Applied Soft Computing Journal 104 (2021) 107223

Contents lists available at ScienceDirect

Applied Soft Computing Journal

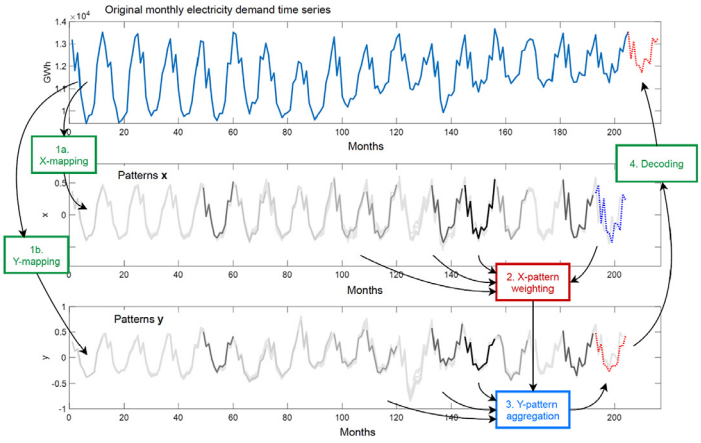
journal homepage: www.elsevier.com/locate/asoc

Pattern similarity-based machine learning methods for mid-term load forecasting: A comparative study

Grzegorz Dudek *, Paweł Pełka

Czestochowa University of Technology, Faculty of Electrical Engineering, 17 Armii Krajowej Ave., 42-200 Czestochowa, Poland

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 28 November 2019
Received in revised form 30 December 2020
Accepted 20 February 2021
Available online 27 February 2021

Keywords:
Pattern similarity-based forecasting models
Mid-term load forecasting
Time series representation

ABSTRACT

Pattern similarity-based frameworks are widely used for classification and regression problems. Repeated, similar-shaped cycles observed in seasonal time series encourage the use of such frameworks for forecasting. In this paper, we use pattern similarity-based models for mid-term load forecasting. An integral part of these models is the use of patterns of time series sequences for time series representation. Pattern representation ensures input and output data unification through trend filtering and variance equalization. This simplifies the forecasting problem and allows us to use models based on pattern similarity. We consider four such models: nearest-neighbor model, fuzzy neighborhood model, kernel regression model, and general regression neural network. Three variants of the approach were proposed. A basic one and two hybrid solutions combining similarity-based and statistical methods (ARIMA and exponential smoothing).

In the experimental part of the work, the proposed models were used to forecast the monthly electricity demand in 35 European countries. The results show the high performance of the proposed models, which outperform both the comparative classical statistical models and machine learning models in terms of accuracy, simplicity, and ease of optimization. Among the proposed variants, a hybrid approach combining similarity-based methods with exponential smoothing turned out to be the most accurate. The study highlights the many advantages of the proposed pattern similarity-based models such as clear operation principles, a small number of parameters to adjust, no training procedure, fast optimization procedure, good generalization ability, ability to work on the newest data without retraining, and delivery of multi-step forecasts.

1. Introduction

Mid-term electrical load forecasting (MTLF) is an essential tool for power system operation and planning. It concerns forecasting the monthly electricity demand and the daily peak loads

* Corresponding author.
E-mail address: grzegorz.dudek@pcz.pl (G. Dudek).

https://doi.org/10.1016/j.asoc.2021.107223
1568-4946/

for the succeeding months. The forecast horizon is usually from a week to a year. Mid-term load forecasts are necessary for maintenance scheduling, fuel reserve planning, hydro-thermal coordination, electrical energy import/export planning and security assessment. In deregulated power systems, MTLF is the basis for negotiation of forward contracts. Accurate forecasting translates into better financial performance for energy companies and other energy market participants.

This work focuses on monthly electricity demand forecasting. The time series of monthly electricity demand usually exhibits a trend and an annual seasonality. The trend is dependent on the dynamics of economic development in a country. The seasonal cycles are dependent on the climate, weather factors and the variability of seasons. Factors which can disrupt the time series include political decisions, unpredictable economic events, structural breaks [1] and transitory effects from external variables [2].

1.1. Related work

Methods of MTLF can be roughly classified as either conditional modeling or autonomous modeling approaches [3]. The conditional modeling approach focuses on economic analysis as well as long-term planning and forecasting of energy policy. Socio-economic conditions that affect energy demand in a given region, and population migrations are taken also into account. Economic growth is described by economic indicators, which are introduced as additional inputs to the forecasting model. These indicators include [3,4] gross national product, the consumer price index, exchange rates and average wages. In addition, variables describing the power system and network infrastructure, such as the number and length of transmission lines and the number of high voltage stations, are introduced as inputs. Economic variables have the greatest impact on the trend, while weather variables, due to their seasonal nature, have an impact on the periodic behavior of the monthly electricity demand [5]. Examples of the conditional modeling approach can be found in [6,7] and [8]. In [6] a knowledge-based expert system for a fast developing utility is proposed. It identifies forecasting algorithms and the key variables, both electrical and nonelectrical, that affect demand forecasts. A set of decision rules relating to these variables is then obtained and stored in the knowledge base. Then, the model that reflects most-accurately the typical system behavior is selected to produce the load forecast. Multiple linear regression and ARIMA models for monthly peak load forecasing are proposed in [7]. The inputs of the models are historical series of electric peak load, weather variables and economic variables such as the consumer price index, and industrial index. In [8], a heuristic model was proposed which approximates the relationship between the actual load and four sets of historical data: population, gross national product, consumer price index and temperature. Additionally, the impact of the reserve margin and load diversity factor are taken into consideration before obtaining the final forecast.

In the autonomous modeling approach, the input variables include only historical loads and weather factors. This approach is more appropriate for stable economies, without rapid changes affecting electricity demand. The selection of weather variables, such as [3] atmospheric temperature, humidity, insolation time, wind speed, etc. depends on the local climate and weather conditions [9]. The autonomous models described in [10] use ARIMA and neural networks (NNs) to forecast monthly peak loads. Input variables include load profiles, weather factors (temperature and humidity) and time index. The model described in [9] uses historical demand and atmospheric temperatures as input variables. Variables expressing the seasons are also introduced. In many cases, autonomous models are simplified using only historical load data as input. Such an approach was used in [5], where the trend of a series of monthly loads was forecasted only on the basis of loads from the previous twelve months. The seasonal component was modeled by Fourier series. In [11], a fuzzy NN model based only on weather variables (atmospheric pressure, temperature, wind speed, etc.), without taking into account historical loads as input variables, was used.

Another categorization of MTLF methods is based on forecasting models which can be classical statistical/econometrics models or artificial intelligence/machine learning (AI/ML) models [12]. Typical examples of the former are ARIMA, linear regression (LR) and exponential smoothing (ETS). The implementation of seasonal cycles in the LR models requires additional operations, such as decomposition of the series into individual month series. In [13], a LR model extended with periodic components implemented by the sine functions of different frequencies was proposed for a nonstationary time series with an irregular periodic trend. Another example of using LR for forecasting power system loads can be found in [14]. This model uses strong daily and yearly correlations to forecast daily load profiles over a period of several weeks to several years. The forecast results are corrected by annual load increases. In [7], the performance of the LR and ARIMA models was compared for the task of forecasting monthly peak loads up to 12 months ahead. The models use the same set of input variables including historical peak load data, weather and economic data. The end result was that ARIMA proved to be about twice as accurate as LR.

The limited adaptability of classical MTLF methods and problems with modeling nonlinear relationships have led to increased interest in AI and ML methods [5]. The most explored ML models in MTLF are NNs. This is due to such attractive features as nonlinear modeling, learning capabilities, universal approximation property and massive parallelism. In [15], two separate NNs are used to forecast the trend of a monthly load time series and seasonal fluctuations. In [16], a NN predicts the future monthly loads on the basis of historical loads and weather variables. To improve learning capability, the NN is trained using a gravitational search algorithm and cuckoo optimization algorithm. An example of using a Kohonen NN for MTLF can be found in [4]. The authors built 12 forecasting networks, one for each month of the year. The input vectors contained historical loads and microeconomic indicators. The NNs proposed in [9] are supported by fuzzy logic and seasonal variables are defined in the form of trapezoidal indicators of the season. The authors train a set of NNs using regularization techniques to prevent overfitting, and aggregate their responses, which results in more accurate forecasts. Other examples of ML models for MTLF are: [11] where a weighted evolving fuzzy NN for monthly electricity demand forecasting was used, [17] where NNs, LR and AdaBoost were used for energy forecasting, [18] where a support vector machine was used, and [19] where a long short-term memory network model was used.

Many of the MTLF methods mentioned above need decomposition of the load time series to deal with the trend and seasonal variations. A typical approach is to decompose the time series into trend, seasonal, and stochastic components. The components expressing less complexity than the original time series can be modeled independently using simpler models. One of the most popular tools for decomposition is STL (seasonal and trend decomposition using Loess) filtering procedure based on a locally weighted polynomial smoother [20]. In [21], STL decomposition was used on the monthly data of total electric energy consumption in various developed and developing countries. The times series were forecasted using ARIMA or ETS, and, additionally, a bootstrap aggregating method was used to improve

accuracy. Another method of decomposition is a wavelet transform which splits up the load time series into subseries in the wavelet domain. A low frequency component called an approximation expresses the trend, while high-frequency components called details express cyclical variations [22]. As an alternative to wavelet decomposition, a Fourier transform, which decomposes the time function into its constituent frequencies, can be used. For example, in [5] the load time series was split into two components: one describing the trend and the other the fluctuations. Then, the fluctuation series was expressed by several Fourier series of different frequencies. Yet another method of load time series decomposition is empirical mode decomposition, which breaks down the time series into so-called intrinsic mode functions. This method of decomposition was used in [23], where to model each of the extracted intrinsic mode functions a deep belief network was used.

1.2. Summary of contribution

The goals of this work are to present a general framework of pattern similarity-based forecasting and to compare various pattern similarity-based forecasting methods (PSFMs) used for MTLF. This work generalizes and summarizes our previous works on PSFMs applied to MTLF [24–26], as well as to short-term load forecasting (STLF) [27,28].

PSFMs, as an alternative to the classical and state-of-the-art ML methods, have many attractive features, useful for forecasting problems. A similarity-based learning framework, as a generalization of the minimal distance methods [29], is extremely practical. This is the basis of many ML and pattern recognition methods used for classification, clustering and regression. The repeated, similar-shaped cycles observed in seasonal time series have encouraged us to apply these methods to forecasting. To do so, first we define the patterns expressing the preprocessed repetitive sequences in a time series. Pattern representation ensures input and output data unification through trend filtering and variance equalization. Consequently, no decomposition of the time series is needed. Due to pattern representation the relationship between input and output data is simplified and the forecasting problem can be solved using simple models. We consider four such models: nearest neighbor model, fuzzy neighborhood model, kernel regression model and general regression NN. All these models use a similarity-based learning framework where a regression function is constructed from aggregation output patterns with weights dependent on the similarity between input patterns. The operating principle is very transparent, which is the big advantage these models have over other forecasting models, which are often black boxes. This advantage is especially valuable in practical applications and real-world scenarios as it does not require any specific knowledge of ML and AI from practitioners. Other advantages of similarity-based methods are the small number of parameters to adjust and their robustness to missing and noisy data. Unlike state-of-the-art ML methods, such as NNs and deep learning, they do not suffer from excessive tuning and training burdens.

