Forecasting Time Series with Multiple Seasonal Cycles Using Neural Networks with Local Learning

Grzegorz Dudek

Department of Electrical Engineering, Czestochowa University of Technology,
Al. Armii Krajowej 17, 42-200 Czestochowa, Poland
dudek@el.pcz.czest.pl

Abstract. In the article a simple neural model with local learning for forecasting time series with multiple seasonal cycles is presented. This model uses patterns of the time series seasonal cycles: input ones representing cycles preceding the forecast moment and forecast ones representing the forecasted cycles. Patterns simplify the forecasting problem especially when a time series exhibits nonstationarity, heteroscedasticity, trend and many seasonal cycles. The artificial neural network learns using the training sample selected from the neighborhood of the query pattern. As a result the target function is approximated locally which leads to a reduction in problem complexity and enables the use of simpler models. The effectiveness of the proposed approach is illustrated through applications to electrical load forecasting and compared with ARIMA and exponential smoothing approaches. In a day ahead load forecasting simulations indicate the best results for the one-neuron network.

Keywords: seasonal time series forecasting, short-term load forecasting, local learning, neural networks.

1 Introduction

Time series may contain four different components: trend, seasonal variations, cyclical variations, and irregular component. Seasonality is defined to be the tendency of time series data to exhibit some pattern that repeats periodically with variation. Sometimes a time series contains multiple seasonal cycles of different lengths. Fig. 1 shows such a time series, where we can observe annual, weekly and daily variations. This series represents hourly electrical load of the Polish power system. From this figure it can be seen that the daily and weekly profiles change during the year. In summer they are more flat than in winter. The daily profile depends on the day of the week as well. The profiles of the weekdays are similar to each other in the same period of the year. To the characteristic features of this time series its nonstationarity and heteroscedasticity should be included as well. These all features have to be captured by the flexible forecasting model.

The most commonly employed methods to modeling seasonal time series include [1]: seasonal autoregressive integrated moving average model (ARIMA), exponential

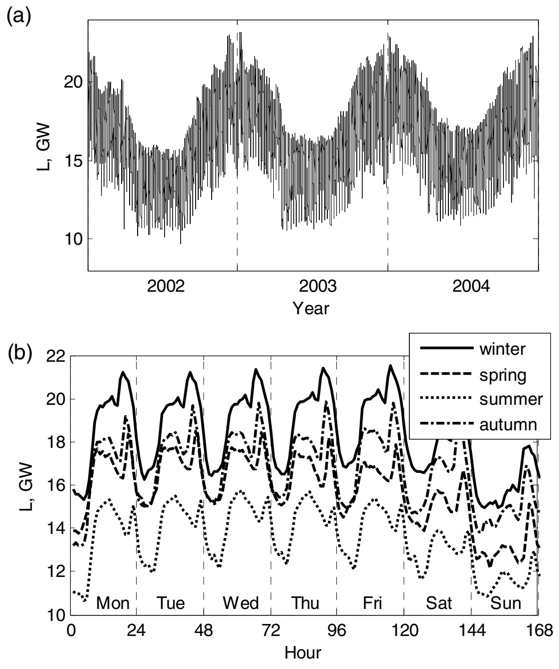

Fig. 1. The load time series of the Polish power system in three-year (a) and one-week (b) intervals

smoothing (ES), artificial neural networks (ANNs), dynamic harmonic regression, vector autoregression, random effect models, and many others.

The base ARIMA model with just one seasonal pattern can be extended for the case of multiple seasonalities. An example of such an extension was presented in [2]. A combinatorial problem of selecting appropriate model orders is an inconvenience in the time series modeling using multiple seasonal ARIMA. Another disadvantage is the linear character of the ARIMA model.

Another popular model – the Holt-Winters exponential smoothing was adapted by Taylor so that it can accommodate two and more seasonalities [2]. An advantage of the ES models is that they can be nonlinear. On the other hand it can be viewed as being of high dimension, as it involves initialization and updating of a large number of terms (level, periods of the intraday and intraweek cycles). In [1] more parsimonious formulation of ES is proposed. New exponentially weighted methods for forecasting time series that consist of both intraweek and intraday seasonal cycles can be found in [3].

