Semi-supervised monitoring of electric load time series for unusual patterns

Nikolaos Kourentzes and Sven F. Crone

***Abstract*— In this paper we propose a semi-supervised neural network algorithm to identify unusual load patterns in hourly electricity demand time series. In spite of several modeling and forecasting methodologies that have been proposed, there have been limited advancements in monitoring and automatically identifying outlying patterns in such series. This becomes more important considering the difficulty and the cost associated with manual exploration of such data, due to the vast number of observations. The proposed network learns from both labeled and unlabeled patterns, adapting automatically as more data become available. This drastically limits the cost and effort associated with exploring and labeling such data. We compare the proposed method with conventional supervised and unsupervised approaches, demonstrating higher accuracy, robustness and efficacy on empirical electricity load data.**

I. Introduction

ACCURATE predictions of electricity load data are required for a large variety of applications, such as trading electricity and scheduling production. In the forecasting literature such data are considered high frequency time series, where predictions are required at hourly or shorter intervals. Although a strict definition of what constitutes a high frequency time series does not exist, typically we refer to data that their sampling rate is daily or less and therefore result in vast amounts of data [1], introducing new issues in data handling, analysis and modeling. Use of conventional statistical modeling approaches, designed for low frequency time series becomes problematic in these cases [2]. In the field of electricity load forecasting several modeling methodologies for such time series have been proposed [3], [4], [5], [6]; however, there have been limited advancements in data monitoring and automatic outlier identification. Outliers in electricity load can be caused by several reasons, such as major events like natural disasters or bank holidays or more subtle like disruptions in industrial production, strikes, etc which are harder to be aware of and model their effect on electricity load, making outlier identification and important problem for this domain.

This omission is crucial for a number of reasons. Time series models often require data cleaning, which involves modifying or removing outliers and obvious errors in the database [7], therefore implicitly assuming that this information is available, which in fact requires costly manual collection. On the other hand, not cleaning the data can have substantial effects on model specification and parameters [8]. Furthermore, outliers can introduce problems in forecasting evaluation, where large errors are introduced due to atypical observations, often masking the real performance differences between the models considered [7], [9]. Taylor et al. acknowledges this issue and removes such days altogether, by replacing them, in order to simplify the application and evaluation of the examined forecasting methods [6]

It is important to distinguish between point outliers, where one or consecutive points in a time series, present abnormal behavior, depending if these are additive or innovative outliers [8], [10] and functional outliers [11]. In the latter case we are interested in analyzing data providing information about curves, surfaces, etc as a whole varying over time. In the context of electricity load these could be complete days. In this paper we focus on the problem of efficiently identifying outlying daily patterns, which can be considered functional outliers, though in this analysis we do not assume that this data are produced from smooth continuous functions, as is the norm in functional data analysis [11], [12]. Figure 1 provides an example of the outliers we are considering. In this example an hourly time series is split into daily profiles and we highlight outliers against normal days. We can observe that outlying days can differ both in the level, making them easy to spot like normal outliers, but also in the shape, which are harder to spot and limits the usefulness of functional boxplots that can be otherwise used to visualize temporal curves and outliers [13]. The difficulty of the later case is increased if the difference in shape does not span the whole day.

Related approaches have employed residual analysis [8], time series clustering and classifications methodologies [14]

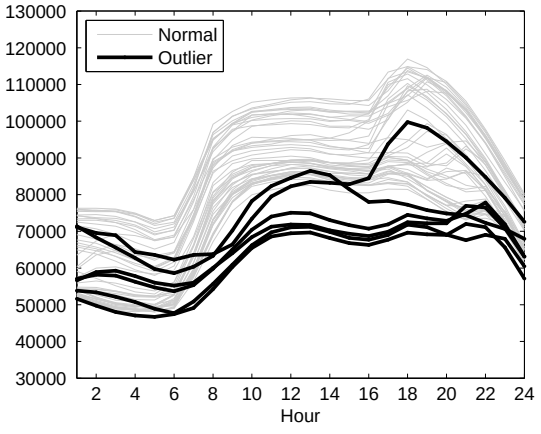

Fig. 1. An example of outlying daily profiles.

Nikolaos Kourentzes and Sven F. Crone are with the Department of Management Science at Lancaster University Management School, Lancaster, LA1 4YX, United Kingdom. (email: {n.kourentzes, s.crone}@lancaster.ac.uk).

belonging to a broader group of outlier detection research using both unsupervised and supervised learning algorithms, such as k-means, self-organizing maps, MLP and RBF networks, etc [15], [16]. The limitations of these approaches is that they either require extensive knowledge of the data, or they do not consider prior information that may be available. In the first case classification algorithms are employed to distinguish between normal and outlying days, which must be first trained on a large number of labeled cases in order to be used to classify future data. Their weakness is the cost and complexity associated with manually labeling high frequency data. On the other hand, unsupervised methods do not require prior knowledge; however, this can also limit their accuracy in identifying correctly the outlying seasonal profiles. In this paper we proposed a semi-supervised neural network method that balances the ability to infer knowledge from the data and make use of prior information, such as known labeled cases.

The contribution of this paper is to propose a novel semi-supervised automatic outlier identification method, based on self-training a neural network classifier, for detecting unusual load patterns in hourly electricity load time series. The innovation lies in the combination of the strengths of self-training and neural network models to robustly and accurately automatically identify outliers in electricity data. The key advantages of the proposed algorithm is that it does not require a large number of labeled samples that may be challenging and costly to collect, which is further reduced by the heuristic we propose to automatically generate an initial set of labeled data.

