xftsp: a Tool for Time Series Prediction by Means of Fuzzy Inference Systems

Federico Montesino, Amaury Lendasse and Ángel Barriga

***Abstract*— A new software tool for time series prediction by means of fuzzy inference systems is reported. This tool, named *xftsp*, implements a novel methodology for time series prediction based on methods for automatic fuzzy systems identification and supervised learning combined with statistical methods for nonparametric residual variance estimation. *xftsp* is designed as a tool integrated in the Xfuzzy development environment for fuzzy systems. Experiments carried out on a number of time series benchmarks show the advantages of *xftsp* in terms of both accuracy and computational requirements as compared against Least-Squared Support Vector Machines, an established technique in the field of time series prediction.**

***Index Terms*— Time series prediction, Fuzzy inference, Supervised learning, Nonparametric regression, Residual variance estimation, Least-squared support vector machines**

I. Introduction

In the past, conventional statistical techniques such as AR, ARMA and derived models have been extensively used for forecasting. However, these techniques have limited capabilities for modeling time series data, and more advanced nonlinear methods including neural networks, evolutionary algorithms and other soft computing techniques have been often applied with success [1].

Fuzzy logic based modeling techniques are appealing because of their interpretability and potential to address a broad spectrum of problems. The application of fuzzy inference systems to time series modeling and prediction dates back to [2], in which the authors develop the well known learn from examples identification algorithm for fuzzy inference systems and use the Mackey-Glass time series as a validation case. Nevertheless, despite its good performance in terms of accuracy and interpretability, fuzzy systems have seen little application in the field of time series prediction as compared to other nonlinear modeling techniques such as neural networks and support vector machines.

Recently, a methodology framework has been proposed for the long-term prediction of time-series by means of fuzzy systems [3]. This paper introduces a tool that implements the cited methodology in an open and modular way. Experimental results are compared against the reference implementation of a methodology commonly applied in the time series prediction field, least-squared support vector machines.

Federico Montesino Pouzols is with the Microelectronics Institute of Seville, CSIC, Scientific Research Council, Avda. Reina Mercedes s/n. Edif. CICA. E-41012 Seville, Spain (phone: +34-955-056-666; fax: +34-955-056-686; email: fedemp@imse.cnm.es).

Amaury Lendasse is with the Laboratory of Computer and Information Science of the Helsinky University of Technology. P.O. Box 5400, FIN-02015 HUT, Finland (phone: +358-9-451 3267; fax: +358-9-451 3277; email: lendasse@cis.hut.fi).

Angel Barriga Barros is with the Department of Electronics and Electromagnetism of the University of Seville, E-41012, Spain (phone: +34-955-056-666; fax: +34-955-056-686; email: barriga@us.es).

This work has been supported in part by project TEC2005-04359/MIC from the Spanish Ministry of Education and Science as well as project TIC2006-635 and grant IAC07-I-0205:33080 from the Andalusian regional Government.

II. Nonparametric Residual Variance Estimation: Delta Test

Nonparametric residual variance estimation (NRVE) is a well-known technique in statistics and machine learning, finding many applications in nonlinear modeling [4].

Delta Test (DT) is a NRVE method for estimating the lowest mean square error (MSE) that can be achieved by a model without overfitting the training set [4]. Given N multiple input-single output pairs, $(\bar{x}_i, y_i) \in R^M \times R$, the theory behind the DT method considers that the mapping between $\bar{x}_i$ and y_i is given by the following expression:

$$y_i = f(\bar{x}_i) + r_i,$$

where f is an unknown perfect fitting model and r_i is the noise. DT is based on hypothesis coming from the continuity of the regression function. When two inputs x and x' are close, the continuity of the regression function implies that outputs $f(x)$ and $f(x')$ will be close enough. When this implication does not hold, it is due to the influence of the noise.

Let us denote the first nearest neighbor of the point $\bar{x}_i$ in the set $\{\bar{x}_1, \ldots, \bar{x}_N\}$ by $\bar{x}_{NN}$. Then the DT, δ, is defined as follows:

$$\delta = \frac{1}{2N} \sum_{i=1}^{N} \left| y_{NN(i)} - y_i \right|^2,$$

where $y_{NN(i)}$ is the output corresponding to $\bar{x}_{NN(i)}$. For a proof of convergence, refer to [5]. DT has been shown to be a robust method for estimating the lowest possible mean squared error (MSE) of a nonlinear model without overfitting. DT is useful for evaluating nonlinear correlations between random variables, namely, input and output pairs. This method will be used for a priori input selection.

