Hinging Hyperplanes for Time-Series Segmentation

Xiaolin Huang, *Member, IEEE*, Marin Matijaš, and Johan A. K. Suykens, *Senior Member, IEEE*

***Abstract*— Division of a time series into segments is a common technique for time-series processing, and is known as segmentation. Segmentation is traditionally done by linear interpolation in order to guarantee the continuity of the reconstructed time series. The interpolation-based segmentation methods may perform poorly for data with a level of noise because interpolation is noise sensitive. To handle the problem, this paper establishes an explicit expression for segmentation from a compact representation for piecewise linear functions using hinging hyperplanes. This expression enables the use of regression to obtain a continuous reconstructed signal and, as a consequence, application of advanced techniques in segmentation. In this paper, a least squares support vector machine with lasso using a hinging feature map is given and analyzed, based on which a segmentation algorithm and its online version are established. Numerical experiments conducted on synthetic and real-world datasets demonstrate the advantages of our methods compared to existing segmentation algorithms.**

***Index Terms*— Hinging hyperplanes, lasso, least squares support vector machine, segmentation, time series.**

I. Introduction

SEGMENTATION is an important issue in time-series analysis and has been applied in many fields such as data management, image processing, smart grid, finance, and medical science. Typically, for a set of time points $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$, where N is the number of data points, and the corresponding signal values are $y(t_1), y(t_2), \ldots, y(t_N)$, the segmentation problem is to find an approximation representation $\hat{f}(t)$, which equals a simple function in each segment, to describe the signal. Several models have been proposed, such as Fourier transform [1], [2], wavelet transform [3], piecewise polynomial representation [4], and piecewise linear (PWL) representation [5]–[11]. Among these models, PWL representation is widely used because of its simplicity. The main advantage of $\hat{f}(t)$ being a PWL function is that $\hat{f}(t)$ is a linear function in each of the segments, which is very useful for change point detection, periodicity analysis, and forecasting.

According to the definition of PWL function, a PWL function $\hat{f}(t)$ can be constructed by applying linear techniques on each segment, when the segmentation points are found. Therefore, the crucial issue for constructing $\hat{f}(t)$ is to find the segmentation points, denoted by the segmentation point vector $S = [s_1, s_2, \ldots, s_M]^T$ with $s_m < s_{m+1}, m = 1, 2, \ldots, M$, where M is the number of segments and m is the segment index. When S is known, there are two ways to find the line between s_m and s_{m+1}. One way is by linear interpolation on interval $[s_m, s_{m+1}]$. In order to do linear interpolation, the values of $\hat{f}(s_m)$ and $\hat{f}(s_{m+1})$ should be known, which means s_m, s_{m+1} should be the sampling time points, i.e., $s_m \in \mathcal{T}, \forall m$. Utilizing linear interpolation on each segment, a continuous PWL function can be constructed, which is determined by S and denoted by $g_S(t)$. In a segmentation problem, one wants to find the best segmentation points, i.e., find a small number of segments to achieve high accuracy. We can describe the problem as minimizing the error between the original and the reconstructed signal with the condition that only M segmentation points are used. If linear interpolation is used in each segment, then the problem can be posed as

$$\min_{s_m \in \mathcal{T}, S \in \mathbb{R}^M} \left\{ \sum_{i=1}^{N} (y(t_i) - g_S(t_i))^2 \right\}. \tag{1}$$

To solve (1), researchers have proposed various algorithms, which can be categorized in the following classes: top-down algorithms [5], bottom-up algorithms [12], dynamic programming [13], and sliding window algorithms [6], [8], [9]. These algorithms perform well in some applications, but when the observed data are corrupted by noise, the results are poor. This weak point comes from the fact that $g_S(t)$ is constructed by linear interpolation. In [11], some techniques are used to make the segmentation method less sensitive to noise, but that algorithm is still based on interpolation, which is essentially sensitive to noise. One way to deal with noise is to use linear regression on each segment instead of interpolation. Following this idea, [4], [8] tried to use linear regression in each segment. However, simply doing linear regression leads to a function which is discontinuous in the segmentation points. Linear interpolation technique but not linear regression is used in most existing segmentation methods because we usually want to get a continuous signal. Let us illustrate the difference with a simple example. In this example, the underlying signal $y(t) = \sin(t/10)^2$ is corrupted by Gaussian noise with mean 0 and standard deviation 0.1, shown in Fig. 1(a). Let the segmentation point vector be $S = [1, 7, 15, 23, 32, 37, 44]^T$, and the reconstructed signals by linear interpolation and linear

Manuscript received June 6, 2012; revised December 20, 2012; accepted March 18, 2013. Date of publication April 26, 2013; date of current version June 28, 2013. 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, projects: G0226.06, G.0302.07, G.0320.08, G.0558.08, G.0557.08, G.0588.09, G.0377.09, and G.0377.12; and IWT Ph.D. Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&ODsquare; Belgian Federal Science Policy Office: IUAP P6/04, IBBT, EU: ERNSI; ERC AdG A-DATADRIVE-B, FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), Contract Research: AMINAL; Helmholtz: viCERP, ACCM, Bauknecht, Hoerbiger.

X. Huang and J. A. K. Suykens are with the Department of Electrical Engineering ESAT-SCD-SISTA, KU Leuven, Leuven B-3001, Belgium (e-mail: huangxl06@mails.tsinghua.edu.cn; johan.suykens@esat.kuleuven.be).

M. Matijaš is with the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb 10000, Croatia (e-mail: marin.matijas@fer.hr).