We demonstrate the performance of the proposed method using electricity load data from the UK. By providing a very small number of labeled outlying cases it correctly identifies other unlabeled outliers in-sample and in unseen out-of-sample data. We compare our approach with unsupervised k-means time series clustering and supervised multilayer perceptron (MLP) based classification, demonstrating the superiority of the proposed method.

The rest of this paper is organized as follows: Section II provides details of the proposed methodology. In section III we provide information regarding the data that we use to assess the performance of the algorithm and the experimental setup, while Section IV presents the empirical evaluation results and discusses its accuracy in contrast to unsupervised and supervised alternatives. We conclude in Section V.

II. Model Description

Semi-supervised learning is as an attractive modeling approach when there is an abundance of mostly unlabeled data [17]. In the context of electricity load forecasting, although it is easy to collect large amounts of data, these are unlabeled and manual exploration is required to identify unusual patterns. Let X be a set of observations, split into X_L with known labels C_L and the remaining unlabeled X_U. An unsupervised algorithm can use both X_L and X_U but makes no use of C_L; while, a supervised algorithm uses only X_L and C_L and is unable to use X_U. A semi-supervised algorithms make use of all [17]. In our implementation that we will discuss here, we follow the paradigm of self-training, where a classifier is first trained on X_L and C_L, with a typically small sample of labeled cases. Consequently the classifier is used to label X_U. The most confidently labeled observations from X_U are added to the training set and the classifier is re-trained. Eventually the classifier uses both the initially provided label cases and its own predictions to train. Self-training has been favorably compared to supervised learning on a variety of tasks [18], [19]. The requirement of a small number of of manually labeled outlying days drastically limits the cost and effort associated with manual exploration of high frequency data sets.

Suppose a time series $Y = (y_1, \ldots, y_n)$ with n observations. This can be split into S vectors $X = (x_1, \ldots, x_S)$, of length s so that $x_i = (y_{(i-1)s+1}, \ldots, y_{is})$, i.e. each x_i containing s observations from Y successively. For example if Y is sampled every hour and $s = 24$, each x_i would be a vector containing observations of a complete day of the time series. If the number of observations is not exactly divisible by s then x_S will have missing values. We define two classes; a seasonal vector x_i can be normal or can be an outlier. We assume that we are provided $S_L < S$ number of labels C_L corresponding to $X_L \subset X$. Only labels of outlying patterns are required for this implementation and typically S_L will be much smaller than S, since these will correspond to a small number of the observation belonging only to one of the two classes. Now we have two sets of seasonal vectors, the labeled X_L and the unlabeled $X_U = X \backslash X_L$ that contains the remaining elements of X.

Often high frequency time series have multiple seasonalities, if such exist, to aid the algorithm we remove the irrelevant seasonalities using a low-pass filter in the form of a moving average. Note that this will also remove any trend in the time series. These can be identified using autocorrelation analysis, periodograms or neural network filters as in [20] in order to avoid limitations of the former.

Before setting up the semi-supervised learner, we need to populate X_L with cases of normal seasonal vectors. We calculate the probability density function (PDF) of the observations in X_U for each y_j across x_i separately using Gaussian kernel density estimation. We then connect the modes of each PDF to form a profile K of normal days. An example can be seen in figures 2 and 3. In this example the kernel density for each hour of the day has been calculated and the modes have been identified. The corresponding values form a seasonal profile as indicated by the kernels. The result is a reasonable profile of the most common daily profile, as it can be seen in 3. We resort to this method as it allows us to be more robust to unlabeled outliers.

The next step is to record the similarity between all X_U and seasonal profile K. For this we use Pearson's correlation, as the objective is to identify seasonal vectors that behave as similar as possible with the estimated profile K. We rank members of X_U according to their similarity and select the ϕS_L top ones, where S_L is the number of items in X_L and $1 \leq \phi \leq \frac{S-S_L}{S_L}$. The selected $x_{u_i} \in X_U$ are

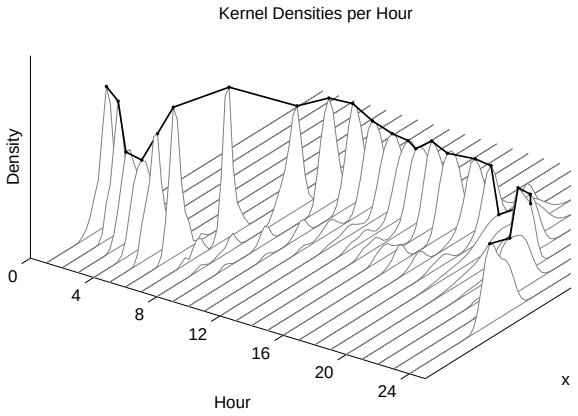

Fig. 2. An example of how the profile is calculated using Gaussian kernels.