III. Prediction Methodology

Consider a discrete time series as a vector, $\bar{y} = y_1, y_2, \ldots, y_{t-1}, y_t$, that represents an ordered set of values, where t is the number of values in the series. The problem of predicting one future value, y_{t+1}, using an autoregressive model (autoregressor) with no exogenous inputs can be stated as follows:

$$\hat{y}_{t+1} = f_1(y_t, y_{t-1}, \ldots, y_{t-M+1})$$

Where $\hat{y}_{t+1}$ is the prediction of model f_1 and M is the number of inputs to the regressor.

Predicting the first unknown value requires building a model, f_1, that maps regressor inputs (known values) into regressor outputs (predictions). When a prediction horizon higher than 1 is considered, the unknown values can be predicted following two main strategies: recursive and direct prediction.

The recursive strategy is based on the application of the same model recursively, using predictions as known data to predict the next unknown values. It is the most simple and intuitive strategy and does not require any additional modeling after an autoregressor for 1 step ahead prediction is built. However, recursive prediction suffers from accumulation of errors. The longer the prediction term is, the more predictions are used as inputs. In particular, for prediction horizons greater than the regressor size, all inputs to the model are predictions.

Direct prediction requires that the process of building an autoregressor be applied for each unknown future value. Thus, for a maximum prediction horizon H, H direct models are built, one for each prediction horizon h:

$$\hat{y}_{t+h} = f_h(y_t, y_{t-1}, \ldots, y_{t-M+1}), \text{ with } 1 \leq h \leq H$$

While building a prediction system through direct prediction is more computationally intensive (as many times as values are to be predicted) it is also straightforward to parallelize. Direct prediction does not suffer from accummulation of prediction errors.

In this paper, the direct prediction strategy is followed. In order to build each autoregressor, a fuzzy inference system is defined as a mapping between a vector of crisp inputs, and a crisp output.

The problem of building a regressor can be precisely stated as that of defining a proper number and configuration of membership functions and building a fuzzy rulebase from a data set of t sample data from a time series such that the fuzzy systems $\mathcal{F}_h(\bar{y})$ closely predict the $h-$th next values of the time series. The error metric to be minimized is the mean squared error (MSE).

xftsp implements a methodology framework for long-term prediction of time series [3]. Within this framework, a fuzzy inference system is defined in an automatic manner for each prediction horizon. Figure 1 shows the stages required to define each autoregressor. These stages are detailed in the following subsections.

A. Variable Selection

As first step, DT estimates are used so as to perform an a priori selection of the optimal subset of inputs from the initial set of M inputs, given a maximum regressor size M. Variable selection requires a selection criterion. The result of the DT applied to a particular variable selection is used as as a measure of the goodness of the selection. The input selection that minimizes the DT estimate is chosen.

In addition, a selection procedure is required. For small (up to around 10-20) regressor sizes, an exhaustive DT evaluation for all the possible selections (a total of $2^M - 1$) is feasible.