Gould et al. [4] introduced the innovation state space models that underlie ES methods for both additive and multiplicative seasonality. This procedure provides a theoretical foundation for ES methods and improves on the current approaches by providing a common sense structure to the models, flexibility in modeling seasonal patterns, a potential reduction in the number of parameters to be estimated, and model based prediction intervals.

ANNs being nonlinear and data-driven in nature, may be well suited to the seasonal time series modeling. They can extract unknown and general information from multi-dimensional data using their self-learning ability. This feature releases a designer from a difficult task of a priori model selection. But new problems appear: the selection of network architecture as well as the learning algorithm. From many types of ANN most often in forecasting tasks the multilayer perceptron is used, which has a property of universal approximation. ANNs are able to deal with the seasonal time series without the prior seasonal adjustment but deseasonalization and also detrending is recommended [5].

The time series decomposition is used not only in ANNs, but also in other models, e.g. ARIMA and ES. The components showing less complexity than the original time series can be modeled independently and more accurate. Usually the time series is decomposed on seasonal, trend and stochastic components. Other methods of decompositions apply the Fourier or wavelet transform. The simple way to remove seasonality is to define the separate time series for each observation in a cycle, i.e. in the case of cycle of length n, n time series is defined including observations in the same position in successive cycles.

This paper considers simple neural forecasting model that approximates the target function using patterns of seasonal cycles. Defining patterns we do not need to decompose a time series. A trend and many seasonal cycles as well as the nonstationarity and heteroscedasticity is not a problem here when using proper pattern definitions. The proposed neural model learns in a local learning procedure which allows to model the target function in the neighborhood of the query pattern. As a result we get a local model which is better fitted in this neighborhood.

2 Patterns of the Time Series Seasonal Cycles

Our goal is to forecast the time series elements in a period of one seasonal cycle of the shortest length. In the case of the time series shown in Fig. 1 this is a daily cycle containing n = 24 elements (hourly loads). The time series is divided into sequences containing one seasonal cycle of length n. In order to eliminate trend and seasonal variations of periods longer than n (weekly and annual variations in our example), the sequence elements are preprocessed to obtain their patterns. The pattern is a vector with components that are functions of actual time series elements. The input and output (forecast) patterns are defined: $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T$ and $\mathbf{y} = [y_1\ y_2\ \ldots\ y_n]^T$, respectively. The patterns are paired ($\mathbf{x}_i$, $\mathbf{y}_i$), where $\mathbf{y}_i$ is a pattern of the time series sequence succeeding the sequence represented by $\mathbf{x}_i$. The interval between these sequences is equal to the forecast horizon τ.

The way of how the $\mathbf{x}$ and $\mathbf{y}$ patterns are defined depends on the time series nature (seasonal variations, trend), the forecast period and the forecast horizon. Functions transforming series elements into patterns should be defined so that patterns carry most information about the process. Moreover, functions transforming forecast sequences into patterns $\mathbf{y}$ should ensure the opposite transformation: from the forecasted pattern $\mathbf{y}$ to the forecasted time series sequence.

The forecast pattern $\mathbf{y}_i = [y_{i,1}\ y_{i,2}\ \dots\ y_{i,n}]$ encodes the successive actual time series elements z in the forecast period $i+\tau$: $\mathbf{z}_{i+\tau} = [z_{i+\tau,1}\ z_{i+\tau,2}\ \dots\ z_{i+\tau,n}]$, and the corresponding input pattern $\mathbf{x}_i = [x_{i,1}\ x_{i,2}\ \dots\ x_{i,n}]$ maps the time series elements in the period i preceding the forecast period: $\mathbf{z}_i = [z_{i,1}\ z_{i,2}\ \dots\ z_{i,n}]$. Vectors $\mathbf{y}$ are encoded using current process parameters from the nearest past, which allows to take into consideration current variability of the process and ensures possibility of decoding. Some definitions of the functions mapping the original space Z into the pattern spaces X and Y, i.e. $f_x : Z \to X$ and $f_y : Z \to Y$ are presented in [6]. The most popular definitions are of the form:

$$f_x(z_{i,t}) = \frac{z_{i,t} - \bar{z}_i}{\sqrt{\sum_{l=1}^{n}(z_{i,l} - \bar{z}_i)^2}}, \qquad f_y(z_{i,t}) = \frac{z_{i+\tau,t} - \bar{z}_i}{\sqrt{\sum_{l=1}^{n}(z_{i,l} - \bar{z}_i)^2}}, \tag{1}$$

where: $i = 1, 2, \dots, N$ – the period number, $t = 1, 2, \dots, n$ – the time series element number in the period i, τ – the forecast horizon, $z_{i,t}$ – the tth time series element in the period i, $\bar{z}_i$ – the mean value of elements in period i.

