Electric Power Systems Research 130 (2016) 139–147

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

Pattern-based local linear regression models for short-term load forecasting

Grzegorz Dudek*

Department of Electrical Engineering, Czestochowa University of Technology, Al. Armii Krajowej 17, 42-200 Czestochowa, Poland

ARTICLE INFO

Article history:
Received 2 March 2015
Received in revised form 2 September 2015
Accepted 4 September 2015
Available online 19 September 2015

Keywords:
Linear regression
Partial least-squares regression
Patterns of seasonal cycles
Short-term load forecasting
Time series

ABSTRACT

In this paper univariate models for short-term load forecasting based on linear regression and patterns of daily cycles of load time series are proposed. The patterns used as input and output variables simplify the forecasting problem by filtering out the trend and seasonal variations of periods longer than the daily one. The nonstationarity in mean and variance is also eliminated. The simplified relationship between variables (patterns) is modeled locally in the neighborhood of the current input using linear regression. The load forecast is constructed from the forecasted output pattern and the current values of variables describing the load time series. The proposed stepwise and lasso regressions reduce the number of predictors to a few. In the principal components regression and partial least-squares regression only one predictor is used. This allows us to visualize the data and regression function. The performances of the proposed methods were compared with that of other models based on ARIMA, exponential smoothing, neural networks and Nadaraya–Watson estimator. Application examples confirm valuable properties of the proposed approaches and their high accuracy.

1. Introduction

Short-term load forecasting (STLF) is necessary for economic generation of power and system security. It refers to forecasts of system load from hours to several days ahead. The accurate load forecasts lead to lower operating cost which contributes to savings in electric utilities. The STLF accuracy is also important for the deregulated electricity markets. The amount of energy which the utility has to buy or sell in the real time market at unfavorable prices depends on the forecast error. Thus STLF is a very important problem for electric utilities, regional transmission organizations, energy suppliers and financial institutions. This is reflected in the literature by many forecasting methods that have been applied, including conventional methods and new computational intelligence and machine learning methods. A large research activity in the field of STLF is related with the problem complexity: the load time series is nonstationary in mean and variance, expresses trend, multiple seasonal variations (daily, weekly and annual) and random noise. In addition, load is affected by many external factors such as weather, time, demography, economy, electricity prices, geographical conditions, consumer types and their habits.

Among the conventional STLF methods the most commonly employed are: the Holt–Winters exponential smoothing (ES) and the autoregressive integrated moving average (ARIMA) models [1]. In ES the time series is decomposed into a trend component (expressed by level and growth terms) and seasonal components which can be combined additively or multiplicatively. ES allows us to model nonlinear and heteroscedastic time series but the exogenous variables cannot be introduced into the model. Other important disadvantages of ES are overparameterization and a large number of starting values to estimate. In [2] to reduce the dimension of the model new ES formulation called parsimonious seasonal ES was proposed. But there are still dozens or hundreds of terms to initialize and update in the model. The recently developed exponentially weighted methods in application to STLF are presented in [3].

ARIMA processes are a very rich class of possible models and allows us to model multiple seasonal cycles. The stochastic nature of load is often modeled with seasonal ARIMA models in practice. A disadvantage of ARIMA models is their linear nature. The order selection process of ARIMA is usually considered subjective and difficult to apply, which is a main obstacle in using these models. To simplify the forecasting problem the time series is often decomposed. The components: trend, seasonal components and irregular component, showing less complexity than the original series, are modeled independently (e.g. [4]). Another time series

* Corresponding author. Tel.: +48 343250896; fax: +48 343250803.
E-mail address: dudek@el.pcz.czest.pl

http://dx.doi.org/10.1016/j.epsr.2015.09.001
0378-7796/

decomposition method using lifting scheme (the second generation wavelet transform) was described in [5].

The most popular computational intelligence methods applied in STLF are neural networks. They have many attractive features such as: universal approximation property, learning capabilities, massive parallelism, robustness in the presence of noise, and fault tolerance. The drawbacks of neural network include: disruptive and unstable training, difficulty in matching the network structure to the problem complexity, weak capacity of extrapolation and many parameters to estimate (hundreds of weights). Some examples of using neural networks in STLF are: [6], where the complexity of the network is controlled by the Bayesian approach, [7], where a new hybrid forecasting method composed of wavelet transform, multilayer perceptron and evolutionary algorithm is proposed, [8], where a generic framework combining similar day selection, wavelet decomposition, and multilayer perceptron is presented, and [9], where the neural network generates the prediction intervals.

Another branch of computational intelligence, fuzzy logic, allows us to enter information by facts and rules formulated verbally by experts and describing the behavior of complex systems by using linguistic expressions. With the help of fuzzy rules the imprecise, incomplete and ambiguous information can be introduced into the STLF models. When it is difficult to gain knowledge directly from the experts, to generate a set of if-then rules the neuro-fuzzy approach is applied, which learns from examples. But the neuro-fuzzy system structure is complex and the number of parameters is usually large (it depends on the problem dimensionality and complexity), so the learning is difficult and does not guarantee convergence to the global minimum. Examples of STLF models based on fuzzy logic are: [10], where the neuro-fuzzy system is used to adjust the results of load forecasting obtained by radial basis function neural network, [11], where two neuro-fuzzy networks are proposed: a wavelet fuzzy neural network using the fuzzified wavelet features as the inputs, and fuzzy neural network employing the Choquet integral as the outputs, [12], where an integrated approach which combines a self-organizing fuzzy neural network learning method with a bilevel optimization method is described, and [13], where the forecasting model combines fuzzy logic, wavelet transform and neural network. Another useful computational intelligence tools for STLF are: support vector machines (SVM) [14,15], ensembles of models [16,17] and artificial immune systems [18] (description of more STLF models you can find on the website http://gdudek.el.pcz.pl/publications).

It is noteworthy that many of the STLF models developed in recent years are hybrid solutions. They combine data preprocessing methods (e.g. wavelet transform) with approximation methods (such as neural and neuro-fuzzy networks or SVM) and optimization or learning methods (e.g. evolutionary and swarm algorithms).

The disadvantages of the above mentioned complex forecasting models with many parameters are: hard and time-consuming training, problems with generalization, unclear structure and uninterpretable parameters. Most often time series with multiple seasonal cycles and trend, expressing nonstationarity in mean and variance cannot be modeled directly and additional treatments such as detrending, deseasonality or decomposition are needed.

In contrast to the complex models commonly used in STLF in this work simple methods of linear regression are proposed. The number of parameters here is small and they can be estimated using simple least squares approach. The key element of the proposed methods is data preprocessing: defining patterns of seasonal cycles. This simplifies the STLF problem eliminating nonstationarity, and filtering trend and seasonal cycles longer than the daily one.

The paper is organized in a theoretical and an empirical part. In the beginning the patterns of daily cycles of load time series are defined. Then the main concepts of the linear regression models for STLF are introduced. In the last section the real load data are used to provide examples of model building and forecasting in practice. The results of the proposed methods are compared to results of other STLF methods: ARIMA, ES, multilayer perceptron and Nadaraya–Watson estimator.

