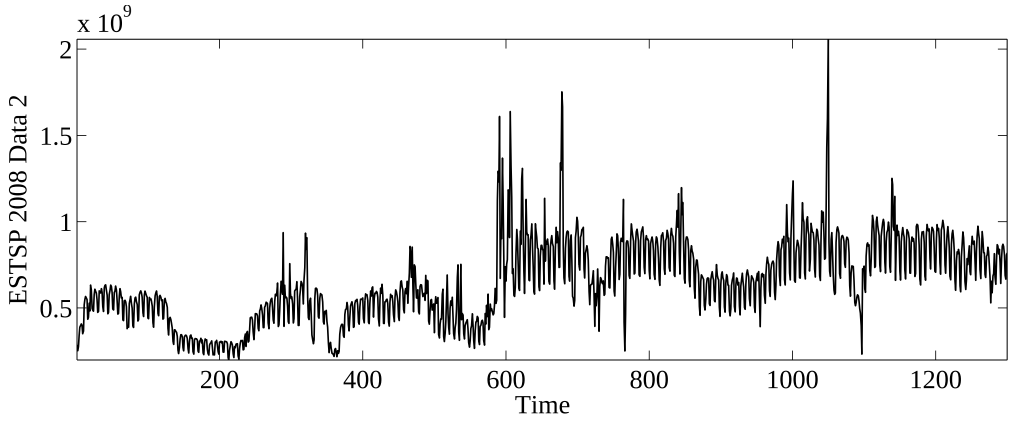

Figure 5: ESTSP 2008-2 competition data.

information available in the learning set, even when preprocessing the test set.

Step function fitting found the exact place of the jump at time point 588. The large scale seasonality was removed using a double square wave. The fitting of the double square is visualized in Figure 6 along with the learning data after removing the jump.

Finally, the standard deviation of the data was removed according to the fitted double square wave, in the high parts and in the low parts separately. Finally, our preprocessed learning data is shown in Figure 7.

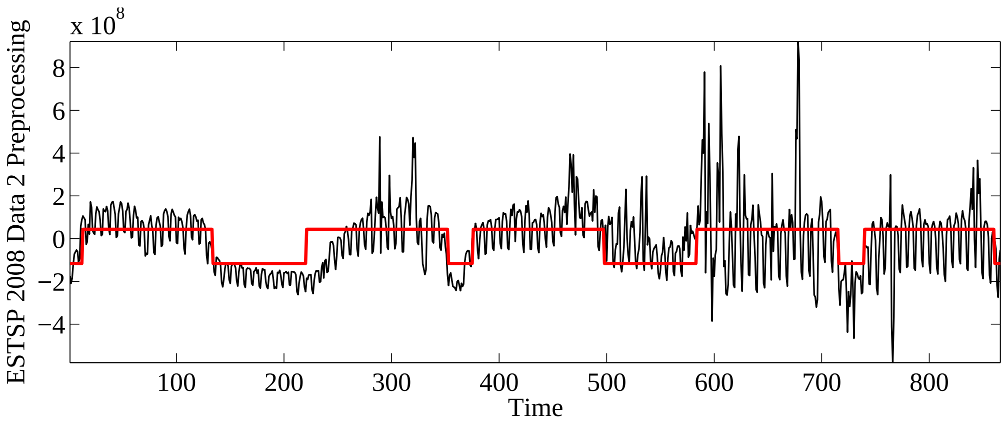

Figure 6: ESTSP 2008-2 data preprocessing. The fitted double square is shown in red.

