Expert Systems with Applications 40 (2013) 4427–4437

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Load forecasting using a multivariate meta-learning system

Marin Matijaš [a,b,*], Johan A.K. Suykens [a], Slavko Krajcar [b]

[a] Department of Electrical Engineering, ESAT-SCD-SISTA, KU Leuven, B-3001, Leuven, Belgium
[b] Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

ARTICLE INFO

Keywords:
Electricity consumption prediction
Energy expert systems
Industrial applications
Short-term electric load forecasting
Meta-learning
Power demand estimation

ABSTRACT

Although over a thousand scientific papers address the topic of load forecasting every year, only a few are dedicated to finding a general framework for load forecasting that improves the performance, without depending on the unique characteristics of a certain task such as geographical location. Meta-learning, a powerful approach for algorithm selection has so far been demonstrated only on univariate time-series forecasting. Multivariate time-series forecasting is known to have better performance in load forecasting. In this paper we propose a meta-learning system for multivariate time-series forecasting as a general framework for load forecasting model selection. We show that a meta-learning system built on 65 load forecasting tasks returns lower forecasting error than 10 well-known forecasting algorithms on 4 load forecasting tasks for a recurrent real-life simulation. We introduce new metafeatures of fickleness, traversity, granularity and highest ACF. The meta-learning framework is parallelized, component-based and easily extendable.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

Load forecasting has a goal to predict future electric energy consumption or power load. It is important for power systems planning, power market operation, power market design, power systems control and security of supply. In power systems planning, long term load forecasting (LTLF) is an important input for decisions on power system development. In power market operation, market participants use load forecasting for managing their costs and strategies. Keeping a low balancing cost is important due to low profit margins in the industry. A conservative estimate by (Hobbs et al., 1999) shows that a decrease of the load forecasting error in terms of mean absolute percentage error (MAPE) by 1% lowers the variable production cost between 0.6 and 1.6 million USD annually for a 10,000 MW utility with MAPE around 4%.

Many market participants have dozens of considerably different load forecasting tasks. As these tasks appeared throughout time they are often solved with different software, algorithms and approaches in order to keep the particular knowledge and lowest possible load forecasting error. Today market participants access many different electricity markets at the same time. Load forecasting tasks are different throughout these electricity markets. It would be beneficial for all but especially for the small market participants when they could have at their disposal a solution that will give them lowest load forecasting error for all of their heterogeneous load forecasting tasks. It would be beneficial if that solution can play a role of expert for them and rank the available algorithms for their particular needs.

To address these problems we propose an approach based on meta-learning for multivariate and univariate load forecasting. Meta-learning algorithms successfully learn on past performance of different approaches and give better results than various single algorithms because meta-learning algorithms can learn the characteristics of each task. Based on the notion that for similar tasks, forecasting algorithms will have similar ranking by performance, meta-learning can predict the ranking of the algorithms without the need to run all the algorithms on a new task which can be computationally expensive.

This paper is organized as follows: in succession we present an overview of the load forecasting and meta-learning fields. The description of the meta-learning system we propose is in Section 2. followed-up by the experiment and the results in Section 3. and concluding remarks in the last section.

1.1. Load forecasting

A good overview of the recent development of load forecasting is present in the recent surveys (Alfares & Nazeeruddin, 2002; Hahn, Meyer-Nieberg, & Pickl, 2009; Tzafestas & Tzafestas, 2001). Although the annual number of scientific papers on load forecasting has increased from around one hundred in 1995 to more than a thousand in recent years (SCOPUS, 2012), the majority of proposed approaches are suited to specific, regional data (Dannecker et al., 2010). Recently Wang, Xia, and Kang (2011) proposed a hybrid

* Corresponding author at: Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia. Tel.: +385 12220935.
E-mail address: marin.matijas@fer.hr (M. Matijaš).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.01.047

two-stage model for Short-Term Load Forecasting (STLF). Based on the influence of relative factors to the load forecasting error their model selects the second stage forecasting algorithm between linear regression, dynamic programming and SVM. Espinoza, Suykens, Belmans, and De Moor (2007) have proposed Fixed-size LSSVM using ARX-NARX structure and showed that it outperformed linear model and LS-SVM in the case of STLF using large time-series. Hong (2011) proposed a new load forecasting model based on seasonal recurrent support vector regression (SVR) that uses chaotic artificial bee colony for optimization. It performed better than ARIMA and the trend fixed seasonally adjusted ε-SVR. Taylor (2012) recently proposed several new univariate exponentially weighted methods of which one using singular value decomposition has shown potential for STLF .

1.2. Why Meta-learning?

We have chosen meta-learning because it can learn on its past knowledge of solving different tasks, and it enables building on the existing wealth of algorithms. It is more complex than approaches it is competing with, like single algorithms and algorithm combinations because they are used to build it. Theoretically, it can be infinitely large by putting meta-learning as components of a bigger meta-learning system. Unlike some systems used as a support for decision making, meta-learning can address new types of tasks, e.g. it can solve a LTLF task if it did not see one before, but has previously solved STLF tasks.

1.3. Meta-learning

Based on the No Free Lunch Theorem for supervised learning (Wolpert, 1996), no single algorithm has the lowest load forecasting error on all load forecasting tasks (Tasks). Examples of three Tasks are STLF of a small industrial facility, a Medium-Term Load Forecast of a whole supply area and a LTLF of the whole country. The selection of the best algorithm for each single Task can be a hard problem due to the size of the search space of possible algorithms. Rice proposed a formalized version of an algorithm selection problem as follows: for a given task in a problem space $x \in P$ with features $f(x) \in F$, find the selection algorithm $S(f(x))$ in algorithm space A, in the way that the selected algorithm $a \in A$ maximizes the performance mapping $z(a(x)) \in Z$ in terms of a performance measure π (Rice, 1976). In the machine learning community this problem has been recognized as a learning task and was named meta-learning, or learning about learning.

In a meta-learning system, features F from the Rice's formulation are called metafeatures and they represent inherent characteristics of a given task x. For a load forecasting task x, F can be composed of the skewness of the load, the kurtosis of the load and the number of exogenous features that are available for the load forecasting. If we gather enough knowledge about different Tasks and load forecasting error of distinct algorithms on them, we can rank algorithms by size of the load forecasting error for each of those Tasks. Based on the characteristics of a new Task, algorithms can be ranked on the assumption that for Tasks with similar characteristics (metafeatures) the same algorithm will return the similar load forecasting error. In this way we do not have to test all algorithms and parameter combinations on every new Task which would take a long runtime. More theoretical background and examples of meta-learning can be found in Giraud-Carrier (2008) and Brazdil, Giraud-Carrier, Soares, and Vilalta (2009).

Efforts of meta-learning which include its application to forecasting have been summarized in Smith-Miles (2008). More recently, Wang, Smith-Miles, and Hyndman (2009) proposed a meta-learning on the univariate time-series using four forecasting methods and a representative database of univariate time-series of different and distinct characteristics. Their results show that ARIMA and NN are interchangeably the best depending on the characteristics of the time-series while ES models and random walk (RW) lagged in forecasting performance. They demonstrated superiority of meta-learning through rule-based forecasting algorithm selection with their CBBP approach being 28.5% better than RW while ARIMA was 27.0% better than RW. On the NN3 and NN5 competition datasets, Lemke and Gabrys have built an extensive pool of meta-features. They have shown that a meta-learning system outperforms approaches representing competition entries in any category. On NN5 competition dataset their Pooling meta-learning had SMAPE of 25.7 which is lower than 26.5 obtained by Structural model, the best performing of 15 single algorithms (Lemke & Gabrys, 2010). If an approach with performance close to or better than the meta-learning system is found, many meta-learning approaches can include those candidates thus becoming better.

We propose to add the ensemble for classification in the meta-learning system for regression and include promising algorithms meta-learning systems did not use so far.