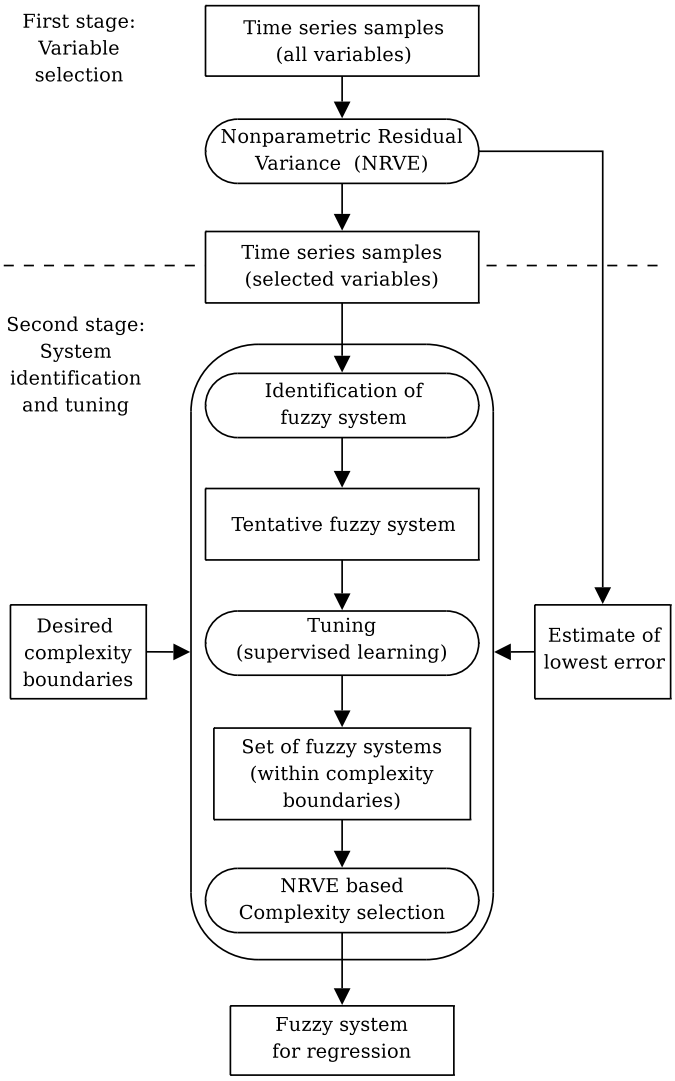

Fig. 1. Methodology Framework of *xftsp*.

We will call this procedure *exhaustive DT search*. Its main advantages is that the optimal selection is found.

For higher regressor sizes, forward-backward search of selections (FBS) [6] can be applied. This procedure combines both forward and backward selection. Although optimality is not guaranteed, a balance between performance and computational requirements is achieved.

B. System Identification and Tuning

This stage comprises two substages that are performed in a coordinated manner until a system that satisfies the error condition derived from the DT estimate is constructed.

1) Stage 2.1: System identification: In this substage, the structure of the inference system (linguistic labels and rule base) is defined. For identification, one or more parameters are usually required that specify the potential complexity of the inference system. Thus, the desired boundaries of complexity for the systems being built are additional inputs to the process.

2) Stage 2.2: System Tuning: We consider an additional tuning step in the methodology as a substage separated from the identification substage. Note that in some cases these two substages can be integrated into a standalone algorithm. The tuning process is driven by one or more error metrics.

C. Complexity selection

As last step, the complexity of the fuzzy autoregressors (measured as the number of linguistic labels per input in our concrete implementation) is selected depending on the DT estimate. The first (simplest) system that falls within the error range defined by the DT NRVE is selected.

The use of DT estimates in a first input selection stage as well as in the identification and tuning stage has been shown to be advantageous in two main aspects [3]:

- It does not only improve the regressor accuracy but also reduces its complexity and increases its interpretability by decreasing the number of inputs to the fuzzy inference system.
- It has been shown to be a robust solution to the problem of selecting the proper system complexity.

IV. *xftsp*

xftsp is a software tool that implements the methodology for time series prediction outlined above in a fully automatic manner. It is implemented in Java and defined as a java package integrated in the Xfuzzy environment for the design of fuzzy inference systems [7]. *xftsp* can be run whether within the Xfuzzy environment or as a standalone console tool. It is released under the same free license (GNU General Public License) as Xfuzzy and is available from the collaborative development web platform for Xfuzzy (https://forja.rediris.es/projects/xfuzzy).

The identification and tuning tasks are implemented using the API of the tools already available in Xfuzzy. Thus, the whole set of identification and tuning algorithms implemented in Xfuzzy [8] is available, as well as the overall framework for the definition of fuzzy sytems.

Figure 2 shows an scheme of the component architecture of Xfuzzy. A number of tools implement the following stages in the fuzzy inference systems development flow: description, tuning, verification and synthesis. The link among all these tools is the use of a common specification language, XFL3, and a common software component for the definition of fuzzy inference systems using XFL3. Within the description stage, Xfuzzy includes graphical tools for defining fuzzy systems. Tools for simulation, monitoring and graphical representation of the system behavior are included for the verification stage. The tuning stage involves identification, supervised learning and simplification tools. Finally, the synthesis stage includes tools generating high-level language descriptions for software and hardware implementations. Each tool can be executed whether as an independent program or as part of a global environment. The the whole set of tools are tied together under a graphical user interface.

