Forecasting time series combining machine learning and Box-Jenkins time series

Montañés E[1]., Quevedo J.R.[1], Prieto M.M.[2], Menéndez C.O.[3]

[1] Artificial Intelligent Center. Viesques E-33271. Gijón
[2] Energy Department. Viesques E-22271. Gijón
[3] Mathematics Department. Llamaquique. E-33007. Oviedo
Asturias (Spain)
[1]{elena,quevedo}@aic.uniovi.es
[2] manuelap@correo.uniovi.es
[3] omar@orion.ciencias.uniovi.es

Abstract. In statistical field Box-Jenkins Time Series is a linear method widely used to forecasting. The linearity makes the method inadequate to forecast real time series, which could present irregular behavior. On the other hand, in artificial intelligent field FeedForward Artificial Neural Networks and Continuous Machine Learning Systems are robust handlers of data in the sense that they are able to reproduce nonlinear relationships. Their main disadvantage is the selection of adequate inputs or attributes better related with the output or category. In this paper we present a methodology that employs Box-Jenkins Time Series as feature selector to Feedforward Artificial Neural Networks inputs and Continuous Machine Learning Systems attributes. We also apply this methodology to forecast some real time series collected in a power plant. It is shown than Feedforward Artificial Neural Networks performs better than Continuous Machine Learning Systems, which in turn performs better than Box-Jenkins Time Series.

1. Introduction

Time series are widely analyzed and forecasted by means of Box-Jenkins Time Series (*BJTS*). This method considers that the time series has been generated by a stochastic process and all techniques which obtain them are conducted to identify this generator. Then, the model is estimated and verified and finally once it has been accepted it is applied to forecast future values of the time series. The model identifies some characteristics of the time series, as the trend and the seasonality and gives an expression which relates the actual value of the time series data with its historical ones which are more relevant. The main disadvantage is that these relations are linear which often conducts to inadequate predictions when treating with real world time series. They also are not able to explain sudden changes in the time series.

Alternatively, the use of artificial intelligent techniques, like Feedforward Artificial Neural Networks (*FANN*) and Continuous Machine Learning Systems (*CMLS*), to forecast time series results promising due to its capacity to handle amounts of data and to reproduce nonlinear relations between inputs and output in *FANN* and between

attributes and category in *CMLS*. They are also capable to predict sudden variations. The main difficult task is to find out which inputs or attributes better define the output or category.

Taking advantage of the best properties of *BJTS* on one hand and of *FANN* and *CMLS* on the other we propose a methodology which combines them in the way that *BJTS* are used to identify adequate inputs or attributes for *FANN* and for *CMLS* respectively. We apply this methodology to several real physical time series collected in a power plant.

The remainder of this paper is as follows: In section 2 *BJTS*, *FANN* and *CMLS* are briefly detailed, in section 3 the combination of the latter methods are explained, the application of the methodology and the discussion of the results is made in section 4 and finally the conclusions and future work are exposed in section 5.

2. Box-Jenkins Time Series, Neural Networks and Machine Learning Systems

In this section *BJTS*, *FANN* and some *CMLS* and their combination are briefly detailed.

2.1. Box-Jenkins Time Series

The most general *BJTS* are Seasonal Autoregressive Integrated Moving Average (*SARIMA*(p^*,d^*,q^*,P^*,D^*,Q^*,s^*)), which assume that the mean is zero. The expression of a *SARIMA* time series is given by eq.(1):

$$f_{p^*}(B)F_{P^*}(B^{s^*})(1-B^{s^*})^{D^*}(1-B)^{d^*}y_t = q_{q^*}(B)Q_{Q^*}(B^{s^*})a_t \quad (1)$$

where f_{p^*}, Φ_{P^*}, q_{q^*} and Θ_{Q^*} are the autoregressive p^*-order, the seasonal autoregressive P^*-order, the moving average q^*-order and the seasonal moving average Q^*-order operators, B is the backward shift operator, y_t is the original time series and a_t is white noise time series.