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/TNNLS.2013.2254720

2162-237X/$31.00 © 2013 IEEE

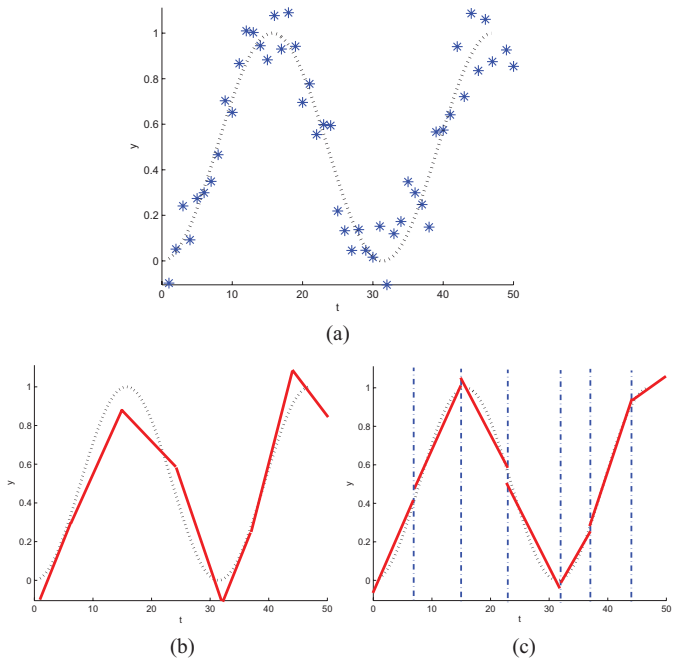


Fig. 1. Example of a noise-corrupted signal. (a) Signal is shown by the dashed line and the observed corrupted data are shown by stars. (b) Result of linear interpolation is very sensitive to noise. (c) Result of linear regression can tolerate some noise but is discontinuous.

regression are shown in Fig. 1(b) and (c), respectively. We can see that the result of interpolation is very sensitive to noise. Because of that noise, accuracy of the regression is better, but in the case of regression the signal is discontinuous [Fig. 1(c)].

In this paper, we propose a new method for segmentation which uses regression to handle the noise, but meanwhile keeps the reconstructed signal continuous. For that purpose, we introduce a compact representation of a continuous PWL function into the field of segmentation. The first compact representation for continuous PWL function was given by Chua [14]. Since then, a series of such models have been established in [15]–[19]. The major goal of establishing these representation models was to extend the representation capability for high-dimensional continuous PWL functions. In a univariate time-series problem, the signal is a 1-D function and the representation capability of hinging hyperplanes (HH) [15] is satisfactory, i.e., any 1-D continuous PWL function can be represented by an HH. Therefore, in this paper, we will use HH function for segmentation after which regression can be used and the continuity can be guaranteed. Moreover, representing a continuous PWL function by HH makes it possible to use some advanced techniques, such as least squares support vector machine (LS-SVM [20]) and l_1-regularization (lasso [21]) to detect the segmentation points.

The remainder of this paper is organized as follows: HH and the related segmentation problems are discussed in Section II. The new segmentation algorithm, using HH, LS-SVM, and lasso, is given in Section III. Section IV discusses the online segmentation method. The proposed algorithms are tested on numerical experiments in Section V. Section VI ends this paper with concluding remarks.

II. Segmentation Using Hinging Hyperplanes

A. Global Regression Using Hinging Hyperplanes

Hinging hyperplanes, proposed in [15], take the form

$$h_{\omega,S}(t) = \omega_0 + \sum_{m=1}^{M} \omega_m \varphi_m(t) \tag{2}$$

where $\varphi_m(t) = \max\{0, t - s_m\}$ is the basis function, called the hinge function because of its geometrical shape. $\omega = [\omega_0, \omega_1, \ldots, \omega_M]^T$ and $S = [s_1, s_2, \ldots, s_M]^T$ are the parameters of HH. Without any loss of generality, one can assume $s_m < s_{m+1}$ and $h_{\omega,S}(t)$ equals a linear function in segment $[s_m, s_{m+1}]$, which means that vector S defines the segmentation points. Equation (2) is naturally continuous for any ω and S, because it is the composition of continuous functions. Hence, no additional constraints are needed and linear regression can be used. Instead of linear regression in each segment, we perform global regression to find the parameters in (2). Then the time-series segmentation problem can be formulated as

$$\min_{\omega,S} \sum_{i=1}^{N} \left(y(t_i) - \omega_0 - \sum_{m=1}^{M} \omega_m \max\{0, t_i - s_m\} \right)^2. \tag{3}$$

The analytic results about convergence rate and the error bound have been given in [15]. Moreover, it has been proved that any 1D continuous PWL function can be represented by HH. Specifically, when S with $s_m \in \mathcal{T}$ is given, $g_S(t)$ can be represented by HH according to the interpolation condition $h_{\omega,S}(s_m) = g_S(s_m) = y(s_m), m = 1, 2, \ldots, M$, which can be posed as the following set of linear equations:

$$\begin{aligned} \omega_0 &= y(s_1) \\ \omega_0 + \omega_1(s_2 - s_1) &= y(s_2) \\ &\vdots \\ \omega_0 + \omega_1(s_M - s_1) + \ldots + \omega_{M-1}(s_M - s_{M-1}) &= y(s_M). \end{aligned}$$

The coefficient matrix of the above equations is lower triangular, and the solution, denoted by $\tilde{\omega}_0, \tilde{\omega}_1, \ldots, \tilde{\omega}_{M-1}$, can be obtained by Gaussian elimination. Then it can be verified that with $\tilde{\omega}_0, \tilde{\omega}_1, \ldots, \tilde{\omega}_{M-1}$, $h_{\tilde{\omega},S}(t) = g_S(t), \forall t \in [s_1, s_M]$. From this equivalence, we find that $g_S(t)$, obtained by solving (1) by any interpolation-based segmentation method, provides a candidate solution for (3). That candidate solution has to satisfy the constraints $s_m \in \mathcal{T}$ and $h_{\omega,S}(s_m) = y(s_m)$, which are not needed in (3). Therefore, solving (3) can give a more accurate result than interpolation-based segmentation methods.

