Short-Term Load Forecasting using Random Forests

Grzegorz Dudek

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

dudek@el.pcz.czest.pl

Abstract. This study proposes using a random forest model for short-term electricity load forecasting. This is an ensemble learning method that generates many regression trees (CART) and aggregates their results. The model operates on patterns of the time series seasonal cycles which simplifies the forecasting problem especially when a time series exhibits nonstationarity, heteroscedasticity, trend and multiple seasonal cycles. The main advantages of the model are its ability to generalization, built-in cross-validation and low sensitivity to parameter values. As an illustration, the proposed forecasting model is applied to historical load data in Poland and its performance is compared with some alternative models such as CART, ARIMA, exponential smoothing and neural networks. Application examples confirm good properties of the model and its high accuracy.

Keywords: Short-term load forecasting, seasonal time series forecasting, random forests.

1 Introduction

Short-term load forecasting (STLF) is necessary for economic power generation and system security. The accurate load forecasts lead to lower operating cost which contributes to savings in electric utilities. The importance of STLF accuracy has become even more evident for the deregulated electricity markets. To correct the forecast inaccuracy the utility has to buy or sell power in the real time market but it comes at the expense of higher real time prices. For these reasons, STLF is an integral part of planning and operation for electric utilities, regional transmission organizations, energy suppliers, financial institutions, and participants in the generation, transmission, and distribution of electricity. The key importance of STLF is reflected in the literature by many forecasting methods that have been applied, including conventional methods and new computational intelligence and machine learning methods. The STLF is a complex problem because the load time series is nonstationary in mean and variance, with trend and multiple seasonal cycles (daily, weekly and annual).

The most commonly employed conventional STLF methods are the Holt-Winters exponential smoothing (ES) [1] and the autoregressive integrated moving average (ARIMA) models [2]. In ES the time series is decomposed into trend and seasonal

components. An disadvantage of the ES models is that they involve initialization and updating of many terms (level, periods of the intraday and intraweek cycles). ARIMA models can be extended for the case of multiple seasonalities but a combinatorial problem of selecting appropriate model orders is an inconvenience. This order selection process is considered subjective and difficult to apply. Another disadvantage is the linear character of the ARIMA models.

The rapid development of computational intelligence has brought new methods of STLF [3]. They are based mostly on artificial neural networks (ANNs) and fuzzy logic but also on support vector machines, clustering methods and expert systems.

The multilayer perceptron (ANN which is most often applied in STLF) is an attractive tool to modeling of nonlinear problems due to its universal approximation property. But learning of ANN is not easy, because of its complex structure and many parameters (hundreds or even thousands of weights to estimate).

Fuzzy logic models allow us to enter input information by rules formulated verbally by experts and describing the behavior of complex systems by using linguistic expressions. Since it is difficult to gain knowledge directly from the experts, to generate a set of if-then rules the learning from examples procedure is applied in neuro-fuzzy networks. 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.

To sum up, the modeling the nonstationary time series with trend and multiple seasonal cycles usually requires complex model with many parameters. The searching of such a model space is a hard and time consuming process. To find the globally optimal solution intelligent searching methods, such as evolutionary algorithms and swarm intelligence, are often applied. The disadvantages of the complex models are their worse generalization ability, unclear structure and uninterpretable parameters.

In this article we study the random forest as a univariate model for STLF. This is a simple model combining regression trees with only few parameters to estimate. Although the random forests have been used for STLF (see [4]), the novelty of this work is data preprocessing. It simplifies the forecasting problem eliminating nonstationarity and filtering trend and seasonal cycles longer than the daily cycle.

The paper is organized in a theoretical and an empirical part. In the beginning we introduce the main concepts of the random forests. Thereafter we present the STLF methodology based on random forests and patterns of the seasonal cycles of time series. In the last section we use real load data to provide an example of model building and forecasting in practice.