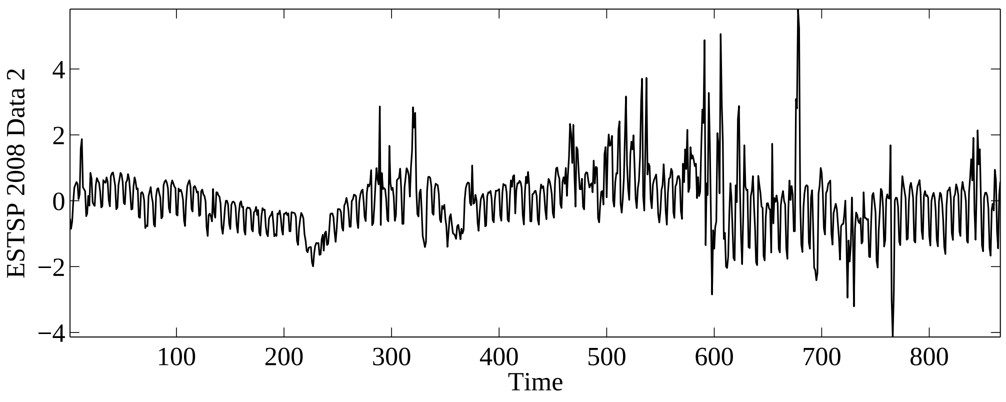

Figure 7: ESTSP 2008-2 data after preprocessing.

4.2. Results

Each data set was associated with a different number of prediction steps and input sample dimensionality along with different number of samples in learning and testing. Table 3 summarizes the specifications of all data sets.

For the competition time series, the input dimensionality means the width of the sliding window used in the original data vector in order to make the input matrix or the regressor. For the long-term prediction, we use the Direct Strategy, which has proven to be accurate and easily implementable choice [16].

For each data set and each prediction horizon, the projection and scaling

Table 3: Summary of all three data sets used in the experiments.

	Samples			
	Learning	Test	Input Variables	Horizon
Finance	357	178	36	1
ESTSP 2007	406	148	60	50
ESTSP 2008-2	717	285	50	100

using the GA and the DT was performed. Then, all four presented methods, the OPELM and the OPKNN both with the LOO and the HQ parameter selection methodologies, were applied. Table 4 summarizes the performance of the methods in each data set.

Table 4: Summary of the average test errors over all prediction horizons for all four methods.

		OPELM		OPKNN	
		LOO	HQ	LOO	HQ
Finance	projection	0.0016	**0.0016**	0.0020	0.0020
	scaling	0.0016	0.0017	0.0019	**0.0018**
	original	0.0016	0.0017	0.0019	0.0019
ESTSP 2007	projection	0.7824	0.7703	**0.9797**	**0.9695**
	scaling	**0.6231**	**0.5915**	1.0161	1.0187
	original	0.6376	0.5935	1.0198	1.0190
ESTSP 2008-2	projection	1.0375	0.9985	0.9935	0.9790
	scaling	**0.9281**	0.9402	**0.9839**	**0.9652**
	original	0.9349	**0.9339**	0.9924	0.9727

From Table 4, we can observe that the OPELM performs generally better than the OPKNN. Furthermore, it is interesting to see that the OPKNN in the case of ESTSP 2007 data is the only one which actually benefits from the projection, in terms of accuracy. Finally, the HQ seems to perform better more often than the LOO, when taking into account both methods and all data sets.

Going more into details regarding each data set, we see from Table 4 that scaling is a little bit better than projection in the financial case. On the other hand, the scaling also has the advantage of easier interpretability of each variables. Table 5 shows the obtained scaled weights for some important variables for Finance data set.

Table 5: Scaling factors for Finance data

Index	Variable	Scaling factor
15	CCR	0.99
22	DD	0.98
9	AMORII	0.84
28	CPO	0.83
33	RCAI	0.76
27	CPER	0.68
2	Transaction	0.62
31	CFI	0.56
12	IF	0.52
7	CA	0.52
4	Vrif Rotation	0.48
25	Debt+1AN	0.45
18	AC	0.44
29	DA	0.40
26	TD	0.38
20	PRC	0.32
35	IS	0.32
8	II	0.21

As can be seen from Table 5, each variable has different importance for the ROA prediction. For example, not surprisingly, we find variable 'CCR' in the first place as accounts receivables help maximizing the ROA by providing a focused collection system while supplying the credit controls necessary to avoid bad debt losses. Therefore, scaling is the best choice we could have for the ROA prediction, not only because the test error is acceptable, but also because of the interpretability, which helps in analyzing the influences of each variable to the target.