2. Proposed meta-learning

2.1. General set-up

While majority of the load forecasting and meta-learning approaches learn on a single level, the proposed meta-learning system learns on two levels: load forecasting task (Task) level and meta-level. The working of the proposed meta-learning system is depicted in Fig. 1. The learning at the forecasting level is represented by the lower right cloud in which names of the forecasting algorithms composing it are written. Feature space and feature selection at the forecasting level should be smaller clouds in that cloud, but are not illustrated in Fig. 1 due to simplicity. Meta-level is represented in Fig. 1 as the arrows between all five clouds. Learning at the meta-level is in the ensemble which is illustrated by a central cloud with seven words representing classification algorithms the ensemble consists of. Metafeatures created for each Task are the input data for the ensemble and they make the basis for learning on the meta-level. For all the Tasks in the meta-learning system (except the new ones) the performance of forecasting algorithms is calculated earlier and is available at the meta-level, it is the output data (label) which the ensemble uses for the classification. Using the notion that for similar Tasks, algorithms will have similar ranking, the proposed meta-learning system associates the algorithm ranking to a new Task based on an ensemble of:

- Euclidean distance,
- CART Decision tree,
- LVQ network,
- MLP,
- AutoMLP,
- ε-SVM and
- Gaussian Process (GP).

Our meta-learning system is modular and component-based which makes it easily extendable. It consists of the following modules:

- Load data,
- Normalization,
- Learn metafeatures,
- Feature selection,
- Forecasting and

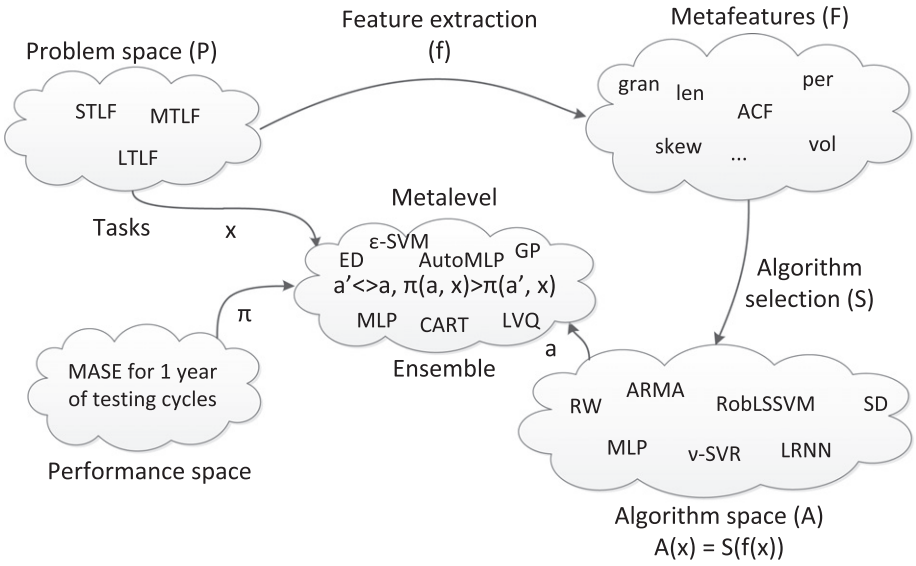


Fig. 1. The working of the proposed meta-learning system seen through Rice's paradigm.

- Error calculation and ranking.

The flowchart of the meta-learning system made of these modules is shown in Fig. 2. In the first module Load data, parameters are set and a new Task is loaded.

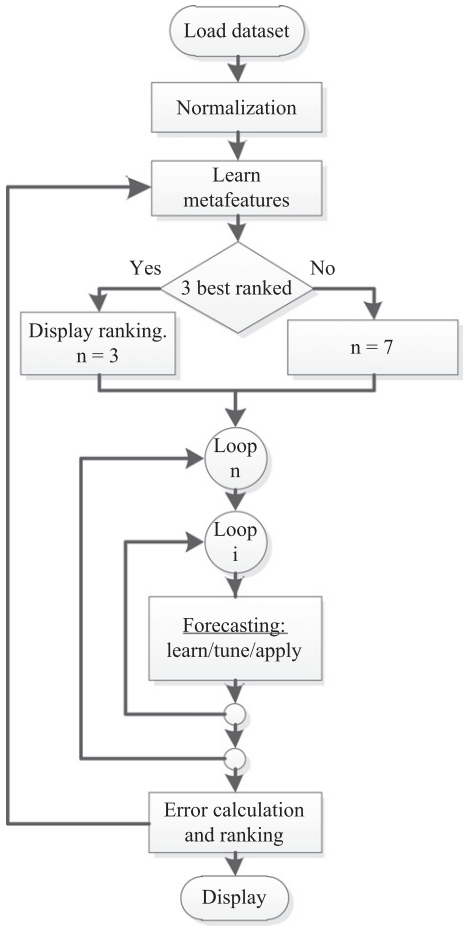


Fig. 2. The flowchart of the meta-learning system that we propose shows the order of the modules, ranking feedback from learning to meta-learning level and two outer loops of the forecasting module.

The second module, Normalization comes in the following variants: Standardization, [0, 1] scaling, [−1, 1] scaling and Optimal combination. We apply normalization to the data in order to get it on the same scale. Neural networks and support vector machines use data normalized at this point as the input data for forecasting.

The learn metafeatures module creates the following metafeatures for each Task: *Minimum*, *Mean*, *Standard deviation*, *Skewness*, *Kurtosis*, *Length*, *Granularity*, *Exogenous*, *Periodicity*, *Highest ACF*, *Traversity*, *Trend* and *Fickleness*. Those features have been selected with ReliefF (Kononenko, 1994) feature ranking for classification that we give in Section 2.2.

Minimum represents the minimum value of the load before normalization. *Mean* represents the mean value of the load. For a load time series Y, mean $\overline{Y}$ is calculated as $\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$, where n is the number of data points of a time-series. *Standard deviation* σ is calculated as

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})^2} \tag{1}$$

Skewness S is the measure of a lack of symmetry and is calculated as

$$S = \frac{1}{n\sigma^3}\sum_{i=1}^{n}(Y_i - \overline{Y})^3 \tag{2}$$

Kurtosis K is the measure of flatness relative to a normal distribution

$$K = \frac{1}{n\sigma^4}\sum_{i=1}^{n}(Y_i - \overline{Y})^4 \tag{3}$$

Length is the number of data points n. *Granularity* g is the distance in time between the two data points in a series Y. Exogenous n_f is the number of exogenous features used in the particular model. *Periodicity* of a time series *per* is the smallest number of data points that repeats in a time-series. It is an index of a highest autocorrelation function (ACF) lag after at least one local minimum of the ACF. If its difference to the global minima of the ACF is not greater than 0.2 or it is not found, *per* is 0. For load time series of hourly *granularity*, *per* is frequently 168 and for majority load time-series of monthly *granularity*, *per* is 12. *Highest ACF* h_{ACF} is the value of the ACF at the periodicity lag. *Traversity trav* is a standard deviation of the difference between time-series Y and Y_{per} where Y_{per} is the *per*-th ACF lag of Y. *Trend tr* is the linear coefficient of the linear regression of Y. *Fickleness fic* is the ratio of the number of times a

time-series reverts across its mean and the length of the time-series n, $fic = \frac{1}{n}\sum_{i=2}^{n} I_{\{sgn(Y_{i-1}-\overline{Y})\neq sgn(Y_i-\overline{Y})\}}$ where $I_{\{sgn(Y_{i-1}-\overline{Y})\neq sgn(Y_i-\overline{Y})\}}$ denotes the indicator function. New metafeatures can be easily added and the forecasting error ranking recalculated without the need to repeat any computationally expensive parts like forecasting.

