Long-Term Time Series Forecasting Using Self-Organizing Maps: the Double Vector Quantization Method

Geoffroy Simon
Université catholique de Louvain
DICE - Place du Levant, 3
B-1348 Louvain-la-Neuve Belgium
Email: simon@dice.ucl.ac.be

Amaury Lendasse
Université catholique de Louvain
CESAME - Avenue Georges Lemaitre, 4
B-1348 Louvain-la-Neuve Belgium
Email: lendasse@auto.ucl.ac.be

Marie Cottrell
Université Paris I - Panthéon Sorbonne
UMR CNRS 8595
SAMOS-MATISSE - Rue de Tolbiac, 90
F-75634 Paris cedex 13 France
Email: marie.cottrell@univ-paris1.fr

Michel Verleysen
Université catholique de Louvain
DICE - Place du Levant, 3
B-1348 Louvain-la-Neuve Belgium
Email: verleysen@dice.ucl.ac.be

Abstract

Kohonen self-organisation maps are a well know classification tool, commonly used in a wide variety of problems, but with limited applications in time series forecasting context. In this paper, we propose a forecasting method specifically designed for long-term trends prediction, with a double application of the Kohonen algorithm. We also consider practical issues for the use of the method.

1. Introduction

The self-organizing maps (SOM), developed by Teuvo Kohonen in the 80's [1], has now become a well-known tool, with established properties [2, 3]. These Kohonen self-organizing maps have been commonly used since their first description in a wide variety of problems, as classification, feature extraction, pattern recognition and other related applications

As shown in a few previous works [4, 5, 6, 7, 8, 9], the SOM may be used to forecast time series at short term. The method presented here shows how long-term forecasting can be achieved when applying the SOM algorithm twice, the goal being to predict global trends and not accurate next values. Furthermore we will explain how we can use favourably the SOM to generalize the method for time series where we want to predict more than a single scalar value. For example, in the case of some a priori known periodicity, we could be interested to predict all values for a period as a whole. Such a case will be called a multi-dimensional forecasting in the later, to be opposed to the prediction of a unique scalar value, the one-dimensional case.

In the following of this paper, we first recall some basic concepts about the SOM classification tool. Then we introduce the proposed forecasting method, the double vector quantization, and explain how to use it with one- and multi-dimensional time series. Next, we introduce the problem of model structure selection and describe a methodology to validate the forecasting capacities of our method. Finally, some experimental results will be shown for both the one- and multi-dimensional cases.

2. The Kohonen Self-Organizing Maps

The Kohonen self-organizing maps (SOM) can be defined as an unsupervised classification algorithm from the artificial neural network paradigm. Any run of this algorithm results in a set, with a priori fixed size, of prototypes. Each one of those prototypes is in fact a vector of the same dimension as the input space. A physical neighbourhood relation links the prototypes. Due to this neighbourhood relation, we can easily graphically represent the prototypes in a 1- or 2-dimensional grid.

After the learning stage each prototype represents a subset of the initial input set in which the inputs share some similar features. Using Voronoï's terminology, the prototype corresponds to a centroïd of a region or zone, each zone being one of the classes obtained by the algorithm. The SOM thus realizes a vector quantization (VQ) of the input space (a Voronoï tessellation) that respects the original distribution of the inputs.

Furthermore, a second property of the SOM is that the resulting prototypes are ordered according to their location in the input space. Similar features in the input space are associated either to the same prototype (as in classical VQ), or to two prototypes that are neighbours on the grid. This last property, known as the topology preservation, does not hold for other standard VQ methods like competitive learning.

Though we could have chosen some other classical VQ method, we decided to use the SOM in our double vector quantization method because the ordered prototypes can be easily represented graphically, allowing a more intuitive interpretation: the 1- or 2-dimensional grid can be viewed as a 1- or 2-dimensional space where the inputs are projected by the SOM algorithm, even if, in fact, the inputs are rather projected on the prototypes themselves (with some interpolation if needed in continuous case). This projection operation for some specific input is proceeded by determining the nearest prototype with respect to some distance metric (usually the Euclidian distance).

3. The double quantization method

The method described here aims to forecast the long-term evolution of a time series. It is based on the SOM algorithm and can be divided into two stages: the characterization and the forecasting. The characterization stage can be viewed as the learning, while the forecasting can be viewed as the use of a model in a generalization procedure.