The function f_x defined using (1) expresses normalization of the vectors $\mathbf{z}_i$. After normalization these vectors have the unity length, zero mean and the same variance. When we use the standard deviation of the vector $\mathbf{z}_i$ components in the denominator of equation (1), we receive vector $\mathbf{x}_i$ with the unity variance and zero mean. Note that the nonstationary and heteroscedastic time series is represented by patterns having the same mean and variance.

Forecast pattern $\mathbf{y}_i$ is defined using analogous functions to input pattern function f_x, but it is encoded using the time series characteristic ($\bar{z}_i$) determined from the process history, what enables decoding of the forecasted vector $\mathbf{z}_{i+\tau}$ after the forecast of pattern $\mathbf{y}$ is determined. To calculate the forecasted time series element values on the basis of their patterns we use the inverse function $f_y^{-1}(y_{i,t})$.

3 Local Learning

The training data can have different properties in different regions of the input and output spaces thus it is reasonable to model this data locally. The local learning [7] concerns the optimization of the learning system on a subset of the training sample, which contains points from the neighborhood around the current query point $\mathbf{x}^*$. By the neighborhood of $\mathbf{x}^*$ in the simplest case we mean the set of its k nearest neighbors. A result of the local learning is that the model accurately adjusts to the target function in the neighborhood of $\mathbf{x}^*$ but shows weaker fitting outside this neighborhood. Thus we get model which is locally competent but its global generalization property is weak. Modeling the target function in different regions of the space requires relearning of the model or even to construct different model, e.g. we can use a linear model for linear fragments of the target function while for the nonlinear fragments we can use a nonlinear model. The generalization can be achieved by using a set of local models that are competent for different regions of the input space. Usually these models are learned when a new query points are presented.

The error criterion minimized in local learning algorithm can be defined as follows:

$$E(\mathbf{x}^*) = \sum_{i=1}^{N} K(d(\mathbf{x}_i, \mathbf{x}^*), h)\delta(\mathbf{y}_i, f(\mathbf{x}_i)), \qquad (2)$$

where: N – number of training patterns, $K(d(\mathbf{x}_i,\mathbf{x}^*),h)$ – kernel function with bandwidth h, $d(\mathbf{x}_i,\mathbf{x}^*)$ – distance between the query pattern $\mathbf{x}^*$ and training pattern $\mathbf{x}_i$, $\delta(\mathbf{y}_i,f(\mathbf{x}_i))$ – error between the model response $f(\mathbf{x}_i)$ and the target response $\mathbf{y}_i$ when input pattern $\mathbf{x}_i$ is presented (this response can be a scalar value).

Various kernel functions might be used, including uniform kernels and Gaussian kernels which are ones of the most popular. The kernel is centered on the query point $\mathbf{x}^*$ and the bandwidth h determines the weight of the ith training pattern error in (2). When we use uniform kernel the training patterns for which $d(\mathbf{x}_i,\mathbf{x}^*) \le h = d(\mathbf{x}_k,\mathbf{x}^*)$, where $\mathbf{x}_k$ is the kth nearest neighbor of $\mathbf{x}^*$, have unity weights. More distant patterns have zero weights, and therefore there is no need to use these points in the learning process. For Gaussian kernels all training points have nonzero weights calculated from the formula $\exp(-d^2(\mathbf{x}_i,\mathbf{x}^*)/(2h^2))$, which means that their weights decrease monotonically with the distance from $\mathbf{x}^*$ and with the speed dependent on h. In order to reduce the computational cost of determination of errors and weights for all training points we can combine both kernels and calculate weights according to the Gaussian kernel for only k nearest neighbors of $\mathbf{x}^*$. The computational cost is now independent of the total number of training patterns, but only on the number of considered neighbors k.