The contribution of this study includes the following two points:

1. A general framework of pattern similarity-based MTLF is given. The framework includes different time series representations and regression models. It can be applied without modification to a wide range of target domains, including STLF and MTLF, as was confirmed in this study. This framework deals with difficult, challenging forecasting problems with time series expressing multiple seasonal periods, trends and significant random fluctuations. It captures long-term as well as short-term dependencies in the data.
2. This work empirically demonstrates that PSFMs are very effective at solving MTLF problems and outperform in terms of accuracy, simplicity and ease of optimization both well-established statistical models and ML models. Unlike their competitors, PSFMs are simple and their operating principle is transparent, which translates into greater confidence in their forecasts. They have one or two hyperparameters but no parameters, which means the optimization procedure is very simple. PSFM generalization can be controlled easily by a bandwidth hyperparameter. PSFMs do not need retraining when new data arrives as new data points can be immediately added to the training set and are available for the model to produce a forecast. PSFMs producing a vector output are able to perform multi-step forecasting without a recursive approach. Additionally, the sizes of the input and output patterns do not affect the number of parameters or the complexity of the training process as is the case for other ML models.

This paper is organized as follows. Section 2 describes monthly electricity demand time series and their representation using patterns. The idea and framework of pattern similarity-based forecasting are presented in Section 3. Four PSFMs are presented in Section 4. Section 5 describes an experimental framework used to evaluate the performance of PSFMs and comparative models. Finally, in Section 6, we conclude the work.

Abbreviations

AI – artificial intelligence
ANFIS – adaptive neuro-fuzzy inference system
ARIMA – autoregressive integrated moving average model
ETS – exponential smoothing
k-NN, *k*-NNw – *k*-nearest neighbor model and its weighted version
LR – linear regression
LSTM – long short-term memory
ML – machine learning
MLP – multilayer perceptron
MTLF – mid-term load forecasting
N-WE – Nadaraya–Watson estimator
NN – neural network
PSFM – pattern similarity-based forecasting model
STL – seasonal and trend decomposition using Loess
STLF – short-term load forecasting

2. Time series and their representation

A monthly electricity demand time series exhibits a trend, annual cycles and a random component. Fig. 1 shows an example of such a time series. As you can see, in this figure, we can observe a nonlinear trend, which jumps at the end of the fifth cycle, and a changing annual pattern over the years. Additionally, the dispersion of the annual cycles changes significantly over time (from $\sigma = 2610$ to 5588 MWh).

One of the main issues in building forecasting models is how to represent the time series to obtain the highest performance from the model. Input and output variables should be defined on the basis of the original time series. These definitions significantly affect the results. In this work, we use input and output variables as patterns of fragments of the monthly load time series. By a pattern we mean a vector with components that are calculated using some function of actual time series elements. For multiple seasonal time series, this function can filter out any trend and seasonal fluctuations and so simplify the data and relationships between them [27]. As a result, the forecasting model working on patterns can be less complex.

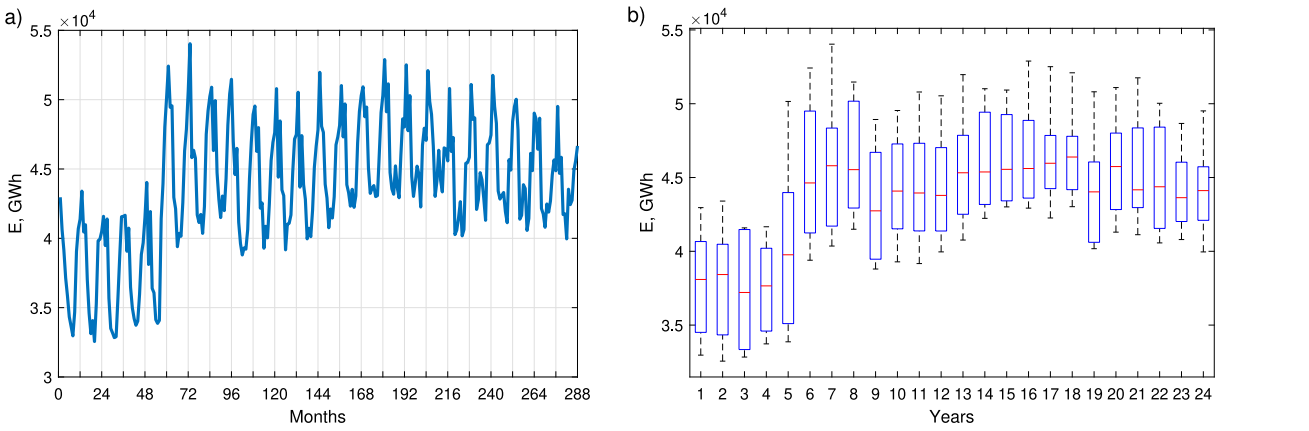


Fig. 1. Monthly electricity demand time series for Germany (a) and its dispersion in successive years (b).

An input pattern $\mathbf{x}_i = [x_{i,1}x_{i,2}\dots x_{i,n}]^T$ of length n is a vector of predictors representing a sequence $X_i = \{E_{i--n+1}, E_{i--n+2}, \dots, E_i\}$ of n successive time series elements E (monthly electricity demands) preceding the forecasted period. The function, which transforms time series elements into patterns, is dependent on the time series properties such as seasonalities, variance and trend. Some definitions of the function mapping the original time series into patterns $\mathbf{x}$ are:

$$x_{i,t} = E_{i-n+t} \tag{1}$$

$$x_{i,t} = E_{i-n+t} - \overline{E}_i \tag{2}$$

$$x_{i,t} = \frac{E_{i-n+t}}{\overline{E}_i} \tag{3}$$

$$x_{i,t} = \frac{E_{i-n+t} - \overline{E}_i}{D_i} \tag{4}$$

where $t = 1, 2, \dots, n$, $\overline{E}_i$ is a mean of sequence X_i, and $D_i = \sqrt{\sum_{j=1}^{n}(E_{i-n+j} - \overline{E}_i)^2}$ is a measure of its dispersion.

Definition (1) just copies sequence X_i into x-pattern without transformation. The pattern components defined using (2) are the differences in demand of a given month and the average demand of sequence X_i. A quotient of these two quantities is expressed by (3). The x-pattern defined by (4) is a normalized version of a vector composed of elements of X_i, i.e. $[E_{i--n+1}E_{i--n+2}\dots E_i]^T$. So, the original time series sequences, which have a different mean and dispersion (see Fig. 1(b)), are unified, and after normalization they are represented by x-patterns which all have zero mean, the same variance and same unity length.

Fig. 2 shows one-year sequences X_i and their x-patterns defined using (2)–(4). Note that all x-pattern definitions boil down the mean of all patterns to the same value (0 or 1) and, additionally, definition (4) boils down the variance of all patterns to the same value. So, the trend is filtered out and patterns differ only in shape.

Output pattern $\mathbf{y}_i = [y_{i,1}y_{i,2}\dots y_{i,m}]^T$ represents a forecasted sequence of length m: $Y_i = \{E_{i+\tau}, E_{i+\tau+1}, \dots, E_{i+\tau+m-1}\}$, where τ is a forecast horizon in months. The output patterns are defined in a similar way to the input patterns:

$$y_{i,t} = E_{i+\tau+t-1} \tag{5}$$

$$y_{i,t} = E_{i+\tau+t-1} - \overline{E}_i^* \tag{6}$$

$$y_{i,t} = \frac{E_{i+\tau+t-1}}{\overline{E}_i^*} \tag{7}$$

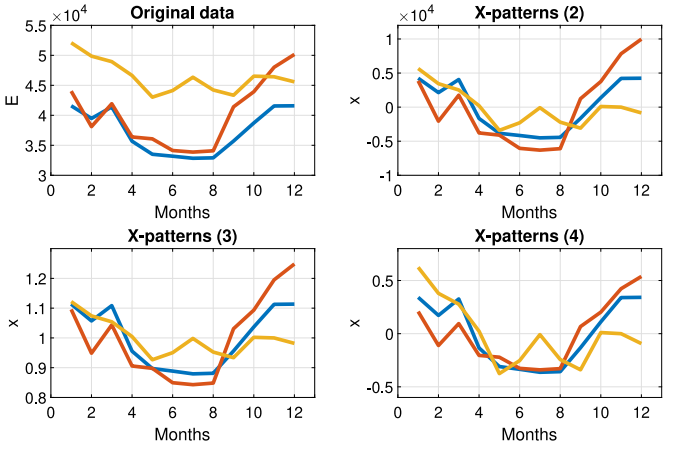


Fig. 2. Three annual periods and their x-patterns (time series for Germany).

$$y_{i,t} = \frac{E_{i+\tau+t-1} - \overline{E}_i^*}{D_i^*} \tag{8}$$

where $t = 1, 2, \dots, m$, and $\overline{E}_i^*$ and D_i^* are coding variables.

The coding variables, $\overline{E}_i^*$ and D_i^*, are the mean and dispersion of the forecasted sequence Y_i, respectively. They are both known from the historical time series, so the training y-patterns can be prepared using them. For a new query x-pattern, the model produces the forecasted y-pattern, $\widehat{\mathbf{y}}$. This pattern corresponds to the forecasted period $Y_q = \{E_{q+\tau}, E_{q+\tau+1}, \dots, E_{q+\tau+m-1}\}$, where q is an index of the last element of sequence $X_q = \{E_{q--n+1}, E_{q--n+2}, \dots, E_q\}$ which is represented by query pattern $\mathbf{x}$. The forecast of the monthly electricity demand in forecasted period Y_q is calculated from $\widehat{\mathbf{y}}$ using transformed Eqs. (5)–(8) (this is known as decoding). For example, when y-patterns are defined using (8), the decoding equation takes the form:

$$\widehat{E}_{q+\tau+t-1} = \widehat{y}_t D_q^* + \overline{E}_q^*, \quad t = 1, 2, \dots, m \tag{9}$$

where $\overline{E}_q^*$ and D_q^* are the mean and dispersion of sequence Y_q.

Unfortunately, the coding variables in (9) are not known, because they are the mean and dispersion of future sequence Y_q, which has been just forecasted. In this case, the coding variables are predicted from their historical values, i.e. in (9), $\overline{E}_q^* = \widehat{\overline{E}}_q^*$ and $D^* = \widehat{D}_q^*$. In the experimental part of the work, the coding variables are predicted using ARIMA (variant V2) and ETS (variant V3).

To avoid forecasting the coding variables we use another approach (denoted as V1). In this approach, instead of using the mean and dispersion of forecasted sequence Y_i as coding variables, we introduce into (6)–(8) the mean and dispersion of sequence X_i, i.e. $\overline{E}_i^* = \overline{E}_i$ and $D_i^* = D_i$. Although this approach does not guarantee that all y-patterns have the same mean value, it unifies output data by taking into account the current process variability, expressed by mean $\overline{E}_i$ and dispersion D_i. When the model returns forecasted y-pattern, the forecast of the monthly demands are calculated from (9) using known coding variables for sequence X_q, i.e. $\overline{E}_q^* = \overline{E}_q$ and $D_q^* = D_q$.