The method is presented for one-dimensional (scalar) data and can be extended naturally to higher-dimension data. Discussion about the method application to one- and multi-dimensional time-series will be provided.

3.1. Method description: the characterization

Though the determination of an optimal regressor in time series forecasting (at least in a non-linear prediction case) is an interesting and open question, we consider here that we know the optimal, or at least an adequate, regressor of the time series. Classically, the regressor can for example be chosen according to some statistical resampling (cross-validation, bootstrap, etc.) procedure.

As for many other time series analysis methods, we convert the inputs into regressors, leading to n-λ+1 data in a λ-dimension space, where λ is the regressor size, or lagging order, and n the number of data at our disposal in the time series. We note the resulting data vectors $x_t = \{x(t\text{-}\lambda\text{+}1), \ldots, x(t)\}$, where $\lambda \le t \le n$, and $x(t)$ is the original time series at our disposal with $1 \le t \le n$.

We then manipulate the obtained vectors x_t and create the so-called "deformations" y_t according to:

$$y_t = x_{t+1} - x_t \,. \qquad (1)$$

Note that each y_t is associated to one x_t. Putting all y_t together in chronological order, we get a second time series, the deformation series in the so-called deformation space to be opposed to the original space containing the x_t. Of course, we have n-λ deformations of dimension λ.

We can now apply the self-organizing map algorithm to each of these two spaces, classifying both the original data x_t and the deformations y_t respectively. Note that in practice any kind of map can be used, but we believe that one-dimensional maps (or strings) are more adequate in this context.

As a result of the vector quantization by the SOM on all x_t data of the original space, we obtain n_1 prototypes X_i with $1 \le i \le n_1$. Let us denote C_i the Voronoï class associated to X_i. For the second application of the SOM on all deformations y_t in the deformation space, we obtain n_2 prototypes Y_j, $1 \le j \le n_2$. Similarly we use the notation C'_j for the Voronoï class associated to Y_j. Note that section 4 will be devoted to the question of choosing the values of n_1 and n_2.

To perform the forecasting, we need some more information than the two sets of prototypes. We therefore compute a matrix f_{ij} based on the relations between the x_t and the y_t with respect to their projections on the X_i and Y_j respectively. We call this matrix f_{ij} the frequency table, where element (i,j) contains the number of x_t projected on prototype X_i for which the associated deformations y_t are projected to the prototype Y_j. In other words, we count the number of x_t in C_i such that the corresponding y_t are in C'_j. Those empirical frequencies are normalized by the number of data in the subset of the original space corresponding to the prototype X_i i.e. the number of x_t in class C_i. More formally:

$$f_{ij} = \frac{\#\left\{x_t \in C_i \text{ such that } y_t = x_{t+1} - x_t \in C'_j\right\}}{\#\left\{x_t \in C_i\right\}}, \qquad (2)$$

with $1 \le i \le n_1$, $1 \le j \le n_2$. The class of each data is determined by projecting them on the best matching prototype according to the Euclidean distance metric used in Kohonen algorithm. In that frequency table, each line i represent the empirical conditional probability that the deformation y_t belongs to the classes C'_j in the deformation space, given the fact that the original vector x_t belongs to class C_i in the original space. The computation of this frequency table completes the characterization part of the method.

3.2. Method description: the forecasting

Once we have the prototypes X_i and Y_j together with the frequency table, we can forecast a time series evolution over a rather long-term horizon k, where horizon 1 is defined to be the next value, i.e. t+1 for instant t.

The methodology for such a forecasting can be described as follows. First, consider a new input $x(t)$ for some instant t. Knowing the $x(t)$ series until time t, we compute the corresponding x_t. Therefore we can find the corresponding prototype in the original space, for

example X_k (this operation is in fact equivalent to determining the class C_k of x_t). We then look in the frequency table and randomly choose a deformation prototype Y_l among the Y_j according to the frequency distribution defined by f_{kl}, $1 \le l \le n_2$. The prediction for instant t+1 is obtained using relation (1):

$$\hat{x}_{t+1} = x_t + Y_l \,. \qquad (3)$$

where $\hat{x}_{t+1}$ is the estimate of x_{t+1} given by our time series prediction model. However in this case we are not interested in a vector $\hat{x}_{t+1}$ of predictions, but only in a single scalar value $\hat{x}(t+1)$. The predicted value $\hat{x}(t+1)$ is finally extracted from the last column of $\hat{x}_{t+1}$.