2 Random Forests

Random forests (RFs) are an ensemble learning method for both classification and regression problems [5]. RF is a collection of decision trees that grow in randomly selected subspaces of the feature space. The principle of RFs is to combine a set of binary decision trees (Breiman's CART – Classification And Regression Trees [6]), each of which is constructed using a bootstrap sample coming from the learning sam-

ple and a subset of features (input variables or predictors) randomly chosen at each node. Thus in contrast to the CART model building strategy, an individual tree in RF is built on a subset of learning points and on subsets of features considered at each node to split on. Moreover trees in the forest are grown to maximum size and the pruning step is skipped.

After individual trees in ensemble are fitted using bootstrap samples, the final decision is obtained by aggregating over the ensemble, i.e. by averaging the output for regression or by voting for classification. This procedure called bagging improves the stability and accuracy of the model, reduces variance and helps to avoid overfitting. The bias of the bagged trees is the same as that of the individual trees, but the variance is decreased by reducing the correlation between trees (this is discussed in [7]). Breiman showed that random forests do not overfit as more trees are added, but produce a limiting value of the generalization error [5]. The RF generalization error is estimated by an out-of-bag (OOB) error, i.e. the error for training points which are not contained in the bootstrap training sets (about one-third of the points are left out in each bootstrap training set). An OOB error estimate is almost identical to that obtained by *N*-fold cross-validation. The large advantage of RFs is that they can be fitted in one sequence, with cross-validation being performed along the way. The training can be terminated when the OOB error stabilizes.

The algorithm of RF for regression in Fig. 1 is shown [7].

1. For k = 1 to K:
   1.1. Draw a bootstrap sample L of size N from the training data.
   1.2. Grow a random-forest tree T_k to the bootstrapped data, by recursively repeating the following steps for each node of the tree, until the minimum node size m is reached.
      1.2.1. Select F variables at random from the n variables.
      1.2.2. Pick the best variable/split-point among the F.
      1.2.3. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_k\}_{k=1, 2, \ldots, K}$.

To make a prediction at a new point **x**:

$$f(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} T_k(\mathbf{x}) \quad (1)$$

Fig. 1. Algorithm of RF for regression.

The two main parameters of RF are: the number of trees in the forest K and the number of input variables randomly chosen at each split F. The number of trees can be determined experimentally. During the training procedure we add the successive trees until the OOB error stabilizes. The RF procedure is not overly sensitive to the value of F. The inventors of the algorithm recommend $F = n/3$ for the regression RFs.

Another parameter is the minimum node size m. The smaller the minimum node size, the deeper the trees. In many publications $m = 5$ is recommended. And this is the

default value in many programs which implement RFs. RFs show small sensitivity to this parameter.

It is noteworthy that using CART model, we get a classifier or an estimate of the regression function, which is a piecewise constant function obtained by partitioning the predictor space. This is a serious limitation of CART. But building an ensemble of CART we get results which are much smoother than from a single tree.

Using RFs we can determine the prediction strength or importance of variables which is useful for ranking the variables and their selection, to interpret data and to understand underlying phenomena. The variable importance can be estimated in RF as the increase in prediction error if the values of that variable are randomly permuted across the OOB samples. The increase in error as a result of this permuting is averaged over all trees, and divided by the standard deviation over the entire ensemble. The more the increase of OOB error is, the more important is the variable.

3 Data Preprocessing

Our goal is to forecast the load curve for the next day. The load time series $\{z_l\}_{l=1, 2, ..., L}$ is divided into daily cycles of length n. To eliminate weekly and annual variations the daily cycles are preprocessed to obtain their patterns. The pattern is a vector with components that are functions of actual time series elements. Two types of patterns are defined: the input patterns **x** and output (forecast) ones **y**. The forecast pattern $\mathbf{y}_i = [y_{i,1}\ y_{i,2}\ \ldots\ y_{i,n}]^{\mathrm{T}}$ encodes the successive actual time series elements z_l in the forecasted daily cycle $i+\tau$: $\mathbf{z}_{i+\tau} = [z_{i+\tau,1}\ z_{i+\tau,2}\ \ldots\ z_{i+\tau,n}]^{\mathrm{T}}$, and the corresponding input pattern $\mathbf{x}_i = [x_{i,1}\ x_{i,2}\ \ldots\ x_{i,n}]^{\mathrm{T}}$ maps the time series elements in the daily cycle i preceding the forecast cycle: $\mathbf{z}_i = [z_{i,1}\ z_{i,2}\ \ldots\ z_{i,n}]^{\mathrm{T}}$. Vectors **y** are encoded using current process parameters from the nearest past, which allows to take into account current variability of the process and enables decoding. Some definitions of the functions mapping the original space Z into the pattern spaces X and Y, i.e. $f_x : Z \to X$ and $f_y : Z \to Y$ are presented in [8]. The most popular definitions are of the form:

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

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

The function f_x (1) expresses normalization of the vectors $\mathbf{z}_i$. After normalization they have the unity length, zero mean and the same variance. Note that the nonstationary and heteroscedastic time series is represented by patterns having the same mean and variance.

The forecast patterns $\mathbf{y}_i$ are defined using analogous function to f_x, but the encoding parameters ($\bar{z}_i$ and dispersion measure in the denominator of (2)) are determined

from the process history. This enables decoding of the forecasted vector $\mathbf{z}_{i+\tau}$ after the forecast of pattern $\mathbf{y}_i$ is determined. We use the inverse function $f_y^{-1}(y_{i,t})$ for this.

From the set of pairs $\{(\mathbf{x}_1,y_1), ..., (\mathbf{x}_N,y_N)\}$, where y_i represents the load at hour t for the next day (this is the t-th component of pattern $\mathbf{y}_i$ for $\tau = 1$), the learning set for RF is generated. For each query point (pattern $\mathbf{x}$ representing the k-th daily period) the learning set is prepared individually from the historical data. It contains M nearest neighbors of the query pattern representing the same days of the week (Monday, ..., Sunday) as the query pattern. This restriction to M nearest neighbors is due to our goal: we do not want to built a global model but a local one, which is competent for the query pattern. So there is no sense to use the distant learning points to train the model. Of course this model is not suitable for other query points and we have to built the separate model for each query point. But the cost and time of model building in this case are not limiting factors.

4 Application Examples

We illustrate the construction of RF forecasting model on the example of STLF using the hourly electrical load data of the Polish power system from the period 2002–2004. (This data can be downloaded from the website http://gdudek.el.pcz.pl/varia/stlf-data.) Our goal is to forecast the load curve for the next day ($\tau = 1$). The test set includes 30 days from January 2004 (without untypical 1 January) and 31 days from July 2004. The training set containing $M = 50$ pairs $(\mathbf{x}_i,y_i)$ is generated individually for each forecasting task (load forecasting at hour t of the day j). In our example there are $(30+31)\cdot 24 = 1464$ forecasting tasks. For each of them the separate RF model is created.

In the first phase of our research we investigate how the model parameters affect an error. In Fig. 1 the OOB errors against K and F are plotted. From these figures it can be seen that MSE drops from $K = 1$ to K about 100 and then stabilize. When F increases to 8 MSE decreases and then gradually increases. Hence it is assumed that the best values of these parameters are: $K = 100$ and $F = 8$, i.e. $n/3$.

Fig. 2 shows OOB error depending on the minimum node size m and the frequencies of m values ensuring the lowest errors. This figure demonstrates the best accuracy for deepest trees in ensemble.

The variable importance for several forecasting tasks in Fig. 3 is shown. It is hard to formulate a rule concerning the variable importance observing these figures. In many cases the variable importance graphs in forecasting tasks for neighboring hours of the same day vary considerably. Some variables have negative importance. This indicates that permutation of these variable values leads to a lower error.

Now we compare in simulations the proposed RF model with CART models in two variants: typical and with fuzzy nodes [9]. In the typical variant the trees were grown until the minimum node size m was reached. This parameter was adjusted individually for each forecasting task. In the fuzzy CART variant the trees were constructed in a classical way and then the crisp tests in nodes were replaced with fuzzy tests. The fuzzy test determines the membership degrees to the branches outgoing from the

node. The slope parameters of the membership functions were tuned for each test node and for each forecasting task individually.