The seasonality s^* is taken by means of the periodogram as the frequency whose amplitude is significantly greater. The parameters d^* and D^* are simultaneously varied until the trend and the seasonality are removed. Then, a new time series called differenced time series x_t is obtained and given by eq.(2).

$$x_t = (1-B^{s^*})^{D^*}(1-B)^{d^*}y_t \quad (2)$$

The autocorrelation and the partial autocorrelation functions (*ACF* and *PACF*) are used to choose p^* and q^* (for the differenced time series) and P^* and Q^* (for the differenced time series taking the values corresponding to s^*-multiples indexes) [2].

The autoregressive, seasonal autoregressive, moving average and seasonal moving average operators are given by eqs. (3)-(6):

$$f_{p^*}(B)x_t = x_t + b_{1,p}x_{t-1} + ... + b_{p,p}x_{t-p} = w_t \quad (3)$$

$$\Phi_{P^*}(B^{s^*})w_t = w_t + b_{1,P}w_{t-s} + ... + b_{P,P}w_{t-P\cdot s} = z_t \quad (4)$$

$$\Theta_{Q^*}(B^{s^*})z_t = z_t + b_{1,Q}z_{t-s} + ... + b_{Q,Q}z_{t-Q\cdot s} = r_t \quad (5)$$

$$q_{q^*}(B)r_t = r_t + b_{1,q}r_{t-1} + ... + b_{q,q}r_{t-q} \quad (6)$$

The coefficients b_{ij} are estimated based on the methodology shown in [3] and [13].

The *ACF* and *PACF* of the residual time series are used to validate the model to check that the residual time series can be regarded as a white noise time series [19].

2.2. Neural Networks

The most common neural networks are *FANN*, which have the neurons structured in layers. The neurons of a layer can only be connected to neurons of the following layer but not to neurons of the same layer. The incorporation of hidden layers and a transfer function furnishes these neural networks with enough flexibility to solve difficult problems, thus reproducing the nonlinear dependence of the variables.

The logistic function was taken as node function, because it is bounded, monotonous increasing and differentiable and these properties assure the convergence of the weight estimation method [8]. One hidden layer was also chosen since the data partially defines a continuous function and one hidden layer is sufficient to approximate this kind of function [6]. The number of neurons in the hidden layer was obtained by iteratively constructing a sequence of *FANNs* $S_d \subset S_{d+1} \subset ... \subset S_{d+m}$, where d is the number of inputs and m is the maximum number of neurons in the hidden layer, which was fixed at 10 and S_i is the set of *FANNs* with a hidden layer and i neurons in this layer. The mean squared error adding and without adding an additional term, which improves the forecasting error, was taken as the error function. The addition of this term is called regularization [15]. The mean of the squared weights was chosen as the additional term. The expression of the resulting error function is given by eq.(7).

$$E\left(w_1,...,w_{N_{pe}}\right) = \frac{1}{2}\sum_{j=1}^{N_{de}}\left[d_j - o_j\right] + \frac{1}{N_{pe}}\sum_{j=1}^{N_{pe}} w_j^2 \quad (7)$$

where w_j are the weights to be estimated, N_{de} is the number of data used to obtain the model, d_j is the desired neural network output, o_j is the actual neural network output and N_{pe} is the number of weights

The method chosen for the weight estimation was the conjugate gradient method [4], [10], [16] and [11] with the Fletcher-Reeves update [9] combined with Charalambous' search [5] for unidimensional minimization. In the conjugate gradient

algorithms a search is performed along conjugate directions, which produces faster convergence than the basic steepest descent algorithm. These methods require only a little more storage than the simpler algorithms, so they are often a good choice for networks with a large number of weights. An iteration of the method is shown in eq.(8) and eq.(9):

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + a_k \boldsymbol{p}_k \tag{8}$$

$$\boldsymbol{p}_k = -\boldsymbol{g}_k + ß_k \boldsymbol{p}_{k-1} \tag{9}$$

where $\boldsymbol{w}_k$ is the vector $(w_1,\ldots,w_{N_{pe}})$ in the k[th] -iteration, $\boldsymbol{p}_k$ is the conjugate gradient direction in the k[th]-iteration, $\boldsymbol{g}_k$ is the gradient direction in the k[th] -iteration and a_k is the optimal size step after linear minimization in the k[th]-iteration.