We can iterate the described procedure, plugging in $\hat{x}(t+1)$ for $x(t)$, computing x_{t+1}, obtaining $\hat{x}_{t+2}$ and extracting $\hat{x}(t+2)$. We then do the same for $\hat{x}(t+3)$, $\hat{x}(t+4)$, …, $\hat{x}(t+k)$. This ends the run of the algorithm to obtain a single prediction of the series at horizon k.

Next, remind that the goal of the method is not to perform a single long-term prediction, but to extract tendencies from possible forecasts. Therefore we repeat many times the whole long-term forecasting procedure at horizon k, as detailed above. As part of the method (random choice of the deformation in the frequency table, according to an empirical law) is stochastic, repeating the procedure leads to different forecasts. Observing the evolution of all those different long-term predictions together with the evolution of their mean, we can infer a global trend for the future of the time series.

We would like to emphasis once again on the fact that the double quantization method is not designed for the problem of determining a precise estimate for instant t+1 but is more specifically devoted to the problem of long-term evolution, which can only be obtained in terms of trends.

3.3. Using the method

As introduced earlier, we can apply our method to both one- and multi-dimensional cases. We will now explain how to manipulate the time series to make this possible.

First, we consider the one-dimensional case. Consider a scalar time series as for example the one shown in Fig 1.1 and Fig 1.2.

The simplest idea is to use a single value from the series, the last known value at time t, to predict the value at time t+1. Such way of working does not use many of the available past information on the series and therefore produces low-quality forecasts. This first approach is illustrated in Fig. 1.1. The solid segments correspond to the past information used for prediction, the dotted ones to the prediction itself. Each horizontal line corresponds to a single forecasting. The predictions sequence is symbolized by the top-down left-right ordered sequence of horizontal lines.

To improve the forecasting, we first transform the data into regressors of size λ. We obtain n-λ+1 data of dimension λ from the original 1-dimensional time series of n data. In our case, each prediction is obtained by forecasting a λ-dimensional vector $\hat{x}_{t+1}$; a simple scalar value of interest is then extracted from this vector. This is illustrated in Fig 1.2. The dotted segments in Fig. 1.2 correspond to the $\hat{x}(t+1)$ values extracted from the $\hat{x}_{t+1}$ predictions.

The choice of an optimal regressor falls out of the scope of this paper. In the following the regressor is supposed to contain the most relevant information from the past evolution of the time series i.e. we suppose it to be optimal.

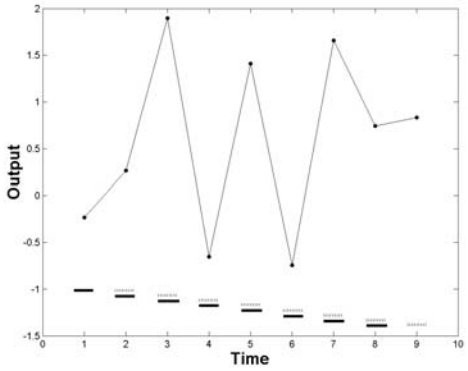

Figure 1.1 **: Using scalar information *t* (solid) for *t*+1 prediction (doted).**

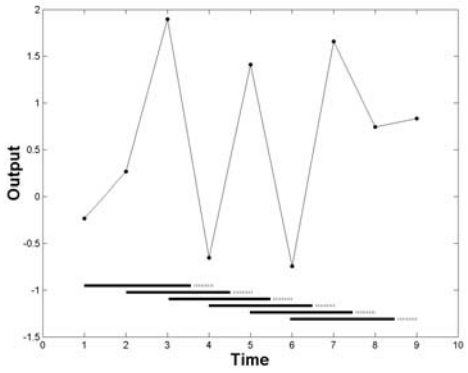

Figure 1.2 **: Applying a regressor to the data: using *t-k*, …, *t* (solid) to predict *t*+1 (doted).**

Secondly, we now consider the multi-dimensional case. This case can be encountered when, for example, we know that there exists some periodicity μ in the time series. In such situation it could be advantageous to forecast in a single step all values for the existing period, what we will call in the later a block (see Fig 2.1 and Fig 2.2).