Consider again the example shown in Fig. 1. We fix the segmentation points $S = [1, 7, 15, 23, 32, 37, 44]^T$ as used in Fig. 1(b) then solve (3), which is a least squares problem for given S. The result is shown in Fig. 2, and one can see that the result of using HH is continuous and insensitive to the noise.

Besides accuracy and runtime, the compression rate is important for segmentation methods. Interpolation-based segmentation methods have to record segmentation points s_m and the corresponding signal values $y(s_m)$. The segmentation points s_m and the coefficients ω_m are needed to store an HH (2). That storing gives the same compression rate as that of interpolation-based segmentation methods.

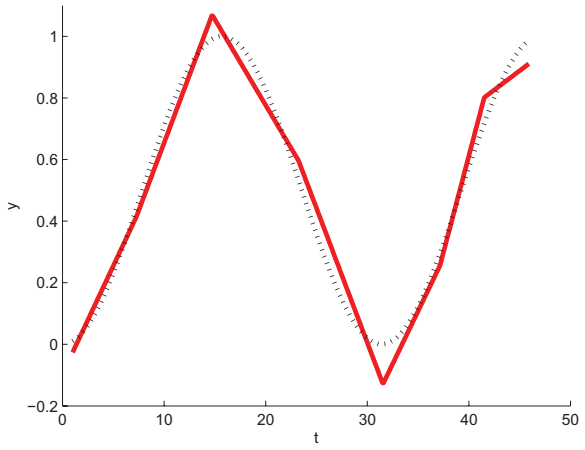


Fig. 2. Signal (dashed line) and the reconstructed signal using HH (solid line) from the data shown in Fig. 1(a). The result is less sensitive with respect to noise [compared to Fig. 1(b)] and and continuous [compared to Fig. 1(c)].

B. Training for Segmentation Points

For a given S, (3) becomes a least squares problem and the corresponding optimal ω can be found. The result of using HH with fixed S can tolerate the noise and has shown some advantages over interpolation-based segmentation algorithms. Besides, the segmentation point vector $S = [s_1, s_2, \ldots, s_M]^T$ can be adjusted to further improve the performance.

To adjust S, some efficient algorithms have been proposed. In [15], a hinge-finding algorithm was established from the geometrical meaning of hinge functions. Then, it has been proved that the hinge-finding algorithm is equal to the fixed-stepsize Newton algorithm in [22] and then a damped modified Newton algorithm has been proposed. We denote the sum of the squared error, which is the objective of (3), by $e_{\text{sse}}(\omega, S)$

$$e_{\text{sse}}(\omega, S) = \sum_{i=1}^{N} r_i(\omega, S)^2$$

where $r_i(\omega, S) = y(t_i) - \omega_0 - \sum_{i=1}^{M} \omega_m \max\{0, t_i - s_m\}$ is the individual residual. Then the training algorithm is formed as a two-step iterative algorithm. The first step is estimating ω with given S. The second step is fixing the obtained ω and updating S by the modified Gauss–Newton method, whose formulation is

$$S' = S - \zeta((J^T J)^{-1} J^T) e_{\text{sse}}(\omega, S) \tag{4}$$

where S' is the new segmentation point vector, ζ is the stepwise length, and J is the Jacobian matrix of $e_{\text{sse}}(\omega, S)$ in this step

$$J = \frac{\partial e_{\text{sse}}(\omega, S)}{\partial S}.$$

Notice that in the strict mathematical sense the derivative does not exist in some points. However, because HH is continuous, we can define the derivatives in such points with a slight influence on the final result, as follows:

$$\frac{\partial \max\{0, t - s_m\}}{\partial s_m} = \begin{cases} -1, & \text{if } t \geq s_m \\ 0, & \text{else.} \end{cases}$$

After getting new S, we turn back to the first step, i.e., estimating ω with fixed S, and then we use (4) again to update S. The above process is repeated until e_{sse} does not decrease. The discussion of the global convergence for this training method can be found in [22]. In this paper, we apply an inexact linear search to find ζ and guarantee the convergence. One can also consider damped step length. As mentioned in Section II-A, s_m represents the segmentation points, i.e., the intersection points of two consecutive lines. Naturally, we want these points to be located in the region of interest, i.e., $t_1 \leq s_m \leq t_N$. If s_m is located outside the region of interest, it has no effect on the error, since $\max\{0, t - s_m\}$ reduces to a linear function in $[t_1, t_N]$, which is equivalent to $\omega_m = 0$. From this observation, one can see that the above training process will not make a segmentation point lie outside the region of interest, and therefore we do not need to consider additional constraints $t_1 \leq s_m \leq t_N$.