Moving on to the ESTSP 2007 data, Figure 8 shows an example of the prediction performance with respect to the prediction horizon for all four models.

From Figure 8, we can see that there is no difference between the two selection methods in case of the OPKNN. Both, the LOO and the HQ give

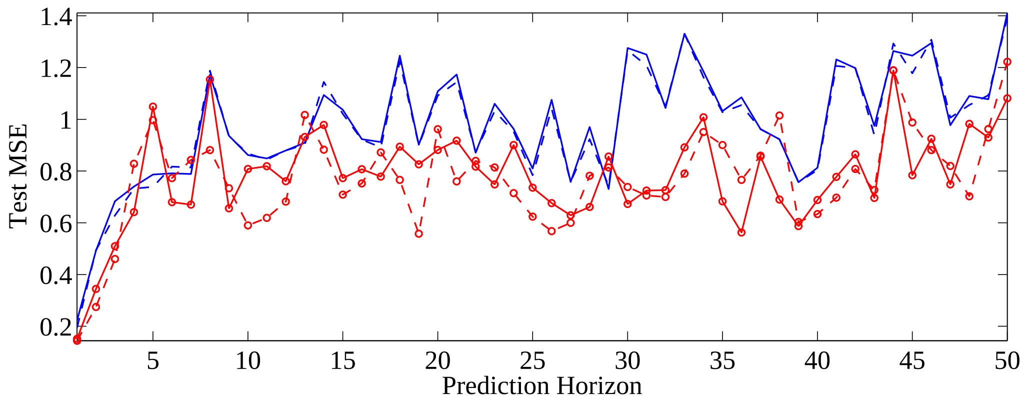

Figure 8: ESTSP 2007 competition data. Prediction errors of all methods. Solid lines are denoting methods optimized with the LOO and dashed ones with the HQ. Lines (red) with circles denote the OPELM methodology and the ones (blue) without any markers are denoting OPKNN.

very similar performance. That is not the case with the OPELM, where clear differences can be seen, but it is not clear which performs better. The performance of all methods are stable for the whole prediction horizon of 50 steps.

Figure 9 shows the values of the Delta Test for all prediction horizons for original, projected and scaled data.

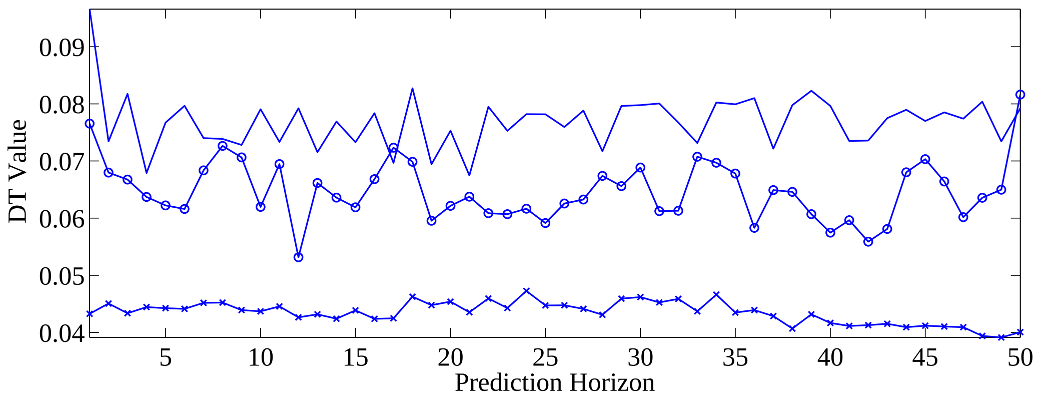

Figure 9: ESTSP 2007 competition data. The Delta Test values for original (solid line), projected (circles) and scaled (crosses) data set for all prediction horizons.

As we can see, the scaling obtains the lowest DT values, while the original data set using all 60 unprocessed inputs has the highest values. The projection lies in between. Comparing this ranking with the results presented in Table 4, both methodologies, the OPELM and the OPKNN, achieve roughly the same performance with the original data and the scaled one, even though the scaling obtains the lowest DT and the original the highest. The correspondence between the DT values and the actual performance of the prediction methodologies warrants further study.