In these figures, we illustrate a series where the size of the blocks is μ=7. Fig. 2.1 shows a prediction paradigm similar to the one in Fig. 1.1, where the scalar values have been replaced by 7-dimensional blocks. In other words, the prediction of the next 7 values ($\hat{x}(t+1)$

to $\hat{x}(t+7)$) is based only on the last 7 known ones (x(t-6) to x(t)). Considering again an original scalar series of n values, we are able to build n/μ-1 regressors (under the hypothesis that n is a multiple of μ). The prediction procedure described in Section 3.2 is then applied in an identical way, with $\lambda=\mu$. Only the last step of this procedure is changed: instead of keeping only one value $\hat{x}(t+1)$ from the vector prediction $\hat{x}_{t+1}$, we keep all μ values of this vector as forecasts.

The next step is to extend this procedure to the case where more than μ past values are considered. This is illustrated in Fig. 2.2, where the size λ of the regressors is a multiple p (here p=2) of μ. In that case again, the procedure from Section 3.2 is applied, with $\lambda=p\mu$. μ predictions ($\hat{x}(t+1)$ to $\hat{x}(t+k)$) are then extracted from the λ-dimensional prediction $\hat{x}_{t+1}$.

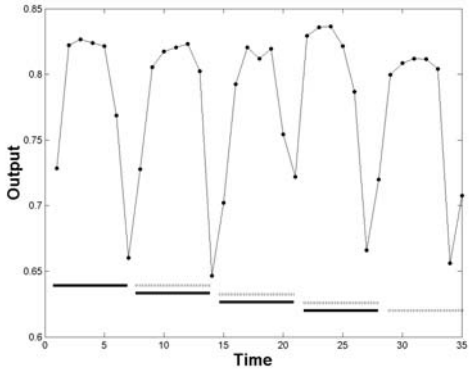

Figure 2.1 **: Predicting a block at instant *t+1* (doted) of μ values (corresponding to the periodicity of the time series) using the information of the block at *t* (solid)**

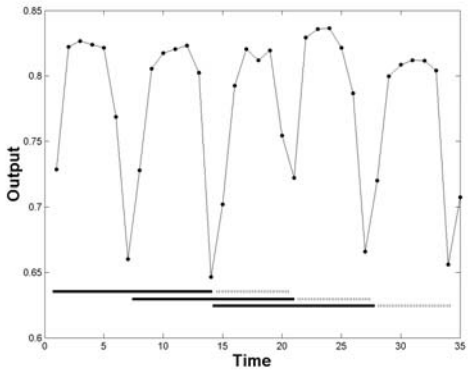

Figure 2.2 **: Manipulating regressors of blocks of data: predicting μ values at instant *t* (doted) using information of the *p* previous blocks of length μ (solid).**

4. Model structure selection

In section 3.1, we introduced the characterization of the double quantization method, which can be viewed as the learning stage of the algorithm. We have then said that a priori values for n_1 and n_2 have to be chosen.

Once we have chosen n_1 and n_2, we can apply the characterization, which results in a model of the time series. But, of course, we can choose other values for n_1 and n_2 and thus obtain another model. Since we can have many different models for the same time series, it becomes necessary to compare these models with respect to a criterion and select the best one as "the" idealized description of the time series.

4.1. Selecting the best model

As usually done in this context of model selection, we use a simple cross-validation technique. We divide the whole set of inputs at our disposal in two different subsets: the learning set, used for the characterization, and the validation set, used for the forecasting. Then we learn the n_1+n_2 prototypes on the learning set, and validate the resulting model during the forecasting stage of our method.

If the idea is quite simple, it is not really so simple in practice. Having no information about the possible values for n_1 and n_2, we have to test a relatively large range of values for n_1 and n_2, leading to a double loop in the model selection procedure and a final number of, at most, n_1 x n_2 models.

To compare the n_1 x n_2 models, we define a quadratic error criterion between the predicted scalar output $\hat{x}(t+1)$ given by the model and the real value $x(t+1)$ in the validation set :

$$\varepsilon_t = \left(x(t+1) - \hat{x}(t+1)\right)^2 . \qquad (4)$$

The validation error achieved by a specific model with n_1 prototypes in its original space and n_2 prototypes in its deformation space is thus the sum-of-squares error over the whole validation set V_{set}:

$$SSE_{n_1 n_2} = \sum_{V_{set}} \left(x(t+1) - \hat{x}(t+1)\right)^2 . \qquad (5)$$

This criterion is sufficient for comparing the models but has a limited intuitive interpretation. We therefore introduce an normalized error criterion:

$$NSSE_{n_1 n_2} = \frac{\sum_{V_{set}} \left(x(t+1) - \hat{x}(t+1)\right)^2}{\sum_{V_{set}} \left(x(t+1) - E[x(t+1)]\right)^2}, \qquad (6)$$

where E[.] denotes the mean and is estimated on the known values of the series. The interpretation of this criterion is more intuitive: if its value is near one, then we are considering a model that does not perform better than giving the mean as prediction, which indeed is a poor informative forecasting.

