Automated Load Curve Data Cleansing in Power Systems

Jiyi Chen, Wenyuan Li, *Fellow, IEEE*, Adriel Lau, Jiguo Cao, and Ke Wang, *Senior Member, IEEE*

***Abstract*—Load curve data refers to the electric energy consumption recorded by meters at certain time intervals at delivery points or end user points, and contains vital information for day-to-day operations, system analysis, system visualization, system reliability performance, energy saving and adequacy in system planning. Unfortunately, it is unavoidable that load curves contain corrupted data and missing data due to various random failure factors in meters and transfer processes. This paper presents the *B-Spline* smoothing and *Kernel* smoothing based techniques to automatically cleanse corrupted and missing data. In implementation, a man–machine dialogue procedure is proposed to enhance the performance. The experiment results on the real British Columbia Transmission Corporation (BCTC) load curve data demonstrated the effectiveness of the presented solution.**

***Index Terms*—Load management, load modeling, power systems, smoothing methods, power quality.**

I. Introduction

A. Background

LOAD CURVE data refers to the electric energy consumption recorded by meters at certain time intervals at delivery points or end user points. Load curve data is the "heartbeat" of electricity systems and is one of several most important data sets collected and retained in utilities. The analysis of load curve data would greatly improve day-to-day operations, system analysis, system visualization, system reliability performance, energy saving, and accuracy in system planning [1]. Two key features in the global vision of smart grid [2] are self-healing from power disturbance events and enabling active participation by consumers in demand response. The collection of valid load curve data is critical for supporting decision making in a smart grid system. For example, smart meters are an important initiative in smart grid and the quality of data is essential for the success of smart meters.

Collecting all load data accurately in fine granularity is a challenging and costly task. There is often missing and corrupted data in the process of information collection and transfer. This is caused by various reasons including meter problems, communication failures, equipment outages, lost data, and other factors. Possible other reasons include unexpected interruption or shutdown in electricity use due to strikes, unscheduled maintenance, and temporal close of production lines. Such events cause a significant deviation in load and do not repeat regularly, resulting in load data being unrepresentative of actual usage patterns. The term *corrupted data* refers to the data that significantly deviates from its regular patterns.

Poor quality load curve data can lead to misleading data analysis and incorrect decision making. It is important that corrupted data is identified and corrected. Currently, most utilities handle corrupted data manually in an *ad hoc* manner. This approach actually does not work, particularly after considerable smart meters come into place, as it is impossible to handle a huge data pool using a manual process.

In this paper, novel statistical approaches are presented to automatically detect corrupted data and replace it with the best estimated data. The accuracy of the detection is also quantified using a confidence interval. Missing data is treated as a special case of corrupted data. This problem is referred to as the *load cleansing problem*. There are two challenges in resolving this problem. First, if a relatively large portion of data is corrupted or missing, most standard statistical methods cannot be applied. Second, due to the uncertainty in electricity consumption, randomness of outage events, and dynamism of customers, it is difficult to judge whether a relatively large deviation represents corrupted data or an underlying change in data patterns.

The contributions of this work are as follows. First, two types of data cleansing problems for load curve data are formalized in Section II, which are called *locally corrupted data detection* and *globally corrupted data detection*, respectively. Though related, these problems are different from the traditional outlier detection and load forecasting problems. Second, a solution is presented by modeling the underlying structure of load curve data and using specific *nonparametric regression* techniques in Section III. This solution provides a common basis for detecting corrupted data, estimating replacing data, and deriving the confidence level for detection. The solution deals with both locally and globally corrupted data in a uniform way and is robust to a relatively large portion of missing data. Third, the implementation techniques are proposed in Section IV, including an incremental training algorithm to incorporate the user feedback at an early stage with minimum user effort. Finally, the proposed solution is tested using real British Columbia Transmission Corporation (BCTC) load curve data in Section V. The results demonstrated the effectiveness and high performance of the proposed solution.

B. Related Work

A closely related area to the load cleansing problem is *outlier detection*, which has been extensively studied in data mining and statistics research.

Manuscript received January 27, 2010; revised May 11, 2010; accepted June 07, 2010. Date of publication July 26, 2010; date of current version August 20, 2010. This work is supported in part by the Collaborative Research and Development Grants, NSERC, 2009. Paper no. TSG-00016-2010.

J. Chen, J. Cao and K. Wang are with Simon Fraser University, Vancouver, BC V5A 1S6, Canada. (e-mail: jiyi_chen@cs.sfu.ca, jiguo_cao@sfu.ca, and wangk@cs.sfu.ca).

W. Li and A. Lau are with British Columbia Transmission Corporation, Vancouver, BC V7X 1V5, Canada. (e-mail: wen.yuan.li@bctc.com and adriel.lau@bctc.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSG.2010.2053052

In the domain of data mining, a broad spectrum of techniques have been developed to detect outliers, among which, the proximity-based techniques such as k-nearest neighbor classification [3], k-means clustering [4], and the neural network methods such as RNNs [5] are frequently used [6]. However, most of these techniques are designed for structured relational data instead of time series. When dealing with outliers in time series, data mining researchers are more interested in determining whether a whole time series is abnormal with respect to a reference of "normal" time series (multiple time series). For example, the techniques in [7]–[11] compute an anomaly score of a coming time series based on its distance to the referenced normal time series. These elegant methods are not applicable to our problem because there is no "normal" time series for reference. Moreover, our goal is to identify corrupted data within a single long time series.

Keogh *et al.* have developed a suite of techniques to detect "discords" in a time series [12]–[15], that is to identify the time series subsequence that is maximally different from all the rest of the time series subsequences. These techniques are not suitable for solving our problem. First of all, they require the prior knowledge on the length of the sliding window whereas the lengths of corrupted data in load curve are uncertain and varied. Secondly, the discords are not necessarily corrupted data or outliers for load curve data and *vice versa*. Consider a load curve with an obvious increasing trend. Subsequences with larger values tend to have larger distances to the rest subsequences, therefore, are more likely to be detected as discords. But such subsequences can be normal because a load curve typically has an increasing trend over time.

