Tabu Search with Delta Test for Time Series Prediction using OP-KNN

Dušan Sovilj, Antti Sorjamaa and Yoan Miche

Helsinki University of Technology - Department of Information and Computer Science
P.O.Box 5400, 02150 HUT - Finland

Abstract. This paper presents a working combination of input selection strategy and a fast approximator for time series prediction. The input selection is performed using Tabu Search with the Delta Test. The approximation methodology is called Optimally-Pruned k-Nearest Neighbors (OP-KNN), which has been recently developed for fast and accurate regression and classification tasks. In this paper we demonstrate the accuracy of the OP-KNN with the Tabu Search using the ESTSP 2008 Competition datasets.

1 Introduction

The amount of information is increasing rapidly in many fields of science. It creates new challenges for storing the massive amounts of data as well as to the methods, which are used in the data mining processes. In many cases, when the amount of data grows, the computational complexity of the used methodology also increases. In this paper, we combine a clever way of selecting the input data and a very fast approximation method for time series prediction task.

In the time series prediction the problem is prediction of future values based on appropriate number of previous values in the series. This number can be decided empirically with trial and error and/or by looking at the periodicity of the series itself. Since the approximation model we are using is very fast, we used both conditions upon decisions of good regressor sizes for the ESTSP 2008 competition datasets.

Further performance improvement can be achieved with adequate input selection strategy. Delta Test is a nonparametric noise estimator which can be used for that purpose [1]. Since exhaustive search is the only way to obtain global optimum, but it is impractical to use with large datasets, a common strategy is to rely on suboptimal Forward-Backward selection. To overcome the problem of local optima we use Tabu Search, which incorporates additional memory and can be extended with various heuristics.

After variable selection with Tabu Search using Delta Test as input selection criterion, actual prediction is done with Optimally-Pruned k-Nearest Neighbors (OP-KNN). It is based on the inspiring work on Extreme Learning Machine (ELM) by Guang-Bin Huang *et al.* in [2] and on Optimally-Pruned ELM by Yoan Miche *et al.* in [3]. Unlike the two previous methods, the OP-KNN is deterministic and simple, while still retaining the speed of the ELM and the accuracy of the OP-ELM. Since we are predicting several values ahead, we focus on using Direct Strategy [4] over Recursive for better long-term prediction accuracy. The Direct Strategy includes building a separate model for every time step. This also includes separate variable selection.

Section 2 presents the Tabu Search and the Delta Test estimation methods. In Section 3, the OP-KNN methodology is explained and finally, Section 4 presents the experimental results using ESTSP 2008 Competition benchmarks.

2 Tabu Search (TS)

Tabu Search [4] resembles the Forward-Backward Selection procedure [5], which only continues along selections that improve the objective function. Tabu Search tries to overcome the problem of local optima with additional use of memory and by considering solutions which do not improve the objective function. Memory is used to keep track of already visited solutions and those are prevented from being explored again, as they are in *tabu* state, hence the name of the method. On the other hand, there are no guarantees that the TS will find the global optimum, but results found with it are usually better than the ones obtained with plain greedy approach.

Consider the optimization problem $f(x), x \in X$, where $f(x)$ is the objective or cost function and x is one possible solution for the problem. The $f(x)$ in our case is the Delta Test. In the context of Tabu Search the neighborhood of solution x, denoted $N(x)$, plays an important role. $N(x)$ is a set of solutions reachable from x via transition moves.

The weakening of search criterion, moving to solutions with worse f values, introduces a problem of cycles, and this is *one* of the reasons for using memory. One part of memory is used to prevent already visited solutions in $N(x)$ to be explored again. Memory is divided into two parts, short term and long term memory. The short term memory strategies are focused on directly modifying the set of neighbors $N(x)$. The long term memory strategies can include solutions, which are not part of the $N(x)$ and which can generate solutions based on the attributes of the good solutions.

The simplest approach is to use only the short term memory, to mark solutions already visited as tabu and to choose solutions from $N(x)$ that have the best f value. This approach is called Simple Tabu Search [4]. The neighborhood $N(x)$ will be influenced by the contents of the memory and the set will change during the search. This is why the TS is sometimes called *dynamic neighborhood search technique*. The size of solutions would make TS impractical to use and it is much more efficient to store transition moves or some other attributes of vector x.