The comparison of the different tested models with different combinations of n_1 and n_2 can be summarized graphically in a plot of the error (*NSSE*) with respect to the values of n_1 and n_2 respectively.

4.2. Evaluating the performances of our method

In the experimental results shown below, we do not divide our input database in two subsets but in three: the learning set for the characterization, the validation set for the forecasting, as above, and we keep unused a third set, the test set, for an estimation of the generalization performances of our method. The aim is to observe how the best model can behave in real conditions.

The first step of our methodology is to choose the best model with the learning and validation sets, as detailed in section 4.1. Once we know the values for n_1 and n_2, we can perform a new learning on a new set of data, recomposed from the previous learning and validation sets. The assumption here is that the best model with n_1 and n_2 prototypes in each space is still the best for the new learning set, and that its performances are strengthened with the information obtained from the enhanced learning set. We then use the forecasting part of the method to forecast the long-term evolution of the time series.

In real applications of our method, these predictions are the final results. In our case, as we kept unused a third part of the time series, we can compare those final results with the real values, and then evaluate the generalisation performances of the method over the test set.

5. Experimental result

We have applied our method on an electrical consumption time series [5], testing the generalisation performances of the selected best model for this multi-dimensional example.

The whole dataset contains about 72 000 hourly data and is plotted in Fig 3. Due to the daily periodicity of the time series, we are interested in daily predictions and thus consider here blocks of μ=24 values, the time window becoming daily instead of hourly. This is an illustration of the multi-dimensional case described in section 3.3.

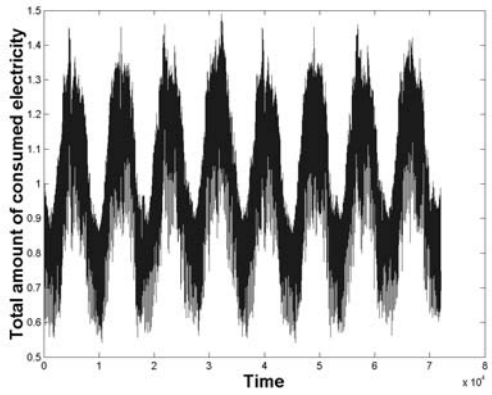

Figure 3 **: The electrical consumption time series.**

Having now at our disposal 3000 data, we use 2000 of them for the learning, 800 for the validation and 200 for the test. Note that each of these three sets contains 24-dimensional vectors.

We apply here many different regressors to the series, using our intuitive understanding of the process. As a result, we choose a final regressor with the 24 hourly values of today, the 24 values of yesterday, the 24 ones of two days ago, of six and of seven days ago. This regressor is maybe not the optimal one, but it is the one that makes the lowest error compared to other regressors we have tested. Since the regressor contains p = 5 data of dimension 24, we work in a 120-dimensional space.

We then run the algorithm again on the learning set with values for n_1 and n_2 varying each from 5 to 200 prototypes with an increment of 5. We thus evaluate the validation error for 1600 different models, and we obtain in Fig. 4 the graph of the *SSE* error. We choose the best n_1 and n_2 to be 160 and 140 respectively; the model results in a *NSSE* of 0.1369.

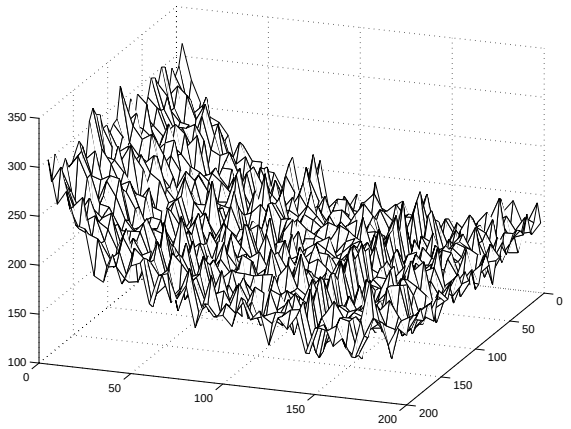

Figure 4 **: Sum-of-squares error versus the number of prototype in each original and deformation space.**