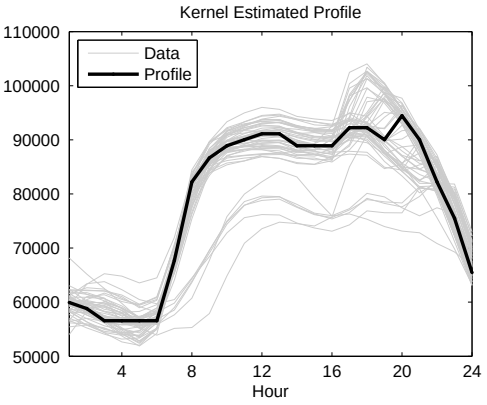

Fig. 3. An example of how the profile is calculated using Gaussian kernels.

labeled as normal cases and moved to X_L. We call this X'_L and $X'_U = X_U \setminus (X_U \cap X'_L)$ are the remaining unlabeled seasonal vectors. Note that X'_L can have unbalanced number of normal and outlying days, depending on the parameter ϕ that was chosen. For this reason and also to limit introducing mislabeled cases, we suggest ϕ to be small. We base this labeling strategy on the smoothness and the cluster assumptions underlying semi-supervised learning, which can be summarized in that points that are close will belong to the same class [17]. The labels $C'_L = (c'_{L_1}, \ldots, c'_{L_{S'_L}})$ corresponding to X'_L are coded as follows: outliers are $[1\ 0]$ and normal seasonal vectors are coded as $[0\ 1]$.

The next step is to develop an initial classifier that is trained on X'_L with targets C'_L. We use MLP for the classifier. MLPs are well researched and have been used successfully in numerous classification tasks; for a survey see [21]. We use a single hidden layer MLP with $h = (1, \ldots, H)$ hidden nodes and provide s inputs, one for each $y_{ij} \in x_i$ and $j = (1, \ldots, s)$. For instance if $s = 24$ we provide 24 inputs, one for each hour of the day. Given the coding of the targets two output nodes $k = (1, 2)$ are required, providing predicted labels $\hat{C}'_L = (\hat{c}'_{L_1}, \ldots, \hat{c}'_{L_{S'_L}})$.

$$\begin{aligned} \hat{c}'_{L_i} &= \left[\hat{c}'_{L_{i1}}\ \hat{c}'_{L_{i2}}\right], \\ \hat{c}'_{L_{ik}} &= \beta_{k0} + \sum_{h=1}^{H} \beta_{kh} g\left(\gamma_{h0} + \sum_{j=1}^{s} \gamma_{hj} y_{ij}\right), \end{aligned} \tag{1}$$

where $\mathbf{w} = (\beta, \gamma)$ are the network weights and $\beta = (\beta_{11}, \ldots, \beta_{2H})$, $\gamma = (\gamma_{11}, \ldots, \gamma_{Hs})$ are the weights for the output and the hidden layer respectively. The β_{k0} and γ_{h0} are the biases of each neuron. The hidden nodes use a nonlinear transfer function $g(\cdot)$, which is usually either the sigmoid logistic or the hyperbolic tangent function. MLPs offer extensive degrees of freedom in modeling for classification tasks. The modeler must choose the appropriate data preprocessing, the number of hidden nodes, the transfer function within nodes, the training algorithm and the cost function. We provide further details of our selections in section III, where we discuss the experimental setup we used to evaluate the proposed method. Figure 4 illustrates the setup of a network for $s = 24$ and 5 hidden nodes that use the hyperbolic tangent transfer function (TanH).

We train the initial classifier on X'_L with targets C'_L and use it to classify the complete sample X. As $X = X'_L \cup X'_U$ the network produces for X'_U labels $\hat{C}'_U = (\hat{c}'_{U_1}, \ldots, \hat{c}'_{U_{S'_U}})$. The next step is to identify the seasonal vectors that have been labeled with the highest confidence. We reverse it to minimizing lack of confidence; therefore we define diffidence $\Psi = (\psi_1, \ldots, \psi_{S'_U})$ as the minimum absolute distance between the predicted $\hat{c}'_{U_i}$ and either target $[1\ 0]$ or $[0\ 1]$, for the seasonal vector being outlying or normal respectively:

$$\psi_i = \min\left\{\left|\hat{c}'_{L_{i1}} - 1\right| + \left|\hat{c}'_{L_{i2}}\right|, \left|\hat{c}'_{L_{i1}}\right| + \left|\hat{c}'_{L_{i2}} - 1\right|\right\}. \tag{2}$$

If $\hat{c}'_{U_i}$ is equal to either $[1\ 0]$ or $[0\ 1]$ the diffidence ψ_i will be equal to zero that is the minimum possible value, or considering the reverse the classifier has maximum confidence on labeling this observation. If $\hat{c}'_{U_i}$ has any different value then

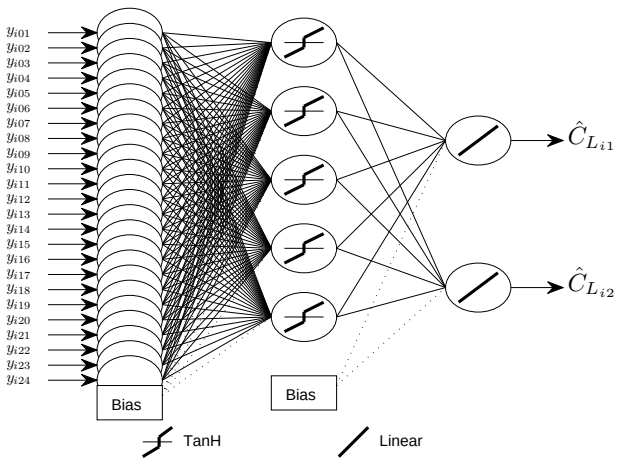

Fig. 4. MLP with inputs for $s = 24$ and 5 hidden nodes.