In the case of variable selection, the neighborhood is easily formed by having two solutions being neighbors if they *disagree* on the selection of exactly one variable v_k. The move that links these two neighbors is just flipping the state of the variable v_k. The size of the search space grows exponentially with the dimensionality of the inputs and for a dataset with a lot of variables it is very difficult to find optimal selection.

2.1 Delta Test (DT)

Delta Test is a Nonparametric Noise Estimator based on a nearest neighbor principle. The nearest neighbor of a point is defined as the (unique) point, which minimizes a distance metric to that point. Distance metric is usually the Euclidean distance, but other metrics can also be used.

In function approximation, we have a set of input points $(x_i)_{i=1}^N$ and associated scalar outputs $(y_i)_{i=1}^N$. The assumption is that there is a functional dependence between them, but with an additive noise term $y_i = f(x_i) + \epsilon_i$. The function f is assumed to be smooth, and the noise terms ϵ_i are independent and identically distributed with zero mean. Noise variance estimation is the study of how to give an *a priori* estimate for $\mathrm{Var}(\epsilon)$ given some data *without* considering any specifics of the shape of f.

Using the previous notation, the Delta Test is usually written as

$$\mathrm{Var}(\epsilon) \approx \frac{1}{2N} \sum_{i=1}^{N} (y_i - y_{P(i)})^2,$$

where $P(i)$ defines the nearest neighbor of x_i in the input space. Hence, using the DT, we consider the estimate of the noise as the mean of the differences in the outputs associated with neighboring points (divided by 2). This is a well-known and widely used estimator, and it has been shown for example in [6] that the estimate converges to the true value of the noise variance in the limit $N \to \infty$.

3 Optimally-Pruned *k*-Nearest Neighbors (OP-KNN)

The OP-KNN is similar to the OP-ELM [3], which is an original and efficient way of training a feedforward neural network. The three main steps of the OP-KNN are summarized in Figure 1.

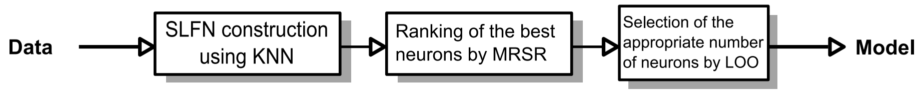
Data
SLFN construction using KNN
Ranking of the best neurons by MRSR
Selection of the appropriate number of neurons by LOO
Model

Fig. 1: The three steps of the OP-KNN algorithm.

3.1 Single hidden Layer Feedforward Neural Network (SLFN)

The first step of the OP-KNN algorithm is similar to the original ELM: building of a Single hidden Layer Feedforward Neural network. The idea of the ELM has been proposed by Guang-Bin Huang *et al.* in [2].

In the context of a single hidden layer network, let us denote the weights between the hidden layer and the output by $\mathbf{b}$. Activation functions used with the OP-KNN differ from the original SLFN choice since the original sigmoid activation functions of the neurons are replaced by the k-Nearest Neighbors, hence the name OP-KNN. For the output layer, the activation function remains as a linear function.

A theorem proposed in [2] states that the output weights $\mathbf{b}$ can be computed from the hidden layer output matrix $\mathbf{H}$: the columns $\mathbf{h_i}$ of $\mathbf{H}$ are the corresponding output of the k-nearest neighbors.

Finally, the output weights $\mathbf{b}$ are computed by $\mathbf{b} = \mathbf{H}^{\dagger}\mathbf{y}$, where $\mathbf{H}^{\dagger}$ stands for the Moore-Penrose inverse [7] and $\mathbf{y} = (y_1, \ldots, y_M)^T$ is the output.

The only remaining parameter in this process is the initial number of neurons N of the hidden layer.

3.2 *k*-Nearest Neighbors (KNN)

The k-Nearest Neighbors model is a very simple, but powerful tool. It has been used in many different applications and particularly in classification tasks. The key idea behind the KNN is that similar training samples have similar output values. In the OP-KNN, the approximation of the output is a weighted sum of the outputs of the k-nearest neighbors as

$$\hat{y}_i = \sum_{j=1}^{k} b_j y_{P(i,j)},$$

where $\hat{y}_i$ represents the output estimation, $P(i,j)$ is the index of the j^{th} nearest neighbor of sample $\mathbf{x}_i$ and b is the result of the Moore-Penrose inverse introduced in the previous section.