The pairs of corresponding x- and y-patterns form a training set: $\Phi = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, $\mathbf{x}_i \in \mathbb{R}^n$, $\mathbf{y}_i \in \mathbb{R}^m$. Note that the successive pairs represent time series sequences covered by two sliding windows: an input window of the lengths n and an output window of the lengths m. The gap between these two windows is $\tau - 1$. The model learns the mapping x-patterns $\rightarrow$ y-patterns. It generates a forecast of a y-pattern, $\widehat{\mathbf{y}}$, for a query pattern $\mathbf{x}$. The forecasts of the monthly electricity demand in period Y are calculated from the forecasted y-pattern using transformed Eqs. (5)–(8).

Note that when using a pattern approach, the forecasting model works on patterns expressing shapes of the time series sequences. In the first step of this approach, the trend, dispersion and additional seasonal variations are filtered out, depending on the definitions of the patterns. Then, the model forecasts the unified data, i.e. y-patterns on the basis of x-patterns. Finally, in the decoding step, the current trend and dispersion are introduced into the forecasted y-pattern to obtain the forecasted monthly demand.

3. Framework of the pattern similarity-based forecasting

Similarity-based methods are very popular in the field of ML and pattern recognition [29,30]. They estimate the class label or the function value for the query sample based on similarities between this sample and a set of training samples by using some similarity function defined for any pair of samples. The proposed forecasting methods can be classified as memory-based approximation methods which use analogies between preprocessed sequences of the time series (patterns). It is assumed that the future behavior of a time series can be deduced from its behavior in similar conditions in the past. This assumption can be expressed in the pattern representation context as follows [27]:

Assumption 1. If query pattern $\mathbf{x}$ is similar to pattern $\mathbf{x}_i$ from the history, then forecasted pattern $\mathbf{y}$ will be similar to pattern $\mathbf{y}_i$ (paired with $\mathbf{x}_i$).

This assumption allows us to predict the y-pattern on the basis of known patterns $\mathbf{x}$, $\mathbf{x}_i$ and $\mathbf{y}_i$. Usually we select many similar patterns $\mathbf{x}_i$ and aggregate patterns $\mathbf{y}_i$ paired with them to determine forecasted y-pattern.

The above assumption underlying PSFMs should be verified for a given time series. To do so, we analyze the relationship between the similarities of x-patterns and the similarities of y-patterns paired with them. We define two random variables: the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, $s(\mathbf{x}_i, \mathbf{x}_j)$, and the similarity between $\mathbf{y}_i$ and $\mathbf{y}_j$, $s(\mathbf{y}_i, \mathbf{y}_j)$, where $i, j = 1, 2, \ldots, N, i \neq j$. Instead of a similarity measure we can use a distance measure between patterns as random variables. All the pairs of these random variables make up the sample. To show the stochastic interdependence of the random variables, the null hypothesis is formulated: Observed differences in numbers of occurrence of the sample elements in the specified categories of the random variables are caused by the random nature of the sample. This hypothesis is verified using the chi-squared test based on a contingency table showing the joint empirical distribution of the random variables [27]. A high value of χ^2 statistic, above the critical value, rejects the null hypothesis in favor of an alternative hypothesis, which justifies the use of PSFMs.

In this study, the similarity between patterns, which are real-valued vectors, is measured using Euclidean distance. Other measures are also possible such as Pearson's correlation coefficient, Tanimoto coefficient, other Minkowski distances or dot product for normalized vectors.

The concept of pattern similarity-based forecasting is depicted in Fig. 3 and can be summarized in the following steps [27]:

1. Mapping the original time series sequences into x- and y-patterns.
2. Weighting training x-patterns depending on their similarity to query pattern $\mathbf{x}$.
3. Weighted aggregation of training y-patterns to determine forecasted pattern $\widehat{\mathbf{y}}$.
4. Decoding pattern $\widehat{\mathbf{y}}$ to obtain forecasted time series sequence Y.

During aggregation of the y-patterns (step 3) we use weights which are dependent on the similarity between query pattern $\mathbf{x}$ and training x-patterns. In this case, the regression model mapping x-patterns into y-patterns takes the form:

$$m(\mathbf{x}) = \sum_{i=1}^{N} w(\mathbf{x}, \mathbf{x}_i)\mathbf{y}_i \tag{10}$$

where $w(.,.)$ is a weighting function, $\sum_{i=1}^N w(\mathbf{x}, \mathbf{x}_i) = 1$.

Note that $m(.)$ is a vector-valued function returning the whole y-pattern. It is a nonlinear function if $w(.,.)$ maps $\mathbf{x}$ nonlinearly. Different definitions of $w(.,.)$ are given in Section 4 for specified PSFMs.

Algorithm 1 summarizes pattern similarity-based MTLF. As input, the algorithm requires the PSFM which is used in step 3 to weight the input patterns using specific *Weighting_function*. This function returns vector $\mathbf{w} = [w_1 w_2 \ldots w_N]^T$ including weights for the successive training x-patterns. The algorithm variant, V1, V2 or V3, determines the method of y-pattern encoding (step 1) and decoding (steps 4–7). In V1, the coding variables for y-patterns are the same as for their corresponding x-patterns (step 4). In V2, the coding variables are predicted for the forecasted period using ARIMA (step 5), and in V3 they are predicted using ETS (step 6). A flowchart of the forecasting procedure in variant V1 is depicted in Fig. 4(a), and in Fig. 4(b) for variants V2 and V3.

4. Pattern similarity-based forecasting models

4.1. Nearest neighbor models

The simplest PSFM is a k-nearest neighbor regression model. It estimates $m(.)$ as the average of the y-patterns in a varying neighborhood of query pattern $\mathbf{x}$. The neighborhood is defined as a set of k nearest neighbors of $\mathbf{x}$ in the training set. The regression function is as follows:

$$m(\mathbf{x}) = \sum_{i \in \Omega_k(\mathbf{x})} w(\mathbf{x}, \mathbf{x}_i)\mathbf{y}_i \tag{11}$$

where $\Omega_k(\mathbf{x})$ is a set of indices of k nearest neighbors of $\mathbf{x}$ in Φ and $w(\mathbf{x}, \mathbf{x}_i) = 1/k$ for each $\mathbf{x}$ and $\mathbf{x}_i$.

Note that in this model the weights of $\mathbf{y}_i$ are all equal to $1/k$. Function (11) is a step function. The number of nearest neighbors k controls the smoothness of the estimator. For $k = 1$ the regression function is exactly fitted to the training points.

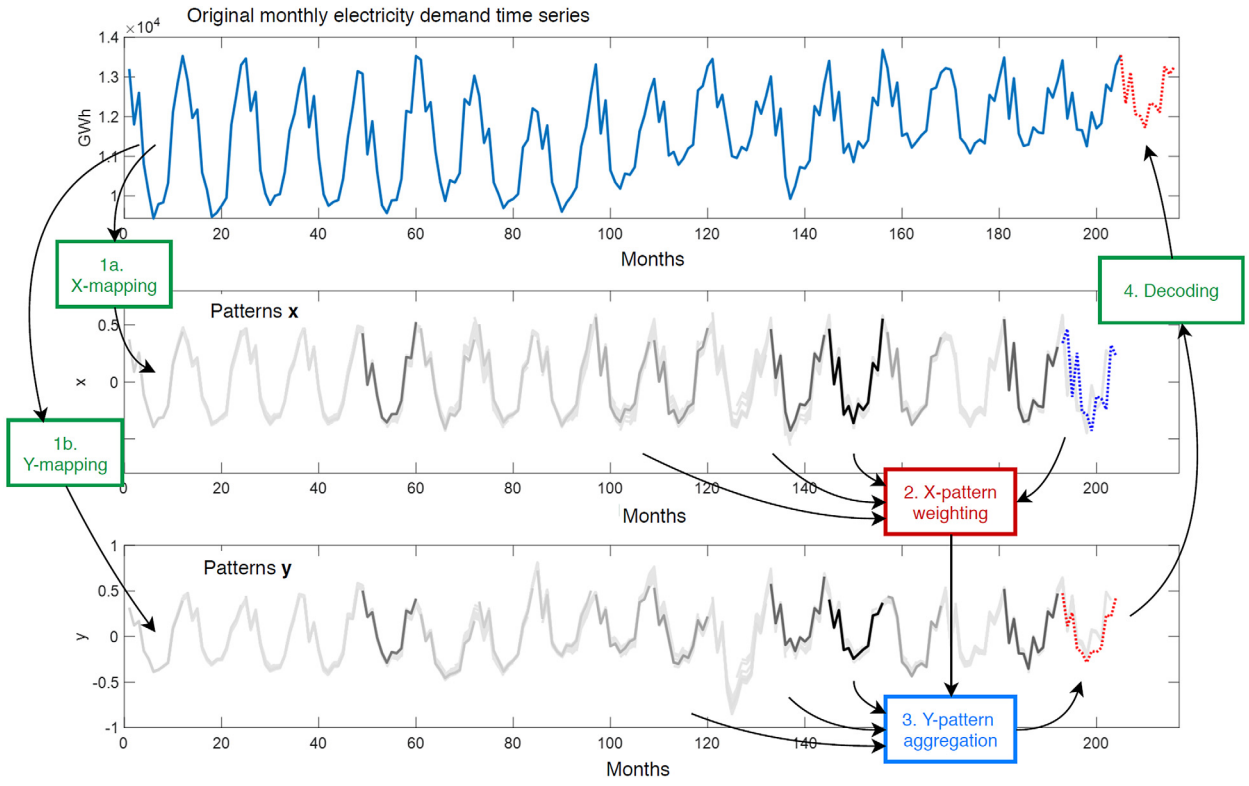
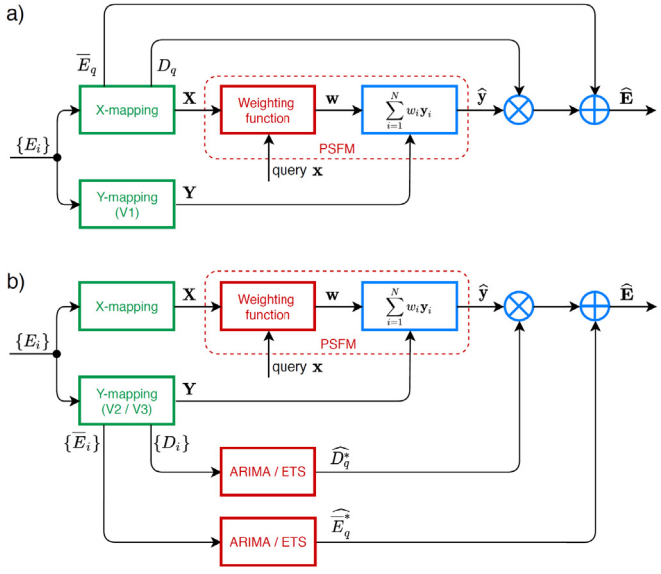


Fig. 3. The concept of pattern similarity-based forecasting.