$\psi_i > 0$, which will be increasing in relation to the absolute distance from the defined coding of the classes. We prefer absolute to square metric in the definition of diffidence as the former is more robust to extreme values [9], [22]. Although alternative coding schemes for binary classification problems can be implemented, using the discussed two outputs coding is useful for defining the diffidence metric.

We rank the newly labeled observations according to their diffidence and calculate the P_Ψ percentile of the distribution of Ψ, which we use as a selection threshold. We define a new set of observations X''_L that includes all labeled seasonal vectors with $\psi_i \leq P_\Psi$. These are the seasonal vectors that are classified with the highest confidence. We replace X'_L by $X'_L \cup X''_L$ and self-training is achieved by inputting the new X'_L again to the classifier as training data, essentially merging the previously labeled training set with the newly labeled set of observations, thus expanding the training set to include previously unknown labels. By repeating this process the semi-supervised neural network is able to use both originally labeled and unlabeled seasonal vectors during training. This process can be repeated iteratively until no more observations can be added to the training set X'_L, where training finishes and the network is ready to be used to classify both observed and future unlabeled seasonal vectors.

Selecting the percentile P_Ψ allows to control the tolerance of the neural network to using labels with low confidence during training and also the speed that new observations are added to the training set. Lower percentiles will make each iteration of self-training to use only a few additional samples, while a high percentage will allow observations with low confidence. The final training set $X'_L \subseteq X$ as some observations may never be labeled with lower diffidence than P_Ψ and subsequently be included in X'_L.

Note that at each iteration when replacing X'_L by $X'_L \cup X''_L$ we implicitly assume that all labels in C'_L from the previous iteration, corresponding to the old X'_L, are correct. However, C'_L is the output of the semi-supervised neural network, which we cannot assess whether it is true or not and is associated with some confidence that can change at each iteration. Similarly, the ϕS_L normal seasonal vectors identified using the proposed heuristic during the initial setup of the classifier are always retained in X'_L even though their confidence may be low. These may cause the network to train on mislabeled observations, resulting in poor performance of the method. This issue can be resolved by replacing X'_L by $X_L \cup X''_L$ at each iteration. Recall that X_L are the initial manually labeled seasonal vectors and X''_L contains the seasonal vectors that have been labeled with the highest confidence only during the last iteration. This way all labels, except those assigned to X_L, are re-assessed for confidence repeatedly and observations that fail to be selected are removed from the training set. However, this approach risks using less sample for the final semi-supervised network, while requiring a high number of training iterations.

Figure 5 provides a summarized view of the described algorithm, illustrating the steps of data preparation, the semi-supervised neural network and the application of the final classifier. Note that the outlined method assumes that there is no trend in the time series; if such exists, it can be removed by an appropriate filter.

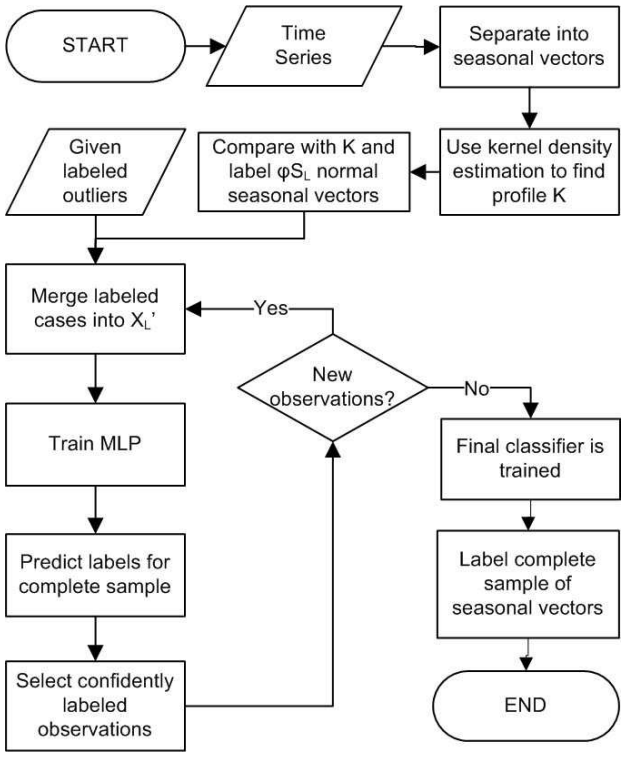

Fig. 5. Flowchart of proposed algorithm.

III. Experimental Design

A. Data

To assess the performance of the proposed semi-supervised neural network we use an electricity load time series from the UK, sampled at hourly intervals, from the 1^st of April 2001 01:00 until the 1^st of November 2008 01:00, amounting to 66505 hourly observations or 2771 days. Although, electricity load data have stable characteristics, exploring and modeling them is not trivial. The data exhibit triple seasonality, having an hour in the day cycle, day of the week and annual patterns. Two leap years are contained, 2004 and 2008, distorting the annual periodicity. The load profile of each day exhibits three distinct patterns for each day, associated to winter, summer and transitional consumption profiles. Figure 6 demonstrates the patterns for summer and winter for Thursday. In order avoid cluttering the figure we plotted transitional profiles together with summer days. Observe that they differ both in average level of consumption and but more importantly in shape, especially after 16:00. UK uses daylight saving, translated into one moving date day in either March or April having 23 hours and another moving date day in October having 25 hours every year. All these characteristics make detection of outliers challenging, as one has to consider changes in both the shape of the daily profiles and in the level of consumption that belong to normal days.