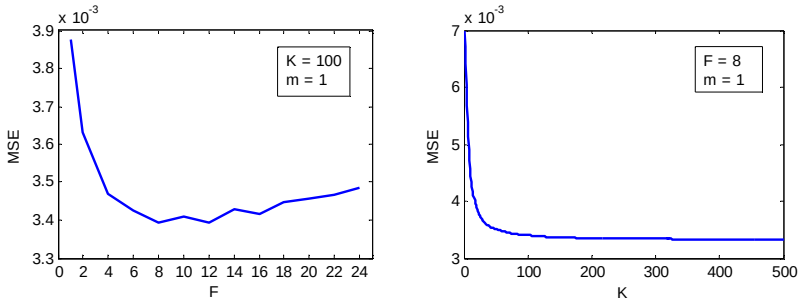

Fig. 2. OOB MSE depending on the number of trees (left) and the number of input variables randomly chosen at each split (right).

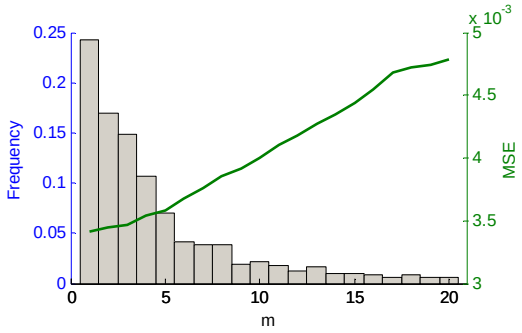

Fig. 3. OOB MSE depending on the minimum node size (line) and the frequencies of *m* values (bars) ensuring the lowest errors.

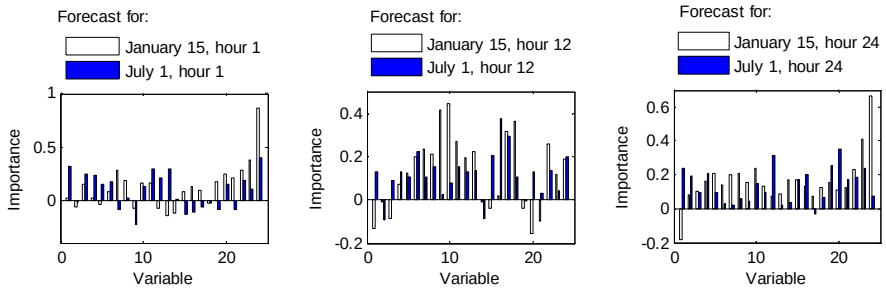

Fig. 4. The variable importance for several forecasting tasks.

We compare the RF model also with popular STLF models such as: ARIMA, exponential smoothing (ES) and artificial neural network (ANN). These models are described in [10]. The time series are preprocessed for ANN in the same way as for RF (patterns of the daily periods are used). The ANN learns using the training sample

selected from the neighborhood of the query pattern (local learning). As shown in [10] for STLF in local learning procedure the one-neuron model is sufficient. To find the best ARIMA and ES models automated procedures implemented in the **forecast** package for the **R** system [11] were used.

MAPE (mean absolute percentage error) is adopted here to assess the performance of the forecasting models. The results of the forecasts (MAPE for the test samples $MAPE_{tst}$ and the interquartile range (IQR) of $MAPE_{tst}$) in Table 1 are presented. In this table the results determined using the naïve method are also shown. The forecast rule in this case is as follows: the forecasted daily cycle is the same as seven days ago. From table 1 it can be seen that the lowest errors were obtained by ANN and RF. Mean errors for RF and ANN are statistically indistinguishable (Wilcoxon signed-rank test was used).

The histograms of the percentage errors (PE) in Fig. 5 are shown. The most favorable error distributions are observed for RF and ANN. The distributions for ARIMA and ES are more flattened and asymmetrical.

Table 1. Results of forecasting.