The second part of this module has the modes learn and work. In the learn mode it will pass onto the following module that all the combinations have to be tried. The learn mode is used only once for the initial meta-learning system creation. In the work mode, all the learn metafeatures algorithms run with equally weighted votes in the ensemble. Ranking is based on a Gaussian Process updated by the best ranked result of the ensemble. Based on the chosen selection, meta-learning is conducted between normalized metafeatures of all Tasks in the database. Later, for example during lower computer load time, it is possible to calculate forecasts for all other algorithms and combinations for a new Task and extend the meta-learning system with that information.

Meta-learning with metafeatures of all the Tasks presents a multiclass classification problem. The number of classes k is equal to the number of algorithms building the meta-learning system which is 7. On the meta-level, each Task is a data point, represented by a d-dimensional vector where d is the number of metafeatures. To solve this multiclass classification problem for which input are values of 13 metafeatures and the label is the forecasting algorithm ranking, we use an ensemble of equally weighted Euclidean distance, CART decision tree, LVQ network, MLP, AutoMLP, ε-SVM and GP. For the latter four algorithms we used 5-fold cross-validation (CV).

Euclidean distance gives the algorithm ranking by taking the best ranked algorithms of the Tasks sorted ascending by the Euclidean distance of the metafeatures.

CART decision tree works by minimizing Gini impurity index (GI) which is a measure of misclassified cases over a distribution of labels in a given set. For each node Gini impurity index is equal to

$$GI = 1 - \sum_{i=1}^{k} r_i^2 \tag{4}$$

where r_i is the percentage of records in class i.

LVQ network is a supervised network for classification. Based on good empirical results we use the topology 13-14-7-(1) with LVQ1 learning rule and 50 training epochs. The LVQ network shown in Fig. 3 illustrates the learning by firing a neuron for a predicted class of a new Task.

MLP is a well-known type of neural networks about which more will be given in the next subsection. We used 6 hidden layer neurons, Levenberg–Marquardt, momentum 0.2 and learning rate 0.3.

AutoMLP (Breuel & Shafait, 2010) is an ensemble of MLPs that uses genetic algorithms and stochastic optimization to find the best network combinations in terms of learning rates and numbers of hidden neurons. We use an implementation with 4 ensembles and 10 generations.

ε-SVM is standard Vapnik's SVM for classification. We optimize C and γ with grid search.

GP (Rasmussen & Williams, 2006) is a kernel based method and it is well-known for probabilistic classification where we employ it. We compared three different versions combining RBF, Epanechnikov and combinations of three Gaussians all using grid search for optimization. We use RBF GP because it had the best performance. The ranking is obtained as:

$$R_{i,j} = \begin{cases} \max count(R_{1,j,k}), & \forall j,\ \forall k, \\ R_{i,j,7}, & i > 1,\ \forall j \end{cases} \tag{5}$$

where $R_{i,j}$ is the ensemble ranking and $R_{i,j,k}$ is the ranking of the forecasting algorithms for ith place, where j is the Task index and k is the classification algorithm index, such that index of GP is 7. Those rankings are on the forecasting level, and they are based on MASE for all of the cycles of each Task. The feature selection module has the following options: Default, All and Optimize lags. The Default option is a result of a long empirical testing by adapting to the longest time-series. All is the option in which the whole feature set is used in the forecasting. This approach does not lead to optimal results because in practice unimportant features increase the load forecasting error. Optimize lags iteratively changes different selection of lags up to the periodicity of the load time-series and forwards it to the forecasting part together with other features. This way the best ARX/NARX feature selection is found for a given time-series with a disadvantage of long runtime because of the dense search in the feature space.

2.2. Forecasting module

The forecasting module is the core of this meta-learning system. It consists of the following algorithms:

1. Random Walk (RW) algorithm,
2. Autoregressive Moving Average (ARMA) algorithm,
3. Similar Days algorithm,
4. Layer Recurrent Neural Network (LRNN) algorithm,
5. Multilayer Perceptron (MLP),
6. ν-Support Vector Regression (ν-SVR) and
7. Robust LS-SVM (RobLSSVM).

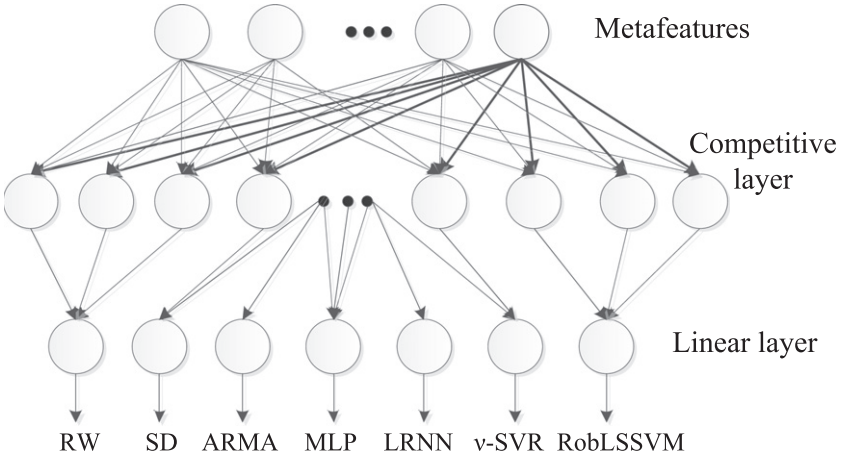


Fig. 3. An illustration of one of the ensemble components, LVQ network in which neuron activation gives an optimal solution for a certain Task.

Except for RW, which is implemented in a vectorized form, other algorithms are implemented in an iterative fashion. Iterations are based on the learning set and the test set. Algorithms 4–7 use a validation set, too. The test set consists of new data points for which the load is unknown and the forecast is calculated. The forecasting in every iteration is a sequence of one-step-ahead point forecasts in which a new step-ahead forecast is made based on the previous prediction. The size of the learning set and the test set are determined at the beginning and can be changed. We tuned the size of the learning set to have runtime usable in real-life applications and the size of test set depends on the type of load forecasting. We give the values in Section 3.1.

All algorithms will perform univariate load forecasting if no exogenous data are provided to the algorithm. If exogenous data are provided, all algorithms except RW and ARMA use the data to perform the multivariate forecasts. In case an algorithm performs the multivariate forecast, no univariate forecast will be calculated with the same algorithm at the particular simulation. The multivariate approaches to load forecasting return the lower forecasting errors in general and are used in majority of the load forecasting. In some cases exogenous data are not available to the forecaster or univariate approaches outperform multivariate ones (Taylor, 2012).

All algorithms that minimize the load forecasting error in this paper use the mean squared error (MSE). This approach is not optimal but it is employed here and in practice because the load forecasting error is the most expensive and it is the hardest to forecast when the load is high, which MSE captures well.

When performing validation, learning based algorithms use the 10-fold CV.

1. Algorithm: RW is a time-series forecasting approach used for the estimate of the upper error bound. For a load time-series Y, the time-series of load predictions $\widehat{Y}$ is calculated as $\widehat{Y}_i = Y_{i-per} + e_i$, where e_i is the white noise which is uncorrelated from time to time. If $per = 0$, median *per* for same *granularity* is used. This RW slightly differs from a common RW approach and uses *per* data points in the past instead of 1 because this whole meta-learning system is made for the real-world application which implies that each forecast is made for a period in the future. In the typical case of the hourly granularity, RW relies on the values of the same hour one week in the past. It is not sensitive to the outliers in the few most recent data points because it uses older data. Learning based algorithms have their predictions checked against it by $|\widehat{Y}_i - \widehat{Y}_{i,RW}| > 6\sigma_d$, where $6\sigma_d$ is a standard deviation of the difference of point forecasts of $\widehat{Y}$ using a learning based algorithm and the RW, $\widehat{Y}_{RW}$. Data were manually inspected for quality in those cases.
2. Algorithm: ARMA, autoregressive moving-average or Box–Jenkins model is calculated using ARMASA Toolbox (Broersen, 2006). The load is preprocessed and univariate $ARMA(p,q)$ is detected automatically. We are using ARMA modelled as:

$$\widehat{Y}_i = \sum_{j=1}^{p} \varphi_j Y_{i-j} + \sum_{j=1}^{q} \theta_j e_{i-j} + e_i \tag{6}$$

where φ_j are the parameters of its autoregressive part $AR(p)$, θ_j are the parameters of the moving average part $MA(q)$ and e_i is the white noise. $AR(p)$ is used for estimation in the identification of the best model, because it is easier to obtain its parameters than those of the autocorrelation function of $MA(q)$. The parameters are obtained using Burg's method that relies on Levinson–Durbin recursion. With the parameters of the ARMA model based on the learning set, a point forecast for a given number of steps ahead is calculated.