2. Patterns of the times series seasonal cycles

Data preprocessing based on patterns simplifies the forecasting time series with multiple seasonal cycles. In our case the patterns of the daily cycles are introduced: the input patterns $\mathbf{x}$ and output ones $\mathbf{y}$. The input pattern is a vector $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T \in X = \mathbb{R}^n$, representing the vector of loads in successive timepoints of the daily period: $\mathbf{L} = [L_1\ L_2\ \ldots\ L_n]^T$, where $n = 24$ for hourly load time series, $n = 48$ for half-hourly load time series and $n = 96$ for quarter-hourly load time series. The functions mapping the time series elements L into patterns are dependent on the time series (trend, seasonal variations), the forecast period and horizon. They should maximize the model quality. In this study the input pattern $\mathbf{x}_i$, representing the ith daily period, is defined as follows:

$$x_{i,t} = \frac{L_{i,t} - \bar{L}_i}{\sqrt{\sum_{l=1}^{n} \left(L_{i,l} - \bar{L}_i\right)^2}}, \tag{1}$$

where $i = 1, 2, \ldots, N$ is the daily period number, N is the number of days in the time series, $t = 1, 2, \ldots, n$ is the time series element number in the period i, $L_{i,t}$ is the tth time series element (load) in the period i, $\bar{L}_i$ is the mean load value in the period i.

According to definition (1), first we subtract the vector $\mathbf{L}_i$ mean from its components and then we divide the resulting vector by its length. As a result we get the normalized vectors $\mathbf{x}_i$ with length 1, zero mean and the same variance. Note that the time series which is nonstationary in mean and variance is represented now by x-patterns having the same mean and variance. The trend and additional seasonal variations (weekly and annual ones in our case) are filtered. The x-patterns contain information only about the shapes of daily curves.

Whilst the x-patterns represent input variables (predictors), i.e. the loads for the day i, the y-patterns represent the output variables, i.e. the forecasted loads for the day $i + \tau$, where τ is a forecast horizon in days. The components of the n-dimensional output pattern $\mathbf{y}_i = [y_{i,1}\ y_{i,2}\ \ldots\ y_{i,n}]^T \in Y = \mathbb{R}^n$, representing the load vector $\mathbf{L}_{i+\tau}$ are defined as follows:

$$y_{i,t} = \frac{L_{i+\tau,t} - \bar{L}_i}{\sqrt{\sum_{l=1}^{n} \left(L_{i,l} - \bar{L}_i\right)^2}}, \tag{2}$$

where $i = 1, 2, \ldots, N$, $t = 1, 2, \ldots, n$.

This is the similar equation to (1) but in this case we do not use the mean load of the day $i + \tau$ $\left(\bar{L}_{i+\tau}\right)$ in the numerator and $\sqrt{\sum_{l=1}^{n} \left(L_{i+\tau,l} - \bar{L}_{i+\tau}\right)^2}$ in the denominator, because these values are not known in the moment of forecasting. We use known values of $\bar{L}_i$ and $\sqrt{\sum_{l=1}^{n} \left(L_{i,l} - \bar{L}_i\right)^2}$ instead. This is very important because when the forecast of pattern $\mathbf{y}_i$ is generated by the model we can determine the forecast of vector $\mathbf{L}_{i+\tau}$ using transformed Eq. (2):

$$\widehat{L}_{i+\tau,t} = \widehat{y}_{i,t} \sqrt{\sum_{l=1}^{n} \left(L_{i,l} - \bar{L}_i\right)^2} + \bar{L}_i, \tag{3}$$

where $\widehat{y}_{i,t}$ is the forecasted tth component of the pattern $\mathbf{y}_i$.

Note that $\bar{L}_i$ and the value of square root in (3) are known at the time of forecasting and can be used for decoding of $\widehat{y}_{i,t}$ to get $\widehat{L}_{i+\tau,t}$. Note also that $\sqrt{\sum_{l=1}^{n} \left(L_{i,l} - \bar{L}_i\right)^2}$ is the carrier of the dispersion of the current daily cycle. Using this square root in the denominator of

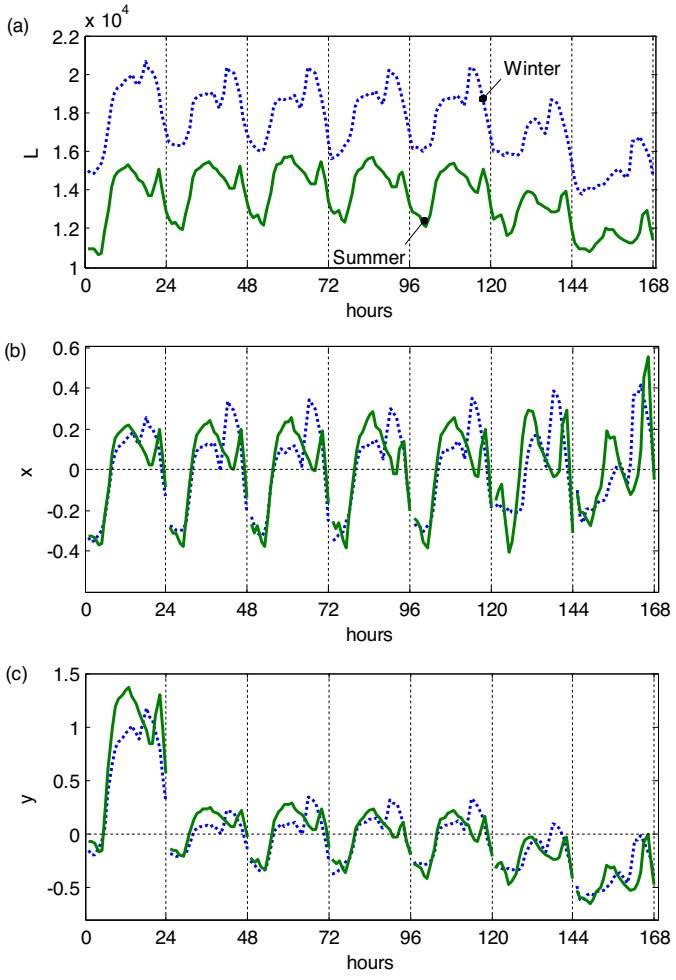

Fig. 1. Two fragments of the load time series (a) and their *x*-patterns (b) and *y*-patterns (c).

(1) we unify the dispersion of *x*-patterns. Using $\bar{L}_i$ in the numerator of (1) we unify the level of patterns. So the *x*-patterns are filtered versions of the daily curves. They carry information about the shape of the daily curves. This is shown in Fig. 1(a) and (b). It can be seen from these figures that the daily cycles of different days of the week $\delta \in \{\text{Monday}, \ldots, \text{Sunday}\}$ and from different periods of the year are represented by *x*-patterns having the same mean and variance. So we can simply compare the shapes of different days.