Many advantages derive from the use of Xfuzzy as the basis of *xftsp*. Besides accelerating development, the data mining and supervised learning algorithms supported by

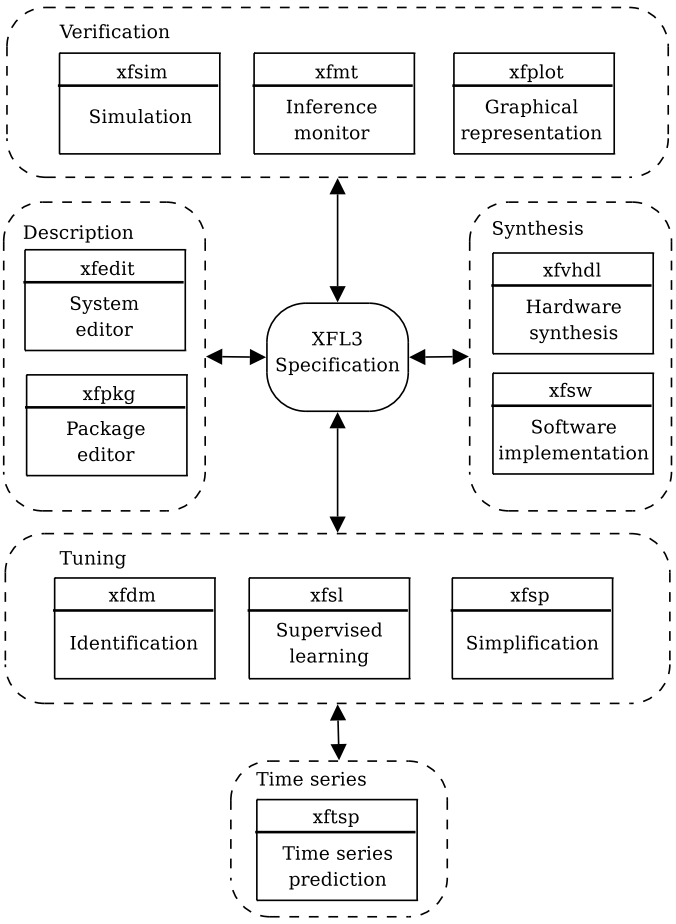

Fig. 2. Overall component architecture of Xfuzzy.

```
xftsp_series_name(sunspot)
xftsp_training_file(sunspot-training.txt)
xftsp_test_file(sunspot-test.txt)
xftsp_selection(DeltaTest, Exhaustive)
xftsp_identification_algorithm(WangMendel)
xftsp_learning_algorithm(Marquardt)
xftsp_option(MaxPredictionHorizon,50)
```

Fig. 3. Example *xftsp* configuration file.

Xfuzzy as well as extensions of these can be rapidly implemented within the time series prediction methodology framework of *xftsp*. Also, by leveraging on the Xfuzzy environment, many additional features are available. For instance, the simplification tool *xftsp* can be used on a system for time series prediction in order to get a simplified rulebase for better interpretability.

An execution of *xftsp* for a time series (defined by means of a training and an optional test file) is driven by a set of configuration options that can be set in a configuation file as well as through a visual interface. An example of configuration file is shown in figure 3.

In principle, any combination of membership functions, operators and inference model can be used for the purposes of time series prediction, but the selection has a significant impact on practical results. As default option, *xftsp* uses the minimum for conjunctions and implications, gaussian

membership functions for inputs, singleton outputs and fuzzy mean as defuzzification method following the Mamdani defuzzification model. These characteristics can be changed by means of configuration options. The default options were selected for the experiments described in the next section. In this particular case a fuzzy autoregressor with M inputs for prediction horizon h is formulated as:

$$\mathcal{F}_h(\bar{y}) = \frac{\sum_{l=1}^{N_h} \min\left(\mu_{R_l^h}, \min_{1\leq v\leq M} \mu_{L_l^{i,h}}(y_v)\right)}{\sum_{l=1}^{N_h} \min_{1\leq v\leq M} \mu_{L_l^{i,h}}(y_v)}$$

Where N_h is the number of rules in the rulebase for horizon h, $\mu_{L_l^{i,h}}$ are gaussian membership functions for the input linguistic labels and $\mu_{R_l^h}$ are singleton membership functions.

As default option, identification is performed using the algorithm by Wang and Mendel [2] (W&M) driven by the DT estimate. Though many modifications to the original algorithm have been proposed throughout the years, for the sake of simplicity we adhere to the original algorithm specification in [2], as implemented in version 3.2 of the Xfuzzy design environment [7].

In the case of the W&M algorithm, the number of labels per input must be specified a priori. The approach implemented in *xftsp* is to explore systems in an increasing order of complexity, from the lowest possible number of labels up to a maximum specified as complexity boundary. The same number of labels is used for each input.