3. Algorithm: Similar Days is the only algorithm used in the proposed meta-learning system that has limits based on granularity. It can be applied only to data of granularity lower than daily. The data are reorganized in sequences of daily data. Based on the Euclidean distance of the features in the learning set, the algorithm finds n_c similar days and calculates the median for each particular hour in the forecasting horizon. We propose the following implementation:

$$\min_i = \sum_{i=1}^{D-1} \sum_{j=1}^{n_f} \sum_{l=1}^{h} \sqrt{X_{D,j,l}^2 - X_{i,j,l}^2} \tag{7}$$

where i is the index of the period for which distance is being minimized to the forecasted period indexed with D, n_f is the number of all features used for Euclidean distance calculation, j is the index over those features, h is the number of data points in each period (day), l is index over those data points and X is the data point in each feature.

The minimization (7) is iteratively repeated n_c times for each data point in D, every time removing data corresponding to the solution from the set and remembering it as o_j where j is incremented by 1. In the end it leaves n_c most similar days with indexes $o_1, o_2, \ldots, o_{n_c}$. We have previously tested the number of similar days between 3 and 14 using different datasets and empirically 5 returns the lowest forecasting error. Forecast is obtained by taking the median value of loads in n_c most similar days for each data point (hour):

$$\widehat{Y}_D = \widetilde{Y}_{o_1, o_2, \ldots, o_{n_c}} \tag{8}$$

where $\widetilde{Y}_{o_1, o_2, \ldots, o_{n_c}}$ is the median of the corresponding hour in most similar days.

4. and 5. Algorithm: For neural networks, where applicable, the following has been used: 1 hidden layer, input delays optimized between 1 and 2, feedback delays optimized between 1 and 2, hidden layer size optimized between 6 and 12, validation set randomly selected and ceiled at 80% of the learning set, hidden layer activation function is hyperbolic tangent sigmoid. For all neural networks, weights and biases have been backpropagated using the Levenberg–Marquardt optimization using early stopping criterion for 6 consecutive validation vector checks.
4. Algorithm: MLP is a well-known type of neural networks known for its absolute generalization ability. It is a static nonlinear model which can be described as: $Y = W\text{tansig}(VX + \beta)$, where $X \in \mathbb{R}^n$ is the input feature set, $Y \in \mathbb{R}^{n_y}$ are the target forecasted values, $\beta \in \mathbb{R}^{n_h}$ is a vector of biases which are threshold values of n_h hidden neurons in the only hidden layer, $W \in \mathbb{R}^{n_h \times n}$ is the interconnection matrix for the output layer and $V \in \mathbb{R}^{n_y \times n_h}$ is the interconnection matrix for the hidden layer.

We are using the batch learning for which the optimization problem can be written as:

$$\min \frac{1}{n} \sum_{j=1}^{n} \|Y_j - f(X_j, \theta)\|_2^2 \tag{9}$$

where $\theta = [W; V; \beta]$. The used NARX model structure is $\widehat{Y}_i = f(Y_{i-1}, \ldots, Y_{i-n_y}, \ldots, X_{i-1}, \ldots, X_{i-n_y})$, with f parameterized by a multilayer perceptron, where n_y is the number of previous data points used in model creation. We couple Matlab implementations of narxnet and timedelaynet in Neural Network Toolbox and choose the one with a better test performance. Due to the NARX nature of the multivariate load forecasting we use MLP as a viable candidate for load forecasts.
5. Algorithm: LRNN are dynamical networks, similar to Distributed Delay, Time Delay and Elman Neural Networks because their hidden layers have a recurrent connection with a tap

delay. We use LRNN with the model structure $\widehat{Y}_i = f(Y_{i-1}, \ldots, Y_{i-n_y}, \ldots, X_{i-1}, \ldots, X_{i-n_y})$. It has one feedback loop and one step delay around the only hidden layer. This recurrent connection makes the dynamic response of the network to the input data infinite.

6. Algorithm: Vapnik and other researchers developed statistical learning theory and introduced SVM and later its version for regression SVR (Vapnik, 1998). v-SVR is the version of SVR proposed by Schölkopf et al. in which v was introduced to control the number of support vectors and the training error thus replacing ε as a parameter (Scholköpf, Smola, Williamson, & Bartlett, 2000). Slack variables $\left(\xi_j, \xi_j^*\right)$ capture everything with an error greater than ε. Via a constant v, the tube size ε is positive number chosen as a trade-off between model complexity and slack variables as:

$$\min\left\{C\left[v\xi + \frac{1}{n}\sum_{j=1}^{n}\left(\xi_j + \xi_j^*\right)\right] + \frac{1}{2}\|w\|^2\right\} \tag{10}$$

subject to :

$$Y_i - w^T X_i - b \leqslant \varepsilon + \xi_j$$
$$w^T X_i + b - Y_i \leqslant \varepsilon + \xi_j^*$$
$$\xi_j, \xi_j^* \geqslant 0.$$

We use the RBF kernel. Parameter optimization for C and γ is made using the grid search in the version 3.11 of the LibSVM (Chang & Lin, 2011).

7. Algorithm: RobLSSVM is a robust LS-SVM for regression (De Brabanter et al., 2009) that we use as part of LSSVMlab 1.8 (De Brabanter et al., 2010) due to its robustness to outliers. Robust LS-SVM was originally proposed in Suykens, De Brabanter, Lukas, and Vandewalle (2002) as an extension to LS-SVM (Suykens, Van Gestel, De Brabanter, De Moor, & Vandewalle, 2002). The optimization problem of this version of weighted LS-SVM can be written as:

$$\min_{w,b,e}\frac{1}{2}w^T w + \frac{1}{2}\gamma\sum_{j=1}^{n}v_j\hat{e}_j^2 \tag{11}$$

such that $Y = w^T\varphi(X) + b + \hat{e}$ where the weights v_j are

$$v_j = \begin{cases} 1, & |e_j/\widehat{s}| \leqslant c_2 \\ \frac{c_2 - |e_j/\widehat{s}|}{c_2 - c_1}, & c_1 \leqslant |e_j/\widehat{s}| \leqslant c_2 \\ 10^{-8}, & \text{otherwise} \end{cases}$$

where $c_1 = 2.5$, $c_2 = 3.0$ and $\widehat{s} = 1.483MAD(e_j)$ is in statistical terms robust estimate of the standard deviation. We use the Myriad reweighting scheme because it has been shown in De Brabanter et al. (2009) that it returns the best results between four candidate reweighting scheme approaches.

For the parameter optimization we use a state-of-the-art two stage Coupled Simulated Annealing-simplex method with five multiple starters. We use the RBF kernel.

2.3. Error calculation and ranking

The last module consists of error calculation, algorithm ranking, update ranking and results display. The error calculation module gives a possibility to choose from 10 measures used in time-series forecasting. The algorithms are ranked based on the chosen measure and the ranking is updated and displayed along with the forecasting results.

3. Experiment and results

The experiment consists of setting up the meta-learning system with the Tasks and then comparing the load forecasting error of the meta-learning system and other approaches used for load forecasting.