Though error $e_{\text{sse}}(\omega, S)$ is nonconvex with respect to S and the globally optimal segmentation points cannot be guaranteed, the above training strategy can improve the accuracy. It also helps us to detect the change points, especially when the sampling points are sparse. We illustrate the performance of training S, on a toy example, in which the signal is a continuous PWL function. The underlying function and the sampling points are shown in Fig. 3(a). The sampling time points are $\mathcal{T} = \{7, 14, 28, \ldots, 7k, \ldots, 175\}$. There is no noise but the change points (the best segmentation points) $t = 40, 80, 120, 160$ are missed when sampling. Using any interpolation-based algorithm, the desired points cannot be detected because there is a constraint $s_m \in \mathcal{T}$. As an example, the result of the feasible sliding window (FSW) algorithm (FSW [9]) with threshold 30 is illustrated in Fig. 3(b), from which one can see that the detected segmentation points are $S = [48, 90, 120, 156, 162]^T$. Now we use HH for segmentation by solving (3). The segmentation points are trained from the initial $S = [48, 90, 120, 156, 162]^T$ by the training strategy described previously. After the training, S becomes $[40.24, 81.20, 120.16, 159.44]^T$, which is more accurate than the results of FSW. The resulting signals of the initial and the trained S are illustrated by the dashed line and the solid line in Fig. 3(c), respectively. From the results one can see the effectiveness of the training strategy and the advantages of using HH over interpolation-based algorithms for segmentation.

III. LS-SVM With Hinging Feature Map

As shown above, using HH is advantageous for segmentation over interpolation-based algorithms. But because (3) is nonconvex with respect to S, the performance depends on the initial selection of S. In this paper, we use HH to present the segmentation problem in a closed form, and some advanced machine learning techniques hence become applicable.

A. Formulation of LS-SVM Using HH

Since the SVM was developed by Vapnik in [23] along with other researchers, it has been applied widely. SVM has shown great performance in classification, regression, clustering, and other applications; however, it has not yet been used for segmentation problems because of the lack of a closed form in interpolation-based methods. In this paper, HH is introduced

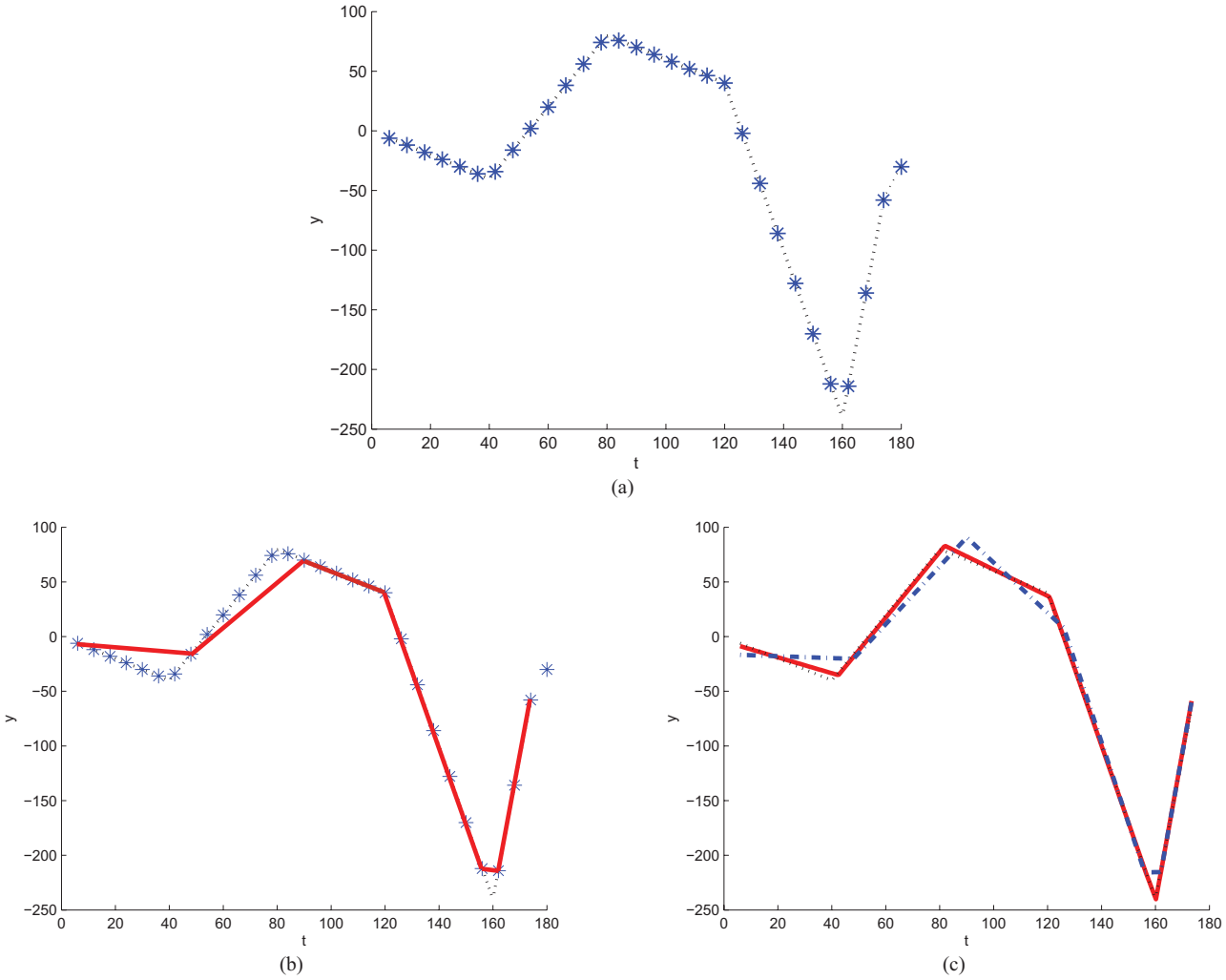


Fig. 3. Example of segmentation points training. (a) Signal (dashed line) and the observed data (stars). Note that the change points $t = 40, 80, 160$ are missed when sampling. (b) Signal (dashed line) and the result of FSW with threshold 30 (red line). (c) Signal (dashed line) and the results corresponding to the initial S (dash-dotted line) and the trained S (solid line).