The various versions of conjugate gradient are distinguished by manner in which the constant $ß_k$ is computed. For the Fletcher-Reeves update, the procedure is given by equation (10):

$$ß_k = \frac{\boldsymbol{g}_k^{\boldsymbol{T}} \boldsymbol{g}_k}{\boldsymbol{g}_{k-1}^{\boldsymbol{T}} \boldsymbol{g}_{k-1}} \tag{10}$$

This is the ratio of the squared of the norm of the current gradient to the squared of the norm of the previous gradient. The storage requirements of this update are lower than others.

The validation model is carried out by taking some data that is not used to obtain the model and checking that the error committed in the forecasting is lower than a given value.

2.3. Continuous Machine Learning Systems

CMLS take a set of examples and learn a set of regression rules or a regression trees from them. The examples are described as a sequence of pairs attribute-value and a category.

These techniques are widely applied to different areas with excellent results; an example can be found in [12]. The versatility, robustness and accuracy of these systems make them adequate to be applied to forecasting.

One of the most important problems of applying this kind of systems is to select a set of informative attributes [1]. Although it is supposed that they are able to deal with irrelevant attributes if its number is high it is recommended to carry out an attribute selection.

The *CMLS* employed in this paper are *m5'* [20], *Cubist* [17] and *RT* [18]. *m5'* is "a rational reconstruction of *m5"*, which produces regression trees. *Cubist* is a commercial system that produces regression rules. *RT* is a learning system that can handle multiple regression problems based on samples of data and the resulting models have the form of a regression tree.

3. Combination of Box-Jenkins Time Series, Feedforward Artificial Neural Networks and Continuous Machine Learning Systems

In this section it is explained the methodology that combines *BJTS*, *FANN* and *CMLS*.

As it was previously exposed the most general *BJTS* is *SARIMA*, whose parameters $p^*,d^*,q^*,P^*,D^*,Q^*,s^*$. These parameters are employed to identify the inputs or attributes of *FANN* and *CMLS*. Firstly of all the meaning of these parameters is briefly detailed:

- The parameter p^* is the relevant delays of the time series respect to the actual instant.
- The parameter d^* is the order of differentiation which allows remove the trend of the time series.
- The parameter q^* is the relevant delays of the white noise time series respect to the actual instant (this time series appear in the model of *BJTS*)
- The parameter s^* is the seasonality.
- The parameter P^* is the relevant delays with s^*-multiples indexes of the time series respect to actual instant.
- The parameter D^* is the order of differentiation which allows remove the seasonality of the time series.
- The parameter Q^* is the relevant delays with s^*-multiples indexes of the white noise time series respect to the actual instant (this time series appear in the model of *BJTS*)

In most time series P^* and Q^* are rarely positive [3]. Besides, the parameters associated to the white noise are not taken into account due to *FANN* and *CMLS* already deal with noise. On the other hand, in Neural Networks and Machine Learning it is accepted the fact that it is desirable that the distribution of the train and the test data be the same. This occurs only when the time series does not have trend, although it has seasonality. This is the reason why D^* is not necessary to be considered. Then, our methodology only takes into account p^*, d^* and s^*.