This iterative identification process for increasing grid partitions of the universe of discourse stops when a system is built such that the training error is lower than the DT estimate or a threshold based on the DT estimate. The selection is made by comparing the error after the next (tuning) stage.

As default option for tuning, *xftsp* applies the Levenberg-Marquardt algorithm [9] for supervised learning driven by the normalized MSE (NMSE) as error metric[1]. All the parameters of the membership functions of every input and output are adjusted using the algorithm implementation in the Xfuzzy development environment [8], i.e., self-tuning inference systems are defined. The Levenberg-Marquardt algorithm is applied by default with the following parameter values: initial Hessian addition 0.1, increase factor 10.0 and decrease factor 0.2.

Finally, the task of long term time series prediction can be highly computationally intensive. Some critical components as for as performance have been identified. In particular, applying DT for input selection raises performance issues. The optimal exhaustive search algorithm has complexity $O(2^n)$, where n is the maximum number of inputs to the regressor. Besides the possibility of alleviating this problem through alternative search algorithms, such as forward-backward [3], an optimized C implementation is provided as a helping tool included with *xftsp* for speeding up the input selection process as far as possible.

[1]Normalization is performed against the square of the range of the series.

V. Application Examples

In this section, we show the results of applying *xftsp* to three time series: the Poland electricity benchmark, the monthly averaged sunspot number and the daily averaged aggregated traffic in the Abilene network backbone. We also compare the accuracy and computational requirements of fuzzy models against least-squared support vector machines (LS-SVM) [10] models with the same autoregressor size and input selection. We show the results from conducting a comparative assessment of *xftsp* and the LS-SVMlab1.5 Matlab/C toolbox. LS-SVM models were built following a direct prediction strategy for the same training subsets and variable selections. We selected RBF kernels, gridsearch as optimization routine and crossvalidation as cost function.

Though one of the major goals of the methodology implemented in *xftsp* is to avoid the requirement of validation and test series, we define two subsets in order to assess the residual noise estimator and algorithms being used.

In order to perform and approximate comparison of computing requirements, both *xftsp* and LS-SVMlab1.5 were executed on the same system configuration: a commodity PC running a distribution of the GNU/Linux operating system on an Intel(R) Core(TM)2 Duo CPU E655 processor at 2.33GHz, with 4 MB of L1 cache memory and 2 GB of RAM. No significant competing load was introduced.

xftsp was run on the Sun Java SE runtime environment version 1.6.0_04, build 1.6.0_04-b12, with the HotSpot virtual machine, build 10.0-b19 in mixed mode. LS-SVMlab was run on Matlab version R2007a, using the optimized C implementation of the LS-SVMlab1.5 toolbox, available from http://www.esat.kuleuven.ac.be/sista/lssvmlab/.

For all the tests, a maximum prediction horizon of 50 is considered, i.e., models are generated for predicting the next 50 unknown values.

A. Sunspot Numbers

The series of sunspot numbers is a periodic measure of the sunspot activity. Values from this series are subject to uncertainty and noise, particularly during the past centuries. We analyze a series of monthly averaged sunspot numbers covering from January 1749 to December 2007, as provided by the National Geographical Data Center from the US National Oceanic and Atmospheric Administration[2]. The series is split into a set of 1000 values for training and a set of 2908 values for testing. The whole series is shown in figure 4.

The training and test errors of LS-SVM models averaged throughout horizons 1 to 50 are shown together with the errors of fuzzy models in table I. Two maximum regressor sizes are shown in the table. Fuzzy autoregressors achieve a higher approximation accuracy for the test subset for all the prediction horizons considered.

[2]The series used here can be obtained from http://www.ngdc.noaa.gov/stp/SOLAR/ftpsunspotnumber.html. The International Sunspot Number is produced by the Solar Influence Data Analysis Center (SIDC) at the Royal Obervatory of Belgium [11].

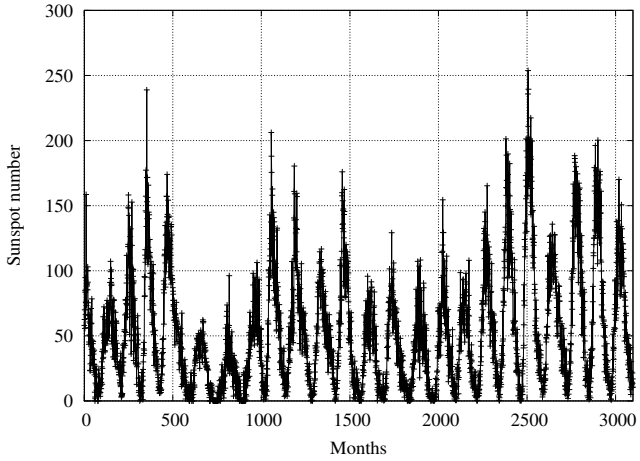

Fig. 4. Sunspot: training (first 1000 samples) and test (last 2098 samples) series.