Figure 10 shows the predictions and the 50 real values of the series in the test set. The predictions are obtained with OPELM with HQ using scaled data, since it showed the best performance.

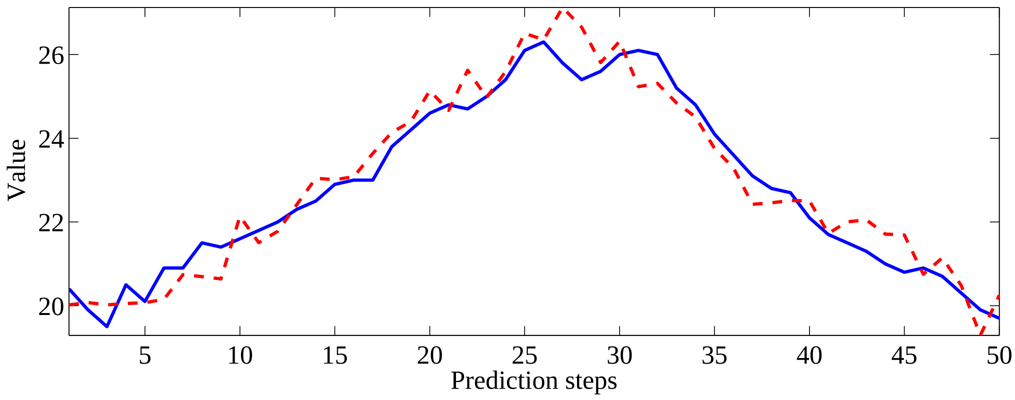

Figure 10: ESTSP 2007 competition data. Prediction for 50 time steps in the test set. Blue solid line represents the real data and red dashed one the prediction.

Finally, we move to ESTSP 2008-2 data set. First, two examples of the DT value evolution with respect to the projection dimension for two prediction horizons are shown in Figure 11.

From the two examples, we can see that the GA achieves lower DT value in horizon 8 than in horizon 78, thus leading to smaller final projection dimension. Also, from both examples, it is clear to see that the first 5 or 8 dimensions decrease the DT value quickly.

Figure 12 shows the number of GA rounds done with respect to the projection dimension.

As the average projection dimension is close to 6, Figure 12 also shows that the dimensions close to 6 have the largest amount of GA optimization rounds performed. The largest selected projection dimension is 14, which is still less than one third of the original dimensionality of the data. It means

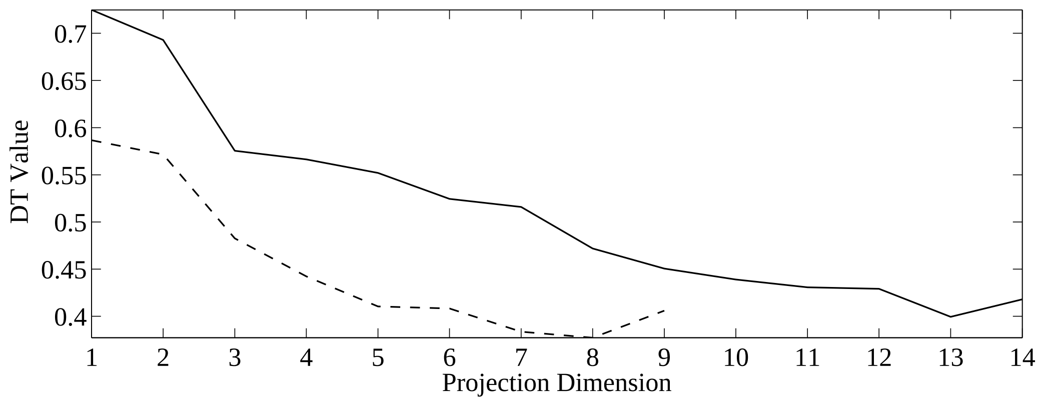

Figure 11: ESTSP 2008-2 data, prediction horizons 8 and 78. Example evolutions of the DT value of each projection dimension. Solid line denotes the horizon 78 and the dashed one the horizon 8.