3.3 Multiresponse Sparse Regression (MRSR)

For the removal of the useless neurons of the hidden layer, the Multiresponse Sparse Regression proposed by Timo Similä and Jarkko Tikka in [8] is used. It is an extension of the Least Angle Regression (LARS) algorithm [9] and hence is actually a variable ranking technique, rather than a selection one.

An important detail shared by the MRSR and the LARS is that the ranking obtained is exact in the case where the problem is linear. In our case this is true, since the neural network built in the previous step is linear between the hidden layer and the output. Therefore, the MRSR provides the exact ranking of the neurons for our problem.

MRSR is hence used to rank the kernels of the model: the target is the actual output y_i while the "variables" considered by MRSR are the outputs of the k-nearest neighbors.

3.4 Leave-One-Out (LOO)

Since the MRSR only provides a ranking of the kernels, the decision over the actual best number of neurons for the model is taken using a Leave-One-Out method. One problem with the LOO error is that it can be very time consuming to calculate if the dataset 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 [10, 11]. The LOO error using the PRESS statistics can be written as

$$\epsilon^{\text{PRESS}} = \frac{y_i - \mathbf{h}_i \mathbf{b}}{1 - \mathbf{h}_i \mathbf{P} \mathbf{h}_i^T},$$

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T\mathbf{H})^{-1}$ and $\mathbf{H}$ as the hidden layer output matrix defined in Section 3.1.

The final decision over the appropriate number of neurons for the model is taken by minimizing the LOO error versus the number of neurons used (properly ranked by the MRSR already).

4 Experiments

In the experiments, we are using the datasets of the ESTSP 2008 Competition. The section is divided into three subsections and each presents one dataset. The first dataset is explained more deeply and the latter two are less detailed.

What is common to all datasets is the selection of the initial regressor sizes for the TS. The OP-KNN was used with several different regressor sizes with all the variables and the ones with the smallest LOO errors were selected for each dataset. For the dataset 3, also the periodicity of the data was taken into account. The inputs and outputs are normalized before running the Tabu Search.

After the selection of the initial regressor sizes, the Tabu Search with the Delta Test as the cost function was used. The set of variables, which had the lowest delta value, was selected individually for each prediction step. This means, that the prediction strategy used was the Direct Strategy [5], where each prediction step needs its own selection of variables and its own model to be built.

Tabu Search was implemented as a Simple Tabu Search, where only recent moves, or changes of variables, were kept in memory. The memory was set to 1/5 of the dimensionality of samples. For example, for sample with 10 dimensions, the memory size would be set to 2 so that only the last 2 applied moves are considered tabu.

Stopping conditions for TS varied from one dataset to another due to the different sizes of the initial regressors. After the selection of the variables, the final model was obtained using the OP-KNN. All the data was used in the training after the preliminary tests showed that the LOO and the test errors behaved in the same way with each other.

4.1 Dataset 1: Multidimensionality

Dataset 1 consists of 3 time series with one of them being the target series, and each of them have 354 values. All 3 series are shown in Figure 2 with the target one being on the top of the figure.