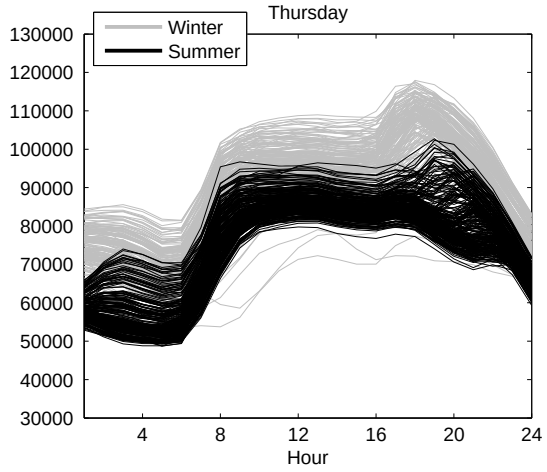

Fig. 6. Summer and winter consumption profiles for Thursday.

Through manual exploration we identified 63 unusual load profiles, which in many cases were associated with bank holidays, though not always. Figure 7 illustrates these unusual profiles. As we did not identify any outliers over weekend days we do not plot Saturdays and Sundays to keep the figure clear.

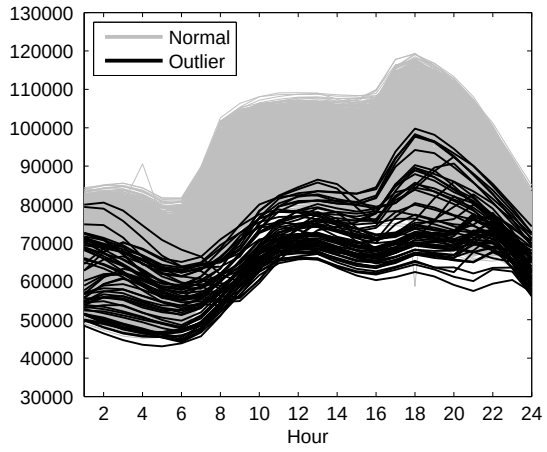

Fig. 7. Outlying daily profiles over complete sample.

For this empirical evaluation we will provide outlying days from 2001 as priorly labeled data. These are only 7 labels. Following the notation used in the previous section, X_L contains the 7 labeled outlying days from 2001, with C_L having the corresponding labels, and X_U containing the remaining 2764 unlabeled days. We identify normal days, by selecting the 50% percentile (median) as cut-off point for including new labels in the training set and $\phi = 3$, i.e. we automatically label 21 normal daily profiles before training the classifier, bringing the total number of training samples to 28. All remaining 2743 days are used to evaluate the performance of the method.

B. Methods

We use the semi-supervised network classifier we presented in section II. To train the network we randomly separate the labeled data to equal size training and validation subsets. The networks are trained using the Levenberg-Marquardt algorithm, which requires setting the μ_{LM} and its increase and decrease steps. Here $\mu_{LM} = 10^{-3}$, with an increase step of $\mu_{inc} = 10$ and a decrease step of $\mu_{dec} = 10^{-1}$. For a detailed description of the algorithm and the parameters see [23]. This training algorithm allows for fast training, essential for self-training. The maximum training epochs are set to 1000. Mean squared error is used as a training cost function and is recorded for both training and validation sets. The training can stop earlier if μ_{LM} becomes equal of greater than $\mu_{max} = 10^{10}$ or the validation error increases for more than 25 epochs. This is done to avoid overfitting. When the training is stopped the network weights that give the lowest validation error are used. The 1000 epochs training limit was never exceeded due to the early stopping criterion. Each MLP is initialized 10 times with randomized starting weights to accommodate the nonlinear optimization. The MLP initialization with the lowest error for each on the validation dataset is used. The inputs are linearly scaled between $[-1, 1]$. To assess the sensitivity on the size of the hidden layer we use from 1 up to 20 hidden nodes.

We compare the results of the proposed method with a conventional supervised MLP classifier. Its setup is identical to the semi-supervised MLP with the exception that no self-training takes place. Furthermore, we use an unsupervised k-means time series clustering approach, with two clusters, one for normal days and one for outlying ones. We experimented with different setups and found the best to be using Euclidean distance as a similarity measure, after the data have been preprocessed to remove annual seasonality. Correlation based similarity did not perform well. The semi-supervised network is the most complex and computationally expensive method, therefore it is useful only if it outperforms both supervised and unsupervised approaches.

C. Experimental Setup

We will employ all models outlined before to predict the labels $\hat{C}'_U$ in X_U. These will be compared with the actual $\hat{C}_U$ to assess the performance of the methods. For this we will employ two criteria. First, the Area Under the Curve (AUC) is used, which has been argued to be more desirable performance measure than accuracy [24], [25]. Because of the large number of normal days we also define Outlier Rate (OR) similarly to sensitivity to highlight the performance of the methods solely for outliers:

$$OR = \frac{TP}{TP + FP + FN}, \tag{3}$$

where TP are the true positives, i.e. correctly identified outliers, FP are the false positives used for undetected outliers and FN are false negatives, which are normal days misidentified as outliers. Therefore, OR measures labeled

TABLE I

AREA UNDER THE CURVE (AUC) RESULTS BY METHOD.