Model	January		July		Mean	
	$MAPE_{tst}$	IQR	$MAPE_{tst}$	IQR	$MAPE_{tst}$	IQR
RF	1.42	1.39	0.92	0.98	1.16	1.17
CART	1.70	1.58	1.16	1.17	1.42	1.39
Fuzzy CART	1.62	1.47	1.13	1.12	1.37	1.35
ARIMA	2.64	2.34	1.21	1.24	1.91	1.67
ES	2.35	1.88	1.19	1.30	1.76	1.56
ANN	1.32	1.30	0.97	1.01	1.14	1.15
Naïve	6.37	5.36	1.29	1.20	3.78	3.82

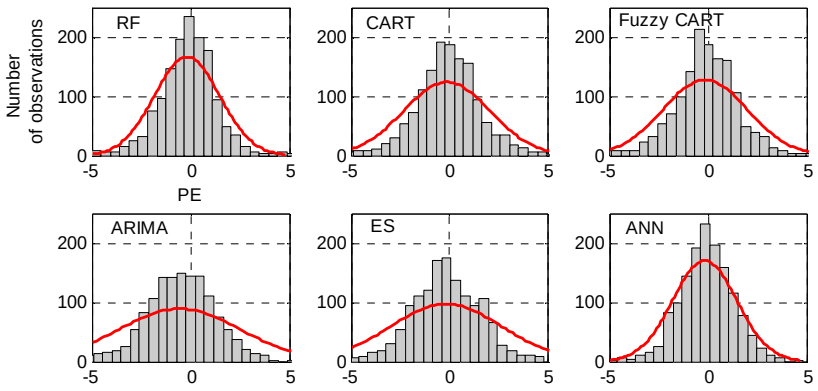

Fig. 5. Histograms of errors.

5 Conclusions

The purpose of the present study was to ascertain the effectiveness of using the RF models in STLF. The proposed approach allows us to forecast time series with multiple seasonal variations. It is due to the data preprocessing and defining the patterns of the seasonal cycles on which the model operates. The target function is approximated locally in the neighborhood of the query point. This simplify the forecasting problem and leads to the better accuracy.

The RF forecasting model is characterized by simplicity. The number of parameters to be estimated is small, which implies a simple procedure of the model optimization. This task is facilitated by the built-in cross-validation mechanism. It is worth noting that the model is not very sensitive to the parameter values.

In application examples the RF model provided as accurate forecasts as ANN and outperformed the crisp and fuzzy CART, ARIMA and ES models. The RF model is simpler to train and tune than the above mentioned models, does not overfit and reduces variance due to averaging the outputs of many simple regression trees (weak learners) over ensemble.

References

1. Taylor J.W., Snyder R.D.: Forecasting Intraday Data with Multiple Seasonal Cycles Using Parsimonious Seasonal Exponential Smoothing. Omega 40(6), 748–757 (2012).
2. Weron R.: Modeling and Forecasting Electricity Loads and Prices. Wiley (2006).
3. Kodogiannis V.S., Anagnostakis E.M.: Soft Computing Based Techniques for Short-Term Load Forecasting. Fuzzy Sets and Systems 128, 413–426 (2002).
4. Ying-Ying Cheng, Chan P.P.K., Zhi-Wei Qiu: Random Forest Based Ensemble System for Short Term Load Forecasting. Proc. Machine Learning and Cybernetics (ICMLC), vol.1, 52–56 (2012).
5. Breiman L.: Random Forests. Machine Learning 45 (1), 5–32 (2001).
6. Breiman L., Friedman J.H., Olshen R.A., Stone C.J.: Classification and Regression Trees. Chapman & Hall (1984).
7. Hastie T., Tibshirani R., Friedman J.: The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Springer (2009).
8. Dudek, G.: Similarity-based Approaches to Short-Term Load Forecasting. In: Zhu, J.J., Fung, G.P.C. (eds.): Forecasting Models: Methods and Applications. iConcept Press, 161–178 (2010).
9. Dudek G.: Short-Term Load Forecasting Using Fuzzy Regression Trees. Przegląd Elektrotechniczny (Electrical Review) 90(4), 108–111 (2014) (in Polish).
10. Dudek G.: Forecasting Time Series with Multiple Seasonal Cycles using Neural Networks with Local Learning. In: Rutkowski L. et al. (eds.): Artificial Intelligence and Soft Computing, ICAISC 2013, LNCS 7894, 52–63 (2013).
11. Hyndman, R.J., Khandakar, Y.: Automatic Time Series Forecasting: The Forecast Package for R. Journal of Statistical Software 27(3), 1–22 (2008).