In the experimental part of this paper we use local learning procedure with uniform kernel.

4 Experimental Results

As an illustrative example of forecasting time series with multiple seasonal cycles using neural networks with local learning we study the short-term electrical load forecasting problem. Short-term load forecasting plays a key role in control and scheduling of power systems and is extremely important for energy suppliers, system operators, financial institutions, and other participants in electric energy generation, transmission, distribution, and markets.

In the first experiments we use the time series of the hourly electrical load of the Polish power system from the period 2002–2004. This series is shown in Fig. 1. The time series were divided into training and test parts. The test set contained 31 pairs of patterns from July 2004. The training part Ψ contained patterns from the period from 1 January 2002 to the day preceding the day of forecast.

We define the forecasting tasks as forecasting the power system load at hour $t = 1, 2, \ldots, 24$ of the day $j = 1, 2, \ldots, 31$, where j is the day number in the test set. So we get 744 forecasting tasks. In local learning approach for each task the separate ANNs were created and learned. The training set for each forecasting task is prepared as follows:

- first we prepare the set $\Omega = \{(\mathbf{x}_i, y_{i,t})\}$, where i indicates pairs of patterns from Ψ representing days of the same type (Monday, …, Sunday) as days represented by a query pair $(\mathbf{x}^*, y_t^*)$,
- then based on the Euclidean distances $d(\mathbf{x}_i, \mathbf{x}^*)$ we select from Ω k nearest neighbors of the query pair getting the training set $\Phi = \{(\mathbf{x}_i, y_{i,t})\} \subset \Omega \subset \Psi$.

For example when the forecasting task is to forecast the system load at hour t on Sunday, model learns on k nearest neighbors of the query pattern which are selected from x-patterns representing the Saturday patterns and tth components of y-patterns representing the Sunday patterns.

ANN (the multilayer perceptron) learns the mapping of the input patterns to the components of output patterns: $f_t : X \to Y_t$. Number of ANN inputs is equal to the x-pattern components. To prevent overfitting ANN is learned using Levenberg-Marquardt algorithm with Bayesian regularization [8], which minimizes a combination of squared errors and net weights. The resulting network has good generalization qualities.

In the first experiment we assume $k = 12$. Since the target function f_t is modeled locally, using a small number of learning points, rather a simple form of this function should be expected, which implies small number of neurons. We tested the networks:

- composed of only one neuron with linear or bipolar sigmoidal activation function,
- with one hidden layer consisting of $m = 2, ..., 8$ neurons with sigmoidal activation functions and one output neuron with linear activation function. Such a network architecture can be seen as a universal approximator .

APE and MAPE (absolute percentage error and mean APE) is adopted here to assess the performance of the forecasting models. The results (MAPE for the training and test samples and the interquartile range (IQR) of $MAPE_{tst}$) of the 9 variants of ANNs are presented in tab. 1. Test errors for these variants are statistically indistinguishable (to check this we use the Wilcoxon rank sum test for equality of APE medians; $\alpha = 0{,}05$). It is observed that for the two-layered networks in many cases most weights tends to zero (weights decay is a result of regularization), thus some neurons can be eliminated. As an optimal ANN architecture that one with one neuron with sigmoidal activation function is chosen. This one-neuron ANN is used in the next experiments.

In the second experiment we examine the network performance depending on the number of the nearest neighbors k, i.e. the size of the training set Φ. We change k from 2 to 50. The results are shown in Fig. 2, where MAPE for the cases when the ANN is trained using all training points representing days of the same type as days represented by query pair, i.e. points from the set Ω, is also shown. As we can see from this figure the test error remains around 1 when $k \in [6, 50]$. For these cases $MAPE_{tst}$ are statistically indistinguishable when using Wilcoxon test. When we train ANN using patterns from the set Ω $MAPE_{tst}$ is statistically distinguishable greater than for $k \in [6, 50]$.