Hidden Nodes (H)	Supervised Learning	Semi-supervised Learning	Unsupervised Learning
1	0.999	0.999	0.818
2	0.999	0.999	0.818
3	0.967	0.967	0.818
4	0.968	0.976	0.818
5	0.975	0.967	0.818
6	0.973	0.976	0.818
7	0.991	0.976	0.818
8	0.997	0.982	0.818
9	0.990	0.991	0.818
10	0.965	0.983	0.818
11	0.968	0.968	0.818
12	0.996	0.998	0.818
13	0.967	0.984	0.818
14	0.996	0.968	0.818
15	0.965	0.996	0.818
16	0.929	0.960	0.818
17	0.905	0.960	0.818
18	0.920	0.968	0.818
19	0.960	0.992	0.818
20	0.994	0.968	0.818
Mean	0.971	**0.979**	0.818
Median	0.971	**0.976**	0.818
St Dev.	0.027	**0.013**	-

outliers as a ratio of all true and wrongly identified outliers. OR can be between zero and 100% and in the first case it is interpreted as no success in detecting outliers, whilst in the second it is interpreted as complete success in accurately identifying all outliers in the sample. The difference between this metric and sensitivity is that this one penalizes the result for false negatives, or in this context for incorrectly labeled normal seasonal vectors. True positives and negatives were identified by manual exploration of the original data.

Last but not least, for any automatic implementation it is important to assess the robustness of a method to different settings. We will focus on both aspects of accuracy (AUC and OR) and robustness in the presentation of the results.

IV. RESULTS

The results for AUC for all experiments are presented in table I. The first column provides the results for the supervised network, the second for the semi-supervised one and the third for the unsupervised k-means clustering. Each row shows the result for a different number of hidden nodes. Mean, median and standard deviation summary statistics are provided at the end of the table. The results for k-means are just replicated across the rows and therefore no standard deviation is provided, as it is not meaningful.

We can observe that the unsupervised results are inferior to either supervised or semi-supervised, on average by about 0.15. Comparing between the networks we can see that for small number of hidden nodes ($H \leq 3$) the semi-supervised approach does not offer any advantages, resulting in the same AUC. Thereafter, in 12 cases the semi-supervised classifier outperforms the supervised one, while on 5 cases, $H = \{5, 7, 9, 14, 20\}$, the supervised performs best. Both mean and median performance favors the semi-supervised approach, which also has substantially lower standard deviation across the AUC results for different number of hidden nodes, demonstrating a superior overall performance.

In this setting that the sample contains mostly unlabeled cases it is difficult to decide the correct number of hidden nodes a-priori or even after simulations. Assuming enough labeled cases, one could withhold some and test the performance of different sizes of hidden layer. However, this case, as well as other cases that the semi-supervised approach is preferable, there is not abundance of labeled instances to withhold. Ideally, the model should be robust to the number of hidden nodes and exhibit good and consistent performance across a wide range of values. In table I we can see that the proposed semi-supervised approach has low sensitivity to the number of hidden nodes and shows small AUC variability. Based on this result we can infer that the method is robust and fine-tuning the number of hidden nodes has small impact, which is also demonstrated in the lower AUC standard deviation. Note that the limited training sample cause problem for the performance of the supervised MLP for large hidden layers, which is overcome in the case of the semi-supervised approach. Robustness to settings is a key advantage for any automatic implementation, which the proposed semi-supervised method possess.

The results in table I consider all 2771 days in the sample, out of which only 2.3% are outliers. In order to assess in more detail the performance of the competing methods in detecting and labeling correcting outlying days we investigate the results in table II that contains the Outlier Rate percent. The OR intuitively is the number of classifiers over all existing outliers and normal days that are misclassified as outliers, or simply correctly detected outliers divided by the number of detected and undetected, correctly or not, outliers. The table has identical structure to table I that we discussed before.

The differences between the models are now more apparent. In agreement with the previous results, we can see that the unsupervised approach is inferior, achieving an OR of only 15.8%. It is useful to stress that given the small number of outliers we could not anticipate good performance of k-means, however this is not due to failings of the method, or its application, but rather due to the problem's characteristics. As expected in the light of the previous results, the first three rows for low number of hidden nodes are identical between the supervised and the semi-supervised models. However, considering OR we can observe that in all but one of the remaining cases the semi-supervised network labeled the outlying days more accurately. For $H = 5$ the supervised MLP is better by 1.5% OR. Consulting the summary statistics we can see that the semi-supervised method is better by about 8% OR overall and has significantly lower standard deviation. Again, we conclude that the semi-supervised neural network is insensitive to the number of hidden nodes, however we can see that the relative difference from the supervised MLP is higher for larger hidden layers. We highlight the results for $H = 15$ that the supervised MLP,

TABLE II

OUTLIER RATE (OR) % RESULTS BY METHOD.