We then learn another model with 160 and 140 parameters vectors in each space with the new learning set, now containing 2000+800 data. The forecasting obtained from this model during the forecasting stage of our algorithm is repeated 1000 times. We show in Fig. 5 a comparison between the mean of those 1000 long-term predictions and the real values. A confidence interval at 95 % level is also provided. This interval covers what we call an envelope in which the time series will most probably stay at long term. A first zoom is provided for convenience (Fig. 5), and a second one (Fig. 6) draws the attention to the first three days of the forecast. Those forecasts are obtained while iterating the predictions from instant t until the final long-term horizon k. For example, from $\hat{x}_{t+1}$, $\hat{x}_{t+2}$ and $\hat{x}_{t+3}$, each of them in dimension 120, we extract blocks of 24 values to obtain the 72 first predicted hourly values for the first three days depicted in figure 6.

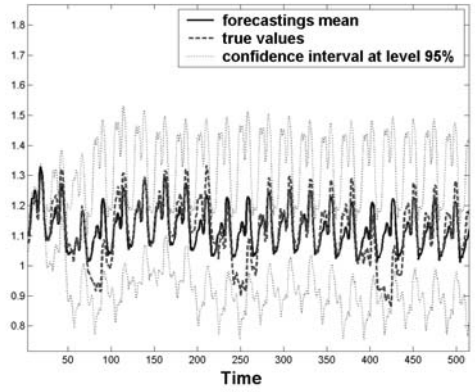

Figure 5 **: Comparison between the true values (dashed) and the mean of the predictions (solid), plus the envelope (confidence interval at 95 % level, doted).**

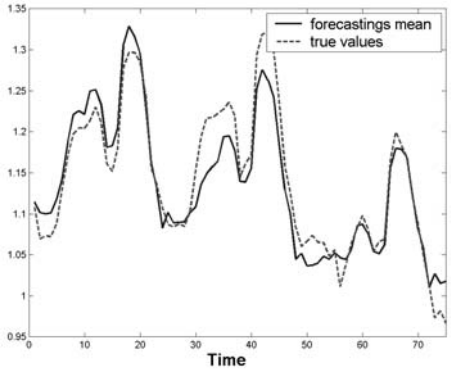

Figure 6 **: Comparison between the true values (dashed) and the predictions mean (solid) for the first 72 hours (3 days).**

Fig. 7 shows 3 predictions obtained by the Monte-Carlo procedure (before taking the mean). This figure tends to empirically prove the consistence of the method. Indeed different predictions seem to have the same shape; this is our main argument for determining long-term trends. Fig. 8 illustrates the robustness of the method: we take in purpose a model that is not the best one (with n_1 = 150 and n_2 = 150) and show the same comparisons as in Fig. 5. Though this model has a *NSSE* of 0.1858, we can see that the results are quite similar. For convenience, Fig. 9 presents a comparison of the forecasting mean between the best model and our second non-optimal choice for the first 72 hours.

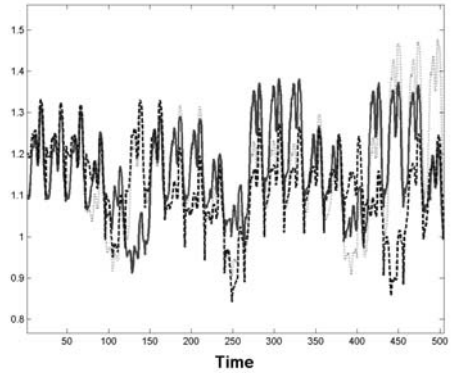

Figure 7 **: Plot of 3 simulations obtained by the generic Monte-Carlo procedure (before taking the mean).**

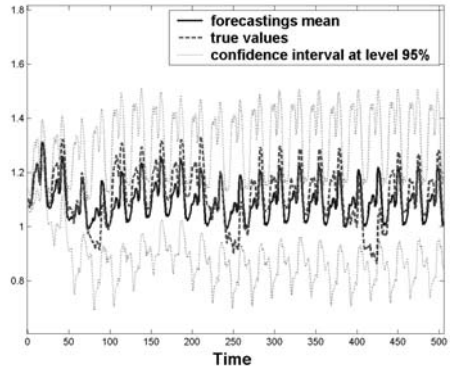

Figure 8 **: Comparison between the true values (dashed) and the predictions mean (solid) plus the envelope (confidence interval at level 95 %, doted) for a sub-optimal model with n_1 = 150 and n_2 = 150.**