3.1. Load forecasting tasks creation

In order to build the meta-learning system, we create heterogeneous load forecasting tasks based on available data from ENTSO–E (2012), Weather Underground (2012) and European Commission (2012). We take 24 time-series of different hourly loads in Europe averaging between 1 and 10,000 MW. An example of one such load is given in Fig. 4. For all of these we estimate missing values, and for 21 we remove the outliers. We do not cut out any part of the time-series such as anomalous days, special events or data that might be flawed. We create time-series of exogenous data other than calendar information for the majority of these loads. For the days of daylight switch, where applicable, we transform the data to have all the days with the same hourly length by removing the 3rd hour or by adding the average of 2nd and 3rd hour.

While univariate time-series is a set of values over time of a single quantity, a multivariate time-series refers to changing values over time of several quantities. In the load forecasting context, the Task that consists of a load, calendar information and temperature is a multivariate Task. Temperature is used as exogenous feature for loads where it is available, due to its high correlation with the load and good empirical results in the industry and research. For one load we additionally use the following exogenous weather time-series: wind chill, dew point, humidity, pressure, visibility, wind direction, wind speed and the weather condition factor. We use past values of weather time-series for forecasting. Based on the availability of exogenous information for each Task, 24 load time-series have been separated in univariate and multivariate Tasks with hourly granularity. For the multivariate Tasks we create new Tasks by aggregating the time-series on a daily and monthly granularity.

The size of the learning set is: 1,000 data points for hourly granularity, 365 for daily and 24 for monthly. Test set size and forecasting horizon are equal for each granularity to: 36 for hourly, 32 for daily and 13 for monthly. Because forecasts are simulated in advance, the part of the load forecast is discarded. For hourly loads which are assumed to be used for creation of daily schedules, the

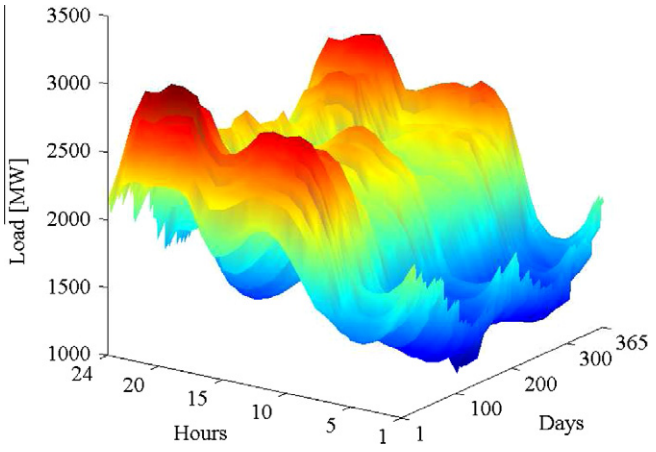


Fig. 4. Hourly load in duration of one year with a stable and frequent periodic and seasonal pattern often found in loads above 500 MW. Animation of load change during the years has been placed in the suplementary material.

forecasts are made at 12:00 of the load time-series and the first 12 hours are discarded. For data of daily and monthly granularity first point forecast is discarded as the load value belonging to the moment in which the forecast is made, is unknown. For data of hourly granularity, one iteration forecasts 36 values and discards first 12 leading to 24 hour forecasts. Similarly, for daily granularity 32 values are forecasted and the first one is discarded. For a monthly granularity, 13 are forecasted and the first value is discarded.

Based on the calendar information we create feature sets for all of the Tasks. Those feature sets consist of different dummy variables for each granularity and feature selection. For a daily granularity a total of 39 calendar features is encoded at the end, for daily granularity 13 and for monthly granularity 14 features. Features with holidays are coded separately for each Task because holidays are different globally. Badly performing feature combinations such as four-seasonal year and working day holidays are not implemented in the meta-learning system. Up to 25 combinations of the lags of the load are added to the feature set to improve the load forecasting performance. To get to a default feature set combination based on calendar information and lags of the load time-series we have conducted extensive empirical testing. With this approach we create a total of 69 Tasks of which 14 are univariate and 55 are multivariate. The average length of load time-series is 10,478 data points and the feature set of Tasks is between 4 and 45 features. We use 65 Tasks to build the meta-learning system and we compare with the other approaches to the 4 that are left. We named those 4 Tasks A, B, C and D. Task C is LTLF and Tasks A, B and D are STLF. Fig. 5 shows non-metric multidimensional scaling in 2 dimensional space using Kruskal's normalized STRESS1 criterion of 13 metafeatures for the 65 Tasks used to build the meta-learning system. Some Tasks are outliers and majority is densely concentrated which characterizes the data points of real-world Tasks. The difference here is that Tasks which are outliers cannot be omitted like single data points, as best performance on each Task is the goal.

3.2. Experiment

The forecasting is conducted for simulation on the data in a previously explained iterative fashion following real-life load

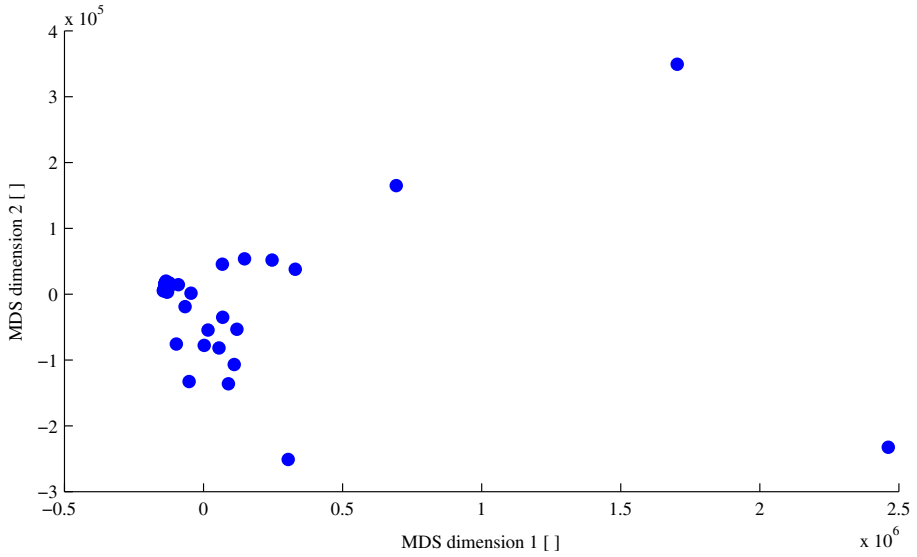


Fig. 5. Multidimensional scaling of Tasks used to build the meta-learning system and their metafeatures shows that some Tasks are outliers.

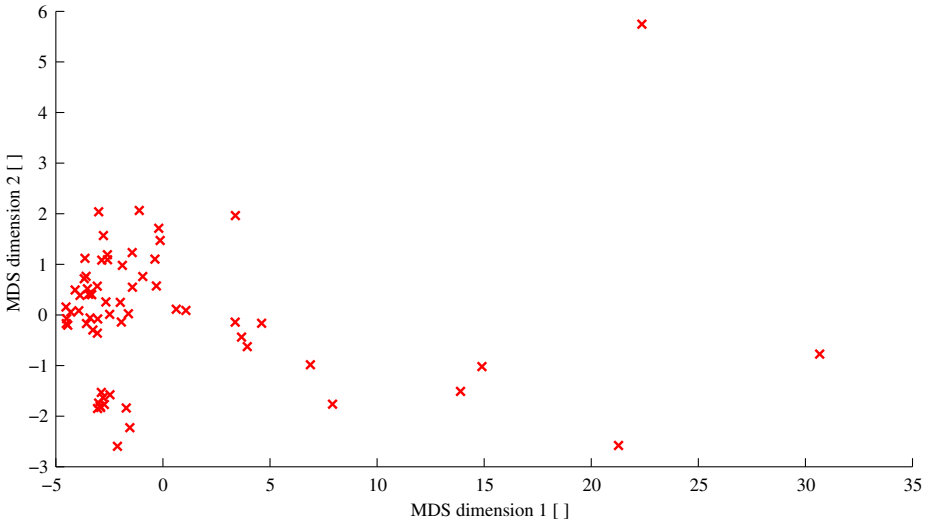


Fig. 6. Multidimensional scaling of Tasks with MASE of all algorithms to 2 dimensions.