Outlier detection in time series has also been studied in the field of statistics [23]. Well known statistical tests for outliers include Z-value, box plot, Rosner test, Dixon test, and Grubbs test. A common assumption for these test methods is the normal distribution of data [16], which is invalid for load curve data. Many other methods [17], [18], [24]–[26] are based on the ARMA model, which impractically assumes that the time series is stationary. Particularly, the traditional time series methods including the ARMA model cannot handle a relatively large portion of missing data.

The time series techniques such as ARMA are often used for *time series forecasting* [27] or *load forecasting* [29]. These issues are different from the load cleansing problem considered here. In load forecasting, historical loads are used to forecast the load at a future time point. In load cleansing, historical loads are used to detect corrupted data at a particular historical time point and decide an appropriate replacement of the corrupted data. In other words, load forecasting trusts all historical data whereas load cleansing deals with historical corrupted data and their replacing data. In addition, the cleansed data will greatly improve the quality of load forecast.

II. Problem Definition

A load curve is a time series where load values are collected at a certain time frequency such as every 5 min or hourly. Typically, load curve data follows certain patterns and behaves a daily, weekly, and seasonal periodicity with an increasing tendency over years. It is also influenced by random factors such as outages, meter failures, communication interruptions or errors, and dynamism of customers. As a result, a load curve consists of not only white noises but also some corrupted data.

Given a time series representing a load curve, the data cleansing includes two tasks: detecting corrupted data and replacing it with representative estimated data. In the data cleansing, the regular patterns of load curve, for example, periodicity, trends, and autocorrelations, must be kept while assuring the quality of replacing data. Missing data can be considered as a special case of corrupted data where a load value of zero is collected. The following two types of corrupted data, namely, *locally corrupted data* and *globally corrupted data*, are considered.

Definition 1: Consider a time series $\{(t_i, y_i)\}_{i=1}^n$ where y_i is the data (observation) at the time t_i. A data point is *locally corrupted* if it deviates markedly from the local patterns of the time series. ■

Definition 2: Consider a time series $\{(t_i, y_i)\}_{i=1}^n$, where y_i is the data (observation) at the time t_i. A data point is *globally corrupted* if it deviates markedly from the global patterns of the time series. ■

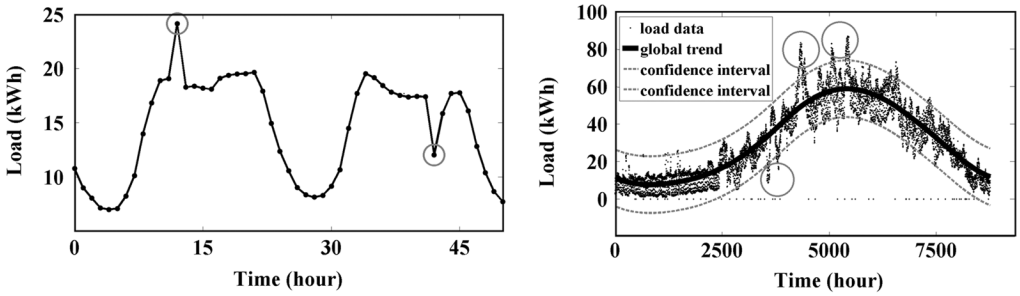
Observations are not necessarily equally spaced in time dimension because some of them may be missing. Also, the detection should focus on a region which contains multiple nearby points that markedly deviate from the local or global patterns, but not on each individual corrupted data point. Therefore, the terms *locally corrupted region* and *globally corrupted region* are used in the paper. A key to the detection of corrupted data is the notion of "deviating markedly" from the patterns. The presented solution is to establish the expected value range of normal data at each time point, which is called the confidence interval at a given confidence level. With the confidence interval, nearby corrupted data points, which locate outside the confidence interval (deviating markedly from the patterns), can be grouped into regions.

Fig. 1(a) shows two locally corrupted regions and Fig. 1(b) shows three globally corrupted regions indicated by circles. The y-axis represents the load data by hourly energy consumption (kWh). The x-axis represents the time dimension in the unit of hours. In Fig. 1(b), the solid line represents global trends of the load data and the two dashed lines represent the confidence interval for normal load data. Based on the confidence interval, three groups of points circled are identified and represent three globally corrupted regions since most points in these groups are outside the interval.

III. Proposed Smoothing Methods

The essence of the proposed solution to the detection of corrupted data is to model the intrinsic patterns or structure of load data. The model found can be used to judge the presence of abnormal deviations from the patterns, and thus to identify corruption of data. Assume that n data points $\{(t_i, y_i)\}_{i=1}^n$ of a load curve have been collected. The underlying data generation process is modeled as a continuous function [19]

$$y_i = m(t_i) + \varepsilon_i \tag{1}$$



(a) (b)

Fig. 1. Examples of locally and globally corrupted data (regions). (a) Locally corrupted data. (b) Globally corrupted data.

where y_i is the data value at time t_i, $m(t)$ is the underlying function and ε_i is the error term. It is assumed that the error term ε_i is normally and independently distributed with the mean of zero and constant variance σ^2. Our main task is to find an appropriate estimate of the function $m(t)$, namely $\hat{m}(t)$, using the collected load data as a sample of observations. Then the point-wise confidence interval can be built based on the estimate of $m(t)$. A data point is judged as corrupted if it locates outside the confidence interval. Corrupted data at time t_i will be replaced by the estimated value $\hat{m}(t_i)$.

A *smoothing parameter* is used to control the curve smoothness. A smoother curve $m(t)$ tends to model global patterns since it is less sensitive to local deviations, whereas a rougher curve $m(t)$ is more capable of modeling local patterns. Different settings of the smoothing parameter can be chosen to model global or local patterns.

A. Nonparametric Regression

The aim of a regression analysis is to estimate the unknown response function (or curve) $m(t)$ from the observed data $\{(t_i, y_i)\}_{i=1}^n$. This approximation procedure is called *smoothing*. The smoothing task can be done essentially in two ways: *parametric regression* and *nonparametric regression*. The former assumes that the curve $m(t)$ has some prespecified functional form, whereas the latter does not. The parametric model is sufficient if the structure of the curve is obvious, such as a straight line. However, a preselected parametric model is too restricted for unknown or more complex relationships. In this work, the nonparametric regression is used because load curve data does not follow a simple and known relationship.