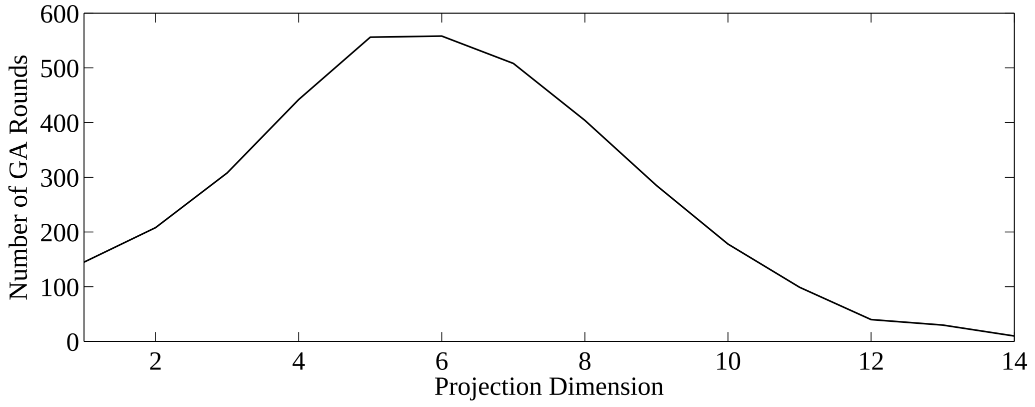

Figure 12: ESTSP 2008-2 data. Number of GA calculation rounds with respect to the projection dimension.

that even the projection ends up using 14 dimensions, it is still huge decrease from the original input space dimensionality.

Figure 13 shows the prediction for 100 steps ahead in the test set, as well as 100 real values of the series. OPELM with LOO is used to produce the estimations.

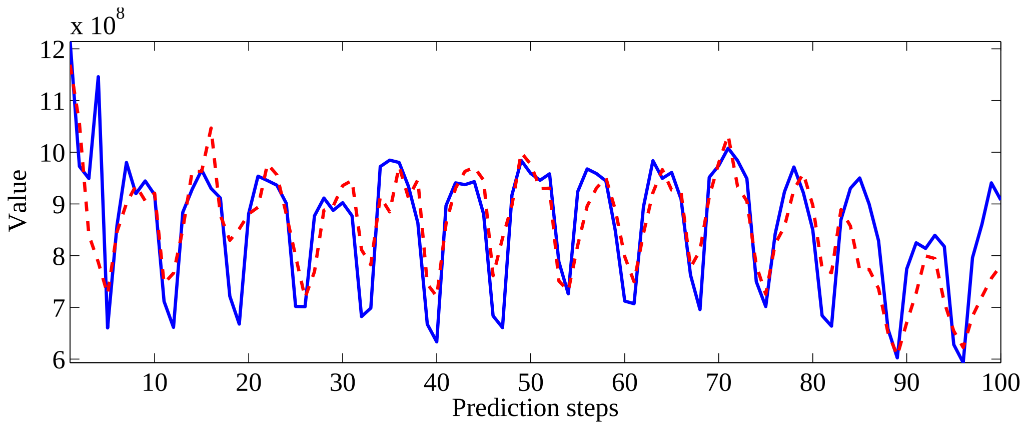

Figure 13: ESTSP 2008-2 data. Prediction for 100 steps ahead in the test set. Blue solid line represents the real data and red dashed one the prediction.

5. Conclusion

In this paper, we have presented a working solution to input processing and compared several methods for the long-term time series prediction. Also, one example from the field of Finance is included to demonstrate the interpretability achieved with scaling, a special case of variable projection.

We have also demonstrated that for each different data set, several input processing and prediction methods should be tried in order to have the optimum performance. It is not clear how to assess the performance *a priori* for any combination of methods when using different data sets. From the obtained results, the predictions from models trained on complete data set are very accurate, and in some cases more reliable than models that include variable projection/scaling as a preprocessing step (OPELM with HQ for ESTSP 2008-2 data). However, for ESTSP 2007 data for both models, input processing is needed to improve the prediction.