and the relationship between the approximation error and the segmentation points is represented explicitly; henceforth, SVMs are applicable for segmentation problems. Among many kinds of SVMs, we use LS-SVM, proposed in [20] and [24], because it involves only linear equality constraints and can be solved very efficiently. LS-SVM has been widely applied in classification, regression, and other fields, including some recent works in [25]–[27].

The formulation of LS-SVM can be written as

$$\min_{\omega,e} \ \frac{1}{2}\sum_{m=1}^{M}\omega_m^2 + \gamma\frac{1}{2}\sum_{i=1}^{N}e_i^2 \tag{5}$$
$$\text{s.t.} \ \ y(t_i) = e_i + \omega_0 + \sum_{m=1}^{M}\omega_m\phi_m(t_i), \quad i = 1, 2, \ldots, N$$

where $e = [e_1, e_2, \ldots, e_N]^T$ is the residual vector and $\gamma > 0$ is the regularization constant. Let $\phi_m(t)$ be the hinge function, i.e., $\phi_m(t) = \max\{0, t - s_m\}, m = 1, 2, \ldots, M$; then the feature map $\phi(t) = [\phi_1(t), \phi_2(t), \ldots, \phi_M(t)]^T$ is named as the hinging feature map and the output of LS-SVM (5) gives an HH $h_{\omega,S}(t) = \omega_0 + \sum_m \omega_m \max\{0, t - s_m\}$. The segmentation training strategy can be modified for (5), which is actually a descent method for tuning kernel parameters for SVM. Like previously, $e_{\text{sse}}(\omega, S)$ is the sum of squared error, and the objective value of (5) can be written as $\frac{1}{2}\gamma\, e_{\text{sse}}(\omega, S) + \frac{1}{2}\sum_{m=1}^{M}\omega_m^2$. When training S, the update formulation is the same as in (4) with the difference that the objective function alters when doing line search.

Using the hinging feature map, we guarantee that the obtained function is continuous PWL, which is suitable for segmentation problems, and, by using LS-SVM, we can find a less sensitive result, which can tolerate some noise. Next, we will try to find reasonable segmentation points based on LS-SVM with the hinging feature map. The idea is to first find all the possible segmentation points and then to reduce the number of segmentation points by using the basis pursuit technique. An efficient method of generating a sparse solution, which contains a number of zero components, is l_1-regularization. This method was originally proposed in [21] and it is well known as lasso. Lasso helps us to reduce the number of segmentation points, and based on it we propose

the following formulation for segmentation:

$$\begin{aligned} \min_{\omega,e} \quad & \frac{1}{2}\sum_{m=1}^{M}\omega_m^2 + \gamma\frac{1}{2}\sum_{i=1}^{N}e_i^2 + \sum_{m=1}^{M}\mu_m|\omega_m| \\ \text{s.t.} \quad & y(t_i) = e_i + \omega_0 + \sum_{m=1}^{M}\omega_m\phi_m(t_i), \quad i = 1, 2, \ldots, N \end{aligned} \tag{6}$$

where $\phi_m(t) = \max\{0, t - s_m\}$ and $\mu = [\mu_1, \ldots, \mu_M]^T$ is the weight vector. Essentially, (6) is an LS-SVM with lasso using hinging feature map. This is a convex optimization problem and the optimal solution can be obtained. According to the value of $|\omega_m|$, the segmentation points s_m with nonzero ω_m are selected to generate the segmentation point vector S which can be trained further.

We consider LS-SVM with hinging feature map which can be handled in either primal or dual space. To solve (6), we transform it into the following constrained quadratic programming (QP):

$$\begin{aligned} \min_{\omega,e,u} \quad & \frac{1}{2}\sum_{m=1}^{M}\omega_m^2 + \gamma\frac{1}{2}\sum_{i=1}^{N}e_i^2 + \sum_{m=1}^{M}\mu_m u_m \\ \text{s.t.} \quad & y(t_i) = e_i + \omega_0 + \sum_{m=1}^{M}\omega_m\phi_m(t_i), \quad i = 1, 2, \ldots, N, \\ & -u_m \le \omega_m \le u_m, \quad m = 1, 2, \ldots, M. \end{aligned} \tag{7}$$

That means any QP solver can be applied to solve (6). Next, we consider the dual formulation. The Lagrangian of (7) is

$$\begin{aligned} &\mathcal{L}(\omega, e, u, \lambda, \alpha, \beta) \\ &= \frac{1}{2}\sum_{m=1}^{M}\omega_m^2 + \gamma\frac{1}{2}\sum_{i=1}^{N}e_i^2 + \sum_{m=1}^{M}\mu_m u_m \\ &\quad - \sum_{i=1}^{N}\lambda_i\left(e_i + \omega_0 + \sum_{m=1}^{M}\omega_m\phi_m(t_i) - y(t_i)\right) \\ &\quad + \sum_{m=1}^{M}\alpha_m(-u_m - \omega_m) + \sum_{m=1}^{M}\beta_m(-u_m + \omega_m) \end{aligned}$$

where λ_i, α_m, and β_m are the Lagrangian dual variables. The optimality condition is given by