The basic idea of nonparametric smoothing is the *local averaging procedure*. Specifically, the curve can be estimated by

$$\hat{m}(t) = \frac{1}{n}\sum_{i=1}^{n} W_i(t) y_i \tag{2}$$

where $\{W_i(t)\}_{i=1}^n$ denotes a sequence of weights which depend on the whole vector $\{t_i\}_{i=1}^n$. Among most well accepted nonparametric smoothing techniques are *Spline* smoothing and *Kernel smoothing*. In this paper, the *B-Spline* smoothing [20], which is commonly used in the Spline smoothing family, and the popular *Nadaraya–Watson estimator* in the Kernel smoothing family [19] are used.

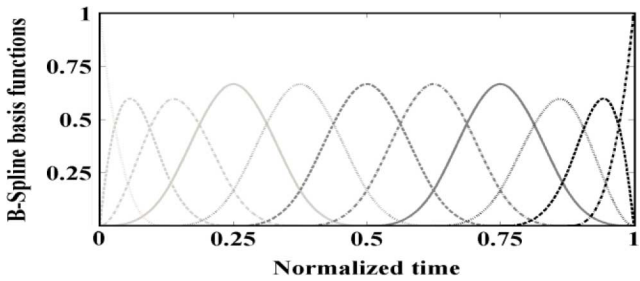


Fig. 2. B-Spline basis functions.

B. B-Spline Smoothing

1) Basis Function System: To estimate the function $m(t)$, the B-Spline smoothing makes use of a *basis function system* consisting of a set of known basis functions $\{\phi_k(t)\}_{k=1}^K$ that are mathematically independent of each other. The idea is to approximate the function $m(t)$ by taking a weighted sum or linear combination of a sufficiently large number K of basis functions $\phi_{\mathrm{k}}(t)$

$$m(t) = \sum_{k=1}^{K} c_k \phi_k(t) \tag{3}$$

or in the form of vectors

$$m(t) = \vec{c}^T \vec{\phi}(t) \tag{4}$$

where $\vec{c} = (c_1, \ldots, c_K)$ is the coefficient vector and $\vec{\phi}(t) = (\phi_1(t), \ldots, \phi_K(t))$ is the vector of basis functions. There are different basis function systems. The B-Spline basis system developed by de Boor [21] is adopted. Fig. 2 shows how a B-Spline basis system looks like. Each function $\phi_k(t)$ in a B-Spline basis system is positive over a short interval of time and is zero in the rest. This property, called *compact support property*, guarantees that mainly local information is considered when estimating the coefficients $\vec{c}$.

2) Estimating Coefficients: To estimate the coefficients $\vec{c}$ from the observations $\{(t_i, y_i)\}_{i=1}^n$, we define an n by K matrix

$$\Phi = \begin{bmatrix} \phi_1(t_1) & \phi_2(t_1) & \cdots & \phi_K(t_1) \\ \phi_1(t_2) & \phi_2(t_2) & \cdots & \phi_K(t_2) \\ \vdots & \vdots & & \vdots \\ \phi_1(t_n) & \phi_2(t_n) & \cdots & \phi_K(t_n) \end{bmatrix} \tag{5}$$

where $\Phi[i,j] = \phi_j(t_i)$ represents the value of the jth basis function at time t_i. By treating all vectors as column vectors, the function values $m(t)$ at time $(t_1,\ldots,t_n)$ are given by $\vec{m} = \Phi\vec{c}$. A simple smoother could be obtained if the coefficients $\vec{c}$ are determined by minimizing the *sum of squared error* (SSE) as

$$\mathrm{SSE} = \sum_{j=1}^{n}\left[y_j - \sum_{k=1}^{K} c_k\phi_k(t_j)\right]^2 \tag{6}$$

or in the form of vectors

$$\mathrm{SSE} = (\vec{y} - \Phi\vec{c})^T(\vec{y} - \Phi\vec{c}) \tag{7}$$

where $\vec{y}$ is the vector form of $\{(t_i, y_i)\}_{i=1}^n$. The SSE can always be decreased by using enough number of basis functions, to make the fitted curve go through all data points. However, a larger K can lead to more risk of overfitting, i.e., the noise that we wish to remove has more chance to be fitted.

To solve the overfitting problem, the coefficients can be estimated by minimizing the *penalized sum of squared errors* (PENSSE) as follows:

$$\mathrm{PENSSE}_\lambda = \mathrm{SSE} + \lambda \times \mathrm{PEN}_2(t) \tag{8}$$

where λ is the smoothing parameter and $\mathrm{PEN}_2(x)$ is the *roughness measure* which is defined as the integral of square of second derivative or *curvature* of the curve $m(t)$

$$\begin{aligned}\mathrm{PEN}_2(t) &= \int [D^2 m(t)]^2 dt \\ &= \int [D^2\vec{c}^T\vec{\phi}(t)]^2 dt \\ &= \vec{c}^T R\vec{c}\end{aligned} \tag{9}$$

where

$$R = \int D^2\vec{\phi}(t) D^2\vec{\phi}(t)^T dt. \tag{10}$$

Generally, the rougher the curve is, the larger curvature it tends to have. The smoothing parameter λ controls the scale of roughness penalty that will be put on and thus controls the smoothness (or roughness) of the curve. By combining (7), (8), and (9), the PENSSE can be expressed in the form of vectors as

$$\mathrm{PENSSE}_\lambda = (\vec{y} - \Phi\vec{c})^T(\vec{y} - \Phi\vec{c}) + \lambda\vec{c}^T R\vec{c}. \tag{11}$$

The estimate of the coefficients $\vec{c}$ can be obtained by setting the derivative of PENSSE with respect to $\vec{c}$ to be zero

$$\hat{\vec{c}} = (\Phi^T\Phi + \lambda R)^{-1}\Phi^T\vec{y}. \tag{12}$$

From (4) and (12), the fitted value vector $\hat{\vec{y}}$ (i.e., estimated values for observations $\vec{y} = (y_1,\ldots,y_n)$) is computed by

$$\hat{\vec{y}} = \Phi\hat{\vec{c}} = \Phi(\Phi^T\Phi + \lambda R)^{-1}\Phi^T\vec{y}, \tag{13}$$

or

$$\hat{\vec{y}} = S\vec{y} \tag{14}$$

where

$$S = \Phi(\Phi^T\Phi + \lambda R)^{-1}\Phi^T. \tag{15}$$

S is referred to as a *hat matrix*, as it converts the dependent variable vector $\vec{y}$ into its fitted value $\hat{\vec{y}}$. The hat matrix S in (14) plays the role of the weight functions $\{W_i(t)\}_{i=1}^n$ in (2). The number of degrees of freedom of the fit is the trace of the hat matrix

$$df = \mathrm{trace}(S). \tag{16}$$

In linear algebra, the *trace* of an n-by-n square matrix A is the sum of the diagonal elements of A. The degrees of freedom will be used in calculating the confidence interval in Section IV-A.