Using the GA with the DT for optimizing the projection matrix is a working combination, providing low-dimensional input sets for further processing, like time series prediction. The combination is able to provide low-dimensional input set, which can be crucial for methods that cannot handle large dimensions effectively. Out of the two tested models, only OPKNN benefits from projection, giving improved results for ESTSP 2007 data and comparable predictions for ESTSP 2008-2.

On the other hand, even though the scaling does not necessarily reduce the dimensionality, the obtained scaled inputs are usable, providing better test errors than the original. Furthermore, the number of parameters to optimize is lower than in projection, which makes the scaling faster.

For further work, the relationship between the DT values and the actual model performance will be studied. Also, the obtained projection performance will be evaluated in even more high-dimensional cases, to ensure the validity of the global search ability of the GA and to study the limits of the search.

References

[1] L. Ljung, System Identification: Theory for the User (2nd Edition), Prentice Hall PTR, 1998.

[2] G. Box, G. Jenkins, G. Reinsel, Time Series Analysis: Forecasting and Control, 3rd Edition, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[3] J. Hansen, R. Nelson, Forecasting and recombining time series components by using neural network, Computational Operational Research 50 (2003) 307–317.

[4] G. Zhang, E. Patuwo, M. Hu, A simulation study of artificial neural network for nonlinear time-series forecasting, Computational Operational Research 28 (2001) 381–396.

[5] A. Weigend, N. Gershendfeld, Times Series Prediction: Forecasting the Future and Understanding the Past, Addison-Wesley Publishing Company, 1994.

[6] M. Clements, P. Franses, N. Swanson, Forecasting economic and financial time-series with non-linear models, International Journal of Forecasting 20 (2) (2004) 169–183.

[7] M. Verleysen, D. François, Lecture Notes in Computer Science, Vol. 3512, Springer, Heidelberg, 2005, Ch. The curse of dimensionality in data mining and time series prediction, pp. 758–770.

[8] P. Brockwell, R. Davis, Introduction to Time Series and Forecasting, 2nd Edition, Springer, Berlin, 2002.

[9] P. D. McNelis, Neural Networks in Finance: Gaining Predictive Edge in the Market, Elsevier Academic Press, 2005.

[10] V. Kodogiannis, A. Lolis, Forecasting financial time series using neural network and fuzzy system-based techniques, Neural Computing & Applications 11 (2) (2002) 90–102. doi:10.1007/s005210200021.
URL `http://dx.doi.org/10.1007/s005210200021`

[11] E. Alpaydin, Introduction to Machine Learning, The MIT Press, 2004.

[12] A. Guillén, D. Sovilj, F. Mateo, I. Rojas, A. Lendasse, Minimizing the delta test for variable selection in regression problems, International Journal on High Performance Systems Architecture 1 (4) (2008) 269–281.

[13] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.

[14] I. Guyon, S. Gunn, M. Nikravesh, L. Zadeh, Feature Extraction: Foundations and Applications, Springer Verlag, 2006.

[15] D. Sovilj, A. Sorjamaa, Y. Miche, Tabu search with delta test for time series prediction using OP-KNN, in: A. Lendasse (Ed.), ESTSP, European Symposium on Time Series Prediction, 2008, pp. 187–196.

[16] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, A. Lendasse, Methodology for long-term prediction of time series, Neurocomputing 70 (16-18) (2007) 2861–2869.

[17] A. Sorjamaa, A. Lendasse, Time series prediction using dirrec strategy, in: M. Verleysen (Ed.), ESANN, European Symposium on Artificial Neural Networks, European Symposium on Artificial Neural Networks, 2006, pp. 143–148.

[18] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.