$\{E_i\} = \{E_1, E_2, ...\}$ – original time series,
$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ – set of training patterns $\mathbf{x}_i \in \mathbb{R}^n$, (1)-(4),
$\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_1, ..., \mathbf{y}_N\}$ – set of training patterns $\mathbf{y}_i \in \mathbb{R}^m$, (5)-(8),
$\mathbf{x} \in \mathbb{R}^n$ – query pattern, (1)-(4),
$\overline{E}_q, D_q$ – mean and dispersion of query sequence X_q,
$\widehat{\overline{E}}_q^*, \widehat{D}_q$ – mean and dispersion forecasted for Y_q,
$\{\overline{E}_i\} = \{\overline{E}_1, \overline{E}_2, ..., \overline{E}_N\}$ – time series of the yearly mean demands,
$\{D_i\} = \{D_1, D_2, ..., D_N\}$ – time series of the yearly demand dispersions,
$\mathbf{w} = [w_1 w_2 ... w_N]^T$ – vector of weights for y-patterns, (12), (15), (17), (19),
$\widehat{\mathbf{y}} \in \mathbb{R}^m$ – forecasted y-pattern, (10),
$\widehat{\mathbf{E}} = [E_{q+\tau} E_{q+\tau+1} \dots E_{q+\tau+m-1}]^T$ – forecasted sequence Y_q, (9).

Fig. 4. Flowcharts of the proposed MTLF procedures in variant V1 (a) and V2/V3 (b).

Algorithm 1 Pattern Similarity-based MTLF

Input:

Monthly electricity demand time series $\{E_i\}$
Algorithm variant (V1, V2 or V3)
Pattern similarity forecasting method PSFM with parameters Θ
Forecasting horizon τ, forecasted sequence length m, input pattern length n

Output:

$\widehat{\mathbf{E}} = [E_{q+\tau} E_{q+\tau+1} \dots E_{q+\tau+m-1}]^T$ (forecasted sequence Y_q)

Procedure:

1: Define patterns **x** and **y** depending on variant V1, V2 or V3
2: Create training set $\Phi = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$
3: PSFM:
$\mathbf{w} = Weighting\_function(\text{query } \mathbf{x}, \Phi, \Theta)$
$\widehat{\mathbf{y}} = \sum_{i=1}^{N} w_i \mathbf{y}_i$
4: **if** V1 **then**
$\overline{E}_q^* = \overline{E}_q$, $D_q^* = D_q$
5: **elseif** V2 **then**
$\overline{E}_q^* = ARIMA(\{\overline{E}_i\})$, $D_q^* = ARIMA(\{D_i\})$
6: **elseif** V3 **then**
$\overline{E}_q^* = ETS(\{\overline{E}_i\})$, $D_q^* = ETS(\{D_i\})$
endif
7: $\widehat{\mathbf{E}} = \widehat{\mathbf{y}} D_q^* + \overline{E}_q^*$

Increasing k causes an increase in bias and a decrease in variance of the estimator. To get rid of jumps in the regression function and make it smoother, we can introduce a weighting function that gives greater weights for closer neighbors and lower weights

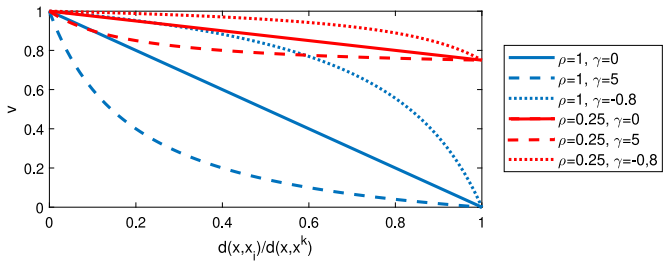


Fig. 5. The weighting function for k-NN.

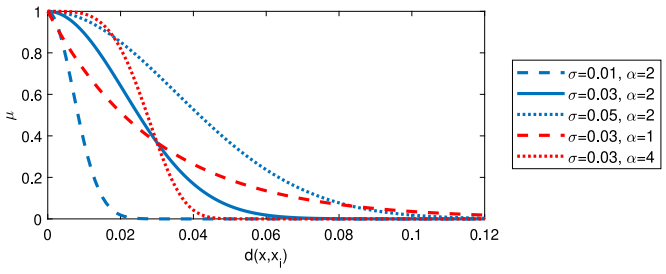


Fig. 6. The membership function for FNM.

for distant ones. Weighting functions are dependent on the distance between a query pattern and its nearest neighbors. They monotonically decrease, reach a maximum value at zero, and a minimum value (nonnegative) for the kth nearest neighbor. Some weighting function propositions can be found in [31]. In this work, we use a weighting function in the form [28]:

$$w(\mathbf{x}, \mathbf{x}_i) = \frac{v(\mathbf{x}, \mathbf{x}_i)}{\sum_{j \in \Omega_k(\mathbf{x})} v(\mathbf{x}, \mathbf{x}_j)} \tag{12}$$

$$v(\mathbf{x}, \mathbf{x}_i) = \rho \left(\frac{1 - d(\mathbf{x}, \mathbf{x}_i)/d(\mathbf{x}, \mathbf{x}^k)}{1 + \gamma d(\mathbf{x}, \mathbf{x}_i)/d(\mathbf{x}, \mathbf{x}^k)} - 1 \right) + 1 \tag{13}$$

where $\mathbf{x}^k$ is the kth nearest neighbor of $\mathbf{x}$ in Φ, $d(\mathbf{x}, \mathbf{x}_i)$ is the Euclidean distance between $\mathbf{x}$ and its ith nearest neighbor in Φ, $\rho \in [0, 1]$ is a parameter determining the differentiation of weights, and $\gamma \geq -1$ is a parameter determining the convexity of the weighting function.

Function (13) is shown in Fig. 5. The interval of weights v is $[1 - \rho, 1]$. So, for $\rho = 1$ the weights are the most diverse and for $\rho = 0$ they are all equal. In the latter case, we get $w(\mathbf{x}, \mathbf{x}_i) = 1/k$. For $\gamma = 0$ the weighting function is linear. For $\gamma > 0$ we get a convex function and for $\gamma < 0$ we get a concave function. The three parameters, k, ρ and γ, allow us to control the model's features flexibly.

4.2. Fuzzy neighborhood model

In the k-NN model, the regression surface is built using k training patterns. The fuzzy neighborhood model (FNM) takes into account all training patterns when constructing the regression surface [24]. In this case, not only k x-patterns belong to the query pattern neighborhood but all training patterns belong to it, with a different degree of membership. The degree of membership of the less distant training pattern $\mathbf{x}_i$ is higher (equal to the maximal value of 1 for $\mathbf{x}_i$ = query x-pattern) than the degree of membership of the more distant pattern. The degree of membership is defined by a membership function which should monotonically decrease with distance $d(\mathbf{x}, \mathbf{x}_i)$. In this study we apply a popular Gaussian membership function in the form:

$$\mu(\mathbf{x}, \mathbf{x}_i) = \exp\left(- \left(\frac{d(\mathbf{x}, \mathbf{x}_i)}{\sigma} \right)^{\alpha} \right) \tag{14}$$

where σ and α are parameters determining the membership function shape (see Fig. 6).

Based on the membership function, the weighting function in FNM takes the form:

$$w(\mathbf{x}, \mathbf{x}_i) = \frac{\mu(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^{N} \mu(\mathbf{x}, \mathbf{x}_j)} \tag{15}$$

Other membership functions than Gaussian-type are also possible, e.g Cauchy-type function with a fatter tail, which gives greater weights for more distant patterns. The model parameters, σ and α, shape the membership function as shown in Fig. 6, and thus control the properties of the estimator. The model tends to increase bias and decrease variance for wider membership functions.

4.3. Kernel regression model

The kernel regression model belongs to the same category of non-parametric methods as k-NN and FNM. It models a nonlinear relationship between a pair of random variables $\mathbf{x}$ and $\mathbf{y}$. The Nadaraya–Watson estimator (N-WE) is the most popular representative of this group. The N-WE estimates regression function $m(.)$ as a locally weighted average, using in (10) a kernel K_h as a weighting function [32]:

$$w(\mathbf{x}, \mathbf{x}_i) = \frac{K_h(\mathbf{x} - \mathbf{x}_i)}{\sum_{j=1}^{N} K_h(\mathbf{x} - \mathbf{x}_j)} \tag{16}$$

A kernel function is centered at data point $\mathbf{x}_i$ and gives the highest value when the distance between this point and query point $\mathbf{x}$ is zero. The kernel function falls with distance, at a speed dependent on smoothing parameter (or bandwidth) h.

When the input variable is multidimensional, the kernel has a product form. In such a case, for a normal kernel, which is often used in practice, the weighting function is defined as [25,28]:

$$w(\mathbf{x}, \mathbf{x}_i) = \frac{\exp\left(- \sum_{t=1}^{n} \frac{(x_t - x_{i,t})^2}{2h_t^2} \right)}{\sum_{j=1}^{N} \exp\left(- \sum_{t=1}^{n} \frac{(x_t - x_{j,t})^2}{2h_t^2} \right)} \tag{17}$$

where h_t is a bandwidth for the tth dimension.

In N-WE, bandwidth has a prominent effect on the estimator shape, whereas the kernel is clearly less important. Note that in (17) we define the bandwidths individually for each dimension. This gives a more flexible estimator which allows us to control the influence of each input on the resulting fitted surface. The bandwidths decide about the bias–variance tradeoff of the estimator. For too low values the estimator is undersmoothed, while for too large values it is oversmoothed.

4.4. General regression neural network model

A General Regression Neural Network (GRNN) was proposed in [33] as a variant of a radial basis function NN. It is a memory-based network where each neuron corresponds to one training x-pattern. GRNN provides smooth approximation of the target function even with sparse data in a multidimensional space. Due to one pass learning it learns very fast when compared to other NNs. Other advantages of GRNN are easy tuning, a highly parallel structure and smooth approximation of a target function even with sparse data in a multidimensional space.

As shown in Fig. 7, GRNN is composed of four layers: input, pattern (radial basis layer), summation and output layer.

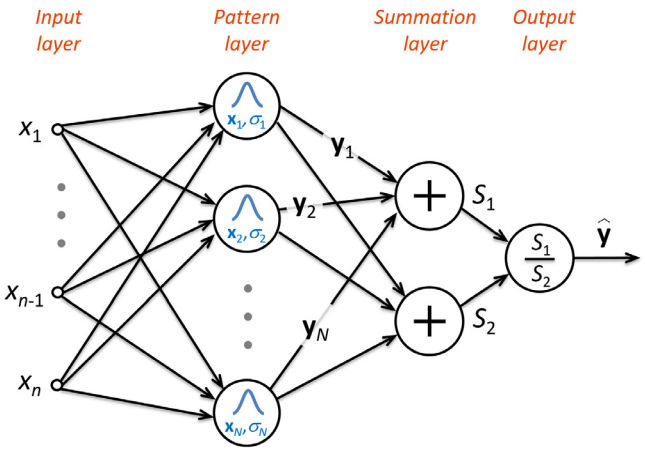


Fig. 7. General regression neural network.

The pattern layer transforms inputs nonlinearly using Gaussian activation functions in the form:

$$G(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|(\mathbf{x} - \mathbf{x}_i)\|^2}{\sigma_i^2}\right) \tag{18}$$

where $\|.\|$ is a Euclidean norm and σ_i is a bandwidth for the ith pattern.