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\omega_m} &= \omega_m - \sum_{i=1}^{N}\lambda_i\phi_m(t_i) - \alpha_m + \beta_m = 0 \\ m &= 1, 2, \ldots, M \\ \frac{\partial\mathcal{L}}{\partial\omega_0} &= \sum_{i=1}^{N}\lambda_i = 0 \\ \frac{\partial\mathcal{L}}{\partial e_i} &= \gamma e_i - \lambda_i = 0, \quad i = 1, 2, \ldots, N \\ \frac{\partial\mathcal{L}}{\partial u_m} &= \mu_m - \alpha_m - \beta_m, \quad m = 1, 2, \ldots, M. \end{aligned}$$

According to the optimality condition, the dual problem of (7) can be written as

$$\begin{aligned} \max_{\lambda,\alpha,\beta} \quad & -\frac{1}{2}\sum_{m=1}^{M}\left(\sum_{i=1}^{N}\lambda_i\phi_m(t_i)\right)^2 - \frac{1}{2}\sum_{m=1}^{M}(\alpha_m - \beta_m)^2 \\ & - \sum_{m=1}^{M}(\alpha_m - \beta_m)\sum_{i=1}^{N}\lambda_i\phi_m(t_i) - \frac{1}{2\gamma}\sum_{i=1}^{N}\lambda_i^2 \\ & + \sum_{i=1}^{N}\lambda_i y(t_i) \\ \text{s.t.} \quad & \sum_{i=1}^{N}\lambda_i = 0 \\ & \alpha_m + \beta_m = \mu_m, \quad m = 1, 2, \ldots, M \\ & \alpha_m, \beta_m \ge 0, \quad m = 1, 2, \ldots, M. \end{aligned} \tag{8}$$

The optimal dual variables can be obtained via solving this QP. Then, one can represent the reconstructed signal as

$$\begin{aligned} h_{\omega,S}(t) &= \sum_{m=1}^{M}\omega_m\phi_m(t) + \omega_0 \\ &= \sum_{m=1}^{M}\left(\alpha_m - \beta_m + \sum_{i=1}^{N}\lambda_i\phi_m(t_i)\right)\phi_m(t) + \omega_0 \\ &= \sum_{i=1}^{N}\lambda_i K(t, t_i) + \sum_{m=1}^{M}(\alpha_m - \beta_m)\phi_m(t) + \omega_0 \end{aligned}$$

where

$$K(t, t_i) = \sum_{m=1}^{M}\phi_m(t)\phi_m(t_i) = \phi(t)^T\phi(t_i) \tag{9}$$

is the kernel function. In the dual representation, there is an additional term which cannot be written in terms of the kernel function. That means the dual problem of LS-SVM with l_1-regularization provides a semiparametric model.

In the regression field, there are several kinds of nonparametric methods, including regression trees and kernel regression. Classification and Regression Trees (CART), proposed by [28], is a widely used method for regression. The corresponding result is piecewise constant and is not continuous. Thus, CART is not suitable to do segmentation for continuous signals. In kernel regression, the popular nonlinear kernels are the radial basis function (RBF) kernel, the polynomial kernel, and the hyperbolic tangent kernel. However, these kernels cannot provide a PWL function and hence are not applicable in segmentation problems. The kernel proposed in this paper, i.e., (9), is constructed from HH and one can verify that this kernel gives a continuous piecewise linear function. In a segmentation problem, we pursue a small number of segments and hence the lasso technique is applied in the primal formulation, which results in the semiparametric model (8).

According to the discussion above, we can get segmentation result $h_{\omega,S}(t)$ via solving the primal problem (7) or the dual

problem (8). Correspondingly, $h_{\omega,S}(t)$ can be represented as

$$h_{\omega,S}(t) = \begin{cases} \nearrow \omega_0 + \sum_{m=1}^{N} \omega_m \phi_m(t) & \text{[P]} \\ \searrow \sum_{i=1}^{N} \lambda_i K(t, t_i) + \sum_{m=1}^{M} (\alpha_m - \beta_m)\phi_m(t) + \omega_0 & \text{[D].} \end{cases}$$

The number of the variables in primal problem (7) is $N + 2M$. Since there are N equality constraints, the number of independent variables involved in (7) is $2M$. Comparatively, the number of independent variables in the dual problem (8) is $N + M - 1$. In segmentation problems, M is usually much smaller than N. Moreover, using the dual variables to represent the function, $N+2M+1$ variables, i.e., $\lambda_1, \ldots, \lambda_N$, $\alpha_1, \ldots, \alpha_M, \beta_1, \ldots, \beta_M, \omega_0$, should be saved and using primal representation, we need to remember only $M + 1$ variables $\omega_0, \ldots, \omega_M$. Therefore, in segmentation problems, we prefer to solve (6) from the primal space.

B. Segmentation Algorithm

In this section, we establish the algorithm for segmentation using HH. This algorithm consists of three parts:

1) initialization;
2) LS-SVM with lasso using hinging feature map;
3) segmentation points training.

The second and the third parts have been discussed previously, and the first part deals with the generation of initial segmentation points for (6) and the selection of parameters γ, μ.