Table 1. Results of 9 variants of ANNs

	Number of neurons								
	1 lin	1 sig	2+1	3+1	4+1	5+1	6+1	7+1	8+1
$MAPE_{trn}$	0.80	0.88	1.12	1.11	1.09	1.09	1.08	1.09	1.10
$MAPE_{tst}$	1.03	0.98	0.98	0.98	1.00	1.00	1.02	1.02	1.01
IQR_{tst}	1.09	1.03	1.02	1.03	1.06	1.02	1.05	0.99	1.04

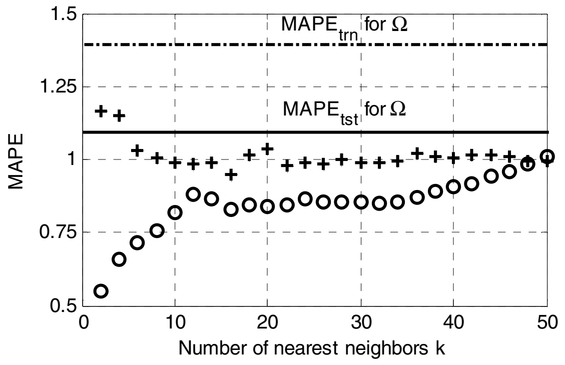

Fig. 2. MAPE for the training sets (rings) and test set (crosses) depending on *k*

In the local learning approach the thorny issue is the ratio of the training points number to the number of free parameters of the network. This ratio for our example even for one-neuron ANN is too small (12/25), which means that the model is oversized (it has too many degrees of freedom in relation to the problem complexity expressed by only a few training points). The regularization which has a form of a penalty for complexity is a good idea to solve this problem. Another idea is the feature selection or feature extraction as a form of dimensionality reduction. The most popular method of feature extraction is the principal component analysis (PCA). This procedure uses an orthogonal transformation to convert a set of multidimensional vectors of possibly correlated components into a set of vectors of linearly uncorrelated components called principal components. The number of principal components is less than or equal to the dimension of original vectors. In the next experiment we transform the 24-dimensional x-patterns into patterns with a smaller number of uncorrelated components using PCA. Fig. 3 shows relationship between MAPE and the number of principal components. From this figure it can be seen that the levels of errors are very similar. $MAPE_{tst}$ are statistically indistinguishable for different number of principal components. Using only first principal component we can built good neural forecasting model for our data. Such a model has only two parameters. The percent variance explained by the corresponding principal components are shown in Fig. 4. The first principal component explains 30% of the total variance.

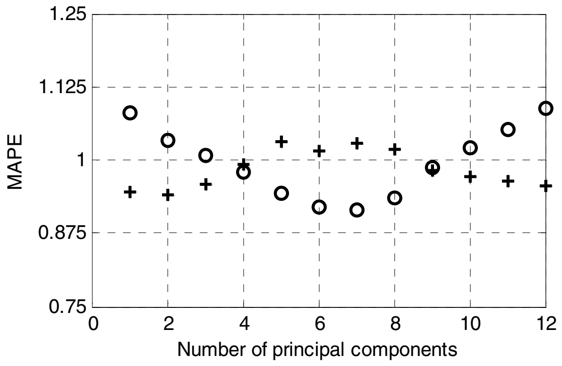

Fig. 3. MAPE for the training sets (rings) and test set (crosses) depending on the number of principal components

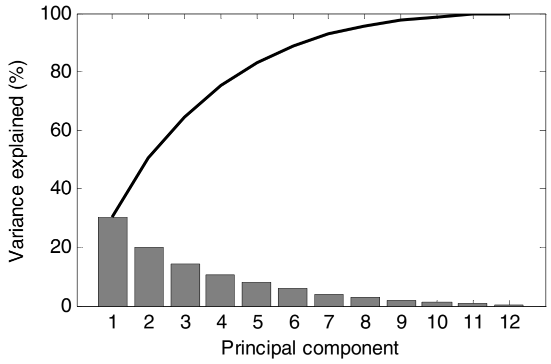

Fig. 4. The percent variance explained by the corresponding principal components

Now we compare the proposed one-neuron ANN with other popular models of the seasonal time series forecasting: ARIMA and ES. These models were tested in the next day electrical load curve forecasting problem on three time series of electrical load:

- PL: time series of the hourly load of the Polish power system from the period 2002–2004 (this time series was used in the experiments described above). The test sample includes data from 2004 with the exception of 13 untypical days (e.g. holidays),

- FR: time series of the half-hourly load of the French power system from the period 2007–2009. The test sample includes data from 2009 except for 21 untypical days,
- GB: time series of the half-hourly load of the British power system from the period 2007–2009. The test sample includes data from 2009 except for 18 untypical days.