The Gaussian functions are centered at different training patterns $\mathbf{x}_i$. The neuron output expresses similarity between the query pattern and the ith training pattern. This output is treated as the weight of the ith y-pattern. So the pattern layer maps the n-dimensional input space into an N-dimensional space of similarity. The weighting function implemented in GRNN is defined as [26]:

$$w(\mathbf{x}, \mathbf{x}_i) = \frac{G(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^{N} G(\mathbf{x}, \mathbf{x}_j)} \tag{19}$$

The performance of GRNN is related to bandwidths σ_i which govern the smoothness of the regression function (10). Note that in the GRNN model each neuron has its own bandwidth σ_i. This gives us the flexibility to control the weight of the ith y-pattern individually. As in the case of the other PSFMs presented above, selection of the optimal values of the bandwidths is a key issue in GRNN.

5. Simulation studies

In this section, we compare PSFMs on a mid-term load forecasting problem using real-world data: monthly electricity demand time series for 35 European countries. The data are taken from the publicly available ENTSO-E repository (www.entsoe.eu/data/power-stats/). The longest time series cover the time period from 1991 to 2014 (11 countries out of 35). Others are shorter: 17 years (6 countries), 12 years (4 countries), 8 years (2 countries), and 5 years (12 countries). The goal is to predict the demand for each of the twelve months in 2014 using historical data. The 35 time series are shown in Fig. 8. As can be seen from this figure, the time series have different levels, trends, variations and yearly shapes.

5.1. Verification of the PSFM assumption

In Section 3, the assumption underlying PSFMs was stated. To confirm this assumption the null hypothesis was formulated. It is verified for each country using the chi-squared test based on a contingency table showing the joint empirical distribution of the random variables, which are Euclidean distances between patterns: $d(\mathbf{x}_i, \mathbf{x}_j)$ and $d(\mathbf{y}_i, \mathbf{y}_j)$. In this analysis, we use patterns defined by (4) and (8), where $n, m = 12$, $\overline{E}_i^* = \overline{E}_i$, and $D_i^* = D_i$.

In the contingency tables, the random variables are divided into five categories containing roughly the same number of observations. Fig. 9 shows the chi-squared statistics for the analyzed countries. Critical value χ^2 is shown by a dashed line. It is equal to 26.30 for the case of five categories adopted for each random variable and significance level $\alpha = 0.05$. As you can see from this figure, all χ^2 values are higher than or very close to the critical value. This allows us to reject the null hypothesis and justifies the use of PSFMs. The strongest relationships between the random variables are observed for Italy, Germany, Belgium and Luxembourg.

5.2. Comparative models

As comparative models, we use classical statistical forecasting models such as ARIMA and ETS, as well as neural and neuro-fuzzy models:

ARIMA – ARIMA$(p, d, q)(P, D, Q)_{12}$ model implemented in function `auto.arima` in R environment (package `forecast`). This function implements automatic ARIMA modeling which combines unit root tests, minimization of the Akaike information criterion (AICc) and maximum likelihood estimation to obtain the optimal ARIMA model [34].

ETS – exponential smoothing state space model [35] is implemented in function `ets` (R package `forecast`). This implementation includes many types of ETS models 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. As in the case of `auto.arima`, `ets` returns the optimal model using AICc [34].

MLP – multilayer perceptron is described in [36]. This model is designed for MTLF. It learns from patterns defined by (4) and (8). It predicts one component of the y-pattern on the basis of x-patterns. For all m components, m MLPs are trained. When all m components of the y-pattern have been predicted by the set of MLPs, the demand forecasts are calculated using (9). The network has one hidden layer with sigmoidal neurons and learns using the Levenberg–Marquardt method with Bayesian regularization to prevent overfitting. The two MLP hyperparameters, which need to be tuned, are the number of hidden nodes and the length of the input patterns n. We use Matlab R2018a implementation of MLP (function `feedforwardnet` from Neural Network Toolbox).

ANFIS – adaptive neuro-fuzzy inference system is proposed for MTLF in [37]. It works on patterns (4) and (8). This is an n-input, single-output model for predicting one y-pattern component. So, for all components, m models are built and trained. ANFIS architecture is functionally equivalent to a Sugeno type fuzzy rule base. Initial membership function parameters in the premise parts of rules are determined using fuzzy c-means clustering. The hybrid learning method applied for ANFIS training uses a combination of the least-squares for consequent parameters and a backpropagation gradient descent method for premise parameters. The ANFIS hyperparameters which were selected are the number of rules and the length of input patterns n. The Matlab R2018a implementation of ANFIS was used (function `anfis` from Fuzzy Logic Toolbox).

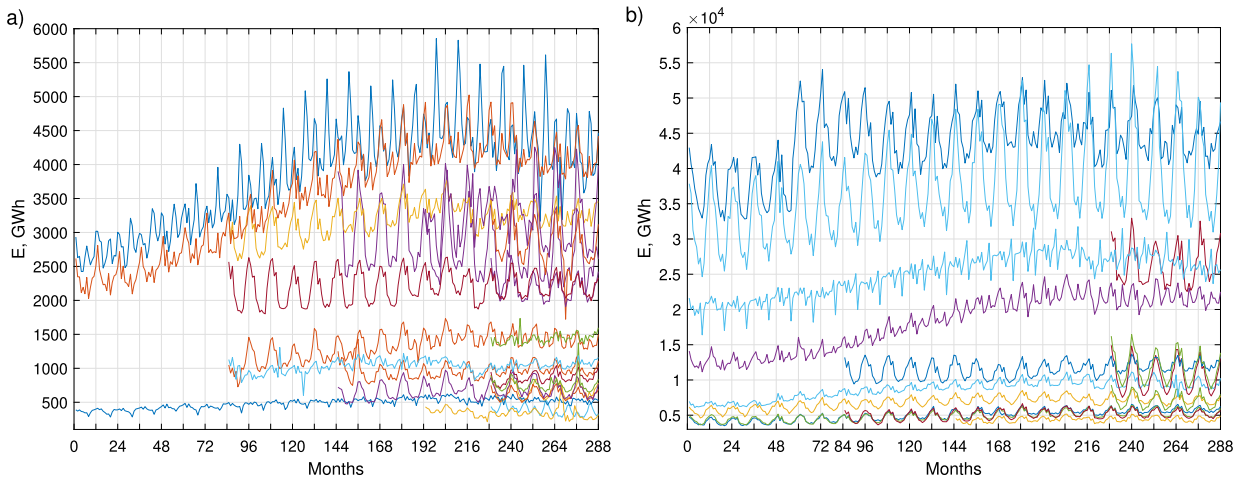


Fig. 8. Monthly electricity demand time series for 35 European countries.

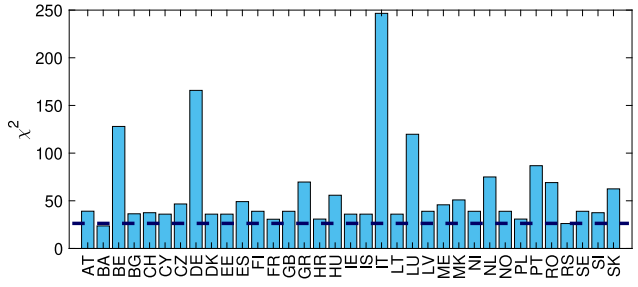


Fig. 9. Chi-squared statistics for verification of PSFM assumption.

LSTM – long short-term memory (LSTM) network, where the responses are training sequences with values shifted by one time step (a sequence-to-sequence regression LSTM network) [38]. For multiple time steps, after one step was predicted the network state was updated. Previous prediction was used as input to the network producing the prediction for the next time step. LSTM was optimized using Adam (adaptive moment estimation) optimizer. The number of hidden nodes was the only hyperparameter to be tuned. Other hyperparameters remain at their default values. The experiments were carried out using Matlab R2018a implementation of LSTM (function `trainNetwork` from Neural Network Toolbox).

5.3. Parameter settings and model variants

Taking into account our earlier experiences with PSFMs reported in [24,27,28,36,36,37] and [25] we use pattern definitions (4) and (8). In most cases these pattern definitions produce the greatest accuracy from the models.

One of the main hyperparameters for all proposed PSFMs, as well as for MLP and ANFIS, was the length of the x-patterns. Although the natural choice for this hyperparameter is the seasonal cycle length, i.e. 12, we tested the models for a range for n from 3 to 24, before selecting the optimal value of n for each model and each time series.

The k-NN model was used in two variants. The first one assigns the same weights to all k neighbors: $w(\mathbf{x}, \mathbf{x}_i) = 1/k$. The second uses the weighting function defined by (12) and (13), where $\rho = 1$ and $\gamma = 0$ (linear weighting function). This variant is denoted as k-NNw. A key hyperparameter in both these variants, besides the x-pattern length, is the number of nearest neighbors k. This was selected from a range from 1 to 50.

In FNM, we set $\alpha = 2$ and control the membership function (14) width with σ. This hyperparameter was calculated from:

$$\sigma = a \cdot d_{med} \tag{20}$$

where d_{med} was the median of pairwise distances between x-patterns in the training set. It was searched for $a = 0.02, 0.04, \ldots, 1$.

Determining σ on the basis of d_{med} calibrates this parameter to data.

The bandwidth parameters in N-WE were searched around the starting values, which were determined using the Scott rule [39] proposed for the normal product density estimators:

$$h_t^S = s_t N^{-\frac{1}{n+4}}, \quad t = 1, 2, \ldots, n \tag{21}$$

where s_t is the standard deviation of the tth component of $\mathbf{x}$ estimated from the training sample.

The searched bandwidths are generated according to:

$$h_t = b \cdot h_t^S, \quad t = 1, 2, \ldots, n \tag{22}$$

where $b = 0.15, 0.2, \ldots, 2$.

Note that the multidimensional optimization problem of searching bandwidths $h_1, h_2, \ldots, h_n$ was replaced by the simple one-dimensional problem of searching b.

For GRNN, we assume in this study the same bandwidths for all neurons. Bandwidth σ was searched according to (20). For MLP, the number of hidden neurons was selected from a range of 1 to 10. The number of rules in the ANFIS model was selected from a range of 2 to 13. The number of hidden nodes in the LSTM model was selected from the set $\{1, 2, \ldots, 10, 15, \ldots, 50, 60, \ldots, 200\}$. For ARIMA and ETS models, we use default parameter settings implemented in functions `auto.arima` and `ets`, respectively.

For each model, the optimal values of its hyperparameters were selected for each of 35 time series in a grid search procedure using cross-validation.

Taking into account the y-pattern encoding variants described in Section 2, three variants of each PSFM are considered:

V1. The basic variant where the coding variables for y-patterns are the mean and dispersion of sequence X_i, i.e. $\overline{E}_i^* = \overline{E}_i$, $D_i^* = D_i$. This variant enables us to calculate the forecast of the monthly loads from (9) without additional forecasting for coding variables.

Table 1
Results comparison among proposed and comparative models.