To generate possible segmentation points, one can use interpolation-based segmentation algorithms, such as the sliding window algorithm described in [6] and FSW in [9]. These algorithms are sensitive to noise, and many points away from the real values will be picked out as the segmentation points, especially when the thresholds are set to be low. All the picked points can be used as the potential segmentation points, and then LS-SVM with lasso can be applied to detect segmentation points. In this paper, we use another simpler approach to find the possible segmentation points, motivated by the following observation: consider three successive sampling points $[t_{i-1}, y(t_{i-1})]^T, [t_i, y(t_i)]^T, [t_{i+1}, y(t_{i+1})]^T$. Then

$$\frac{y(t_i) - y(t_{i-1})}{t_i - t_{i-1}} - \frac{y(t_{i+1}) - y(t_i)}{t_{i+1} - t_i}$$

measures the difference of two slopes in the left and right part of t_i. Based on this fact, we calculate

$$d_i^{(2)} = (t_i - t_{i-1})y(t_{i+1}) - (t_{i+1} - t_{i-1})y(t_i) + (t_{i+1} - t_i)y(t_{i-1}). \quad (10)$$

It is not hard to verify that

$$\left|d_i^{(2)}\right| = (t_i - t_{i-1})(t_{i+1} - t_i) \times \left|\frac{y(t_i) - y(t_{i-1})}{t_i - t_{i-1}} - \frac{y(t_{i+1}) - y(t_i)}{t_{i+1} - t_i}\right|.$$

Algorithm 1 Segmentation Algorithm Using HH (SAHH)

(**Initialization**)
- Set M_0, R_μ, δ (the threshold for detecting nonzero components) and ε (the tolerance for training);
- Compute $d_i^{(2)} = (t_i - t_{i-1})y(t_{i+1}) - (t_{i-1} + t_{i+1})y(t_i) - (t_{i+1} - t_i)y(t_{i-1})$;
- Pick out M_0 points with maximal absolute value of $d_i^{(2)}$ as the segmentation points S^0;
- Carry out rid search and ten-fold cross validation for γ using LS-SVM (5);
- Set $\mu_1 = \frac{\gamma}{R_\mu}$, $\mu_m = \frac{\mu_1}{s_m - s_{m-1}}, m = 2, \ldots, M_0$;

(**LS-SVM with lasso using hinging feature map**)
- Solve (6), denote the results as $\omega^\star$;
- Set $\mathcal{M} = \{m : |\omega^\star| > \delta\}$ and $S^1 = S^0(\mathcal{M})$;

(**Segmentation points training**)

repeat
- Fix S^1 and solve LS-SVM (5). Denote the result as ω^1 and the objective value as e^1;
- Fix ω^1 and use modified Newton–Gaussian formulation (4) to do line search for $\frac{1}{2}\gamma e_{\text{sse}}(\omega, S) + \frac{1}{2}\sum_{m=1}^{M} \omega_m^2$, and denote the optimized result as S^2 and the value as e^2;

until $(e^1 - e^2)/e^1 < \varepsilon$;
- Algorithm ends and returns S^1 and ω^1.

For equidistant sampling problems, $\left|d_i^{(2)}\right|$ is proportional to the difference of two consecutive slopes. In nonequidistant sampling problems, $(t_i - t_{i-1})(t_{i+1} - t_i)$ measures the distance between t_i and the adjacent points. When t_i is far away from the adjacent points, it has a high probability to be a segmentation points. In this paper, we choose M_0 points with maximal absolute value of $d_i^{(2)}$ as the initial segmentation points and then use (6) to find the suitable points.

What remains is to determine the values of γ and μ, which balance the accuracy and sparseness. One way to tune γ is by using ten-fold cross validation and a grid search. It is time consuming since there are linear constraints in (6). To speed up the search process, we ignore the l_1-regularization term and consider the LS-SVM problem (5). Since (5) or its dual problem can be solved efficiently, grid search with ten-fold cross validation is applicable to find a proper value for γ. In this paper, we use LS-SVMLab v1.8 [29] to determine γ.

The parameter μ_m reflects the wish that we want ω_m to be zero, i.e., using a linear function to describe the signal around s_m. Generally, if there is another segmentation point near s_m, i.e., $s_m - s_{m-1}$ is small, it is possible that we do not use s_m as a segmentation point. According to this observation, we set

$$\mu_1 = \frac{\gamma}{R_\mu} \quad \text{and} \quad \mu_m = \frac{\mu_1}{s_m - s_{m-1}} \qquad \forall m \geq 2 \quad (11)$$

where $R_\mu \in \mathbb{R}^+$ is determined by the user. The discussion above is summarized in Algorithm 1, named segmentation algorithm using HH (SAHH).

In SAHH, there are some user-defined parameters. The meanings and the typical values of these parameters are

listed below. The sensitivity to these parameters will be evaluated numerically in Section V.

1) δ: the threshold for detecting nonzero components. In our algorithm, we set $\delta = 10^{-4}$.
2) ε: the tolerance for training segmentation points. We set it to be $\varepsilon = 10^{-4}$.
3) M_0: the number of initial segmentation points. In our algorithm, we set $M_0 = \lceil \frac{N}{4} \rceil$, where $\lceil \cdot \rceil$ maps a real number to the nearest integer greater than or equal to it.
4) γ: the cost of the sum of squared error. We apply grid search with ten-fold cross validation to find a suitable γ, which can be implemented by LS-SVMLab v1.8 [29].
5) R_μ: the tradeoff between accuracy and sparseness. As shown in (11), small R_μ corresponds to large μ_m in (6), which means we put more emphasis on sparseness. One can choose R_μ according to different requirements. The typical range of R_μ is between 0.01 and 10.