First of all it is necessary to remove the trend, so the time series is differentiated according to eq. (11).

$$x_t = (1-B)^{d^*} y_t \tag{11}$$

Then, p^* delays respect to the actual instant and p^* delays respect to the seasonality s^* are considered as inputs or attributes of the *FANN* and *CMLS* as expressed in eq. (12).

$$(t\text{-}c_{p^*},\ t\text{-}\ c_{p^*-1},\ldots,c_1,\ t\text{-}c_{s^*+p^*-1},\ t\text{-}\ c_{s^*+p^*-2},\ldots,c_{s^*}) \tag{12}$$

4. Model Application and Results

Pressures, temperatures and mass flow rates simultaneously registered every 10 minutes from a seawater refrigerated power plant condenser subject to tide effects.

between 8-6-98 and 14-6-98 (1008 data) are used to evaluate the model. The last two days (288 data) were removed from the *test*.

Model implementation was carried out using the *Matlab 5.3 NAG Foundation Toolbox* [14] and *Neural Network Toolbox* [7] and *Cubist* [17], *M5'* [20] and *RT* [18].

Table 1 shows the *BJTS* and the inputs or attributes of *FANN* and *CMLS*.

Table 1. *BJTS* and the inputs or attributes of *FANN* and *CMLS*. The value c_i in $(c_1, c_2,..., c_m)$ is the number of delays with respect to the actual instant t

	Time Serie $(p^*,d^*,q^*,P^*,D^*,Q^*,s^*)$	*FANN and CMLS* $(c_1, c_2,..., c_m)$
M_c	*(1,0,0,0,1,0,144)*	*(1,144)*
T_{d1}	*(1,0,0,0,1,0,144)*	*(1,144)*
p_{tes1}	*(1,0,0,0,1,0,144)*	*(1,144)*
p_{stlp}	*(2,0,0,0,1,0,144)*	*(1,2,144,145)*
T_{stlp}	*(1,0,0,0,1,0,144)*	*(1,144)*
p_{cp}	*(1,0,0,0,1,0,144)*	*(1,144)*
T_{hot}	*(1,0,0,0,1,0,144)*	*(1,144)*
Dp	*(1,0,0,0,1,0,144)*	*(1,144)*
L_{ch}	*(2,0,0,0,1,0,144)*	*(1,2,144,145)*
$T_{w,i}$	*(2,0,0,0,0,0,0)*	*(1,2)*
$T_{w,o}$	*(1,0,0,0,1,0,144)*	*(1,144)*

In the *BJTS*, the seasonality obtained from the periodogram was a day (s^* =144). The seasonality is removed by differencing the original time series (D^*=1). However, it was not necessary to remove the trend ($d^*=0$). The moving averaged components are zero ($q^*=0$ and $Q^*=0$). Only nonseasonal, autoregressive components were identified ($p^*>0$ and $P^*=0$).

The Medium Average Deviation (*MAD*) is a common measure of the performance, but it is adequate to use the Relative Medium Average Deviation (*RMAD*) which is the *MAD* divided by the *MAD* of the system that always predicts the average function (*Av. MAD*). This measure removes the effect of the dimension of the variables. In Table 2 it is shown the *MAD* of the system that always predicts the average function for each time series.

Table 2. *MAD* of the system that always predicts the average function

	Mc	T_{d1}	p_{tes1}	p_{stlp}	T_{stlp}	p_{cp}	T_{hot}	Dp	L_{ch}	$T_{w,i}$	$T_{w,o}$
MAD	47.22	3.20	0.03	0.40	2.30	6.97	2.39	61.82	0.79	0.25	1.91

In Table 3 it is shown the *RMAD* forecasting errors for *BJTS*, *FANN* without and with term regularization and some *CMLS* when the real values of the delays are used to forecast the next value of the time series (*TO 1*). It is also shown the best (*BEST*) *RMAD* for each time series.