In the case of *y*-patterns using transformation (2) we unify the patterns for each type of the day of the week δ separately. The patterns of different days can be incomparable. This is because in (2) we use the mean and dispersion of the *i*th day to encode $\mathbf{L}_{i+\tau}$. So the *y*-patterns of Mondays for $\tau = 1$ are located higher than *y*-patterns of other days because we use $\bar{L}_i$ of Sundays in (2), which are usually lower than $\bar{L}_{i+\tau}$ of Mondays. For analogous reasons *y*-patterns of Saturdays and Sundays are located at a lower level than *y*-patterns of weekdays. This is shown in Fig. 1(c).

Because the position (level) of *y*-pattern depends on the day type δ, the forecasting models are built for the particular day type using training set Φ containing patterns corresponding to this day type. For example when we build the model for Monday and for $\tau = 1$ (next day forecast) we learn it on the training set containing *x*-patterns of Sundays and *y*-patterns of corresponding Mondays. If $\tau = 2$ (two days ahead forecast) we use *x*-patterns of Saturdays and *y*-patterns of Mondays in the training set. This approach and the unification of input and output variables using patterns simplify the forecasting model in which we do not need to implement weekly and annual cycles. The information about the position of the daily period in the weekly and annual cycles, which is contained in $\bar{L}_i$, we introduce to the forecast $\widehat{L}_{i+\tau,t}$ in (3) by adding $\bar{L}_i$ as well as we introduce the information about current dispersion of the time series multiplying $\widehat{y}_{i,t}$ by $\sqrt{\sum_{l=1}^{n}\left(L_{i,l}-\bar{L}_i\right)^2}$. So when we forecast the load time series using the pattern-based approach, first we filter out the information about the position of the days *i* and $i+\tau$ in the weekly and annual cycles ((1) and (2)). Then we build the model on patterns and we generate the forecast of the *y*-pattern. Finally we introduce the information about the position of the forecasted day in the weekly and annual cycles using (3).

More functions defining patterns you can find in [19]. A fragment of time series represented by *x*-patterns does not have to coincide with the daily period (e.g. *x*-pattern can represent loads at hours 13 to 24 of the day $i-1$ and hours 1 to 12 of the day *i*). It can include several adjacent daily cycles (e.g. two days preceding the day of forecast) or a part of one cycle (e.g. $t = 1, 2, \ldots$, 12 h of the day *i*). It does not have to include the contiguous sequence of elements. We can select elements to the input pattern, e.g. loads at hours 2, 5, 14 and 22. We can also use the feature extraction methods to create new pattern components from the original time series.

3. Linear regression models for STLF

The relationship between *x*- and *y*-patterns can be nonlinear. In our approach this function is approximated locally in the neighborhood around the current input pattern for which we want to get the forecast (we call this pattern a query pattern $\mathbf{x}^*$). By the neighborhood of $\mathbf{x}^*$ in the simplest case we mean the set of its *k* nearest neighbors defined as the *k* *x*-patterns from the history which are closest to $\mathbf{x}^*$ in terms of Euclidean distance and which represent the same day type δ as $\mathbf{x}^*$. In the neighborhood of $\mathbf{x}^*$ the target function mapping *x*-patterns to *y*-patterns is less complex than in the entire range of $\mathbf{x}$ variation. It is assumed that for small *k* this function can be approximated locally using linear regression (in the experimental part of the work it is assumed $k = 12$). To simplify the regression model the problem of approximating the vector-valued function $g: X \to Y$ is decomposed into a set of problems of approximating the scalar-valued functions $g_t: X \to Y_t$, $t = 1, 2, \ldots, n$. Now instead of multivariate linear regression model the multiple linear regression models can be used, one for each component of $\mathbf{y}$.

The idea of the proposed pattern-based linear regression in Fig. 2 is presented and summarized in the following steps:

1. Mapping the original time series elements to patterns $\mathbf{x}$ and $\mathbf{y}$ using (1) and (2).
2. Selection of the *k* nearest neighbors of the query pattern $\mathbf{x}^*$ and creation of the training set $\Phi = \{(\mathbf{x}_i, y_{i,t})\}$, where $\mathbf{x}_i$ are the nearest neighbors of $\mathbf{x}^*$ representing the same day of the week as $\mathbf{x}^*$.
3. Construction of the linear regression model *M* mapping $X \to Y_t$ based on Φ.
4. Determination of the forecasted $\widehat{y}_t$ value for $\mathbf{x}^*$ using *M*.
5. Decoding $\widehat{y}_t$ to get the forecast $\widehat{L}_t$ using (3).

In step 1 the load time series is preprocessed. The input and output patterns are determined for the successive daily periods, up to the current period. Pattern $\mathbf{x}_i$ represents *i*th daily period from the history, and $\mathbf{y}_i$ paired with it represents $(i+\tau)$th daily period. The query pattern $\mathbf{x}^*$ represents the current daily period. When the day type of $\mathbf{x}^*$ is δ and we forecast the load at hour *t* for the next day, we find *k* patterns $\mathbf{x}_i$ most similar to pattern $\mathbf{x}^*$ (step 2). We include these *x*-patterns and *t*th components of paired with them *y*-patterns into the training set Φ. Then, in step 3 the linear model is built. The estimation of the model parameters and model optimization procedures (described in Sections 3.1–3.5) are performed

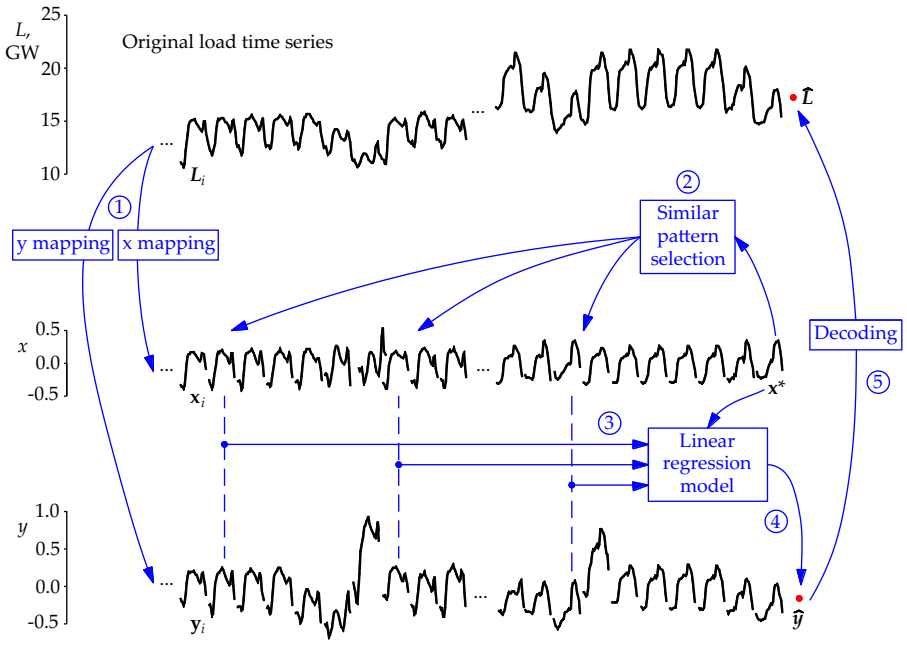

Fig. 2. The flowchart of pattern-based linear regression for STLF.

on the training set Φ. The response of the model to the query pattern $\mathbf{x}^*$ is the forecasted value of the y-pattern tth component $\widehat{y}_t$ (step 4). In step 5 to get the forecasted load at hour t for the next day we decode $\widehat{y}_t$ according to (3). We use for this current, known parameters of the time series: $\bar{L}_i$ and $\sqrt{\sum_{l=1}^{n}\left(L_{i,l}-\bar{L}_i\right)^2}$.