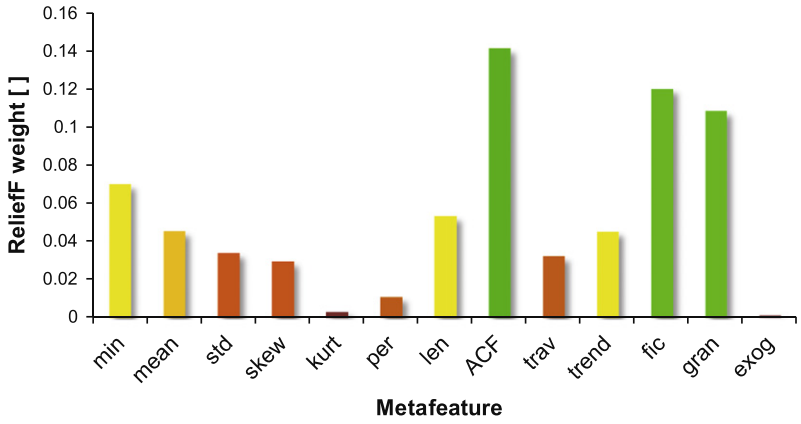


Fig. 7. ReliefF weights for chosen metafeatures show that *highest ACF*, *fickleness* and *granularity* are important.

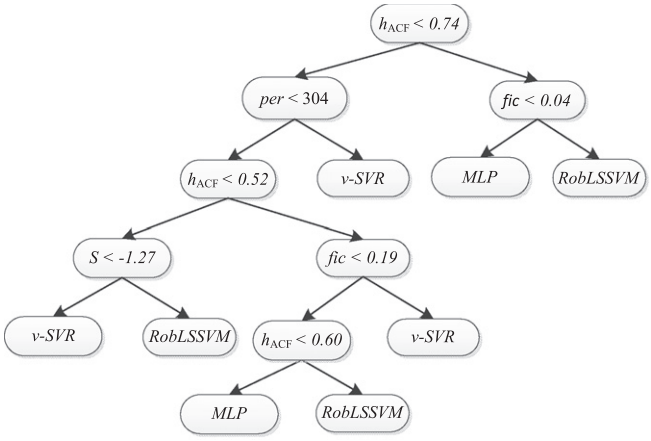


Fig. 8. The CART decision tree shows that the proposed metafeatures are used more than those frequently encountered which might indicate them as good candidates in application of meta-learning to load forecasting.

forecasting practice. For Tasks A and B a full year (365 cycles) is forecasted, for Task C, 1 year (1 cycle) is forecasted and for Task D which has forecasts of exogenous variables, 10 days (cycles) are forecasted ex-ante. We use the default feature selection.

Although RMSE and MAPE are most widely used performance measures in time-series forecasting (Sankar & Sapankevych, 2009) and in load forecasting (Hahn et al., 2009), for meta-learning the system and later for performance comparison we use MASE (Hyndman & Koehler, 2006) and NRMSE instead, due to problems with the same scale (RMSE) and division by zero (MAPE). Amongst different versions of NRMSE, we have selected RMSE over standard deviation as it best depicts different scales in one format. It is defined as the following:

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{j=1}^{n}(Y_j - \overline{Y}_j)^2}} \tag{12}$$

We use $\pi = \frac{2}{3}MASE + \frac{1}{3}NRMSE$ where both are previously normalized as the main performance measure.

We rank the forecasting results on 65 Tasks to find the best performing algorithm on each Task. Fig. 6 shows non-metric multidimensional scaling of MASE ranking amongst 65 Tasks to 2 dimensions where we can see that the ranking is similar in many Tasks. We use that ranking as the label for the CART decision tree in Fig. 8 on the metafeatures to create a decision tree for the meta-learning. Based on the ranking of the algorithms and metafeatures for 65 Tasks we apply the ReliefF for classification with 4 nearest neighbors to 14 candidate metafeatures (one additional being *Maximum*). We discard *Maximum* as it has negative weight. ReliefF weights are presented in Fig. 7. *Highest ACF*, *fickleness* and *granularity* are the most important metafeatures based on ReliefF. *Highest*

Table 1
Accuracy on Meta-Level.

Approach	ED	CART	LVQ	MLP	AutoMLP	ε-SVM	GP	Ensemble
Accuracy [%]	64.6	76.9	73.9	72.3	70.8	74.6	72.3	**80.0**

Table 2
Load forecasting error comparison.

Approach	Task A			Task B			Task C			Task D		
	MASE	NRMSE	MAPE	MASE	NRMSE	MAPE	MASE	NRMSE	MAPE	MASE	NRMSE	MAPE
RW	0.94	0.341	5.30	0.86	0.270	4.79	1.90	0.963	7.83	3.39	1.086	21.20
ARMA	0.82	0.291	4.83	1.07	0.315	6.14	1.72	1.113	7.72	2.05	0.669	12.43
SD	0.89	0.340	5.00	1.74	0.523	8.56	-	-	-	4.95	1.455	31.21
MLP	0.28	0.125	1.48	0.38	0.136	1.88	0.37	0.341	**0.57**	0.52	0.183	3.08
Elman NN	0.45	0.129	2.42	0.38	0.120	2.07	0.78	0.538	3.66	0.73	0.259	4.46
LRNN	0.47	0.222	2.59	0.33	0.106	1.81	1.01	0.711	4.45	0.76	0.279	4.79
ε-SVR	0.30	0.110	1.78	0.35	0.101	1.96	1.60	1.040	7.19	0.49	0.150	2.88
ν-SVR	0.24	0.096	1.41	0.27	0.086	1.54	1.60	1.039	7.19	0.45	0.139	2.61
LSSVM	0.16	0.072	0.98	0.20	0.065	**1.15**	0.43	0.311	2.08	0.43	0.143	2.49
RobLSSVM	0.15	0.065	**0.91**	0.20	0.065	**1.15**	0.44	0.340	2.11	0.40	0.139	**2.18**
Meta-Learning	0.15	0.065	**0.91**	0.20	0.065	**1.15**	0.37	0.341	**0.57**	0.40	0.139	**2.18**