In ARIMA the time series were decomposed into n series, i.e. for each t a separate series was created. In this way a daily seasonality was removed. For the independent modeling of these series ARIMA$(p, d, q)\times(P, D, Q)_m$ model was used:

$$\Phi(B^m)\phi(B)(1-B^m)^D(1-B)^d z_t = c + \Theta(B^m)\theta(B)\xi_t, \tag{3}$$

where $\{z_t\}$ is the time series, $\{\xi_t\}$ is a white noise process with mean zero and variance σ^2, B is the backshift operator, $\Phi(.)$, $\phi(.)$, $\Theta(.)$, and $\theta(.)$ are polynomials of order P, p, Q and q, respectively, m is the seasonal period (for our data $m = 7$), d and D are orders of nonseasonal and seasonal differencing, respectivelly, and c is a constant.

To find the best ARIMA model for each time series we use a step-wise procedure for traversing the model space which is implemented in the **forecast** package for the **R** system for statistical computing [9]. This automatic procedure returns the model with the lowest Akaike's Information Criterion (AIC) value.

ARIMA model parameters, as well as the parameters of the ES model described below, were estimated using 12-week time series fragments immediately preceding the forecasted daily period. Untypical days in these fragments were replaced with the days from the previous weeks.

The ES state space models [10] are classified into 30 types depending on how the seasonal, trend and error components are taken into account. These components can be expressed additively or multiplicatively, and the trend can be damped or not. For example, the ES model with a dumped additive trend, multiplicative seasonality and multiplicative errors is of the form:

$$\begin{aligned}
&\text{Level:} && l_t = (l_{t-1} + \phi b_{t-1})(1 + \alpha\xi_t),\\
&\text{Growth:} && b_t = \phi b_{t-1} + \beta(l_{t-1} + \phi b_{t-1})\xi_t,\\
&\text{Seasonal:} && s_t = s_{t-m}(1 + \gamma\xi_t),\\
&\text{Forecast:} && \mu_t = (l_{t-1} + \phi b_{t-1})s_{t-m},
\end{aligned} \tag{4}$$

where l_t represents the level of the series at time t, b_t denotes the growth (or slope) at time t, s_t is the seasonal component of the series at time t, μ_t is the expected value of the forecast at time t, α, β, $\gamma \in (0, 1)$ are the smoothing parameters, and $\phi \in (0, 1)$ denotes a damping parameter.

In model (4) there is only one seasonal component. For this reason, as in the case of the ARIMA model, time series is decomposed into n series, each of which represents the load at the same time t of a day. These series were modeled independently using an automated procedure implemented in the **forecast** package for the **R** system [9]. In this procedure the initial states of the level, growth and seasonal components are estimated as well as the smoothing and damping parameters. AIC was used for selecting the best model for a given time series.

In Table 2 results of PL, FR and GB time series forecasts are presented. In this table the results of forecast determined by the naïve method are also shown. The forecast rule in this case is as follows: the forecasted daily cycle is the same as seven days ago. The Wilcoxon test indicates statistically significant differences between $MAPE_{tst}$ for each pair of models and each time series, so we can indicate the one-neuron ANN as the best model for this data and ES as the second best model.

Table 2. Results of forecasting

Model	PL		FR		GB	
	$MAPE_{tst}$	IQR	$MAPE_{tst}$	IQR	$MAPE_{tst}$	IQR
ANN	1.44	1.41	1.64	1.70	1.65	1.70
ARIMA	1.82	1.71	2.32	2.53	2.02	2.07
ES	1.66	1.57	2.10	2.29	1.85	1.84
Naïve	3.43	3.42	5.05	5.96	3.52	3.82

The last experiment concerns time series forecasting up to seven daily periods ahead. In such tasks the y-patterns are defined using $\tau = 1, 2, \ldots, 7$. For each horizon τ the one-neuron ANN is trained using the same local learning scheme as for $\tau = 1$ described above. The forecast errors for PL, FR and GB time series in Fig. 5 are presented. For FR and GB data ANN gave the lowest errors. For PL data and $\tau > 2$ ES model is better, and for $\tau > 3$ also ARIMA model is better. The actual and forecasted fragments of the time series are shown in Fig. 6.