The proposed linear models do not include exogenous variables such as weather factors or price of electricity. To take them into account additional model can be built that corrects load forecast generated by the base linear model depending on the exogenous variables. This is the subject of future work. An example of such approach is presented in [10]. Atypical days such as public holidays are not handled for the proposed models. This is because there is often no information about atypical daily curve of the forecasted day in the previous day which is represented by the x-pattern. So the proposed forecasting models cannot predict atypical y-pattern using this x-pattern as input.

3.1. Multiple linear regression

The multiple linear regression (MLR [20]) model for STLF is of the form:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \varepsilon, \tag{4}$$

where $\beta_0, \beta_1, \ldots, \beta_n$ are coefficients and ε is a random disturbance or error.

The coefficients are estimated using least-squares fit. Notice that in the local approach the number of points used to build a model (k) can be less than their dimensionality and the number of free parameters of the model. In such a case the model is oversized: it has too many degrees of freedom in relation to the problem complexity expressed by only a few training points. In m-dimensional space (in model (4) $m = n+1$), we need at least m points to define a hyperplane. When $m > k$ we get an infinite number of solutions of regression model (4), i.e. the least squares coefficients β_j are not uniquely defined.

It is worth notice also that the components of x-patterns representing subsequent elements of time series are usually strongly correlated (see Fig. 3). Correlations between predictors indicate

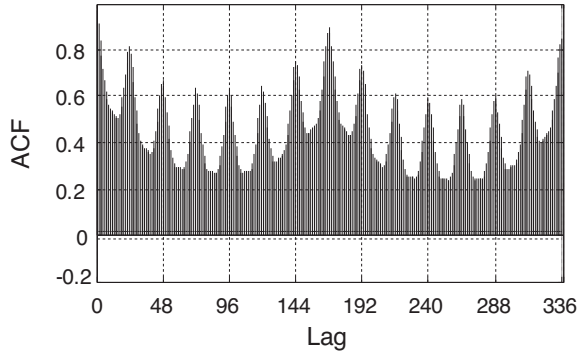

Fig. 3. The autocorrelation function of the load time series of the Polish power system.

that some of them are linear combination of others (multicollinearity). Building model on collinear predictors leads to imprecise estimate of coefficients and missing importance of predictors. If predictors carry similar information about the response variable, some of them can be ignored. To select the most informative predictors the stepwise and lasso regression procedures described in Sections 3.2 and 3.3 are used, respectively. Another cure for collinearity is the ridge regression which reduces the absolute value of coefficients. As a result their estimates have lower variance which may lead to better prediction. However this does not lead to dimensionality reduction. Yet another way to deal with collinearity and excessive dimensionality is the creation of new predictors combining the original ones. Two ways of extraction of predictors: principal component regression and partial least-squares regression are presented in Sections 3.4 and 3.5, respectively. New predictors are uncorrelated and to explain the variability of the response less predictors is needed.

3.2. Stepwise regression

Stepwise regression [21] is a method for selection of the best subset of predictors to the linear model by adding and removing

predictors in a systematic way. The linear model in this case is of the form:

$$y = \beta_0 + w_1\beta_1 x_1 + \cdots + w_n\beta_n x_n + \varepsilon, \tag{5}$$

where $w_1, \ldots, w_n$ are binary weights.

The goal is to find the binary weight vector $\mathbf{w} = [w_1\ w_2\ \ldots\ w_n]^T$ and the coefficient values β_j for predictors with nonzero weights. The criterion of adding or removing the predictor is based on the p-value for an F test of the change in sum of squared error. When we analyze the introduction of the tth predictor to the model, the null hypothesis is that this predictor would have a zero weight if added to the model. If the null hypothesis is rejected, the predictor is added to the model ($w_t = 1$). When we analyze removing of the tth predictor from the model, the null hypothesis is that this predictor has a zero weight. If there is insufficient evidence to reject the null hypothesis, the predictor is removed from the model ($w_t = 0$).

The selection procedure starts with empty set of predictors in the model. At each step it adds the predictor with the smallest p-value until there is no predictors having p-value less than an entrance tolerance (0.05 was used). Then the elimination procedure is activated which removes from the model the predictor with the largest p-value, if it is greater than an exit tolerance (0.1). The selection and elimination procedures are repeated alternately until there is no predictor for removal.

Some other criteria can be used to add and remove predictors like [20]: R-squared, Mallows C_p, Akaike information criterion or Bayes information criterion. The solutions generated by the stepwise regression are suboptimal and there is no guarantee that a different initial model or different sequence of steps will not lead to a better model.

3.3. Regularized least-squares regressions

In the regularized least-squares regression the minimized criterion is composed of the usual regression criterion and a penalty term dependent on the coefficient values. Thus the coefficients are shrunk toward zero. This can greatly reduce the variance, resulting in a better mean-squared error. The ridge regression estimates coefficients by minimizing the criterion containing sum of squared coefficients as a penalty term [22]:

$$\widehat{\beta} = \underset{\beta}{\operatorname{argmin}} \left(\sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{t=1}^{n} x_{i,t}\beta_t \right)^2 + \lambda \sum_{t=1}^{n} \beta_t^2 \right), \tag{6}$$

where $\lambda \geq 0$ is a parameter that controls the amount of shrinkage.

The large value of λ leads to more shrinkage. We get different coefficient estimates for different values of λ. But the coefficient values are never set to zero exactly, and therefore cannot perform predictor selection in the linear model.

The alternative way of regularization is lasso (least absolute selection and shrinkage operator). This is a shrinkage method like ridge, with subtle but important difference: the penalty term in lasso is a sum of absolute values of coefficients. The lasso estimate is defined as [22]:

$$\widehat{\beta} = \underset{\beta}{arg\,\min} \left(\frac{1}{2} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{t=1}^{n} x_{i,t}\beta_t \right)^2 + \lambda \sum_{t=1}^{n} |\beta_t| \right). \tag{7}$$

The nature of penalty term in lasso causes some coefficients to be shrunken to zero exactly, thus the lasso is able to perform predictor selection in the linear model. As λ increases more and more coefficients are set to zero (see Fig. 6, right). In the experimental part of the work the value of λ in the ridge and lasso regressions was tuned in the leave-one-out procedure.