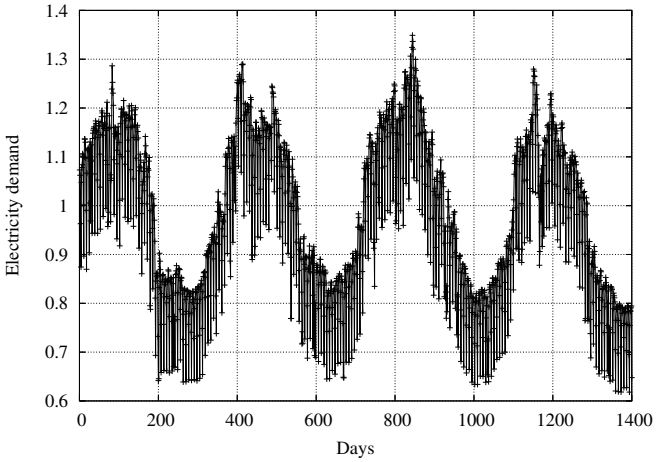

Fig. 5. PolElec: training series (1400 samples).

TABLE I

TRAINING AND TEST ERRORS OF FUZZY INFERENCE MODELS AND LS-SVM, AVERAGED FOR HORIZONS 1 THROUGH 50. ALL ERRORS ARE GIVEN AS NMSE. MAXIMUM REGRESSOR SIZE SPECIFIED BETWEEN PARENTHESIS.

	xftsp		LS-SVMlab1.5	
Series	Training	Test	Training	Test
Sunspot (9)	$1.69{\cdot}10^{-2}$	$2.64{\cdot}10^{-2}$	$1.34{\cdot}10^{-2}$	$3.28{\cdot}10^{-2}$
Sunspot (12)	$1.59{\cdot}10^{-2}$	$2.63{\cdot}10^{-2}$	$9.64{\cdot}10^{-3}$	$3.02{\cdot}10^{-2}$
PolElec (7)	$1.70{\cdot}10^{-2}$	$1.78{\cdot}10^{-2}$	$1.16{\cdot}10^{-2}$	$3.57{\cdot}10^{-2}$
PolElec (14)	$1.58{\cdot}10^{-2}$	$1.82{\cdot}10^{-2}$	$1.04{\cdot}10^{-2}$	$3.24{\cdot}10^{-2}$
AbileneI (7)	$1.44{\cdot}10^{-2}$	$1.73{\cdot}10^{-2}$	$8.58{\cdot}10^{-3}$	$2.47{\cdot}10^{-2}$
AbileneI (12)	$1.22{\cdot}10^{-2}$	$1.50{\cdot}10^{-2}$	$6.77{\cdot}10^{-3}$	$2.15{\cdot}10^{-2}$

TABLE II

RUNNING TIME (IN SECONDS) REQUIRED TO BUILD TIME SERIES MODELS FOR HORIZONS 1-50. ALL TESTS WERE RUN ON THE SAME SYSTEM, WITH NO SIGNIFICANT COMPETING LOAD. MAXIMUM REGRESSOR SIZE SPECIFIED BETWEEN PARENTHESIS.

Series	*xftsp*	LS-SVMlab1.5
Sunspot (9)	$1.04 \cdot 10^4$	$3.10 \cdot 10^5$
Sunspot (12)	$1.22 \cdot 10^4$	$2.42 \cdot 10^5$
PolElec (7)	$1.05 \cdot 10^4$	$3.04 \cdot 10^5$
PolElec (14)	$2.30 \cdot 10^4$	$9.91 \cdot 10^5$
AbileneI (7)	$1.75 \cdot 10^3$	$1.40 \cdot 10^5$
AbileneI (12)	$4.69 \cdot 10^3$	$1.27 \cdot 10^5$

Table II shows the running times for *xftsp* and LS-SVMlab. As can be seen, *xftsp* is at least 1 order of magnitude faster than LS-SVMlab1.5. Most of the time required by *xftsp* to complete a model is spent in the supervised learning sub-stage. The maximum duration of this stage can be bounded by setting two complementary configuration options: the maximum number of iterations of the learning algorithm to perform and the training error decrease beyond which no more iterations are performed.