C. Kernel Smoothing

In Kernel smoothing, the shape of the weight function $W_i(t)$ in (2) is described by a density function (*Kern*) called a *kernel* (e.g., a normal distribution density function) [19]. The weight sequence $\{W_i(t)\}_{i=1}^n$ is defined by

$$W_i(t) = \frac{\mathrm{Kern}_h(t - t_i)}{n^{-1}\sum_{i=1}^n \mathrm{Kern}_h(t - t_i)} \tag{17}$$

where

$$\mathrm{Kern}_h(t) = \frac{1}{h}\mathrm{Kern}\left(\frac{t}{h}\right) \tag{18}$$

is the kernel with the scale factor h. A well-accepted *kernel estimator* is the *Nadaraya–Wastson estimator*

$$\hat{m}_h(t) = \frac{n^{-1}\sum_{i=1}^n \mathrm{Kern}_h(t - t_i)y_i}{n^{-1}\sum_{i=1}^n \mathrm{Kern}_h(t - t_i)}. \tag{19}$$

The *shape* of the weights is determined by *Kern* and the *size* of the weights is parameterized by h, which is called the *bandwidth*. In our case, *Kern* is chosen to be the probability density function of the *standard normal distribution*

$$\mathrm{Kern}(t) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}t^2} \tag{20}$$

such that Kern_h is the density function of a normal distribution with a mean of 0 and a variance of h^2. According to (2) and (17), the fitted value vector $\hat{\vec{y}}$ can be rewritten in the form of (14), i.e., $\hat{\vec{y}} = S\vec{y}$, where the hat matrix S is defined as

$$S = \begin{bmatrix} n^{-1}W_1(t_1) & n^{-1}W_2(t_1) & \cdots & n^{-1}W_n(t_1) \\ n^{-1}W_1(t_2) & n^{-1}W_2(t_2) & \cdots & n^{-1}W_n(t_2) \\ \vdots & \vdots & & \vdots \\ n^{-1}W_1(t_n) & n^{-1}W_2(t_n) & \cdots & n^{-1}W_n(t_n) \end{bmatrix} \tag{21}$$

where $W_i(t_j)$ is the weight of observation at time t_i for estimating the data value at time t_j. Again, the number of degrees of freedom of the fit is computed by the trace of the hat matrix, i.e., (16).

The bandwidth h controls the roughness of the fitted curve. This plays the same role as the smoothing parameter λ does in

the B-Spline smoothing. The larger the h is, the more neighboring information is taken into account and the smoother the fitted curve would be. For simplicity, both h and λ will be referred to as the smoothing parameter.

IV. Implementation

The modeling methods for the underlying structure of a load curve were presented in previous section. Two remaining issues are how to detect corrupted data and how to set the parameters in the model.

A. Confidence Interval

To detect corrupted data, the idea of the point-wise confidence interval is utilized. An observation within the confidence interval is considered normal and an observation outside the confidence interval is considered corrupted. As mentioned earlier, the error term ε_i in (1) is assumed to be normally and independently distributed with the mean of zero and constant variance σ^2. Under this assumption, the predicted confidence interval of a data point can be computed by its estimated value plus or minus a multiple of predicted errors. The estimated predicted errors is given by [28]

$$s_i^2\{\text{pred}\} = \text{MSE} + s_i^2(\hat{y}_i) \tag{22}$$

where the MSE is the *mean square error*

$$\text{MSE} = \frac{1}{n-df}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{23}$$

and $s_i^2(\hat{y}_i)$ is the sampling variance to the fit at time t_i and is given by the entry (i,i) of the matrix $\text{Var}[\hat{\vec{y}}]$

$$\text{Var}[\hat{\vec{y}}] = S * S^T * \text{MSE} \tag{24}$$

where S is the hat matrix introduced in Section III [20]. For the given α significance level, the $100*(1-\alpha)\%$ confidence interval at time t_i is estimated by

$$[\hat{y}_i - Z_{1-\alpha/2} * s_i\{\text{pred}\}, \hat{y}_i + Z_{1-\alpha/2} * s_i\{\text{pred}\}] \tag{25}$$

where $Z_{1-\alpha/2}$ is the $100*(1-\alpha/2)$ percentile of the standard normal distribution, which can be obtained by looking up from a standard normal table. In the following experiment, the 0.05 significance level $(\alpha = 0.05)$ is chosen and the corresponding $Z_{1-\alpha/2}$ is 1.96. This means that a normal observation would fall in the confidence interval with a probability of 95%, or equivalently, the probability that a data point locates outside the interval is corrupted is 95%.

B. K—The Number of Basis Functions

For the B-Spline smoothing introduced in Subsection III.B, the number of basis functions, i.e., K, needs to be determined. As we mentioned in the previous section, a larger K can make the curve fit the actual data better, and also result in a higher risk of overfitting and more computations. There is no gold criterion for the selection of K [20]. One strategy is to only let the smoothing parameter λ control the smoothness of the curve. The following rule is used in our study, which leads to good results: in modeling local patterns where only local data (e.g., data of one week) is used, K is selected as the number of observations, whereas in modeling global trends where more data (e.g., data of a whole year) is considered, K is selected as large as the computation allows.