3.4. Principal components regression

Principal component regression (PCR) [20,22] produces new predictors (principal components) which are linear combinations of the original ones and are linearly uncorrelated. The first principal component has the largest sample variance. Subsequent principal components have the highest variances possible under the constraint that they are orthogonal to the preceding components. The principal components are used in place of the original predictors in the regression model:

$$y = \beta_0 + \beta_1 z_1 + \cdots + \beta_c z_c + \varepsilon, \tag{8}$$

where z_j is the jth principal component, $c \leq n$ is the numbers of components included into the model.

There is no need to use all principal components in the model but only the first few ones (c) because usually they explain most of the variability in the response variable. So the components with the lowest variance can be discarded.

3.5. Partial least-squares regression

Partial least-squares regression (PLSR) has some relationship to PCR. It also constructs new predictors by linear combination of original ones, but unlike PCR it uses also the response variable to do so. The new predictors called latent variables are the best orthogonal linear combinations of x_t for predicting y (they explain best the response). PLSR searches for such orthogonal directions to project x-points that have the highest variance and highest correlation with the response. The number of predictors used in the final model is a parameter of PLSR like in PCR. These both methods are useful when there is more predictors than observations and when there is multicollinearity among predictors. The algorithms of partial least-squares and connections between PLSR and PCR can be found in [23].

4. Simulation examples

In the first example the proposed linear models are examined in the tasks of load forecasting of the Polish power system for the next day ($\tau = 1$). The hourly load time series is from the period 2002–2004 (see Fig. 4; these data can be downloaded from the website http://gdudek.el.pcz.pl/varia/stlf-data). The test samples are from January 2004 (without atypical 1 January; atypical days such public holidays are not handled for the proposed methods) and July 2004, i.e. we forecast loads in the successive days of January and July. Models are constructed using 12 nearest neighbors of the query point $\mathbf{x}^*$ from the history, i.e. they are selected from the period from 1 January 2004 until the day before the day of the forecast. The Euclidean distance is used to select the nearest x-patterns. For each hour of the day of forecast a separate model is built. So for our test samples $(30+31)\cdot 24 = 1464$ models were constructed. Because $m > k$ ($m = 25$, $k = 12$) before using model (4) ten predictors were selected using stepwise regression (only with selection procedure and without elimination procedure). The algorithm starts with empty set of predictors and it adds at each step the predictor with the smallest p-value until the number of predictors reaches 10. The similar approach for reducing the initial number of predictors in the ridge and lasso regressions was used.

The frequencies of predictors selected in the stepwise and lasso regressions and the predictor number frequencies in Fig. 5 are shown. Among all 24 predictors included in x-patterns the most often selected predictor represents the load at hour 24. It means that this predictor, which is the nearest in time to the forecasted variable among all predictors, carries much information about this variable. Fig. 3 suggests that good candidate as a predictor could be that one with lag 168 having high value of autocorrelation function.

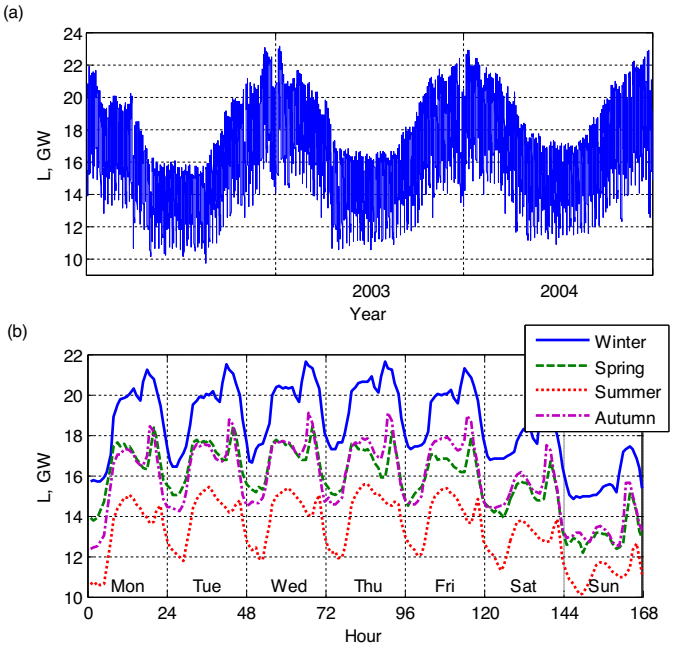

Fig. 4. The hourly electricity demand in Poland in three-year (a) and one-week (b) intervals.

But in the proposed approach *x*-patterns were defined on the basis of the last daily period from the history.

The final models of stepwise regression most often were based on only one predictor, and in many cases they included only an intercept. In the case of lasso in over 30% of cases the final model included only an intercept. In such a case for some query pattern $\mathbf{x}^*$ we get the forecast as $\widehat{y} = \beta_0$. It means that the *y*-values of the nearest neighbors of $\mathbf{x}^*$ are similar to each other, i.e. the approximating hyperplane is parallel to all *x*-axis of the coordinate plane. Remember that this liner model is valid only for this query point. For another query point we determine another set of neighbors and the hyperplane changes.

In Fig. 6, the ridge and lasso traces (simultaneous graph of the regression coefficients plotted against parameter λ) for one of the forecasting task are shown. Lasso selects three predictors in this case: x_7, x_{10} and x_{18} at the optimal value of $\lambda = 0.0092$.

A small number of predictors selected in stepwise and lasso regressions confirms the assumption that the same information about the response variable is repeated in many predictors. Thus there is no sense to generate many of orthogonal components in PCR and PLSR models. The preliminary tests performed on the different load time series were shown that although the training error decreases with the number of principal components, the test error increases, in general. So the number of components was limited to

Table 1
Forecast errors and their interquartile ranges in the first example.

Linear model	January		July		Average	
	$MAPE_{tst}$	IQR_{tst}	$MAPE_{tst}$	IQR_{tst}	$MAPE_{tst}$	IQR_{tst}
MLR	2.37	2.44	2.63	2.42	2.50	2.45
Stepwise	1.52	1.44	1.14	1.20	1.33	1.28
Ridge	1.59	1.50	1.23	1.23	1.41	1.29
Lasso	1.51	1.39	1.06	1.02	1.28	1.18
PCR	1.36	1.21	0.94	0.99	1.15	1.09
PLSR	1.18	1.29	1.00	1.03	1.09	1.14

only one, but it is worth remembering that this new component compresses information extracted from all original predictors. The local regressions using PCR and PLSR for one of the forecasting task in Fig. 7 are shown. In this case MAPE (Mean Absolute Percentage Error) for PCR was 1.15 and for PLSR was 1.49.

In Table 1 the forecast errors (MAPE = 100·mean(|(forecast – actual value)/actual value|)) for the test samples are presented. MAPE is traditionally used as an error measure in STLF. As a measure of error dispersion interquartile ranges (IQR) were used. As we can see from this table the lowest errors were achieved by PLSR and PCR. It is noteworthy the for MLR errors are about twice larger than for other models. The density functions of percentage errors (PE = 100·(forecast – actual value)/actual value) are presented in Fig. 8. These functions for PLSR and PCR are very similar as well as for stepwise and lasso regressions.