Table 3. *RMAD* for *BJTS*, *FANN* whitout and with term regularization, some *CMLS* (*Cubist, M5'* and *RT*) and the best *RMAD* when *TO 1*

	BJTS	*FANN*		*MLS*			*BETTER*
		-Reg	*+Reg*	*Cubist*	*M5'*	*RT*	
Mc	13.18%	10.89%	14.72%	10.99%	10.67%	16.79%	10.67%
T_{d1}	10.52%	9.31%	12.03%	9.39%	9.55%	15.31%	9.31%
p_{tes1}	11.39%	8.71%	9.38%	7.68%	7.86%	12.61%	7.68%
p_{stlp}	9.70%	7.03%	14.01%	7.17%	7.11%	15.13%	7.03%
T_{stlp}	51.89%	31.21%	30.65%	29.92%	31.167%	35.60%	29.92%
p_{cp}	13.65%	12.35%	14.26%	12.41%	12.32%	18.59%	12.32%
T_{hot}	12.29%	11.30%	12.06%	11.13%	11.14%	16.97%	11.13%
Dp	22.43%	17.41%	19.04%	17.61%	17.84%	22.96%	17.41%
L_{ch}	20.87%	16.78%	22.05%	15.80%	16.66%	20.46%	15.80%
$T_{w,i}$	12.31%	12.31%	13.49%	16.57%	16.62%	23.99%	12.31%
$T_{w,o}$	14.02%	12.45%	15.08%	12.84%	12.73%	18.28%	12.45%
Av.	17.48%	13.61%	16.07%	13.77%	13.97%	19.70%	13.27%

In Table 4 it is shown the *RMAD* forecasting errors for *BJTS*, *FANN* without and with term regularization and some *CMLS* when the forecasted values of the delays are used to forecast the next value of the time series (*TO N*). It is also shown the best (*BEST*) *RMAD* for each time series.

Table 4. *RMAD* for *BJTS*, *FANN* whitout and with term regularization, some *CMLS* (*Cubist, M5'* and *RT*) and the best *RMAD* when *TO N*

	BJTS	*FANN*		*MLS*			*BETTER*
		-Reg	*+Reg*	*Cubist*	*M5'*	*RT*	
Mc	60.13%	90.58%	53.16%	86.07%	119.74%	117.30%	53.16%
T_{d1}	58.99%	97.97%	53.90%	133.03%	121.88%	150.34%	53.90%
p_{tes1}	62.66%	89.13%	64.67%	129.63%	116.96%	121.63%	62.66%
p_{stlp}	70.19%	57.50%	68.89%	111.89%	66.96%	122.42%	57.50%
T_{stlp}	111.74%	97.84%	96.30%	99.95%	123.15%	235.94%	96.30%
p_{cp}	56.75%	79.77%	54.11%	117.16%	94.21%	126.79%	54.11%
T_{hot}	56.36%	93.05%	53.85%	124.96%	119.29%	129.08%	53.85%
Dp	88.55%	90.14%	87.24%	99.33%	94.77%	114.56%	87.24%
L_{ch}	84.00%	73.58%	79.92%	97.73%	90.86%	152.25%	73.58%
$T_{w,i}$	86.62%	48.63%	47.90%	138.96%	98.75%	106.38%	47.90%
$T_{w,o}$	63.82%	94.77%	57.86%	92.60%	99.04%	137.10%	57.86%
Av.	72.71%	83.00%	65.25%	111.94%	104.15%	137.62%	63.46%

Looking at results in Table 3 it is noticed that *FANN* without regularization term outperforms *BJTS* for all time series. However *FANN* with regularization and *BJTS*

are similar. *Cubist* and *M5'* do the same except for $T_{w,i}$, although the difference is insignificant. *Cubist* and *M5'* are also better than *FANN* with regularization. The performance of *FANN* without regularization, *Cubist* and *M5'* are similar. However, the results of *RT* are worse even than *BJTS*.

Looking at results in Table 4 it is noticed that *FANN* with regularization term outperforms *BJTS* and *CMLS* for all time series. However *CMLS* do not have learned well given that most of the *RMAD* are above the 100%, that is the *MAD* is worse than the *MAD* of the function that predicts the average. *FANN* without regularization are worse than *BJTS* and *FANN* with regularization.

5. Conclusions

This paper describes the employment of the widely used forecasting technique *BJTS* to identify the inputs of *FANN* or the attributes of *CMLS*. This methodology is then applied to some time series collected in a seawater refrigerated power plant condenser subject to tide effects.