TABLE I
Incremental Training Algorithm

```
Algorithm Incremental_Training (T, k)
1.   Partition T into T1,T2,...,Tk;
2.   User_Label(T1);
3.   Labeled_T ← T1;
4.   j ← 2; OSP ← NaN;
5.   While (User not satisfied And j ≤ k)
6.        OSP ← Find_OSP (Labeled_T) ;
7.        Smoothing (OSP, Tj);
8.        User_Confirm (Tj);
9.        Labeled_T ← Labeled_T ∪ Tj;
10.       j++;
11.  End
12.  Unlabeled_T ← T – Labeled_T;
13.  Smoothing(OSP, Unlabeled_T);
```

C. Smoothing Parameter

For both B-Spline smoothing and Kernel smoothing, the choice of the smoothing parameter depends not only on the data but also on the type of patterns being modeled. If global trends of a time series are focused, a relatively smoother fitted curve with a larger smoothing parameter is preferred; if local patterns are focused, a rougher curve with a relatively smaller smoothing parameter will fit the data better. In the literature, different criteria such as minimizing *mean squared error*, mean integrated squared error (MISE), cross-validation error sum of squares (CV), and generalized cross-validation error (GCV) have been proposed to find an optimal smoothing parameter [19], [20], [22]. Unfortunately, none of these guarantees to produce the "best result" because ultimately it relies on the user's needs.

To address this issue, an incremental training algorithm is proposed in this paper. The idea is to let the user label some small portion of the data that is corrupted and using the labeled data to search for the "optimal" smoothing parameter. In the process, care must be taken to minimize the user effort required for labeling the data. This algorithm is given in Table I. The algorithm applies to both B-Spline smoothing and Kernel smoothing. The range of the smoothing parameter is divided into 10 different levels. For the B-Spline smoothing, for locally corrupted data detection, the 10 levels are defined as

$$\lambda = 10^{(i-1)/2-9}, \quad i = \{1,2,\ldots,10\} \tag{26}$$

and for globally corrupted data detection, they are

$$\lambda = 10^{i-11}, \quad i = \{1,2,\ldots,10\}. \tag{27}$$

For the Kernel Smoothing, these are defined as

$$h = (1+i/2) * \text{space}, \quad i = \{1,2,\ldots,10\} \tag{28}$$

where the *space* is the normalized time lag between two data points in the time series. The incremental training procedure is as follows.

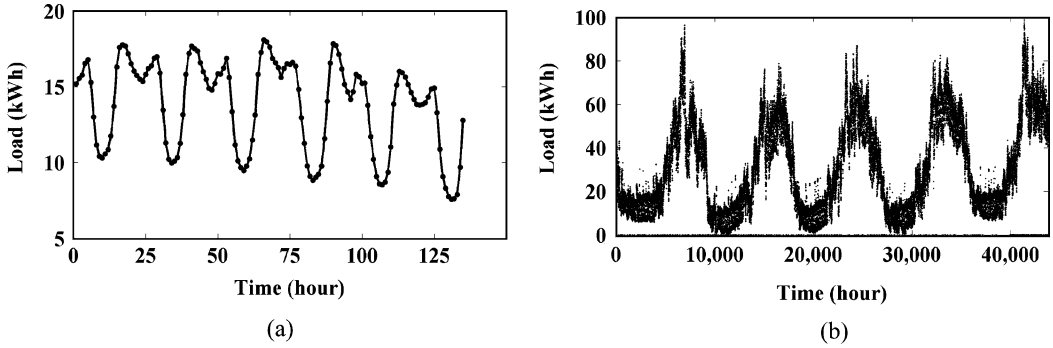


Fig. 3. Local patterns and global patterns in load curve data. (a) Local patterns. (b) Global patterns.

First, the length of the whole load time series T is divided into k partitions $\mathrm{T}_1, \ldots, \mathrm{T}_\mathrm{k}$ (see Line 1 in Table I). Initially, the user only needs to label the first partition T_1 (see Line 2). A smoothing method (either B-Spline smoothing or Kernel smoothing) presented in Section III runs on T_1 for *each* of the predefined 10 smoothness levels to identify the *trained optimal smoothing parameter* (denoted by OSP), which results in the most accurate detection of the labeled corrupted data in the first partition and is done by the function Find_OSP on Line 6 in Table I. Next, with this trained optimal smoothing parameter, the smoothing method runs to identify the candidates of corrupted data in the second partition (see Line 7). These candidates and smoothing results with the confidence interval are then presented to the user for confirmation (see Line 8). The user's confirmation enables us to label the corrupted data in the second partition. Thus, the labeled data is expanded by one partition (see Line 9).

Till now, the first iteration is completed, in which the labeled data in the first partition is utilized to help label the corrupted data in the second partition. In each iteration, the trained optimal smoothing parameter OSP is updated by rerunning the smoothing program on the labeled partitions, and the labeled data is expanded by one partition (Lines 6–9). The training process can be stopped at any iteration as long as the user is satisfied with the accuracy of the current model. If the user stops at the end of the jth run, the first $(j+1)$ partitions have been labeled and the most updated trained optimal smoothing parameter OSP will be used to construct the model to label the remaining $(k-j-1)$ partitions (see Line 10 and 11).

The essence of the incremental training algorithm is the dialogue between human being and the computer program. The initial labeling and feedback from the user provide the information for the computer program to find a better smoothness level. As more partitions are labeled, the updated model gradually becomes more robust. Another feature of this algorithm is that it can be implemented in an *online environment* where data continuously arrives in real time since only one new partition of data is required to process with the incremental training algorithm.

The incremental training algorithm in Table I is used mainly for detecting locally corrupted data. To detect globally corrupted data, the Smoothing(OSP, $\mathrm{T_j}$) is replaced with Smoothing(OSP, T). In other words, the smoothing program runs on the *whole time series* T. This change is necessary because modeling global patterns requires the examination of the whole time series. However, the user confirmation and labeling at Lines 8 and 9 are still conducted on one partition at a time.

V. Experiments

In this section, the proposed solution is tested. The real load data from BCTC was used for our experimental evaluation. The data set includes hourly residential energy consumption in a certain area for the five years from April 2004 to March 2009. Fig. 3(a) shows the data in one week and Fig. 3(b) shows five-year data distribution. The data exhibits both local patterns (by day) and global patterns (by year).