In the second example best linear models, PCR and PLSR, were examined in the STLF problems on four time series:

- PL: time series of the hourly load of the Polish power system from the period of 2002–2004 (this time series was used in the first example). The test sample includes data from 2004 with the exception of 13 atypical days (e.g. public holidays),
- FR: time series of the half-hourly load of the French power system from the period of 2007–2009. The test sample includes data from 2009 except for 21 atypical days,
- GB: time series of the half-hourly load of the British power system from the period of 2007–2009. The test sample includes data from 2009 except for 18 atypical days,
- VC: time series of the half-hourly load of the power system of Victoria, Australia, from the period of 2006–2008. The test sample includes data from 2008 except for 12 atypical days.

Our models were compared with other popular models of STLF: ARIMA, exponential smoothing (ES) and multilayer perceptron (MLP) as well as with the nonparametric regression model: Nadaraya–Watson estimator (N–WE).

In ARIMA and ES the time series were decomposed into *n* series, i.e. for each *t* a separate series was created. This eliminates the daily seasonality and simplifies the forecasting problem. The ARIMA and

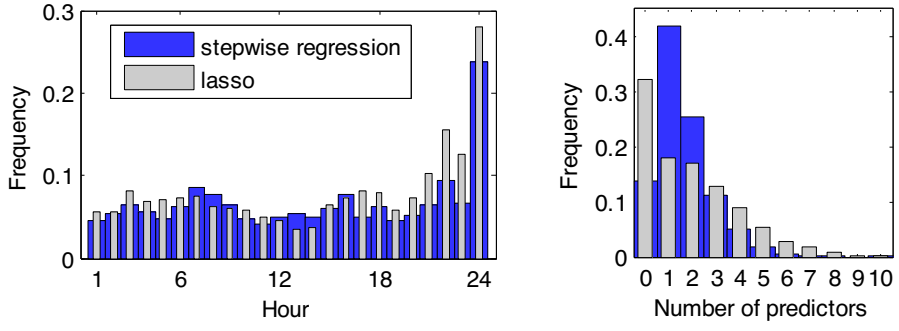

Fig. 5. The frequencies of predictors (left) and the frequencies of the predictor numbers (right) in stepwise and lasso regressions.

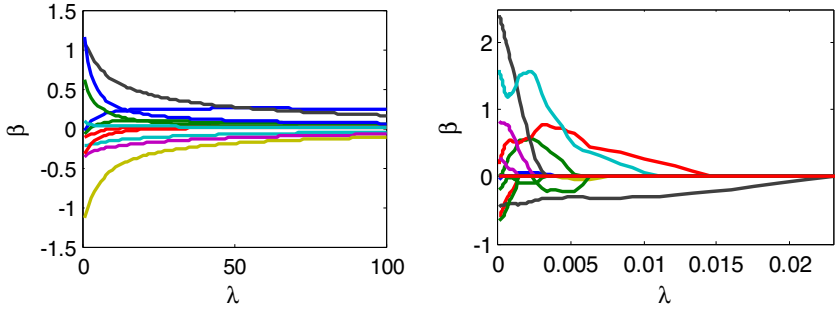

Fig. 6. The ridge (left) and lasso (right) traces for the forecasting task of July 1, 2004, hour 12.

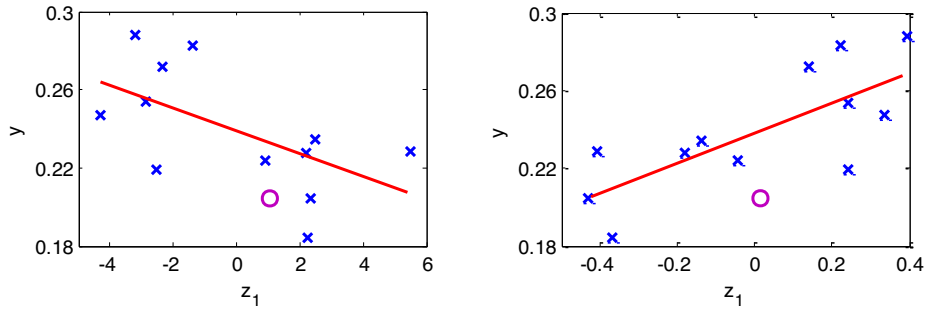

Fig. 7. The local regressions using PCR (left) and PLSR (right) for the forecasting task of July 1, 2004, hour 12 (***o***–query point).

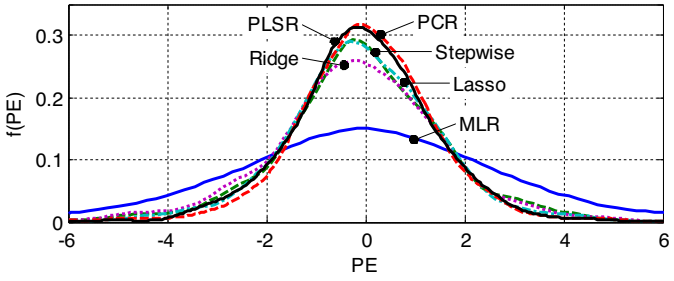

Fig. 8. The probability density functions of percentage errors.

ES parameters were estimated for each forecasting task (forecast of system load at time t of day i) using 12-week time series fragments immediately preceding the forecasted day. Atypical days in these fragments were replaced with the days from the previous weeks. Due to using short time series fragments for parameter estimation (much shorter than the annual period) and due to time series decomposition into n series we do not have to take into account the annual and daily seasonality in the models. In such a case the number of the parameters is much smaller and they are easier to estimate compared to models with triple seasonality.

For each forecasting task the seasonal ARIMA$(p, d, q) \times (P, D, Q)_v$ model was created (where the period of the seasonal pattern appearing $v = 7$, i.e. one week period) as well as the ES state space model. ES models are classified into 30 types [24] depending on how the seasonal, trend and error components are taken into account (they can be expressed additively or multiplicatively, and the trend can be damped or not). To estimate parameters of ARIMA and ES stepwise procedures for traversing the model spaces implemented in the forecast package for the R environment for statistical computing [25] were used. These automatic procedures return the optimal models with the lowest Akaike information criterion value.

The MLP model is learned locally [26] using training patterns selected from the neighborhood of the query pattern. These are the same 12 patterns that are used to construct the proposed linear models. For each forecasting task a separate MLP is learned. To prevent overfitting MLP is learned using Levenberg–Marquardt algorithm with Bayesian regularization [27]. Since the target function is approximated locally using a small number of learning points, rather a simple form of this function should be expected. This implies small number of neurons. Based on the research reported in [26] the network composed of only one neuron with bipolar sigmoid (or hyperbolic tangent) activation function was chosen as an optimal architecture.