Model	Median APE	MAPE	IQR	RMSE
k-NN	3.11	5.19	4.13	385.68
k-NNw	2.89	4.99	4.06	368.79
FNM	2.88	4.88	4.43	354.33
N-WE	2.84	5.00	4.14	352.01
GRNN	2.87	5.01	4.30	350.61
k-NN+ARIMA	2.88	4.71	4.21	352.42
k-NNw+ARIMA	2.89	4.65	4.02	346.58
FNM+ARIMA	2.87	4.61	3.83	341.41
N-WE+ARIMA	2.85	4.59	3.74	340.26
GRNN+ARIMA	2.81	4.60	3.77	345.46
k-NN+ETS	2.72	4.58	3.55	333.27
k-NNw+ETS	2.71	4.47	3.43	327.94
FNM+ETS	2.64	4.40	3.34	321.98
N-WE+ETS	2.68	4.37	3.20	320.51
GRNN+ETS	2.64	4.38	3.35	324.91
MLP	2.97	5.27	3.89	378.81
MLP+ARIMA	3.12	4.83	4.16	362.03
MLP+ETS	3.11	4.80	4.12	358.07
ANFIS	3.56	6.18	4.91	488.75
ANFIS+ARIMA	3.66	6.05	5.40	473.80
ANFIS+ETS	3.54	6.32	4.57	464.29
LSTM	3.73	6.11	4.46	431.83
ARIMA	3.32	5.65	5.27	463.07
ETS	3.50	5.05	4.17	374.52

Table 2
Descriptive statistics of percentage errors PE.

Model	Mean	Median	Std	Skewness	Kurtosis
k-NN	−1.96	−1.27	10.83	−4.88	49.39
k-NNw	−1.87	−1.08	10.43	−5.14	44.56
FNM	−2.03	−1.22	9.34	−4.16	35.14
N-WE	−1.91	−1.18	10.82	−5.41	48.94
GRNN	−1.87	−1.16	10.60	−5.34	48.11
k-NN+ARIMA	−1.76	−0.75	8.10	−2.66	20.96
k-NNw+ARIMA	−1.74	−0.82	7.92	−2.50	19.56
FNM+ARIMA	−1.75	−0.84	7.89	−2.76	21.57
N-WE+ARIMA	−1.75	−0.85	7.82	−2.68	21.38
GRNN+ARIMA	−1.75	−0.81	7.81	−2.59	20.48
k-NN+ETS	−1.26	−0.20	9.11	−4.47	38.22
k-NNw+ETS	−1.25	−0.20	9.00	−4.40	37.30
FNM+ETS	−1.26	−0.11	8.80	−4.75	41.71
N-WE+ETS	−1.26	−0.17	8.68	−4.63	40.75
GRNN+ETS	−1.26	−0.11	8.61	−4.42	38.38
MLP	−1.37	−0.68	11.88	−7.52	109.64
MLP+ARIMA	−1.64	−0.92	7.45	−1.64	12.16
MLP+ETS	−1.71	−1.03	7.32	−1.55	11.83
ANFIS	−2.51	−1.43	11.37	−4.35	34.93
ANFIS+ARIMA	−1.94	−0.65	9.63	−1.67	13.29
ANFIS+ETS	−1.30	−0.40	12.65	−0.96	39.37
LSTM	−3.12	−1.81	9.49	−2.86	22.21
ARIMA	−2.35	−1.03	13.62	−9.01	119.20
ETS	−1.04	−0.31	7.97	−1.89	13.52

V2. The variant where the coding variables are the mean and dispersion of sequence Y_i. For the query pattern, they are both forecasted independently using ARIMA model on the basis of their historical values. The denotations in this variant are extended by "+ARIMA", e.g. "*k*-NN+ARIMA", "ANFIS+ARIMA".

V3. As in variant V2, the coding variables are the mean and dispersion of sequence Y_i. But in this case they are forecasted for the query pattern using ETS. The denotations in this variant are extended by "+ETS", e.g. "*k*-NN+ETS", "ANFIS+ETS".

These three variants are also used for MLP and ANFIS models (they also use pattern representation).

5.4. Results

PSFMs are deterministic models which return the same results for the same data. NN-based models, i.e. MLP, ANFIS and LSTM, due to the stochastic nature of the learning processes, return different results for the same data. In this study, these models were trained 100 times and the final errors were calculated as averages over 100 independent trials.

Fig. 10 shows the forecasting errors on the test sets (mean absolute percentage error, MAPE) for each country. The rankings of the models are shown in Fig. 11. The ranking shown on the left is based on the median of APE and the ranking shown on the right is based on the average ranks of the models in the rankings for individual countries. Table 1 summarizes the accuracy of the models showing the median of APE, MAPE, the interquartile ranges of APE and the root mean square error (RMSE).

As can be seen from Figs. 10 and 11 and Table 1, the most accurate models are the five PSFMs with ETS (variant V3). There is no significant difference in errors between them. PSFMs in the basic variants (V1) and in +ARIMA variants (V2) were ranked lower then PSFMs+ETS in both rankings. The largest errors among PSFMs were observed for the simple *k*-NN model with equal weights.

The comparative models were ranked as the least accurate. Among them, the best one turned out to be MLP and the worse ones were ANFIS-based models and LSTM ($MAPE > 6\%$). Note the enormous errors for Montenegro (ME) in Fig. 10, $MAPE > 25\%$. These are caused by the very irregular character of the time series for ME and the abnormal value of demand for March and April 2013, which is about twice the value typical for these months. No model managed to deal with this time series satisfactorily.

To evaluate further the forecast errors, we present descriptive statistics of percentage errors (PE) in Table 2. The mean value of PE indicates forecast bias. For each model, we obtained a negative mean PE, indicating a negatively biased forecast, i.e. overprediction. The negatively biased forecasts were confirmed in all cases using the *t*-test, which rejected the hypothesis that mean PE is equal to zero ($\alpha = 0.05$). The PE distributions are similar in shape to the normal shape but statistical tests for the assessment of normality (Jarque–Bera test and Lilliefors test) do not confirm this. Among the proposed PSFMs, the least biased are those models in the V3 variant, and the most biased are those models in the basic variant V1.

The negative values of skewness for all models shown in Table 2, indicate left-skewed PE distributions. High kurtosis values indicate leptokurtic distributions where the probability mass is concentrated around the mean.

Examples of forecasts generated by the models for four countries are depicted in Fig. 12. For PL, the PSFMs produce the most accurate forecasts. Among the PSFMs, basic variant V1 is slightly better than variants V2 and V3. Note that for PL the classical models, ARIMA nad ETS, give outlier forecasts. ARIMA produces overestimated forecasts, while ETS produces underestimated forecasts. ARIMA also gives the most inaccurate forecasts for DE and FR. For GB, the forecasts generated by all models are underestimated. This results from the fact that demand in GB in 2014 went up unexpectedly despite the downward trend observed from 2010 to 2013. In FR, the reverse situation caused a slight overestimation of forecasts. Note that for both MLP and ANFIS jumps in the forecasted curve are observed. This is because these models predict only one component of the y-pattern, and to forecast all m components we use m independent models. So, the relationships between the components are ignored. In the case of PSFMs which generate a multi-output response, these relationships are kept because the forecasted annual cycle is formed by weighted averaging the shapes of historical annual cycles.

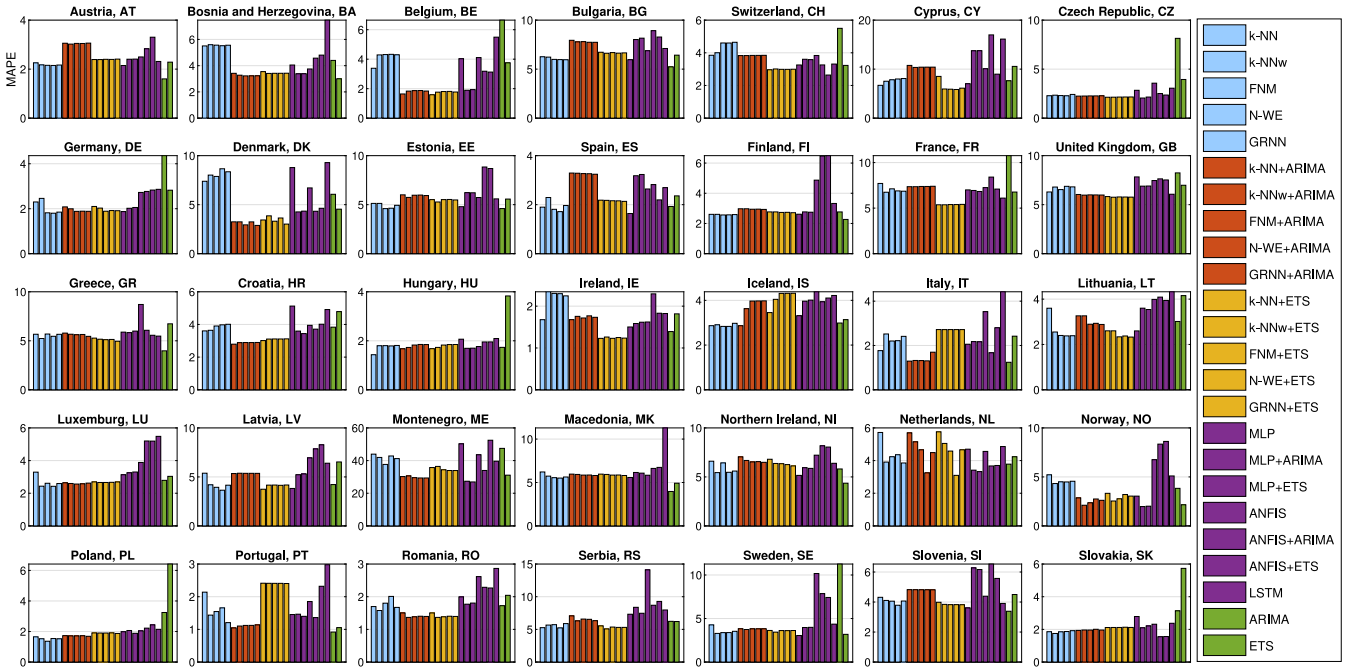


Fig. 10. MAPE for each country.

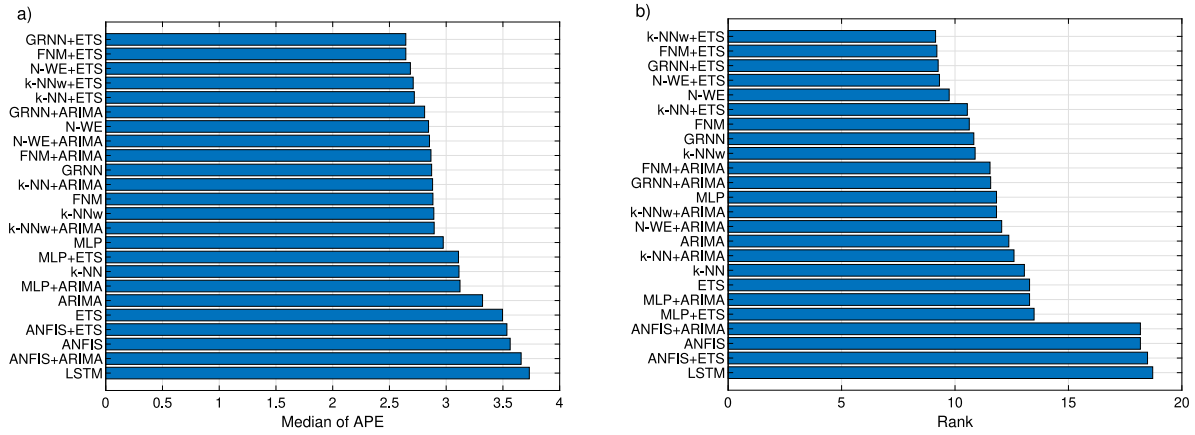


Fig. 11. Rankings of the models: (a) ranking based on the median of APE, (b) ranking based on the average model ranks in the individual rankings for each country.