For both locally and globally corrupted data detection, we follow the same experiment procedure. First, the incremental training process described in Table I is carried out to obtain the trained optimal smoothing parameter. The data used in this training process is called the *training data set*. Second, the smoothing method with the trained optimal smoothing parameter runs on a separate prelabeled *testing data set* for accuracy evaluation. The testing data set simulates the data on which the user wants to detect corrupted data except that the labeling would not be available in real applications. The standard *precision* (P), *recall* (R), and *F-measure* (F) are used as the accuracy metrics. Precision is the percentage of correctly detected corrupted regions with regard to the total detected regions; recall is the percentage of correctly detected regions with regard to prelabeled corrupted regions; the F-measure is a harmonic mean of precision and recall, i.e.,

$$F = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \tag{29}$$

A larger F-measure indicates more accurate detection.

A. Detecting Locally Corrupted Data

To facilitate the evaluation for locally corrupted data detection, 25 weeks' data was selected from the five-year BCTC data with 168 hourly data points per week. For simplicity, each corrupted data point was considered as one region. The data in the first 12 weeks was used to carry out the incremental training following the procedure in Table I where the data in each week was treated as one partition $\mathrm{T_j}$. The trained optimal smoothing parameter (OSP) was obtained from the incremental training process. The data in the remaining 13 weeks served as the testing data for evaluation, in which all corrupted data points were manually labeled in advance. The model on the testing data was con-

TABLE II
LOCALLY CORRUPTED DATA DETECTION RESULTS

B-Spline Smoothing				Kernel Smoothing			
Level	P	R	F	Level	P	R	F
1	53%	74%	0.62	1	55%	74%	0.63
2	50%	70%	0.58	2	67%	78%	0.72
3	52%	70%	0.59	3	71%	74%	0.72
4	52%	70%	0.59	**4	90%	78%	0.84
5	73%	70%	0.71	5	89%	74%	0.81
**6	86%	78%	0.82	*6	94%	74%	0.83
*7	85%	74%	0.79	7	94%	74%	0.83
8	78%	61%	0.68	8	88%	65%	0.75
9	82%	61%	0.7	9	88%	65%	0.75
10	81%	57%	0.67	10	88%	61%	0.72

structed using the OSP obtained from the incremental training process. The precision, recall and F-measure were recorded. For comparison, the other 9 predefined smoothness levels were also tested on the testing data. The results produced by all 10 smoothness levels are shown in Table II.

In Table II, the left half shows the results obtained by using B-Spline smoothing and the right half shows those obtained by using Kernel smoothing. The "Level" column represents the smoothness level, with 1 being the least smooth and 10 being the smoothest; the "P" column represents precision; the "R" column represents recall; the "F" column represents F-measure. The row with $*$ shows the result produced by the trained optimal smoothing parameter, whereas the row with $**$ shows the best result that can be produced by any of the 10 predefined smoothness levels. It can be seen from Table II that the trained optimal smoothing parameter (labeled by $*$) produces the result that is very close to the best one (labeled by $**$). It is believed that the difference will be even smaller if more prelabeled data for both training and testing are used.

It can be observed that as the smoothness level increases, the F-measure increases and reaches the peak, and then begins to decrease. To understand this behavior, the fitted curve and confidence interval produced by the B-Spline smoothing for one week's testing data at the smoothness level 1, 6, and 10 are presented in Fig. 4. The solid line with points represents real observations, the dashed line with points represents the fitted curve, and the dashed lines without points represent the upper bound and lower bound of the confidence interval. Fig. 4(a) shows the raw data with 5 labeled locally corrupted data points. When the smoothing parameter is selected at Level 1 (least smooth), as shown in the Fig. 4(b), the curve tends to fit both normal data and corrupted data. When the smoothing parameter increases to Level 6, indicated in Fig. 4(c), the curve becomes to reveal the real local patterns of the data and filter out corrupted data. Fig. 4(d) indicates that an overly smooth curve fails to identify some corrupted data because of failure to model the local patterns.

A closer look reveals the following interesting behaviors of precision and recall with respect to different smoothness levels. The precision metric largely increases as the fitted curve gets smoother (i.e., at a higher smoothness level). This increase is contributed by the decrease of the number of wrongly detected corrupted data. As a matter of fact, a smoother curve causes more fitting errors, thus, a larger MSE and a wider confidence interval. In this case, fewer data points fall outside the confidence interval and these data points are most likely true corrupted data. Therefore, a higher smoothness level helps increase the precision metric.

In contrast, the recall metric behaves differently. When the smoothing parameter is set at a low smoothness level, the curve is rough and tends to overfit the data. In this case, the recall metric is low because corrupted data is also fitted instead of being detected. As the smoothness level increases, the curve is getting smoother and gradually approximating the intrinsic structure of the real data, thus, more abnormal deviations from the function are detected. After reaching the "peak point" (i.e., Level 6 for the B-Spline smoothing and Level 4 for the Kernel smoothing), further increasing smoothness level produces an overly smooth curve, thus, a large MSE. This causes a too wide confidence interval that fails to exclude the corrupted data and therefore leads to a decrease of the recall metric.

B. Detecting Globally Corrupted Data

A globally corrupted region is a region that includes a group of nearby corrupted data. For globally corrupted data detection, the data in all the five years was considered. The data in the first two years was used as training data and the rest three-year data was used as testing data. Fifteen globally corrupted regions in the testing data were manually prelabeled. The incremental training algorithm ran on the training data to obtain the trained optimal smoothing parameter (OSP). Then the smoothing program was carried out on the testing data with the OSP. The results using the B-Spline smoothing are shown in Fig. 5. The circles in Fig. 5(a) represent prelabeled globally corrupted regions in the three-year testing data; the circles in Fig. 5(b) and (c) represent correctly detected corrupted regions; the rectangles represent corrupted regions that were not detected.

In Fig. 5(b), the fitted curve and confidence interval produced by the OSP (Level 7) are presented. As showed, 14 out of 15 globally corrupted regions are detected. For comparison, Fig. 5(c) gives the result with the smoothness Level 4, in which only 5 out of 15 are correctly detected.