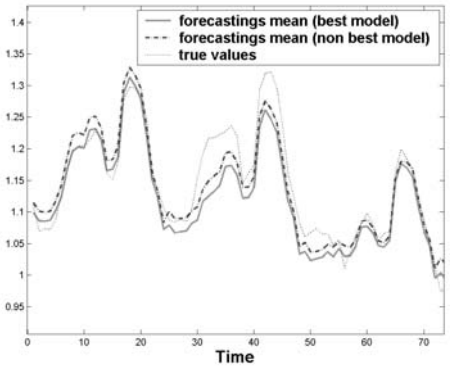

Figure 9 **: Forecastings mean comparison between the best model (solid) and another sub-optimal model (dashed) for the first 72 hours (3 days) with the real values (doted).**

6. Conclusion

In this paper, we have presented a time-series forecasting method based on a double classification of the regressors and of their differences (deformations), using the SOM algorithm. The use of SOMs makes it possible to apply the method both on one- and multi-dimensional time series, as discussed in section 3.3 and illustrated in section 5. A model selection procedure aimed to an automatic choice of the parameters (number of SOM classes) in the method has also been presented, in addition to some empirical proofs of the consistence and robustness of the method.

The proposed method is not designed to obtain an accurate forecast of the next values of a series, but rather aims to determine long-term trends.

Further work includes an additional Monte-Carlo procedure for the characterization stage, in order to avoid being trapped in a local minimum as a result of the SOM initialisation, and the use of more efficient validation procedures as (k-fold) cross-validation, leave-one-out or bootstrap.

7. Acknowledgements

We would like to thank Professor Osowsky from Warsaw Technical University for providing us the Polish Electrical Consumption data used in our example. G. Simon is funded by the Belgian F.R.I.A. M. Verleysen is Senior Research Associate of the Belgian F.N.R.S. The work of A. Lendasse is supported by the Interuniversity Attraction Poles (IAP), initiated by the Belgian Federal State, Ministry of Sciences, Technologies and Culture. The scientific responsibility rests with the authors.

8. References

[1] Kohonen T., Self-organising Maps, *Springer Series in Information Sciences*, Vol. 30, Springer, Berlin, 1995.

[2] Cottrell M., Fort J. C., Pagès G., Theoretical aspects of the SOM algorithm, *Neurocomputing*, 21, p. 119-138, 1998.

[3] Cottrell M., de Bodt E., Verleysen M., Kohonen maps versus vector quantization for data analysis, in *Proc of ESANN*, M. Verleysen Ed., D Facto, Brussels, 1997.

[4] Cottrell M., de Bodt E., Grégoire Ph., Simulating Interest Rate Structure Evolution on a Long Term Horizon: A Kohonen Map Application, *Proceedings of Neural Networks in The Capital Markets*, Californian Institute of Technology, World Scientific Ed., Pasadena, 1996.

[5] M.Cottrell, B.Girard, P.Rousset, Forecasting of curves using a Kohonen classification, Journal of Forecasting, 17, p. 429-439.

[6] Walter J., Ritter H., Schulten K., Non-linear prediction with self-organising maps, *Proc. of IJCNN*, San Diego, CA, 589-594, July 1990.

[7] Juha Vesanto, *Using the SOM and Local Models in Time-Series Prediction*, In *Proceedings of Workshop on Self-Organizing Maps (WSOM'97)*, Espoo, Finland, pp. 209-214, 1997.

[8] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski, "Recurrent SOM with Local Linear Models in Time Series Prediction", Proc. of ESANN'98, 6th European Symposium on Artificial Neural Networks, D-Facto, Brussels, Belgium, pp. 167-172, April 1998.

[9] Lendasse A., Verleysen M., de Bodt E., Cottrell M., Grégoire P., Forecasting Time-Series by Kohonen Classification, European Symposium on Artificial Neural Networks 1998, Bruges (Belgium), April 1998, pp221-226, D-Facto Publications (Brussels), ISBN 2-9600049-8-1.