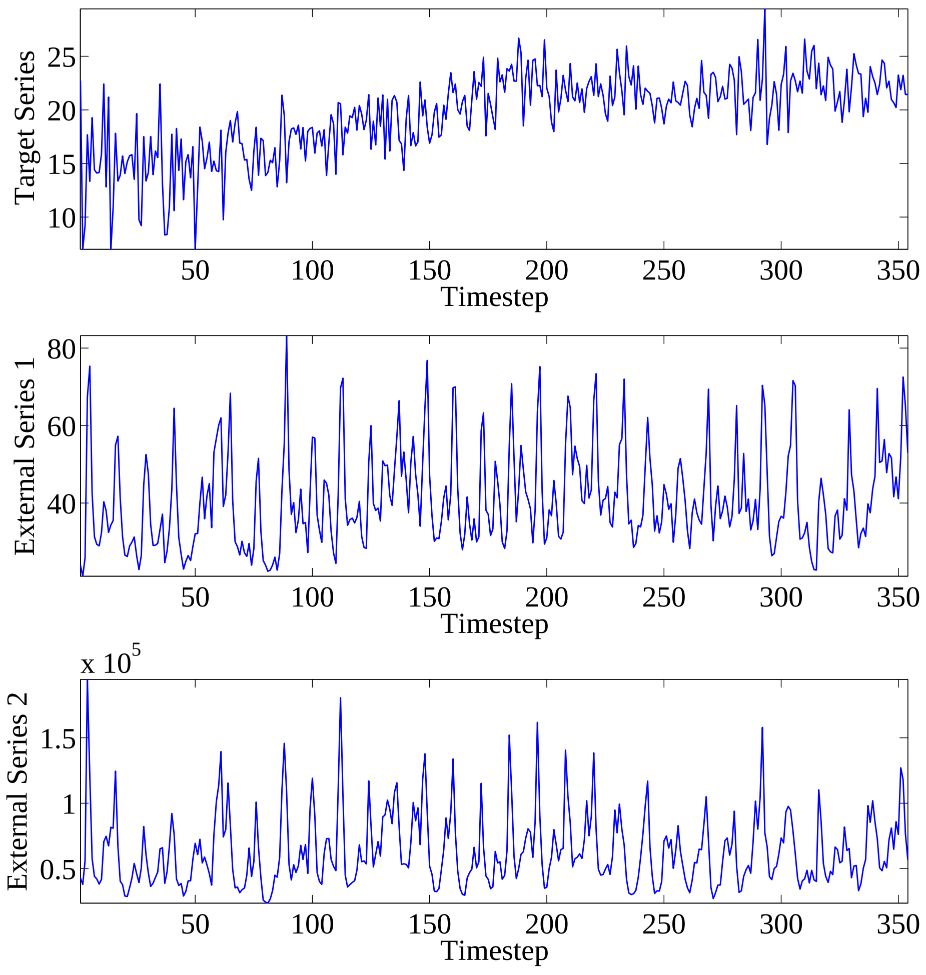

Fig. 2: Dataset 1. Top one is the series itself and the two lower ones are the external series.

First step was the removal of the trend present in the target series by transforming it using first order difference. The OP-KKN results showed that regressor of size 20 is the optimum for the target series. From both external series we chose 12 variables as extra inputs, based on the autocorrelations with the target series. As the number of selected variables is lower than for the target series, some of the values from the beginning of the two external series are removed. The alignments of two external 12 variables are done in such a way so that we always use the latest measurements as possible. Thus, the final regressor size was 44 and the final number of training samples was 316.

Since the search space has $2^{44} - 1$ instances, the TS was run 24 times from random starting points with 1 hour each. Roughly 10^6 solutions were examined in every run, a very small percentage of the whole space, and the found delta value is probably not the global optimum. Figure 3 shows that the variable selection with Tabu Search using Delta Test does improve overall performance of the OP-KNN with respect to the LOO error. In the same figure the Delta values found by the TS are also shown.

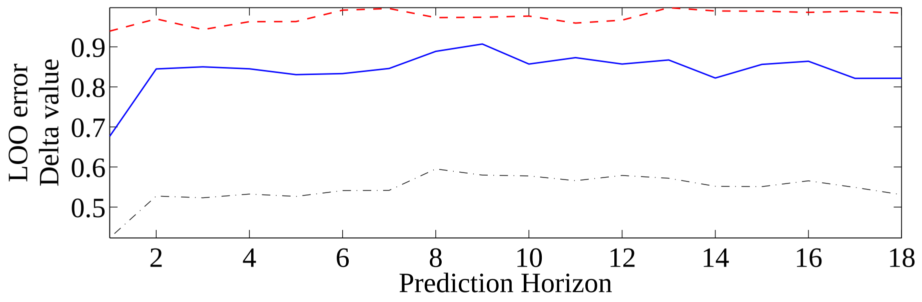

Fig. 3: LOO errors of 18 OP-KNN models with all variables (dashed line) and the selected variables with the Tabu Search (solid line). For comparison, also the Delta Test values found with the Tabu Search (dot-dashed line).

The Delta Test gives an estimate that one can achieve in terms of the training error without overfitting the model. Using the TS the LOO error was decreased more than 25% with respect to the Delta value. The Delta values are fairly high for this dataset (over 0.5), which suggests that the target series is very noisy, and therefore, hard to predict.

The predictions for the 18 steps ahead are shown in Figure 4.