5.5. Discussion

PBFM variants. The basic variant, V1, uses coding variables determined from history, i.e., the means and dispersions of sequences X_i. In such a case, the stability of the relationship between means/dispersions of sequences X_i and Y_i is very important. When the trend is falling, the y-pattern is encoded with the mean of the previous sequence X_i which is higher than the mean of Y_i. For a forecasted y-pattern the same higher value of the mean coding variable is expected. But when the time series, instead of continuing to fall, starts to rise, the relationship between the means of sequences X_i and Y_i observed in the past is no longer valid. This results in a wrong forecast which continues the falling trend. The opposite situation occurs when the trend is rising and it starts to fall in the final part. This problem is observed for many countries, e.g. BA, BE, IT, DK, and IE (see the increased errors for PSFMs in variant V1 for these countries in Fig. 10). A similar problem arises with dispersion. To prevent such situations we use variants V2 and V3, where the coding variables are predicted using ARIMA and ETS. In these cases, final accuracy is dependent on the accuracies of the three models: (1) PSFM, which predicts y-pattern, (2) ARIMA/ETS, which predicts the mean of sequence Y_q, and (3) ARIMA/ETS, which predicts the dispersion of sequence Y_q. When the coding variables are predicted with low accuracy, the final error can be higher than in the case of the basic PSFM variant (see graphs for AT, BG and ES in Fig. 10).

The simulation study found that variant V3 performed best, i.e., a combination of PSFM with ETS. Slightly higher errors were shown by variant V2 (PSFM+ARIMA), and the highest errors were for basic variant V1.

Standard ML vs. PSFM learning. In the standard approach, ML methods such as MLP and ANFIS create global forecasting models. During learning all training patterns are treated in the same way, without any of them being proffered. That is, the model is optimized globally by minimizing the loss function, which is the sum of errors for all training patterns. So, each training pattern has the same impact on regression function shaping. Such a model is globally accurate but may be inaccurate around the current query point. Outlier patterns, in this case, can disrupt learning and lead to a loss of generalization.

In PSFMs, the regression function is constructed locally, around the query x-pattern. It is built from the training y-patterns

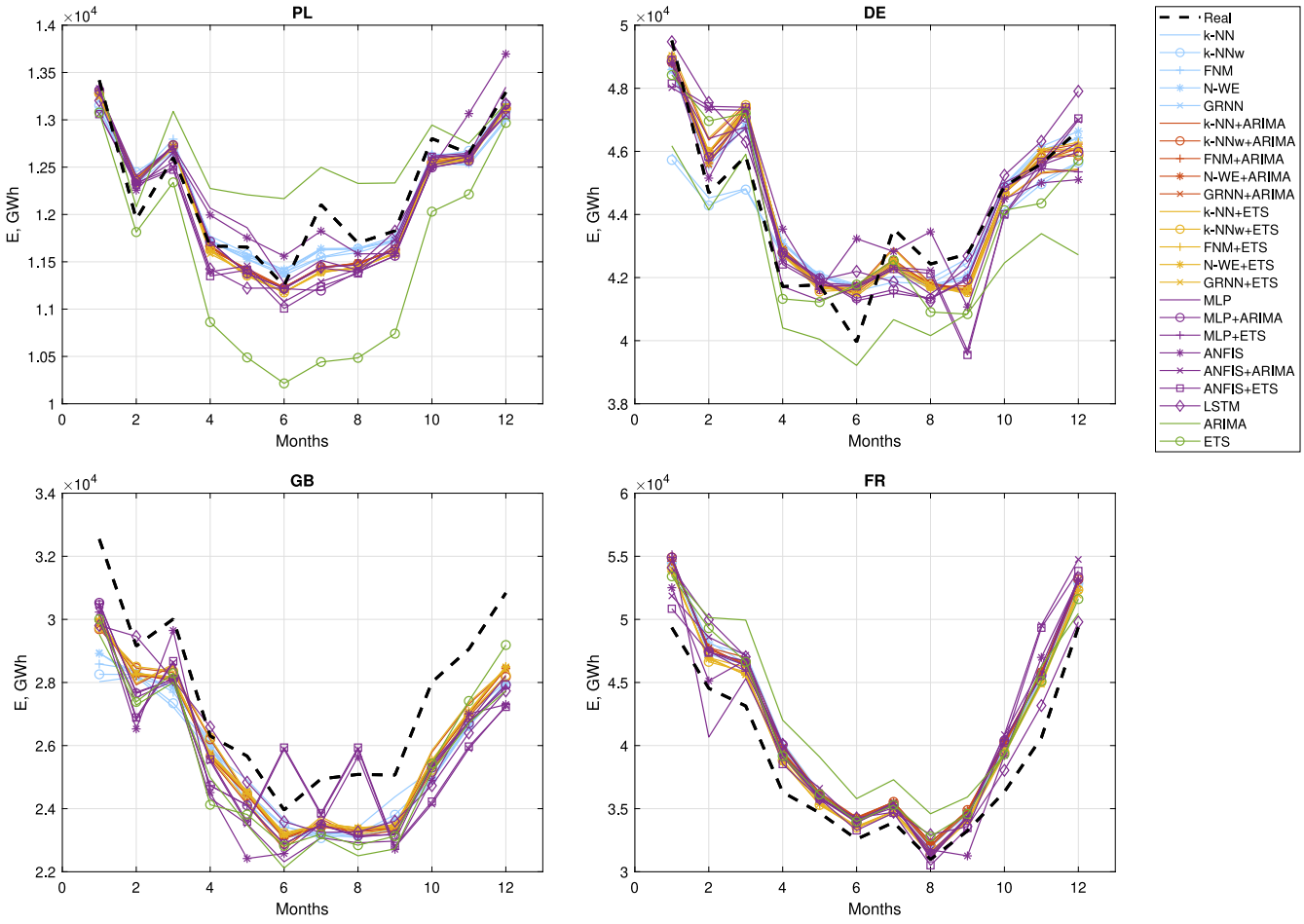


Fig. 12. Real and forecasted electricity demand for PL, DE, GB and FR.

corresponding to the x-patterns which are most similar to the query pattern. In the regression model (10) such y-patterns have larger weights, unlike the y-patterns corresponding to the more distant x-patterns. Thus the regression surface around the query pattern is not affected by distant patterns as in the case of standard learning. This leads to more accurate modeling around the query pattern, and thus to better forecasts. This also limits the impact of outliers when they are among the x-patterns (x-pattern outliers have low weights as they are distant from the query pattern). Eliminating the impact of outliers among the y-patterns requires a different approach (not included in the current version of PSFMs). In this approach, the y-pattern outlier can be detected by comparison with other training y-patterns from the neighborhood of its corresponding x-pattern. A y-pattern which is distant from other neighboring y-patterns can be detected as an outlier and removed or its weight can be decreased. The exact mechanism for dealing with outliers in PSFMs will be the subject of further research.

Note that PSFMs are non-parametric regression models. They directly use data to construct the forecast (see (10) where the model combines y-patterns and weights which are calculated based on x-patterns). They do not require training like parametric ML models, where the parameters (weights in NNs) are learned from the training set. The number of parameters to adjust in parametric ML is sometimes huge (hundreds or thousands), which entails time consuming training. Moreover, hyperparameters such as the number of neurons, learning rate, activation function type, number of epochs (and many others depending on the model architecture) need additional optimization. PSFMs need only to select their hyperparameters. In our case, there are only two hyperparameters: the input pattern length and the bandwidth parameter deciding the weighting function shape. Thus the optimization procedure of PSFMs is much faster than the learning and optimization of parametric ML models.

Note also that parametric ML needs retraining when we want to include new data in the model. This is necessary in forecasting models because new data, i.e. last time series sequence just available usually contains most information about the forecasted sequence. There is no retraining in PSFMs. In this case when a new sequence is added to the training set it can be immediately used in a regression model (10).

LSTM vs. PSFM. LSTM is a recurrent NN with connections between nodes forming a directed graph along a temporal sequence. It is able to exhibit temporal dynamic behavior and capture long-term dependencies in sequential data by using their internal state (memory) to process sequences of inputs. Information learned from the previous time steps is contained in the cell state and so called hidden state. Information can be added to or removed from the cell state using three nonlinear gates. At each time step, the past state of the network and the current input are used to compute output and updated cell state. The complex structure of LSTM makes it very hard to understand and analyze. There are a lot of parameters (input and recurrent weights of each gate) and hyperparameters to adjust. The learning process is time consuming and sensitive to vanishing and exploding gradients. New LSTM architectures, with a multilayered (stacked) structure equipped with additional mechanisms such as dilation and residual connections [40], are even more complex and so even more

difficult to understand. As other ML models, LSTM creates a global model.

In contrast to LSTM, PSFMs have a simple and understandable operating principle (no internal states, gates, recurrent cycles). Instead of hundreds of parameters and several hyperparameters to adjust, they have only two hyperparameters. They process each query pattern individually, constructing a local model for it.

Statistical forecasting models vs. PSFM. In ARIMA and ETS the model is optimized globally, i.e., its parameters are fine-tuned to ensure the lowest error for all time series points (however, we can limit the length of the time series to the last M observations to take into account only the specificity of the last period in the model). The ARIMA model is limited to the last time series points (orders of the AR and MA models). Thus the forecasted value is "constructed" based on the last observations. The ETS model is based on all observations but with an impact which exponentially decreases with time, i.e., the last observations have higher weights. In contrast, in PSFMs, the local model is constructed for each query pattern and it uses time series points not limited to the last observed ones like in ARIMA (see Assumption 1, where the history is not limited to the last period) and not weighted over time like in ETS. To construct the local regression function, PBFM takes into account observations that can be distant in time from the query pattern (if such distant observations are similar in shape to the query pattern). However, the time distance can be introduced easily in PBFM (10) by additional y-pattern weights decreasing with time. Weights, depending on seasonality, can also be introduced in PSFMs for multi-seasonal time series. The outliers in ARIMA and ETS distort the selection of the parameters and lead to suboptimal models. In PSFMs, the effect of outliers is partially reduced as described above.

Another difference between statistical models and PSFMs is that the former produce one-step ahead forecasts. Multi-step forecasting is performed recursively by taking a prediction for the prior time step as an input for making a prediction for the following time step. In contrast, PSFM forecasts the y-pattern representing the entire forecasted sequence in one-shot.

Limitations and disadvantages of PSFMs. The performance of PSFM is dependent on the fulfillment of Assumption 1. For a given time series, this assumption can be verified using appropriate statistical tests (see Sections 3 and 5.1). The pattern similarity-based framework model was designed for seasonal time series where the seasonal cycles are similar in shape, and the relationship between coding variables is stable in time. When this relationship changes with time, the coding variables should be predicted (variants V2 and V3 of the algorithm). This increases the complexity of the model.

PSFMs in the variants presented in this work belong to the univariate autonomous modeling approach, where only historical load data are used as inputs. By not including other inputs, such as weather and economic factors, objections can be raised but we have to remember that these exogenous variables are usually not available and have to be forecasted. This is a major challenge in all MTLF approaches with exogenous inputs. Many researchers use statistical measures such as correlation coefficients, personal experience and intuition to assess the validity, effectiveness and contribution of such exogenous variables to energy and load forecasting [3]. This can lead to low accuracy forecasts of weather and economic variables, and consequently, to larger load forecast errors. However, the regression model (10) can be extended to incorporate additional input variables. These can be introduced by additional weights dependent on the similarity between patterns representing external variables. Such an approach was shown in [41], where the model for STLF was extended to include weather variables.