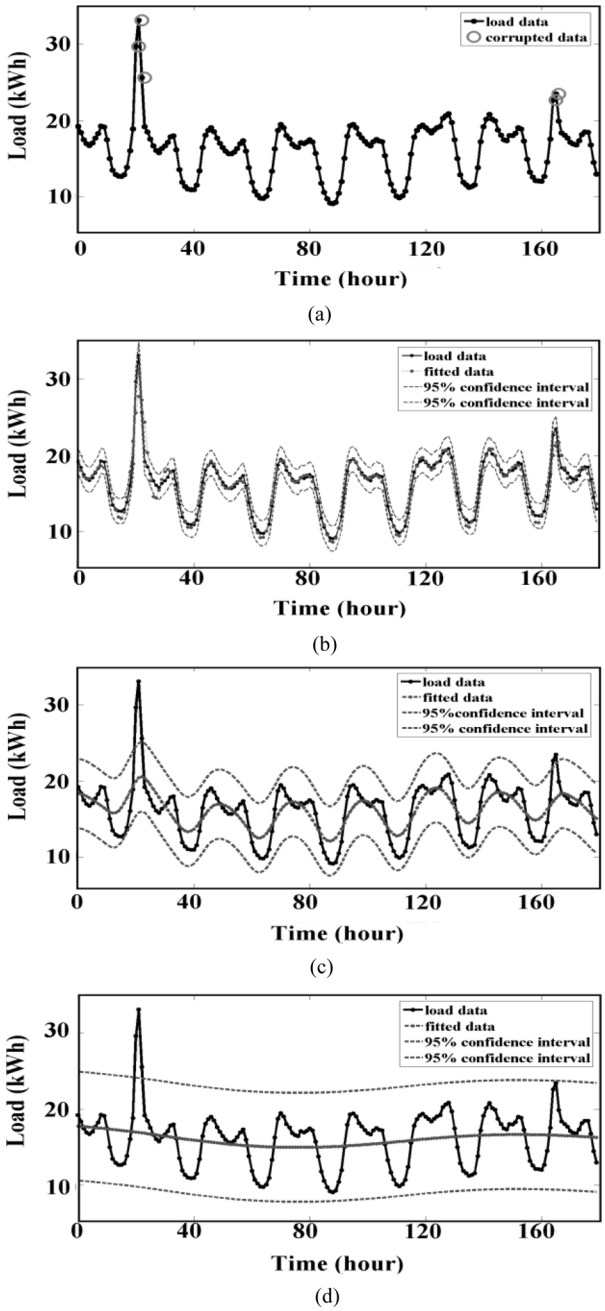


Fig. 4. Locally corrupted data detection—B-Spline smoothing. (a) Real observations with 5 corrupted data points in circles. (b) Result for smoothness Level 1. (c) Result for smoothness Level 6. (d) Result for smoothness Level 10.

As expected, when the smoothing parameter is at a low smoothness level, such as in Fig. 5(c), the curve tends to model too much detailed information or local patterns. As a result, globally corrupted data is fitted as normal observations rather than being detected. In contrast, when the curve becomes smoother, as modeled in Fig. 5(b), the performance improves accordingly as the local information is ignored and more general data behaviors are modeled. Compared to locally corrupted data detection, a higher smoothness level (i.e., a smoother curve) is usually preferred to detect globally corrupted data. It is also interesting to note that the smoother fitted curve in Fig. 5(b) correctly preserves the global trends exhibited by the annual seasonality.

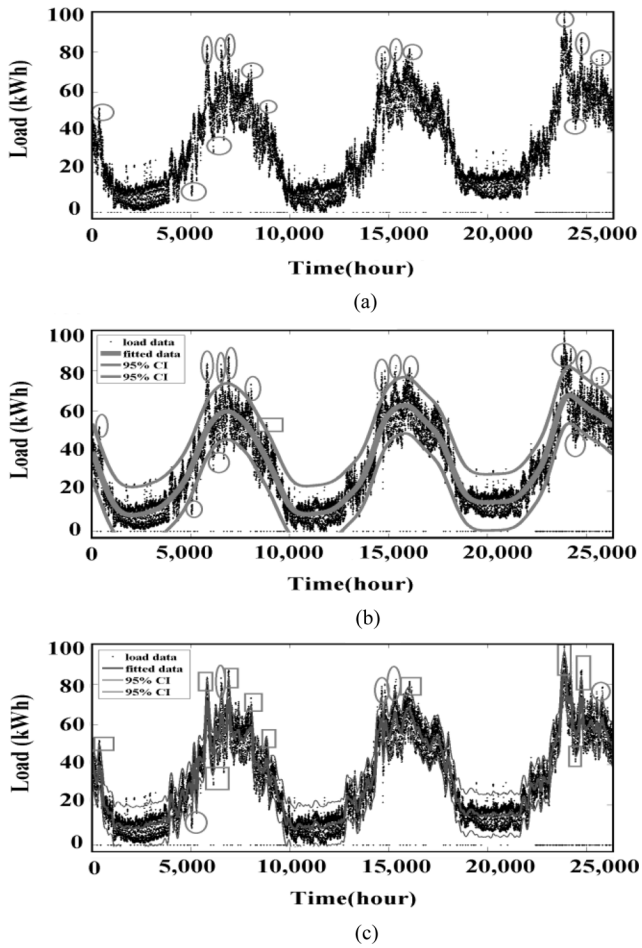


Fig. 5. Globally corrupted data detection for the three-year testing data. The predefined 10 smoothness levels are different in locally and globally corrupted data detection (refer to Section IV-C). (a) Three-year testing data with 15 globally corrupted regions. (b) B-Spline smoothing result with smoothness Level 7. (c) B-Spline smoothing result with smoothness Level 4.

VI. Conclusion

Load curve data is noisy and contains corrupted data. Detecting and correcting the corrupted data in load curves is the first step for further data analysis and is particularly important for the smart grid that will be in place in the future. In this paper, this problem is addressed as *load cleansing problem*. A practical solution based on well founded nonparametric regression techniques has been presented. The solution handles both locally corrupted data and globally corrupted data in a uniform way through the setting of a single smoothing parameter. A challenge in implementation is how to determine the best smoothness level for the smoothing parameter, which ultimately requires user involvement. One novelty of the presented solution is taking into account of user input while minimizing user effort. The user effort is minimized by performing the proposed smoothing methods incrementally in multiple runs so that the user can provide input based on partial results in previous runs. The automatic detection and the user input form a man–machine dialogue mechanism where each does its utmost. This greatly improves the overall performance. The evaluation on the real BCTC load curve data demonstrated the effectiveness of the presented solution. Although this paper focuses on load curve data

for electric utilities, we believe that our solution is equally applicable to time series data in other domains.