[19] I. Oh, J. Lee, B. Moon, Local search-embedded genetic algorithm for feature selection, in: ICPR, 2002, pp. II:148–151.
URL `http://dx.doi.org/10.1109/ICPR.2002.1048259`

[20] I. Oh, J. Lee, B. Moon, Hybrid genetic algorithms for feature selection, IEEE Trans. Pattern Anal. Mach. Intell 26 (11) (2004) 1424–1437.

[21] W. F. Punch, E. D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, R. Enbody, Further research on feature selection and classification using genetic algorithms, in: S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA'93), Morgan Kaufmann Publishers, San Mateo, California, 1993, pp. 557–564.

[22] M. Raymer, W. Punch, E. Goodman, L. Kuhn, A. Jain, Dimensionality reduction using genetic algorithms, IEEETEC: IEEE Transactions on Evolutionary Computation, A Publication of the IEEE Neural Networks Council 4.

[23] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19).

[24] F. Mateo, D. Sovilj, R. G. Gironés, A. Lendasse, RCGA-S/RCGA-SP methods to minimize the delta test for regression tasks, in: J. C. at el. (Ed.), Bio-Inspired Systems: Computational and Ambient Intelligence, 10th International Work-Conference on Artificial Neural Networks, IWANN 2009, Salamanca, Spain, June 10-12, 2009. Proceedings, Part I, Vol. 5517 of Lecture Notes in Computer Science, Springer, 2009, pp. 359–366.
URL `http://dx.doi.org/10.1007/978-3-642-02478-8`

[25] E. Liitiäinen, F. Corona, A. Lendasse, Nearest neighbor distributions and noise variance estimation, in: ESANN 2007, European Symposium on Artificial Neural Networks, Bruges (Belgium), 2007, pp. 67–72.

[26] C. R. Rao, S. K. Mitra, Generalized Inverse of Matrices and Its Applications, John Wiley & Sons Inc, 1972.

[27] Y. Miche, P. Bas, C. Jutten, O. Simula, A. Lendasse, A methodology for building regression models using extreme learning machine: OP-ELM, in: ESANN 2008, European Symposium on Artificial Neural Networks, Bruges, Belgium, 2008.

[28] T. Poggio, F. Girosi, A theory of networks for approximation and learning, Laboratory, Massachusetts Institute of Technology 1140.

[29] B. Caputo, K. Sim, F. Furesjo, A. Smola, Appearance-based object recognition using svms: Which kernel should i use?, in: Proc of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, 2002.

[30] T. Similä, J. Tikka, Multiresponse sparse regression with application to multidimensional scaling, in: Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, Vol. 3697/2005, 2005, pp. 97–102.

[31] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, in: Annals of Statistics, Vol. 32, 2004, pp. 407–499.

[32] R. Myers, Classical and Modern Regression with Applications, 2nd edition, Duxbury, Pacific Grove, CA, USA, 1990.

[33] G. Bontempi, M. Birattari, H. Bersini, Recursive lazy learning for modeling and control, in: European Conference on Machine Learning, 1998, pp. 292–303.

[34] H. Akaike, A new look at the statistical model identification, IEEE Transactions on Automatic Control 19 (6) (1974) 716–723.

[35] G. Schwarz, Estimating the dimension of a model, Annals of Statistics 6 (1978) 461–464.

[36] R. J. Bhansali, D. Y. Downham, Some properties of the order of an autoregressive model selected by a generalization of akaike's epf criterion, Biometrika 64 (3) (1977) 547–551.

[37] E. J. Hannan, B. G. Quinn, The determination of the order of an autoregression, Journal of the Royal Statistical Society, B 41 (1979) 190–195.

[38] Y. Miche, A. Lendasse, A faster model selection criterion for op-elm and op-knn: Hannan-quinn criterion, in: ESANN'09: European Symposium on Artificial Neural Networks, Bruges, Belgium, 2009.

[39] Q. Yu, E. Séverin, A. Lendasse, Variable selection for financial modeling, in: CEF 2007, 13th International Conference on Computing in Economics and Finance, Montréal, Quebec, Canada, 2007, pp. 237–241.

[40] J. K. Kapler, Measuring the economic rate of return on assets, Review of Industrial Organization 17 (4) (2000) 457–463.