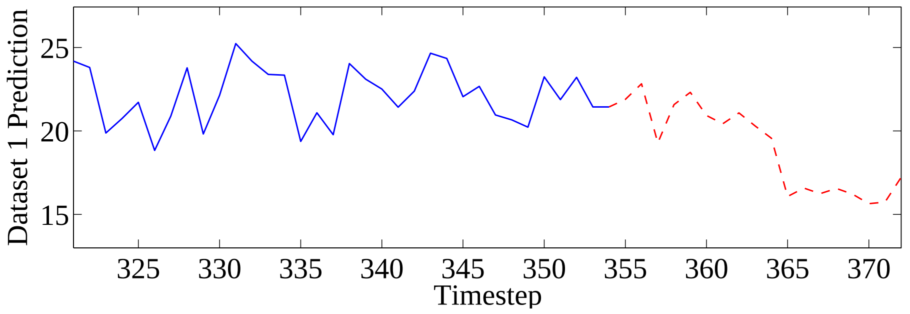

Fig. 4: Prediction of 18 steps for target series of Dataset 1. Solid line represents last known values and dashed one the prediction.

4.2 Dataset 2: Easy Sailing

Dataset 2 consists of 1300 values and it is shown in Figure 5.

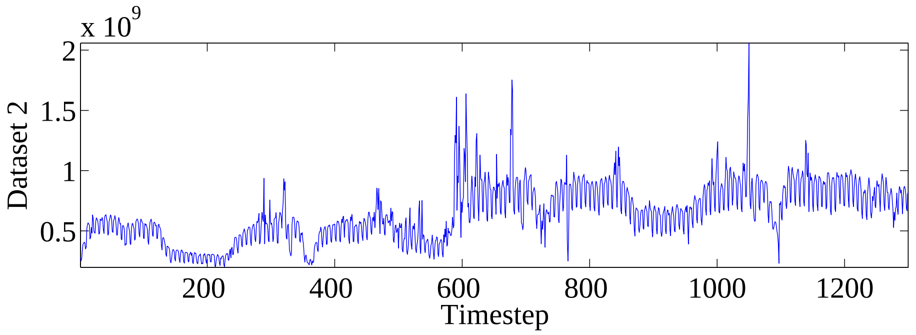

Fig. 5: Dataset 2.

Because the data seems to have two distinct levels, which changes just before the timestep 600, we decided to normalize the dataset in two parts: first part including the values before the 600 and the second part after it. At the same time, we have the whole set normalized for the TS.

The initial regressor size for the TS was selected to 20, which leaves us with roughly 1180 samples for the TS phase and the training of the OP-KNN. The stopping criterion for the TS was set to 10 percent of the search space, where the running times were around 1.5 hours for each prediction step.

The predictions for 100 steps ahead are shown in Figure 6.

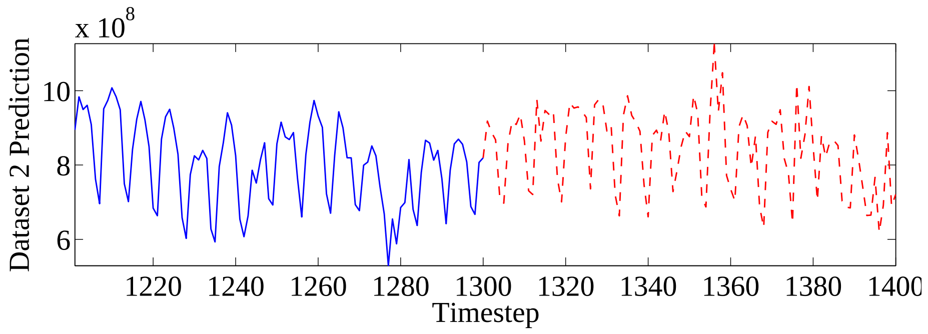

Fig. 6: Dataset 2 prediction of 100 steps ahead using the OP-KNN with Tabu selected variables. Solid line represents the known values and the dashed one is the prediction.

4.3 Dataset 3: Measurements Aplenty

Data 3 includes vast amount of data, more than 31 000 samples, shown in Figure 7.

Since the dataset had a clear periodicity of 24, we decided to cut the dataset into slices of 24 rather than build the traditional regressor by taking the samples

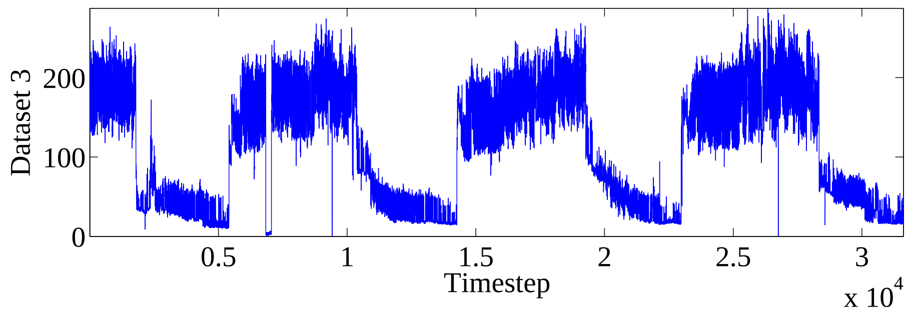

Fig. 7: Dataset 3.

using moving window through the dataset. After the slicing, our dataset has roughly 1300 samples of 24 dimensions.