REFERENCES

[1] W. Li, *Risk Assessment of Power Systems: Models, Methods, and Applications*. New York: IEEE Press—Wiley, 2005.

[2] Smart Grid, U.S. Dept. Energy [Online]. Available: http://www.oe.energy.gov/smartgrid.htm

[3] R. Sridhar, R. Rajeev, and S. Kyuseok, "Efficient algorithms for mining outliers from large data sets," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 427–438, Jun. 2000.

[4] A. Nairac, N. Townsend, R. Carr, S. King, P. Cowley, and L. Tarassenko, "A system for the analysis of jet engine vibration data," *Integr. Comput.-Aided Eng.*, vol. 6, no. 1, pp. 53–66, Jan. 1999.

[5] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *Proc. 4th Int. Conf. Data Warehousing Knowledge Discovery*, 2002, pp. 170–180.

[6] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artif. Intell. Rev.*, vol. 22, no. 2, pp. 85–126, Oct. 2004.

[7] L. Wei, E. Keogh, and X. Xi, "SAXually explicit images: Finding unusual shapes," ICDM 2006.

[8] D. Yankov, E. Keogh, and U. Rebbapragada, "Disk aware discord discovery: Finding unusual time series in terabyte sized datasets," in *Proc. 6th Int. Conf. Data Mining*, 2007, pp. 711–720.

[9] P. K. Chan and M. V. Mahoney, "Modeling multiple time series for anomaly detection," in *Proc. 5th Int. Conf. Data Mining*, 2005, pp. 90–97.

[10] M. V. Mahoney and P. K. Chan, "Trajectory boundary modeling of time series for anomaly detection," presented at the KDD Workshop Data Mining Methods Anomaly Detection, Chicago, IL, 2005.

[11] S. Salvador, P. Chan, and J. Brodie, "Learning states and rules for time series anomaly detection," in *Proc. 17th Int. FLAIRS Conf.*, 2005, pp. 300–305.

[12] E. Keogh, J. Lin, and A. Fu, "HOT SAX: Finding the most unusual time series subsequence: Algorithms and applications," in *Proc. 5th Int. Conf. Data Mining*, 2005, pp. 226–233.

[13] Y. Bu, T.-W. Leung, A. W.-C. Fu, E. Keogh, J. Pei, and S. Meshkin, "WAT: Finding top-K discords in time series database," presented at the 2007 SIAM Int. Conf. Data Mining, Minneapolis, MN.

[14] A. W.-C. Fu, O. T.-W. Leung, E. Keogh, and J. Lin, "Finding time series discords based on Haar Transform," in *Proc. 2nd Int. Conf. Adv. Data Mining Appl*, 2006, pp. 31–41.

[15] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Towards parameter-free data mining," in *Proc. 10th Int. Conf. Knowledge Discovery and Data Mining*, 2004, pp. 206–215.

[16] A. Fallon and C. spada, "Detection and accommodation of outliers in normally distributed data sets," [Online]. Available: http://www.cee.vt.edu/ewr/environmental/teach/smprimer/outlier/outlier.html

[17] A. J. Fox, "Outliers in time series," *J. Roy. Stat. Soc. B, Methodological*, vol. 34, pp. 350–363, 1972.

[18] G. M. Ljung, "On outlier detection in time series," *J. Roy. Stat. Soc. B, Methodological*, vol. 55, pp. 559–567, 1993.

[19] W. Hrdle, *Applied Nonparametric Regression*. Cambridge, U.K: Cambridge Univ. Press, 1990.

[20] J. O. Ramsay and B. W. Silverman, *Functional Data Analysis*, 2nd ed. New York: Springer, 2005.

[21] C. de Boor, *A Practical Guide to Splines*. New York: Springer, 2001.

[22] M. P. Wand and M. C. Jones, *Kernel Smoothing*. London, U.K.: Chapman & Hall, 1995.

[23] V. Barnett and T. Lewis, *Outliers in Statistical Data*, 3rd ed. New York: Wiley, 1994, pp. 397–415.

[24] B. Abraham and N. Yatawara, "A score test for detection of time series outliers," *J. Time Ser. Anal.*, vol. 9, pp. 109–119, 1988.

[25] B. Abraham and A. Chuang, "Outlier detection and time series modeling," *Technometrics*, vol. 31, pp. 241–248, 1989.

[26] W. Schmid, "The multiple outlier problem in time series analysis," *Australian. J. Stat.*, vol. 28, pp. 400–413, 1986.

[27] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 4th ed. Hoboken, NJ: Wiley, 2008.

[28] M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li, *Applied Linear Statistical Models*, 5th ed. New York: McGraw-Hill/Irwin, 2004.

[29] J. W. Taylor, "An evaluation of methods for very short-term load forecasting, using minute-by-minute British data," *Int. J. Forecasting*, vol. 24, pp. 645–658, 2008.

Jiyi Chen received the B.S. degree at Peking University, China. He is currently working toward the M.S. degree in the School of Computing Science, Simon Fraser University, Vancouver, BC, Canada.

His research interests include data mining and time series analysis.

Wenyuan Li (SM'89–F'02) received the Ph.D. degree from Chongqing University, China, in 1987.

He is currently the Principal Engineer with BC Hydro, Vancouver, BC, Canada. He has authored four books and published more than 120 papers in power systems.

Dr. Li is a Fellow of the Engineering Institute of Canada. He is the recipient of several IEEE awards.

Adriel Lau received the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada.

He is currently a Senior Engineer with BC Hydro, Vancouver, BC, Canada.

Dr. Lau is a Registered Professional Engineer

Jiguo Cao received the Ph.D. degree in statistics from McGill University, Canada, in 2006.

He is an Assistant Professor in the Department of Statistics and Actuarial Science, Simon Fraser University, Burnaby, BC, Canada. His research is partly supported by a discovery grant from the Natural Science and Engineering Research Council of Canada.

Ke Wang (M'07–SM'07) received the Ph.D. degree from the Georgia Institute of Technology, Atlanta.

He is a Professor in the School of Computing Science, Simon Fraser University, Vancouver, BC, Canada. His research interests include database and data mining.