The use of *BJTS* as a feature selector to *FANN* and *CMLS* results promising since these latter techniques reaches significantly more accurate than *BJTS*. *Cubist* (a *CMLS*) performs the best although *M5'* (another *CMLS*) and *FANN* also give good results.

FANN without regularization, *Cubist* and *M5'* outperforms *BJTS* when the real values of the delays are used to forecast the next value of the time series. However, *FANN* with regularization is the only system that outperforms BJTS when the forecasted values of the delays are used to forecast the next value of the time series. In this latter case *CMLS* do not learn correctly.

As future work we are interested in exploiting the power of *FANN* and *CMLS* to deal with different variables at the same time, that is, to incorporate other variables as inputs or attributes, fact that is computational unacceptable for *BJTS*.

Acknowledgements

The research reported in this paper has been supported in part under MCyT and Feder grant TIC2001-3579. FICyT and Hidroeléctrica del Cantábrico also made this research possible under the grants PA-TDI96-01, PC-TDI98-01, and PA-TDI99-05.

References

1. Blum A.L., Langley. P.:Selection of relevant features and examples in machine learning. Artificial Intelligence. (1997) 245-271
2. Brockwell P.J., Davis R.A, Introduction to Time Series and Forecasting. New York: Springer-Verlag, 1996.
3. Box G.E.P., Jenkins G.M., Reinsel C. Time Series Analysis, Forecasting and Control. New Jersey: Holden-Day (Revised Edition), 1976.

4. Ciarlet P.G. Introduction L`analyse Numèrique Matricielle et L`optimisation. París: Masson, 1982.
5. Charalambous C. Conjugate Gradient Algorithm for Efficient Training of Artificial Neural Networks. IEEE Proceedings, 1992;139(3):301-310.
6. Cybenko G. Approximations by Superpositions of a Sigmoidal Function. Mathematics of Control, Signals y Systems, 1992;2(4):303-314.
7. Demuth H., Beale M. Neural Network Toolbox for Use with Matlab. Massachusetts: The MathWorks, 1998.
8. Fine T. L. Feedforward Neural Network Methodology. New York: Springer Verlag, 1999.
9. Fletcher R., Reeves C.M. Function Minimization by Conjugate Gradient. Computer Journal, 1964;7:149-154.
10. Fletcher R. Practical Methods of Optimisation. New York: John Wiley, 1992.
11. Glowinski R. Numerical Solutions of Non-linear Problems and Variational Inequalities. New York: Springer-Verlag, 1987.
12. Goyache, F.; Coz, J. J. del; Quevedo, J. R.; López, S.; Alonso, J.; Ranilla, J.; Luaces, O.; Alvarez, I.; Bahamonde, A.: Using artificial intelligence to design and implement a morphological assessment system in beef cattle. Animal Science, Vol. 73 (2001), pp. 49-60.
13. Marquardt D. W. An Algorithm for Least-squares Estimation of Nonlinear Parameters. J. Soc. Indust. Appl. Math., 1963;11:431.
14. Numerical Algorithms Group. NAG Foundation Toolbox for Use with Matlab. Massachusetts: The MathWorks, 1995.
15. Poggio T., Girosi F. Networks for Approximation and Learning. In Proc. IEEE, 1990;1481-1497.
16. Polak E. Optimisation Algorithms and Consistent Approximations. New York: Springer-Verlag, 1997.
17. Quinlan, J.R.: Cubist. http://www.rulequest.com/cubist-info.html
18. Torgo. L.: Functional models for regression tree leaves. In Proc. of the 14th International Conference on Machine Learning, Nashville, TN. Morgan Kaufmann. (1997) 385-393
19. Uriel E. and Peiró A. Introducción al análisis de series temporales. Madrid: AC, 2000.
20. Wang, Y., and Witten, I.H.. Inducing model trees for continuous classes. In Poster Papers 9th European Conf. on Machine Learning. Prague, Czech Republic. (1997) 128-137.