The pattern-based STLF model using Nadaraya–Watson estimator was proposed in [28]. This is a representative of pattern-similarity based forecasting models [29]. The model parameters: smoothing parameters or bandwidts $h_1, h_2, \ldots, h_n$ are estimated in the grid search procedure using the same training sample as in linear models and MLP. In the grid search the starting point is determined using the Scott's rule and then the neighborhood of this point is searched in the iteration process (see [28] for details). To avoid overfitting the model was optimized using leave-one-out cross-validation.

In Table 2, errors for one day ahead load forecasting are presented. The errors generated by the naïve model of the form: the forecasted daily curve is the same as seven days ago, are also shown in this table. The best results are marked with an asterisk and the second best results are marked with a double asterisk (best results were confirmed by Wilcoxon rank sum test with 5% significance

Table 2
Forecast errors and their interquartile ranges ($MAPE_{tst}/IQR_{tst}$) in the second example.

Model	PL	FR	GB	VC
PCR	1.35/1.33**	1.71/1.78	1.60/1.68**	3.00/2.70
PLSR	1.34/1.32**	1.57/1.61*	1.54/1.61*	2.83/2.60**
ARIMA	1.82/1.71	2.32/2.53	2.02/2.07	3.67/3.42
ES	1.66/1.57	2.10/2.29	1.85/1.84	3.52/3.35
MLP	1.44/1.41	1.64/1.70**	1.65/1.70**	2.92/2.69
N–WE	1.30/1.30*	1.66/1.67	1.55/1.63*	2.82/2.56*
Naïve	3.43/3.42	5.05/5.96	3.52/3.82	4.88/4.55

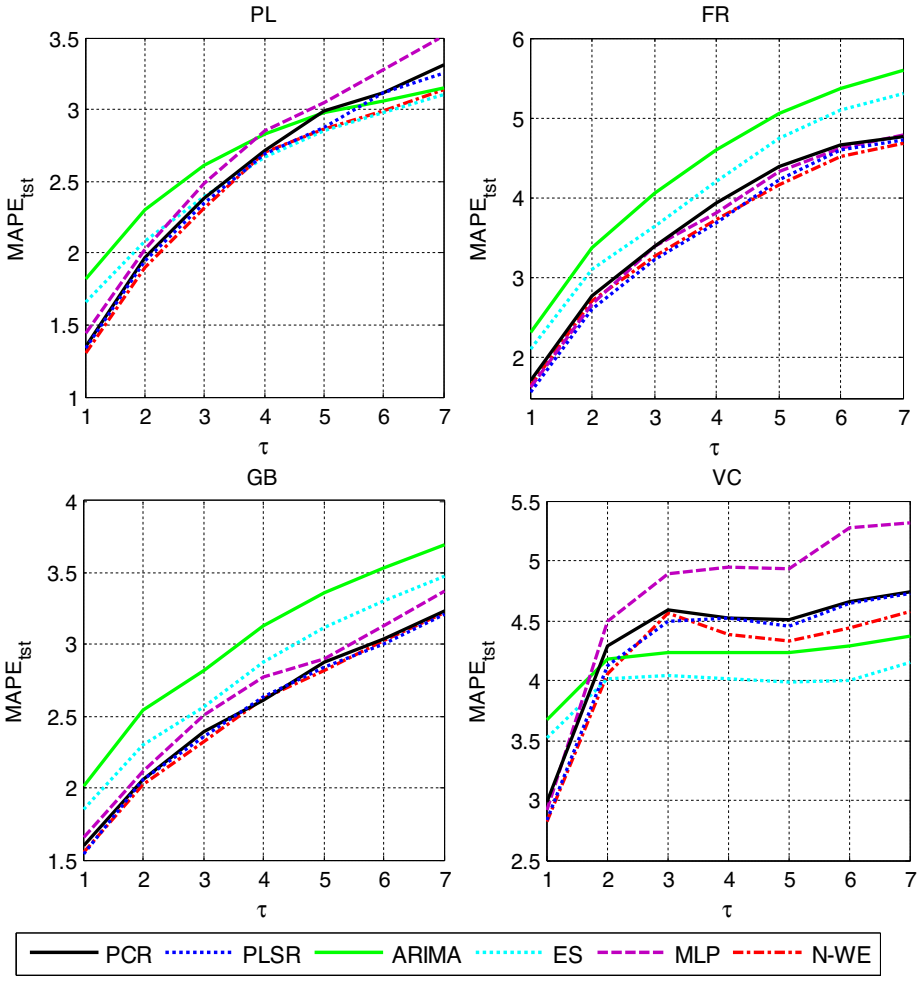

Fig. 9. Errors for different forecast horizons.

level). As we can see from this table PLSR takes the first place among tested models for FR and GB data and second place for PL and VC data. Note that N–WE generates the best results for three datasets, but the difference in errors between this model and PLSR is small except FR data, where PLSR is better. The conventional forecasting models: ARIMA and ES work significantly worse than pattern-based models. To see how PCR and PLSR work on more recent data they were tested on time series of hourly load of the Polish power system from the period of 2012–2014 (test sample includes data from 2014 with the exception of 14 atypical days). The results for PLSR did not differ from those for PL data: $MAPE_{tst} = 1.34$. For PCR results were a little worse: $MAPE_{tst} = 1.44$.

In Fig. 9, the errors for forecast horizons up to 7 days are compared. For longer horizons the linear regression models generated good results compared to the reference models. For VC data and horizons more than two days the conventional models ARIMA and ES outperformed other models.

In Fig. 10, the time efficiency of the forecasting models is compared. The times presented in this figure include time of the model optimization and forecasting for all hours of the next day for PL data (computing environment: Intel Core 2 Quad CPU Q9550 2.83 GHz, 4 GB RAM, Matlab R2012b). In the MLR, ridge and lasso regressions, where stepwise regression is used first to reduce the initial number of predictors, the algorithm spends most of the time in stepwise phase (about 62 s). The most time efficient models are PLSR and PCR. When using these models the process of model building and forecasting for 24 h of the next day takes less than half of a second.

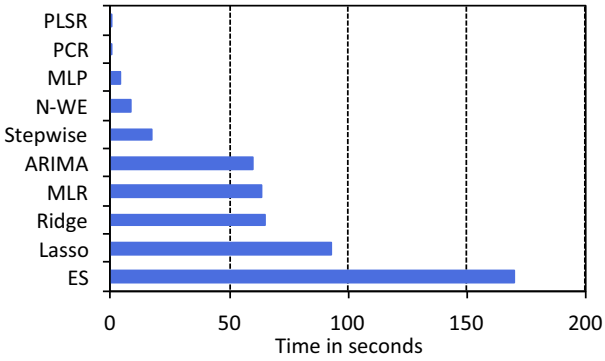

Fig. 10. The total time of building the forecasting models for 24 h of the next day for PL data.