When using PSFMs, the time series should be long enough to include many seasonal cycles. These cycles, expressed in patterns, should provide sufficient variability in shapes. A lack of variability limits the model performance. This is because regression function (10) creates the forecast by combining the historical y-patterns. When their variability is low or they are not representative, the model is not flexible enough and thus cannot produce an accurate forecast.

6. Conclusion

In this work, we presented a framework of pattern similarity-based forecasting and explored PSFMs for MTLF. The key component of these models is the use of patterns of the time series sequences for time series representation. We defined input and output patterns which unify input and output data. The patterns carry information about the shapes of the annual cycles, filtering out the trend and normalizing data. They simplify relationships between data, making the forecasting model simpler and faster to train.

PSFMs belong to a class of lazy learning regression models where the regression function is built by the aggregation of output patterns with weights dependent on the similarity between input patterns. The simplest PSFM is the k-NN model with equal weights. More sophisticated models, such as k-NNw, FNM, N-WE and GRNN, which use weighting functions, produce more accurate forecasts.

The simulation study showed the high accuracy of PSFMs when compared to the classical models, such as ARIMA and ETS, as well as MLP, ANFIS and LSTM models. The best performance was achieved by hybrid PBFMs combining ML with ETS for forecasting the coding variables. It is also worth noting that basic PSFM variants (V1) outperform the comparative models, producing more accurate forecasts while being much simpler and easier to optimize.

The advantages of PSFMs can be summarized as follows:

1. A forecasting model based on pattern similarity is transparent and its operating principle is understandable. This is very important in practical applications because it translates into a greater confidence in the forecasts. Moreover, PSFMs do not require any specific knowledge of ML or AI from practitioners.
2. PSFMs are simple. They have only two hyperparameters, or even one when we assume that the pattern length is equal to the length of the seasonal cycle. Simple models are easy to optimize and do not suffer from excessive tuning and training burdens like other ML models, especially deep learning ones.
3. In PSFMs, model generalization (bias–variance tradeoff) can be controlled easily by a bandwidth parameter. In other ML models, generalization is dependent on all parameters and hyperparameters, which makes generalization difficult to achieve.
4. PSFMs do not need retraining when new data arrives. New data can be immediately added to the training set and used to produce the forecast.
5. PSFMs produce a vector output, thus they are able to multi-step forecast without a recursive approach. The components of the output pattern are the forecasts for the successive time periods.
6. The sizes of the input and output patterns do not affect the number of parameters or the complexity of the training process as is the case for other ML models.
7. PSFMs are distinguished by their robustness to incomplete data. The models can work using patterns with missing components.

8. PSFMs can be easily extended by introducing additional weighting functions in the regression model (10). This enables us to introduce additional input variables and temporal dependencies.
9. PSFMs are not limited to forecasting time series with a single seasonality. They were designed to forecast time series with multiple seasonal cycles such as STLF [27,28], where there are three seasonalities, i.e. yearly, weekly and daily.

Possible directions for further research on PSFMs are the introduction of additional input variables to PSFMs, PSFM ensembling, the introduction of confidence degrees in the training data to the models, and probabilistic forecasting.

CRediT authorship contribution statement

Grzegorz Dudek: Conceptualization, Methodology, Formal analysis, Validation, Visualization, Writing - original draft, Writing - review & editing, Supervision. **Paweł Pełka:** Software, Investigation, Formal analysis, Resources, Data curation, Validation, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] E. Dogan, Are shocks to electricity consumption transitory or permanent? Sub-national evidence from Turkey, Util. Policy 41 (2016) 77–84.
[2] F. Apadula, A. Bassini, A. Elli, S. Scapin, Relationships between meteorological variables and monthly electricity demand, Appl. Energy 98 (2012) 346–356.
[3] M. Ghiassi, D.K. Zimbra, H. Saidane, Medium term system load forecasting with a dynamic artificial neural network model, Electr. Power Syst. Res. 76 (2006) 302–316.
[4] M. Gavrilas, I. Ciutea, C. Tanasa, Medium-term load forecasting with artificial neural network models, in: IEEE Conf. Elec. Dist. Pub., vol. 6, 2001.
[5] E. González-Romera, M.A. Jaramillo-Morán, D. Carmona-Fernández, Monthly electric energy demand forecasting with neural networks and Fourier series, Energy Convers. Manage. 49 (2008) 3135–3142.
[6] M.S. Kandil, S.M. El-Debeiky, N.E. Hasanien, Long-term load forecasting for fast developing utility using a knowledge-based expert system, IEEE Trans. Power Syst. 17 (2) (2002) 491–496.
[7] P. Bunnoon, K. Chalermyanont, C. Limsakul, Mid term load forecasting of the country using statistical methodology: Case study in Thailand, in: International Conference on Signal Processing Systems, 2009, pp. 924–928.
[8] N.A. Mohammed, Modelling of unsuppressed electrical demand forecasting in Iraq for long term, Energy 162 (2018) 354–363.
[9] E. Doveh, P. Feigin, L. Hyams, Experience with FNN models for medium term power demand predictions, IEEE Trans. Power Syst. 14 (2) (1999) 538–546.
[10] M.M. Elkateb, K. Solaiman, Y. Al-Turki, A comparative study of medium-weather-dependent load forecasting using enhanced artificial/fuzzy neural network and statistical techniques, Neurocomputing 23 (1998) 3–13.
[11] Chang Pei-Chann, Fan Chin-Yuan, Lin Jyun-Jie, Monthly electricity demand forecasting based on a weighted evolving fuzzy neural network approach, Electr. Power Energy Syst. 33 (2011) 17–27.
[12] L. Suganthi, A.A. Samuel, Energy models for demand forecasting — A review, Renew. Sustain. Energy Rev. 16 (2) (2012) 1223–1240.
[13] E.H. Barakat, Modeling of nonstationary time-series data. Part II. Dynamic periodic trends, Electr. Power Energy Syst. 23 (2001) 63–68.
[14] H.M. Al-Hamadi, S.A. Soliman, Long-term/mid-term electric load forecasting based on short-term correlation and annual growth, Electr. Power Syst. Res. 74 (2005) 353–361.
[15] E. González-Romera, M.A. Jaramillo-Morán, D. Carmona-Fernández, Monthly electric energy demand forecasting based on trend extraction, IEEE Trans. Power Syst. 21 (4) (2006) 1935–1946.
[16] J.F. Chen, S.K. Lo, Q.H. Do, Forecasting monthly electricity demands: An application of neural networks trained by heuristic algorithms, Information 8 (1) (2017) 31.
[17] T. Ahmad, H. Chen, Potential of three variant machine-learning models for forecasting district level medium-term and long-term energy demand in smart grid environment, Energy 160 (2018) 1008–1020.
[18] W. Zhao, F. Wang, D. Niu, The application of support vector machine in load forecasting, J. Comput. 7 (7) (2012) 1615–1622.
[19] J. Bedi, D. Toshniwal, Empirical mode decomposition based deep learning for electricity demand forecasting, IEEE Access 6 (2018) 49144–49156.
[20] M. Theodosiou, Forecasting monthly and quarterly time series using STL decomposition, Int. J. Forecast. 27 (4) (2011) 1178–1195.
[21] E.M. de Oliveira, F.L.C. Oliveira, Forecasting mid-long term electric energy consumption through bagging ARIMA and exponential smoothing methods, Energy 144 (2018) 776–788.
[22] D. Benaouda, F. Murtagh, J.L. Starck, O. Renaud, Wavelet-based nonlinear multiscale decomposition model for electricity load forecasting, Neurocomputing 70 (1–3) (2006) 139–154.
[23] X. Qiu, Y. Ren, P.N. Suganthan, G.A.J. Amaratunga, Empirical mode decomposition based ensemble deep learning for load demand time series forecasting, Appl. Soft Comput. 54 (2017) 246–255.
[24] P. Pełka, G. Dudek, Prediction of monthly electric energy consumption using pattern-based fuzzy nearest neighbour regression, in: Proc. Conf. Computational Methods in Engineering Science CMES'17, ITM Web Conf., vol. 15, 2017, pp. 1–5.
[25] G. Dudek, P. Pełka, Medium-term electric energy demand forecasting using Nadaraya–Watson estimator, in: Proc. Conf. on Electric Power Engineering EPE'17, 2017, pp. 1–6.
[26] P. Pełka, G. Dudek, Medium-term electric energy demand forecasting using generalized regression neural network, in: Proc. Conf. Information Systems Architecture and Technology ISAT 2018, AISC 853, Springer, Cham, 2018, pp. 218–227.
[27] G. Dudek, Pattern similarity-based methods for short-term load forecasting – Part 1: Principles, Appl. Soft Comput. 37 (2015) 277–287.
[28] G. Dudek, Pattern similarity-based methods for short-term load forecasting – Part 2: Models, Appl. Soft Comput. 36 (2015) 422–441.
[29] W. Duch, Similarity-based methods: A general framework for classification, approximation and association, Control Cybernet. 29 (4) (2000) 937–968.
[30] Y. Chen, E.K. Garcia, M.R. Gupta, A. Rahimi, L. Cazzanti, Similarity-based classification: Concepts and algorithms, J. Mach. Learn. Res. 10 (2009) 747–776.
[31] C.G. Atkenson, A.W. Moor, S. Schaal, Locally weighted learning, Artif. Intell. Rev. 11 (1997) 75–113.
[32] W.K. Härdle, M. Müller, S. Sperlich, A. Werwatz, Nonparametric and Semiparametric Models, Springer, 2004.
[33] D.F. Specht, A general regression neural network, IEEE Trans. Neural Netw. 2 (6) (1991) 568–576.
[34] R.J. Hyndman, G. Athanasopoulos, Forecasting: Principles and Practice, second ed., OTexts, Melbourne, Australia, 2018, . (Accessed on 4 October 2019).
[35] R.J. Hyndman, A.B. Koehler, J.K. Ord, R.D. Snyder, Forecasting with Exponential Smoothing: The State Space Approach, Springer, 2008.
[36] P. Pełka, G. Dudek, Pattern-based forecasting monthly electricity demand using multilayer perceptron, in: Proc. Conf. Artificial Intelligence and Soft Computing ICAISC 2019, in: LNAI 11508, Springer, Cham, 2019, pp. 663–672.
[37] P. Pełka, G. Dudek, Neuro-fuzzy system for medium-term electric energy demand forecasting, in: Proc. Conf. Information Systems Architecture and Technology ISAT 2017, AISC 655, Springer, Cham, 2018, pp. 38–47.
[38] P. Pełka, G. Dudek, Pattern-based long short-term memory for mid-term electrical load forecasting, in: 2020 International Joint Conference on Neural Networks, IJCNN, Glasgow, United Kingdom, 2020, pp. 1-8.
[39] D.W. Scott, Multivariate Density Estimation: Theory, Practice, and Visualization, Wiley, 1992.
[40] S. Smyl, A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting, Int. J. Forecast. 36 (1) (2020) 75–85.
[41] T. Popławski, G. Dudek, J. Łyp, Forecasting methods for balancing energy market in Poland, Int. J. Electr. Power Energy Syst. 65 (2015) 94–101.