Because the dataset exhibits heavy long-term seasonality, we decided to use sample-wise normalization and predict the normalized series, the means and the standard deviations separately. It means that in order to get 200 step ahead prediction for the original dataset, we needed to predict 200 steps for the normalized samples and 9 steps ahead for the means and standard deviations to be added and multiplied with the normalized prediction.

In the prediction of the normalized samples, we used two previous days as the initial regressor size for the TS. Hence, we have a regressor size 48. For the means and standard deviations the regressor size was selected to be 15. For normalized samples we had only 1 run with TS lasting 6 hours, while for the means and standard deviations the stopping criterion was 10 percent of solution space (same as in Dataset 2).

After similar steps than with the dataset 1, we obtain the prediction for 200 steps shown in Figure 8.

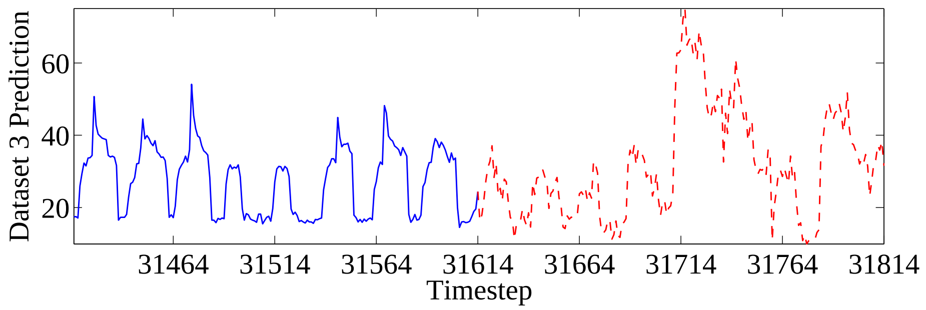

Fig. 8: Dataset 3 with the prediction of 200 steps using the OP-KNN with Tabu selected variables. The solid line denotes the known values and the dashed one the predicted.

5 Conclusions

In this paper we have shown the efficiency of the combination of a clever and adaptable input variable selection method, the Tabu Search, and a fast prediction method, the OP-KNN.

The results are satisfying and obtained with a small calculation time. The Tabu Search can be given a predefined amount of time for its search, which covers the search space better than plain greedy methods. However, it is also able to give acceptable results in reduced time, if necessary.

The OP-KNN is very fast to train and to use in the prediction task. The accuracy of the method can be improved with input selection, which on the other hand will increase the total calculation time. This creates the need for fast input selection methodology.

For further work, the Tabu Search will be improved in terms of speed and used heuristics with better finetuning for different datasets. The OP-KNN methodology will be tested more extensively with different input selection methods as well as with different regression tasks.

References

[1] E. Eirola, E. Liitiäinen, A. Lendasse, F. Corona, and M. Verleysen. Using the delta test for variable selection. In *ESANN 2008, European Symposium on Artificial Neural Networks, Bruges (Belgium)*, April 2008.

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

[3] Y. Miche, P. Bas, C. Jutten, O. Simula, and 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*, April 23-25 2008.

[4] Fred Glover and Manuel Laguna. *Tabu Search.* Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[5] Antti Sorjamaa, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16-18):2861–2869, October 2007.

[6] E. Liitiäinen, F. Corona, and A. Lendasse. Nearest neighbor distributions and noise variance estimation. In *ESANN 2007, European Symposium on Artificial Neural Networks, Bruges (Belgium)*, April 25-27 2007.

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

[8] T. Similä and J. Tikka. Multiresponse sparse regression with application to multidimensional scaling. In *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005*, volume 3697/2005, pages 97–102. 2005.

[9] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. In *Annals of Statistics*, volume 32, pages 407–499. 2004.

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

[11] G. Bontempi, M. Birattari, and H. Bersini. Recursive lazy learning for modeling and control. In *European Conference on Machine Learning*, pages 292–303, 1998.