5. Conclusions

The major contribution of this work is to propose new simple univariate linear regression models based on patterns of daily cycles for STLF. Patterns allows the forecasting problem to be simplified by filtering out the trend, annual and weekly cycles. The relationship between input and output patterns is approximated locally in the neighborhood of the query pattern using linear regression. Thus we resign from the global modeling of the target function in the entire range creating the locally competent model for the region around the query point. Since the local complexity is lower

than the global one, we can use a simple model. This model brings good results for the current query point, but we have to construct new models for other query points.

The similar approach based on patterns and local modeling was used earlier in other STLF models: MLP and N–WE. Although these models are nonlinear, the proposed linear models have better extrapolation property. Because the linear models are not as flexible as neural networks and nonparametric regression models there is no problem with overfitting. The cumbersome and time-consuming procedures to prevent overfitting are unnecessary. In the application examples the STLF methods based on patterns and local modeling outperform conventional models: ARIMA and exponential smoothing especially for shorter horizons.

Using principal component regression or partial least-squares regression the number of predictors can be reduced to only one which allows us to visualize the regression function. In this case the models have only two parameters simply estimated using least-squared approach. This is a great advantage in comparison to the complex STLF models based on ARIMA, exponential smoothing, neural and neuro-fuzzy networks or SVM, where there are dozens or hundreds of parameters and their estimation requires advanced optimization methods.

Acknowledgments

I am very grateful to James W. Taylor with the Saïd Business School, University of Oxford for supplying British and French data, and Shu Fan and Rob J. Hyndman with the Business and Economic Forecasting Unit, Monash University for supplying Australian data.

References

[1] R. Weron, Modeling and Forecasting Electricity Loads and Prices, Wiley, Chichester, 2006.
[2] J.W. Taylor, R.D. Snyder, Forecasting intraday data with multiple seasonal cycles using parsimonious seasonal exponential smoothing, Omega 40 (6) (2012) 748–757.
[3] J.W. Taylor, Short-term load forecasting with exponentially weighted methods, IEEE Trans. Power Syst. 27 (1) (2012) 458–464.
[4] J. Nowicka-Zagrajek, R. Weron, Modeling electricity loads in California: ARMA models with hyperbolic noise, Signal Process. 82 (2002) 1903–1915.
[5] C.-M. Lee, C.-N. Ko, Short-term load forecasting using lifting scheme and ARIMA models, Expert Syst. Appl. 38 (2011) 5902–5911.
[6] H.S. Hippert, J.W. Taylor, An evaluation of Bayesian techniques for controlling model complexity and selecting inputs in a neural network for short-term load forecasting, Neural Netw 23 (2010) 386–395.
[7] N. Amjady, F. Keynia, Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm, Energy 34 (2009) 46–57.
[8] Y. Chen, P.B. Luh, C. Guan, Y. Zhao, L.D. Michel, M.A. Coolbeth, P.B. Friedland, S.J. Rourke, Short-term load forecasting: similar day-based wavelet neural networks, IEEE Trans. Power Syst. 25 (1) (2010) 322–330.
[9] Hao Quan, D. Srinivasan, A. Khosravi, Short-term load and wind power forecasting using neural network-based prediction intervals, IEEE Trans. Neural Netw. Learn. Syst. 25 (2) (2014) 303–315, http://dx.doi.org/10.1109/TNNLS.2013.2276053.
[10] Z. Yun, Z. Quan, S. Caixin, L. Shaolan, L. Yuming, S. Yang, RBF neural network and ANFIS-based short-term load forecasting approach in real-time price environment, IEEE Trans. Power Syst. 23 (3) (2008) 853–858.
[11] M. Hanmandlu, B.K. Chauhan, Load forecasting using hybrid models, IEEE Trans. Power Syst. 26 (1) (2011) 20–29.
[12] H. Mao, X.-J. Zeng, G. Leng, Y.-J. Zhai, J.A. Keane, Short-term and midterm load forecasting using a bilevel optimization model, IEEE Trans. Power Syst. 24 (2) (2009) 1080–1090.
[13] D.K. Chaturvedi, A.P. Sinha, O.P. Malik, Short term load forecast using fuzzy logic and wavelet transform integrated generalized neural network, Int. J. Electr. Power Energy Syst. 67 (2015) 230–237.
[14] Q. Wu, Power load forecasts based on hybrid PSO with Gaussian and adaptive mutation and Wv-SVM, Expert Syst. Appl. 37 (2010) 194–201.
[15] E. Ceperic, V. Ceperic, A. Baric, A Strategy for short-term load forecasting by support vector regression machines, IEEE Trans. Power Syst. 28 (4) (2013) 4356–4364.
[16] M. De Felice, Short-term load forecasting with neural network ensembles: a comparative study, IEEE Comput. Intell. Mag. 6 (3) (2011) 47–56.
[17] Rui Zhang, Zhao Yang Dong, Yan Xu, Ke Meng, Kit Po Wong, Short-term load forecasting of Australian National Electricity Market by an ensemble model of extreme learning machine, IET Gener. Transm. Distrib. 7 (4) (2013) 391–397.
[18] G. Dudek, Artificial immune system for forecasting time series with multiple seasonal cycles, Trans. Comput. Collec. Intell. XI LNCS 8065 (2013) 176–197.
[19] G. Dudek, Pattern similarity-based methods for short-term load forecasting—Part 1: Principles, Appl. Soft Comput. 37 (2015) 277–287, http://dx.doi.org/10.1016/j.asoc.2015.08.040, ISSN 1568-4946.
[20] S. Chatterjee, A.S. Hali, Regression Analysis by Example, John Wiley & Sons, Inc., Hoboken, New Jersey, 2006.
[21] N. Draper, H. Smith, Applied Regression Analysis, Wiley, New York, 1981.
[22] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning. Data Mining, Inference, and Prediction, Springer-Verlag, New York, 2009.
[23] R. Rosipal, N. Kramer, Overview and recent advances in partial least squares Subspace, Latent Structure and Feature Selection LNCS, 3940, Springer Berlin Heidelberg, 2006, pp. 34–51.
[24] R.J. Hyndman, A.B. Koehler, J.K. Ord, R.D. Snyder, Forecasting with Exponential Smoothing: The State Space Approach, Springer, - Springer series in statistics, Berlin/London, 2008.
[25] R.J. Hyndman, Y. Khandakar, Automatic time series forecasting: the forecast package for R, J. Stat. Softw. 27 (3) (2008) 1–22.
[26] G. Dudek, Forecasting time series with multiple seasonal cycles using neural networks with local learning Artificial Intelligence and Soft Computing, ICAISC 2013, LNCS, vol. 7894, Springer Berlin Heidelberg, 2010, pp. 52–63.
[27] F.D. Foresee, M.T. Hagan, Gauss-Newton approximation to Bayesian regularization, in: Proc. Inter. Joint Conference on Neural Networks, 1997, pp. 1930–1935.
[28] G. Dudek, Short-term load forecasting based on kernel conditional density estimation, Przegl. Elektrotech. 86 (8) (2010) 164–167.
[29] G. Dudek, Pattern-Similarity Machine Learning Models for Short-Term Load Forecasting, Academic Publishing House “Exit”, Warsaw, 2012 (in Polish).