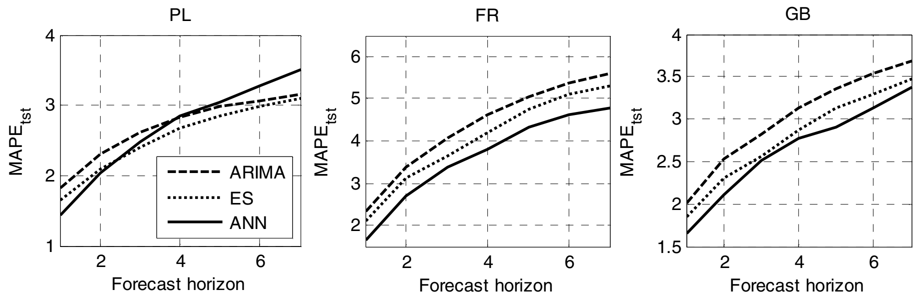

Fig. 5. The forecast errors for different horizons

Note that in the case of ARIMA and ES the model parameters are estimated on the basis of the time series fragment (12 weeks in our example) directly preceding the forecasted fragment. ANN learns on the training set composed of patterns represented daily periods from longer history. In local learning case the training patterns are selected using criterion based on the similarity to the current input pattern.

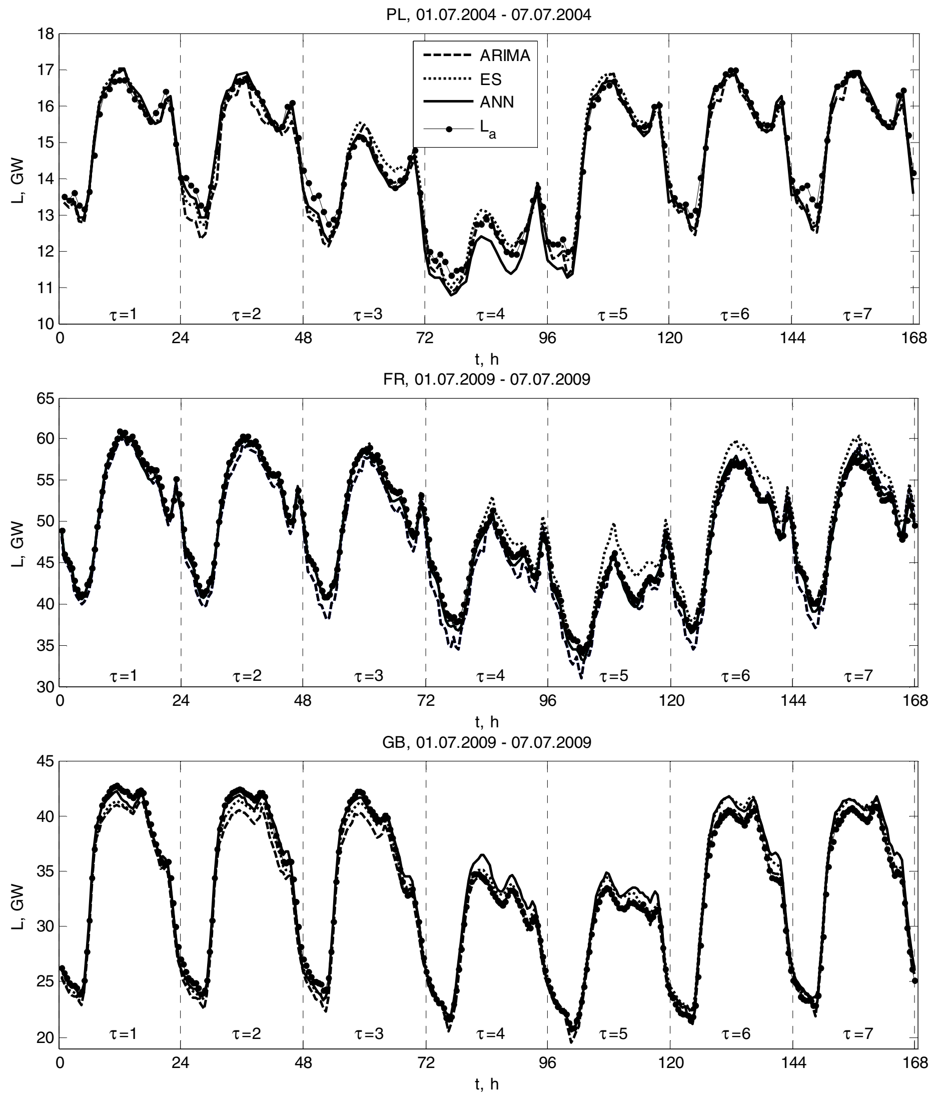

Fig. 6. The fragments of load time series and their forecasts for different horizons

5 Conclusions

In this article we examine a simple neural model with local learning for forecasting seasonal time series. At the initial stage of the forecasting procedure data are preprocessed to get patterns of the time series seasonal periods. An approach based on the patterns of the seasonal cycles simplify the problem of forecasting of the nonstationary and heteroscedastic time series with trend and many seasonal variations. After simplification the problem can be modeled using simpler tools. The existence of many seasonal cycles is not a problem when we use forecasting model based on patterns. We resign from the global modeling, which does not necessarily brings good results for the current query point. Instead, we approximate the target function locally in the neighborhood of the query point. The disadvantage of the local learning is the need to learn the model for each query point. But since the local complexity is lower than the global one, we can use a simple model that is quickly learned.

This approach is acceptable when we have enough time (some seconds) to learn model and prepare forecast. The learning speed is penalized by the selection of the nearest neighbors. As shown by simulation studies to model the local relationship between input and output patterns the one-neuron model is sufficient. This model turned out to be better than the conventional models (ARIMA and exponential smoothing) in one-day ahead forecasting of the electrical load time series and competitive in forecasting over longer time horizons.