Hidden Nodes (H)	Supervised Learning	Semi-supervised Learning	Unsupervised Learning
1	94.030	94.030	15.842
2	95.455	95.455	15.842
3	89.394	89.394	15.842
4	92.188	93.750	15.842
5	90.909	89.394	15.842
6	81.081	92.308	15.842
7	92.537	93.750	15.842
8	86.301	87.143	15.842
9	87.324	93.939	15.842
10	79.730	92.424	15.842
11	92.188	93.651	15.842
12	81.818	87.500	15.842
13	86.765	95.313	15.842
14	80.769	92.188	15.842
15	31.980	80.769	15.842
16	85.714	92.063	15.842
17	80.952	92.063	15.842
18	82.813	93.651	15.842
19	92.063	96.875	15.842
20	74.118	93.651	15.842
Mean	83.906	**91.966**	15.842
Median	86.533	**93.038**	15.842
St Dev.	13.502	**3.647**	-

which is also the initial classifier for the semi-supervised network, has an OR=32% that is improved through self-training by the semi-supervised model to 81%.

We found minimal differences in the results of the semi-supervised model if we allowed it to re-assess the labeling confidence of all seasonal vectors or not, as discussed in section II. Selecting different percentiles for P_Ψ had impact on the rate that observations entered the training set and final number of observations considered, but had small impact on the quality of the final classifier. Figure 8 provides examples for a network with 10 hidden nodes. The algorithm is run for different percentiles P_Ψ, given in brackets; for example $P_\Psi(20\%)$ is the 20% percentile. The number of training iterations are marked on the horizontal axes and the vertical axis the percent of the total available sample used for training at each iteration. The final training point for each P_Ψ is circled and its respective OR is provided. We can see that in most cases four to five training iterations were required and the final training samples between percentiles differ substantially. On the other hand, disregarding $P_\Psi(100\%)$, the maximum difference in OR is only 3%. The network that uses $P_\Psi(100\%)$ learns new labels too fast, which results in low performance. The relevant supervised MLP OR is 79.7% as we can see in table II. Figure 8 provides results for networks that do not re-assess the labeling confidence of observation that are already in the training sample. Figure 9 shows the same simulations for networks that re-assess the labeling confidence and can add as well as remove instances from the training sample. While the resulting OR for all cases is identical, the training sample profiles are substantially different, resulting overall in lower sample usage.

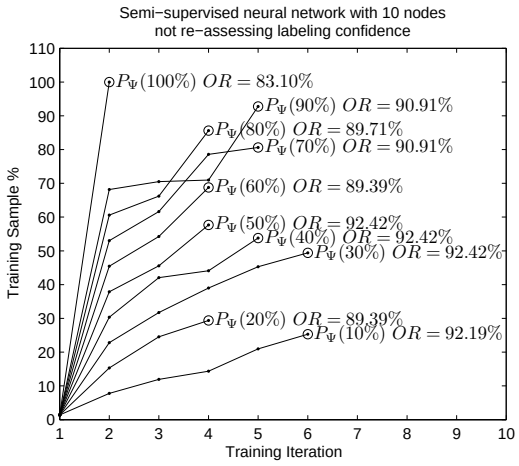

Fig. 8. Training behaviour of semi-supervised network with 10 hidden nodes for different P_Ψ. The network is not allowed to re-assess labeling confidence of instances in the training set.

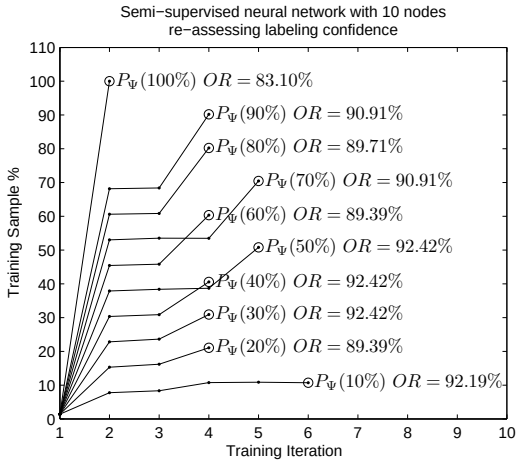

Fig. 9. Training behaviour of semi-supervised network with 10 hidden nodes for different P_Ψ. The network is allowed to re-assess labeling confidence of instances in the training set.

V. CONCLUSIONS

In this paper we presented a semi-supervised neural network algorithm to detect and identify unusual load profiles in hourly electricity load time series. The key advantage of the proposed method is that it does not need a large sample of labeled data that may be challenging and costly to collect due to the nature of the time series. We used the Area Under the Curve (AUC) to assess the overall accuracy and defined a metric targeted at measuring outlier detection, the Outlier Rate (OR) and showed empirically that the algorithm, in comparison to conventional supervised and unsupervised approaches, performs consistently more accurate and robust, for several design parameters, such as the number of hidden nodes H, the training sample selection threshold P_Ψ and the design decision of whether to reevaluate the output labels of the algorithm from the previous training iterations or not. We showed that robustness was achieved through the semi-supervised learning scheme, rather than as a property of

the MLPs. The robustness to design parameters is crucial for applications that have a few labeled cases and therefore semi-supervised methods are used, since in practice we don't have many labeled cases to evaluate their performance. Also, we proposed a heuristic to populate the initial sample of labels with additional cases of normal seasonal vectors, thus reducing further the required number of labels. We found that the heuristic was producing useful labels as they were retained in the final training set, even when the algorithm was allowed to remove them.