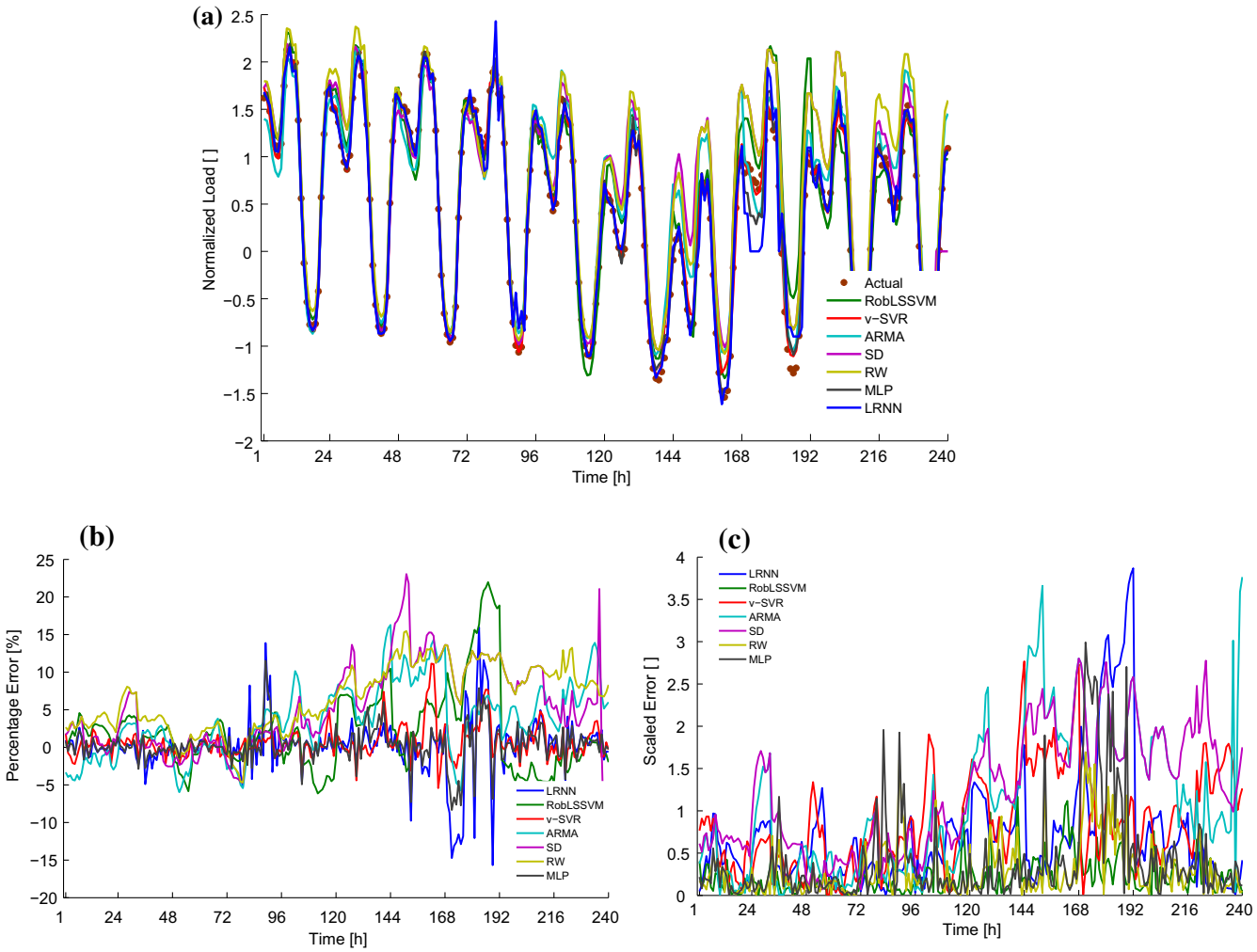


Fig. 9. An example of forecasting of the meta-learning system for 10 cycles: (a) Typical difference between actual and forecasted load using the 7 algorithms that build the meta-learning system. (b) The error comparison in terms of percentage error which is used for the calculation of MAPE. (c) The error comparison in terms of scaled error which is used for MASE calculation and is basis for the performance comparison. RW below 1 on average shows that our selection of RW is a better choice for load forecasting compared to the one typically used in time-series forecasting.

ACF is related to autocorrelation of the time-series and it is known that some algorithms are more suitable for auto-correlated data. Some algorithms work better with data that is more chaotic and reverts more around its mean value. In load forecasting practice it is established that granularity of the data affects model selection. We obtain the algorithm ranking learning on the metafeatures using the ensemble. We run the CART decision tree on metafeatures of 65 Tasks for the meta-learning system and present results in Fig. 8. Results suggest the importance of *highest ACF*, *periodicity* and *fickleness* at the meta-level. Before applying the ensemble to Tasks A to D, we test the performance on the meta-level alone using leave one out cross validation on the training data and compare the result of the ensemble against all candidates for it in Table 1. We used Pearsons $\chi^2 2 \times 2$ table test between the pairs of the approaches. The ensemble had the best accuracy and it is statistically significant in the boundary compared to the Euclidean distance and ε-SVM ($p = 0.05$). Between other pairs there is no significant statistical difference ($p > 0.05$). We use the ensemble to find the optimal forecasting algorithm for Tasks A to D.

3.3. Results

Finally, a comparison with 10 other algorithms has been conducted. Additionally to the algorithms used for the creation of the meta-learning system, simpler approaches like Elman Network, ε-SVR and LSSVM are used for the comparison. Optimizations that are used for those are same as for the algorithms in a meta-learning system related to them. The results of the comparison on the

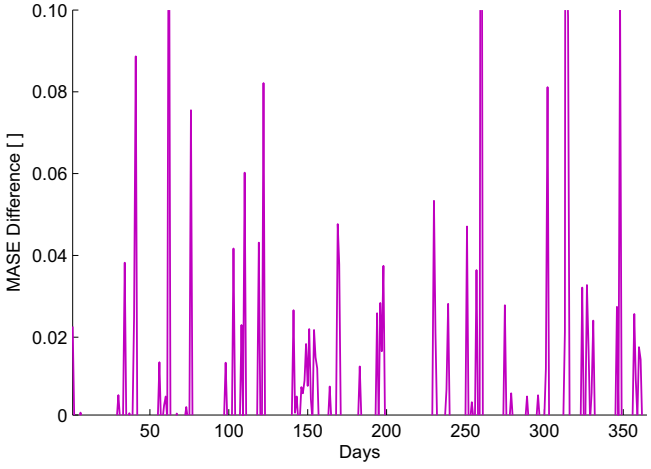


Fig. 10. MASE Difference for Task A between meta-learning system and best solution averaged per cycle.

Table 3
Error statistics of Task A.

Selection	Standard deviation		Skewness		Time [s]	
	MASE	MAPE	MASE	MAPE	per cycle	total
Meta-Learning	0.097	0.62	4.84	4.90	106	38,843
All combinations	0.082	0.52	3.69	3.92	370	135,004
$\frac{\text{Meta-Learning}}{\text{All combinations}}$ [%]	**118**	**119**	**131**	**125**	**29**	

Tasks A to D are present in Table 2 and the MAPE of the best results is bolded. The result of the meta-learning system is equal to one of the algorithms that build it because we used the same instance. For Tasks A to D, the proposed meta-learning system returns lower forecasting error than any single algorithm would do over all of those Tasks. Although RobLSSVM and LSSVM have comparable performance to the meta-learning on many Tasks, it pays off to use the meta-learning in the long run because of the performance differences which Task C demonstrates well. Example of the forecasting for few cycles is presented in Fig. 9. It shows a typical relation between actual and forecasted load at a high level. Periodicity of 24 shows that the Task is STLF. In terms of percentage error we can see that LRNN, SD and RobLSSVM deviate a lot for certain data points. For scaled error, a good reference is value of 1, below it performance is good and above it performance is bad. RobLSSVM and RW have better performance than SD and ARMA in both standard deviation and absolute value. The comparison of error of a meta-learning system and the best algorithm for each cycle for Task A is given in Fig. 10. The magenta lines indicate the difference between MASE of the meta-learning system (''Meta-Learning'') and the best MASE amongst all algorithms (''All combinations'') for each cycle. It can be seen that difference was above 0.10 in only 7 out of 365 cycles which may indicate that the meta-learning system made a good selection. We present a summary and error statistics for Task A in Table 3. Bolded ratios in the last row have a statistical significance of $p<0.05$. In 82% of cycles the meta-learning would have performed best for Task A which indicates a very good selection. The relative MASE is 1.059 and relative MAPE is 1.057 where they are defined as ratio of ''Meta-Learning'' and ''All combinations'' of MASE and MAPE, respectively. These relative indicators show how close are the performances of ''All combinations'' and ''Meta-Learning'' on Task A.

Our meta-learning system has been implemented in MATLAB R2011b. It has been parallelized for increased performance and scalability using MATLAB Toolbox Parallel Computing. The number of used CPU cores is selected in the first module. If it is set to 1, it does not work in parallel mode and uses only 1 core. Our meta-learning system has been tested on 32 and 64 bit versions of the following operating systems: Linux, Windows, Unix and Macintosh. It has been tested on different configurations running from 1 to 16 cores at the same time. The maximum memory usage was 2 GB RAM. We tuned the runtime of the system according to the industry needs. From Table 3 it can be seen that one usual cycle of forecasting (36 data points) takes on average 106 s and that the runtime of ''All combinations'' is over three times longer than that of the ''Meta-Learning''.