Acknowledgments. The author would like to thank Professor James W. Taylor from the Saïd Business School, University of Oxford for providing French and British load data. The study was supported by the Research Project N N516 415338 financed by the Polish Ministry of Science and Higher Education.

References

1. Taylor, J.W., Snyder, R.D.: Forecasting Intraday Time Series with Multiple Seasonal Cycles Using Parsimonious Seasonal Exponential Smoothing. Department of Econometrics and Business Statistics Working Paper 9/09, Monash University (2009)
2. Taylor, J.W.: Short-Term Electricity Demand Forecasting Using Double Seasonal Exponential Smoothing. Journal of the Operational Research Society 54, 799–805 (2003)
3. Taylor, J.W.: Exponentially Weighted Methods for Forecasting Intraday Time Series with Multiple Seasonal Cycles. International Journal of Forecasting 26(4), 627–646 (2010)
4. Gould, P.G., Koehler, A.B., Ord, J.K., Snyder, R.D., Hyndman, R.J., Vahid-Araghi, F.: Forecasting Time-Series with Multiple Seasonal Patterns. European Journal of Operational Research 191, 207–222 (2008)
5. Zhang, G.P., Qi, M.: Neural Network Forecasting for Seasonal and Trend Time Series. European Journal of Operational Research 160, 501–514 (2005)
6. Dudek, G.: Similarity-based Approaches to Short-Term Load Forecasting. In: Zhu, J.J., Fung, G.P.C. (eds.) Forecasting Models: Methods and Applications, pp. 161–178. iConcept Press (2010), `http://www.iconceptpress.com/www/site/download.paper.php?paperID=100917020141`
7. Bottou, L., Vapnik, V.: Local learning algorithms. Neural Computation 4, 888–900 (1992)
8. Taylor, J.W., Snyder, R.D.: Forecasting Intraday Time Series with Multiple Seasonal Cycles Using Parsimonious Seasonal Exponential Smoothing. Department of Econometrics and Business Statistics Working Paper 9/09, Monash University (2009)
9. Hyndman, R.J., Khandakar, Y.: Automatic Time Series Forecasting: The Forecast Package for R. Journal of Statistical Software 27(3), 1–22 (2008)
10. Hyndman, R.J., Koehler, A.B., Ord, J.K., Snyder, R.D.: Forecasting with Exponential Smoothing: The State Space Approach. Springer (2008)