The objective of the proposed method is twofold. From one hand is to provide a dedicated monitoring tool for electricity load time series, which is commonly overlooked in research and practice alike. In several cases, tools from conventional low frequency time series exploration are applied to electricity load, although they are not designed to deal with high frequency time series and can provide erroneous results. Its second purpose is to aid in time series forecasting. Unusual load profiles are commonly excluded from modeling an electricity load time series and are either replaced by a normal profile day or removed altogether. Even if a forecasting method allows for special treatment of the outliers, one has to be able to detect them fast and reliably as new data become available. The advantage of the proposed algorithm is that it can both provide such information and also learn from useful new cases to improve its discriminatory power. Although in this study we used MLPs at the core of the algorithm due to their properties, other algorithms with similar learning capabilities, such as Support Vector Machines, can be explored.

The superior performance comes at the cost of additional complexity and computational cost. An important question that needs to be addressed is when the semi-supervised approach is required. The key decision variable is the availability and the cost of labeled cases to form the initial training set. Electricity load time series typically involve very large samples, which are difficult to manipulate and explore, making manual labeling time consuming and costly. Moreover, these time series exhibit particular problems, such as daylight saving, leap years, multiple overlaying seasonalities, etc. that increase the complexity and the difficulty of manual analysis further.

Next steps in this research are to explore in detail the relation of the parameter ϕ, used to initialize the labeled training sample, with the performance of the method, as well as explore the interactions and possible heuristics for choosing the remaining design parameters. The ultimate evaluation of the value of this algorithm is its impact on forecasting accuracy and decision making; therefore, we plan to integrate this into a forecasting model and measure directly its impact to forecasting accuracy.

REFERENCES

[1] R. F. Engle, "The econometrics of ultra-high-frequency data," *Econometrica*, vol. 68, no. 1, pp. 1–22, 2000.

[2] C. W. J. Granger, "Extracting information from mega-panels and high-frequency data," *Statistica Neerlandica*, vol. 52, no. 3, pp. 258–272, 1998.

[3] L. J. Soares and M. C. Medeiros, "Modeling and forecasting short-term electricity load: A comparison of methods with an application to brazilian data," *International Journal of Forecasting*, vol. 24, no. 4, pp. 630–644–, 2008.

[4] H. Hahn, S. Meyer-Nieberg, and S. Pickl, "Electric load forecasting methods: Tools for decision making," in *International Conference on Information Systems, Logistics and Supply Chain*, vol. 199, Lyon, France, 2009, pp. 902–907.

[5] J. R. Trapero and D. J. Pedregal, "Frequency domain methods applied to forecasting electricity markets," *Energy Economics*, vol. 31, no. 5, pp. 727–735, September 2009.

[6] J. W. Taylor, L. M. de Menezes, and P. E. McSharry, "A comparison of univariate methods for forecasting electricity demand up to a day ahead," *International Journal of Forecasting*, vol. 22, no. 1, pp. 1–16, 2006.

[7] C. Chatfield, *The Analysis of Time Series: An Introduction*, 6th ed. Chapman & Hall/CRC, 2004.

[8] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. New Jersey: Prentice Hall Inc., 1994, vol. 3rd.

[9] L. Tashman, "Out-of-sample tests of forecasting accuracy: An analysis and review," *International Journal of Forecasting*, vol. 16, pp. 437–450, 2000.

[10] F. P. H. and van Dijk D., *Non-linear time series models in empirical finance*. Cambridge University Press, 2000.

[11] J. O. Ramsay and B. W. Silverman, *Functional Data Analysis*. Springer Science+Business Media Inc., 2002.

[12] R. J. Hyndman and H. L. Shang, "Rainbow plots, bagplots and boxplots for functional data," Monash University, Department of Econometrics and Business Statistics, Monash Econometrics and Business Statistics Working Papers 9/08, Nov. 2008.

[13] Y. Sun and M. G. Genton, "Functional boxplots," *Journal of Computational and Graphical Statistics*, vol. to appear, 2011.

[14] P. K. Chan and M. V. Mahoney, "Modeling multiple time series for anomaly detection," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, ser. ICDM '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 90–97.

[15] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *SIGMOD Rec.*, vol. 29, pp. 427–438, May 2000.

[16] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artif. Intell. Rev.*, vol. 22, pp. 85–126, October 2004.

[17] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*. The MIT Press, 2006.

[18] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ser. ACL '95. Stroudsburg, PA, USA: Association for Computational Linguistics, 1995, pp. 189–196.

[19] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05) - Volume 1 - Volume 01*, ser. WACV-MOTION '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 29–36.

[20] S. F. Crone and N. Kourentzes, "Feature selection for time series prediction - a combined filter and wrapper approach for neural networks," *Neurocomput.*, vol. 73, no. 10-12, pp. 1923–1936, 2010.

[21] G. P. Zhang, "Neural networks for classification: a survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 30, no. 4, pp. 451–462, Nov 2000.

[22] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, pp. 679–688, 2006.

[23] M. Hagan, H. Demuth, and M. Beale, *Neural Network Design*. Boston: PWS Publishing, 1996.

[24] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine Learning*, vol. 45, pp. 171–186, 2001, 10.1023/A:1010920819831.

[25] C. L. Jin, C. X. Ling, J. Huang, and H. Zhang, "Auc: a statistically consistent and more discriminating measure than accuracy," in *In Proceedings of 18th International Conference on Artificial Intelligence (IJCAI-2003*, 2003, pp. 329–341.