4. Conclusions

In this paper we proposed a meta-learning system for univariate and multivariate time-series forecasting as general framework for load forecasting. In a detailed comparison with other approaches to load forecasting it returns lower load forecasting error. We introduced classification ensemble in meta-learning for regression and applied Gaussian Processes and Robust LS-SVM in meta-learning. We designed this meta-learning system to be parallelized, modular, component-based and easily extendable. As a minor contribution of this paper we introduce four new metafeatures: *highest ACF*, *granularity*, *fickleness* and *traversity*.

Our empirical tests showed that those new metafeatures can indicate more challenging loads in terms of forecasting which we were looking for. Our decision trees and ReliefF test favor *highest ACF* and *fickleness* metafeatures. We parallelized our implementation to make it easily scalable.

The meta-learning approach is a promising venue of research in the areas of forecasting and expert systems. New kernel methods for load forecasting might be a good way towards further improving the performance of the meta-learning system. New optimization methods in feature selection and forecasting algorithms might lead to more efficiency. Forecasting approaches such as hybrids, ensembles and other combinations are a fertile area for further research.

Besides research opportunities, it can be used in industry for everyday operation to lower operating costs thus saving money to society. It can help those who need to forecast by selecting the most appropriate algorithm for their task. It can also be propagated to new end-user services based on large scale forecasting which would contribute to the development of smart grid and electricity market.

Although we developed this meta-learning system primarily from the perspective of load forecasting, it can be adapted to other areas involving heterogenous time-series regression such as finance, medicine, logistics and security systems.

Acknowledgements

This work was supported in part by the scholarship of the Flemish Government; Research Council KUL: GOA/11/05 Ambiorics, GOA/10/09 MaNet, CoE EF/05/006 Optimization in Engineering(OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; Flemish Government:FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G.0302.07 (SVM/Kernel), G.0588.09 (Brain-machine) research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC), G.0377.12 (Structured models), IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare; Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); IBBT; EU: ERNSI; ERC AdG A-DATADRIVE-B, FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940); Contract Research: AMINAL; Other:Helmholtz: viCERP, ACCM, Bauknecht, Hoerbiger. Johan Suykens is a professor at KU Leuven, Belgium.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.eswa.2013.01.047.

References

Alfares, H. K., & Nazeeruddin, M. (2002). Electric load forecasting: Literature survey and classification of methods. *International Journal of Systems Science, 33*(1), 23–34. http://dx.doi.org/10.1080/00207720110067421.

Brazdil, P., Giraud-Carrier, C., Soares, C., & Vilalta, R. (2009). Metalearning: Applications to Data Mining. In D. M. Gabbay, & J. Siekmann (Eds.). (1st ed.). Berlin: Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-73263-1.

Breuel, T. M., & Shafait, F. (2010). AutoMLP: Simple, effective, fully automated learning rate and size adjustment. *The learning workshop, Snowbird, USA*.

Broersen, P. M. T. (2006). *ARMASA toolbox with applications*. *Automatic autocorrelation and spectral analysis*. London: Springer-Verlag, pp. 223–250.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology, 2*(3). http://dx.doi.org/10.1145/1961189.1961199. 2:27:1–27:27.

Dannecker, L., Boehm, M., Fischer, U., Rosenthal, F., Hackenbroich, G., & Lehner, W. (2010). FP7 Project MIRABEL D 4.1: State-of-the-art report on forecasting. (p. 2). Dresden.

De Brabanter, K., Karsmakers, P., Ojeda, F., Alzate, C., De Brabanter, J., Pelckmans, K., De Moor, B., Vandewalle J., & Suykens J. A. K. (2010). LS-SVMlab Toolbox Users Guide version 1.8 (Leuven, Belgium) (pp. 1–115). http://www.esat.kuleuven.be/sista/lssvmlab/.

De Brabanter, K., Pelckmans, K., De Brabanter, J., Debruyne, M., Suykens, J. A. K., Hubert, M., et al. (2009). Robustness of kernel based regression: A comparison of iterative weighting schemes. In C. Alippi, M. Polycarpou, C. Panayiotou, & G. Ellinas (Eds.), *Lecture notes in computer science: Artificial neural networks–ICANN 2009* (pp. 100–110). Berlin: Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-04274-4 11.

ENTSO–E. (2012). http://www.entsoe.net/ Accessed 28.12.12.

Espinoza, M., Suykens, J. A. K., Belmans, R., & De Moor, B. (2007). Electric load forecasting using kernel-based modeling for nonlinear system identification. *IEEE Control Systems Magazine, 27*(5), 43–57. http://dx.doi.org/10.1039/c1em10127g.

European Commission. (2012). *Demography report 2010* (pp. 1–168).

Giraud-Carrier, C. (2008). *Metalearning-A Tutorial* (pp. 1–38).

Hahn, H., Meyer-Nieberg, S., & Pickl, S. (2009). Electric load forecasting methods: Tools for decision making. *European Journal of Operational Research, 199*(3), 902–907. http://dx.doi.org/10.1016/j.ejor.2009.01.062.

Hobbs, B. F., Jitprapaikulsarn, S., Konda, S., Chankong, V., Loparo, K. A., & Maratukulam, D. J. (1999). Analysis of the value for unit commitment of improved load forecasts. *IEEE Transactions on Power Systems, 14*(4), 1342–1348.

Hong, W.-C. (2011). Electric load forecasting by seasonal recurrent SVR (support vector regression) with chaotic artificial bee colony algorithm. *Energy, 36*(9), 5568–5578. http://dx.doi.org/10.1016/j.energy.2011.07.015.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting, 22*(4), 679–688. http://dx.doi.org/10.1016/j.ijforecast.2006.03.001.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. *Lecture Notes in Computer Science: Machine Learning–ECML, 784*, 171–182.

Lemke, C., & Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing, 73*(10–12), 2006–2016. http://dx.doi.org/10.1016/j.neucom.2009.09.020.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Cambridge, USA: MIT Press, pp. 1–219.

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers* (15), 65–118.

Sankar, R., & Sapankevych, N. I. (2009). Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine, 4*(2), 24–38.

Scholköpf, B., Smola, A. J., Williamson, R., & Bartlett, P. (2000). New support vector algorithms. *Neural Computation, 12*(5), 1207–1245http://www.ncbi.nlm.nih.gov/pubmed/10905814.

SCOPUS. (2012). http://www.scopus.com/ Accessed 28.12.12.

Smith-Miles, K. A. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys, 41*(1), 125. http://dx.doi.org/10.1145/1456650.1456656.

Suykens, J. A. K., De Brabanter, J., Lukas, L., & Vandewalle, J. (2002). Weighted least squares support vector machines: Robustness and sparse approximation. *Neurocomputing, 48*, 85–105.

Suykens, J. A. K., Van Gestel, T., De Brabanter, J., De Moor, B., & Vandewalle, J. (2002). *Least squares support vector machines* (1st ed.). Singapore: World Scientific.

Taylor, J. W. (2012). Short-term load forecasting with exponentially weighted methods. *IEEE Transactions on Power Systems, 27*(1), 458–464.

Tzafestas, S., & Tzafestas, E. (2001). Computational intelligence techniques for short-term electric load forecasting. *Journal of Intelligent and Robotic Systems, 31*(1–3), 7–68.

Vapnik, V. N. (1998). *Statistical learning theory* (1st ed.). Wiley, pp. 1–736.

Wang, X., Smith-Miles, K., & Hyndman, R. (2009). Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing, 72*(10–12), 2581–2594. http://dx.doi.org/10.1016/j.neucom.2008.10.017.

Wang, Y., Xia, Q., & Kang, C. (2011). Secondary forecasting based on deviation analysis for short-term load forecasting. *IEEE Transactions on Power Systems, 26*(2), 500–507.

Weather Underground. (2012). http://www.wunderground.com/ Accessed: 28.12.12.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation, 8*(7), 1341–1390 [MIT Press 238 Main St., Suite 500, Cambridge, MA 02142–1046 USA].