B. Poland Electricity

This time series (PolElec henceforward) represents the normalized average daily electricity demand in Poland in the 1990´s. The benchmark consists of a training set of 1400 samples, shown in figure 5, and a test set of 201 samples, shown in figure 6. It has been shown that the dynamics of this time series is nearly linear [12]. Besides the yearly periodicity, a clear weekly periodicity can be seen on smaller time scales (see figure 6). Accuracy and timing results are also summarized in tables I and II.

C. Aggregated Incoming Traffic in the Abilene Network

This series, AbileneI henceforward, represents the total amount of aggregated incoming traffic in the routers of the Internet2 backbone network during several years. The AbileneI series consists of 1458 daily averages (in bps) covering from the 4th of January of 2003 to the 31st of December of 2006. The data are available from the Abilene Observatory at http://www.internet2.edu/observatory/. The daily averages for years 2003 and 2004 (the first 728 values) were selected as training set, whereas the daily averages for years 2005 and 2006 (the last 730 values) were selected as test set. Accuracy and timing results are summarized in tables I and II as well.

VI. DISCUSSION

The methodology followed in this paper and its particular implementation in the *xftsp* tool have been experimentally shown to perform well for long-term time series prediction. *xftsp* does not require a validation stage and thus the whole available data set can be used as input training data.

In addition to the interpretability of the methodology implemented in *xftsp*, fuzzy inference based models have been shown to consistently outperform LS-SVM models in terms of accuracy. For the kind of time series considered in this paper, noisy time series for which there are no deterministic models available, fuzzy models are consistently around 3 times more accurate than LS-SVM models.

A remarkable property of the models generated by *xftsp* is their generalization capability. Test errors have been found to be of the same order of magnitude than training errors and are usually very close. While LS-SVM are usually praised for their good generalization performance, it can be concluded

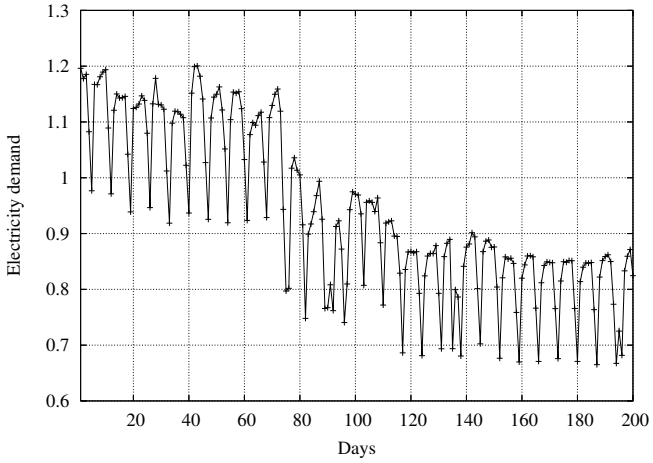

Fig. 6. PolElec: test series (201 samples).

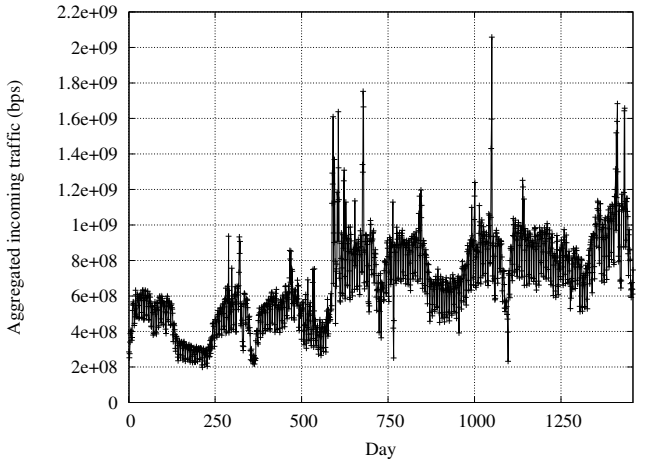

Fig. 7. AbileneI: daily averaged aggregated incoming traffic in the Abilene backbone for 1458 days. Training series (first 728 values) and test series (last 730 values).

from table II that fuzzy autoregressors developed with *xftsp* clearly outperform LS-SVM based autoregressors in terms of generalization capability.

An important contribution of *xftsp* is the interpretability of the models it generates. As an example, let us consider the model with maximum regressor size 7 for 7 steps ahead prediction of the AbileneI series. Three inputs are selected to predict y_{t+7}: y_t, y_{t-2} and y_{t-4}. Let us suppose that the last 7 daily average traffic measurements that are available correspond to the traffic for a week from Monday through Sunday. Then, the fuzzy autoregressor predicts the average traffic for next Sunday based on the averages from last Sunday, Friday and Wednesday. Two linguistic terms are defined for each input variable. The rulebase of this predictor consists of 8 rules. A sample rule from this system would read as follows:

IF Wednesday was High AND Friday was Low AND
Sunday was High THEN NextSunday ← "915Mbps"

where the variables "Wednesday", "Friday", "Sunday" and "NextSunday" refer to daily traffic averages, "Low" and "High" are the two linguistic terms defined for the inputs, and "915Mbps" is used as linguistic label for a singleton output centered approximately at the 915 Mbps value.

In most cases, the most accurate system has a low number of linguistic terms and rules (below 15 or 10). However, in some cases the number of rules can be of a few tens. In general, it can be concluded that systems with the minimum number of linguistic terms provide a reasonable approximation to the most accurate system. Thus, it is easy to obtain simple approximate models that ease the understanding of the time series dynamics.

Finally, alternative options for fuzzy inference systems identification, tuning and simplification have been proposed to date, and the ones used in this paper could be improved as well. This is an area of future research in identification and tuning techniques for time series prediction.

VII. Conclusion

The architecture and usage of a new tool for time series prediction by means of fuzzy inference systems, *xfstp*, has been described. The tool has been shown to perform well and to clearly outperform the well-established LS-SVMlab matlab toolbox in terms of both accuracy and speed. By having a modular architecture, *xftsp* enables further research on the application of fuzzy systems identification and tuning techniques for time series modeling and prediction.

References

[1] C. Chatfield, *The Analysis of Time Series. An Introduction*. CRC Press, Jul. 2003, sixth edition, ISBN: 1-58488-317-0.

[2] L. Wang and J. M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 4, pp. 1414–1427, Dec. 1992.

[3] F. M. Pouzols, A. Lendasse, and A. Barriga, "Fuzzy Inference Based Autoregressors for Time Series Prediction Using Nonparametric Residual Variance Estimation," in *17th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'08), IEEE World Congress on Computational Intelligence*, Hong Kong, China, Jun. 2008.

[4] A. J. Jones, "New Tools in Non-linear Modelling and Prediction," *Computational Management Science*, pp. 109–149, Sep. 2004.

[5] E. Liitiäinen, A. Lendasse, and F. Corona, "Non-parametric Residual Variance Estimation in Supervised Learning," in *WANN 2007, International Work-Conference on Artificial Neural Networks*, San Sebastián, Spain, Jun. 2007, pp. 63–71.

[6] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, and A. Lendasse, "Methodology for Long-Term Prediction of Time Series," *Neurocomputing*, vol. 70, no. 16-18, pp. 2861–2869, Oct. 2007.

[7] F. J. Moreno-Velo, I. Baturone, S. Sánchez-Solano, and A. Barriga, "Rapid Design of Fuzzy Systems With Xfuzzy," in *12th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'03)*, St. Louis, MO, USA, May 2003, pp. 342–347.

[8] F. J. Moreno-Velo, I. Baturone, A. Barriga, and S. Sánchez-Solano, "Automatic Tuning of Complex Fuzzy Systems with Xfuzzy," *Fuzzy Sets and Systems*, vol. 158, no. 18, pp. 2026–2038, Sep. 2007.

[9] R. Battiti, "First and Second Order Methods for Learning: Between Steepest Descent and Newton's Method," *Neural Computation*, vol. 4, no. 2, pp. 141–166, Mar. 1992.

[10] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002, ISBN: 981-238-151-1.

[11] R. V. der Linden and the SIDC Team, "Online Catalogue of the Sunspot Index," RWC Belgium, World Data Center for the Sunspot Index, Royal Observatory of Belgium, years 1748-2007, http://sidc.oma.be/html/sunspot.html, Jan. 2008.

[12] A. Lendasse, J. Lee, V. Wertz, and M. Verleyssen, "Forecasting Electricity Consumption using Nonlinear Projection and Self-Organizing Maps," *Neurocomputing*, vol. 48, no. 1, pp. 299–311, Oct. 2002.