IV. Online Segmentation Method Using HH

SAHH can handle segmentation problems, and its performance is shown in Section V. In SAHH, all the data are used, and hence its computation time is similar to the ones using all the data together, such as the top-down algorithm and the bottom-up algorithm. The online method is required because, in some applications, the allowed computation time is very short or the data arrive successively. In order to establish the online algorithm, we learn from the idea of FSW algorithm in [9]. FSW calculates the maximum vertical distance between the newly arriving data and the currently active line. If the distance exceeds a threshold, denoted by $d_{\max}$, a new segmentation point is added and FSW applies linear interpolation to reconstruct the signal in the new segment.

Based on HH (2), a new online segmentation method is proposed. It uses the FSW framework for initially detecting segmentation points, then linear regression is applied instead of interpolation to determine the coefficients, and segmentation points are adjusted as well. Let us consider the case where $s_1, s_2, \ldots, s_{M-1}$ and $\omega_0, \omega_1, \ldots, \omega_{M-1}$ have been determined. Along with a new point arriving, the number of time points N increases, and the new data point is denoted by $[t_N, y(t_N)]^T$. Suppose a new segmentation point is needed, then we should consider how to calculate ω_M and adjust s_M.

In segment $[s_M, t_N]$, $h_{\omega,S}(t)$ equals an affine function with slope $\sum_{m=1}^{M} \omega_m$. Thus, the optimal slope can be obtained by keeping $\omega_1, \ldots, \omega_{M-1}$ unchanged and adjusting ω_M only. Suppose N_1 is the starting point of the current segment. Because $s_M \geq t_{N_1} > s_{M-1}$, the segmentation position s_M and the value of ω_M will not affect the approximation performance for $t \leq t_{N_1}$ since $\phi_M(t) = 0, \forall t \leq t_{N_1}$. Therefore, we can focus on the data between N_1 and N. For these data, when s_M is given, ω_M is computed by

$$\begin{aligned} \min_{\omega_M, e} \quad & \frac{1}{2}\omega_M^2 + \gamma \frac{1}{2} \sum_{i=N_1}^{N} e_i^2 \\ \text{s.t.} \quad & \tilde{y}(t_i) = e_i + \omega_M \phi_M(t_i), \quad i = N_1, \ldots, N \end{aligned} \tag{12}$$

where $\phi_M(t_i) = \max\{0, t_i - s_M\}$ and $\tilde{y}(t_i) = y(t_i) - \omega_0 - \sum_{m=1}^{M-1} \omega_m \phi_m(t_i)$. In (5), the regularization constant γ is obtained by grid search with ten-fold cross validation, but doing grid search on all data is not feasible for online algorithm. To get a reasonable value, we apply grid search for γ on a small part of data, e.g., the first 100 points, and use the result as the value of γ in (12). The noise level may change, hence one can modify γ online as well. For example, when $t = 200$, we can use the data points between $t = 100$ and $t = 200$ to do grid search and get a new γ. Modifying γ can improve the accuracy but takes more time. In this paper, we simply use the first 100 data points to determine γ and do not change it online.

Algorithm 2 Online Segmentation Algorithm Using HH (Online SAHH)

- Give $d_{\max}$ (error threshold), Δ (distance for the grid search);
- Let $i = M = \text{SID} = 1$, $l_{\text{up}} = \infty, l_{\text{low}} = -\infty$, $s_1 = 1$, $p_M = y(s_1), \omega_0 = y(t_1)$, where SID is a note of the present segment identifier;

repeat

- $i = i + 1$, $l_{\text{up}} = \min\{l_{\text{up}}, \frac{y(t_i)+d_{\max}-p_M}{t_i - s_M}\}$, $l_{\text{low}} = \max\{l_{\text{low}}, \frac{y(t_i)-d_{\max}-p_M}{t_i - s_M}\}$;

if $l_{\text{up}} < l_{\text{low}}$ **then**

- Set $N_1 = \text{SID} + 1$, $N = i$;
- Use (13) to calculate $\omega(s)$ and $\text{err}(s)$ for $s = t_{N_1}, t_{N_1} + \Delta, t_{N_1} + 2\Delta, \ldots, t_N$;
- $s_M = \arg\min \text{err}(s)$, $\omega_M = \omega(s_M)$;
- $i = \max\{\text{SID}, \lceil s_M \rceil\}$, $M = M + 1$;
- $s_M = t_i$, $p_M = y(s_M)$, $l_{\text{up}} = \infty, l_{\text{low}} = -\infty$;

else

if $l_{\text{low}} \leq \frac{y(t_i)-p_M}{t_i - s_M} \leq l_{\text{up}}$ **then**

- SID $= i$;

end

end

until $i > N$;

The optimal solution of (12) can be obtained by

$$\omega_M = \frac{\sum_{i=N_1}^{N} \tilde{y}(t_i)\phi_M(t_i)}{1/\gamma + \sum_{i=N_1}^{N} \phi_M(t_i)^2}. \tag{13}$$

Essentially, we are seeking the best basis function $\omega_M \max\{0, t - s_M\}$ to approach the residuals $\tilde{y}(t_i)$. The best ω_M for given s_M can be obtained by (13). And training s_M furthermore improves the accuracy. For adjusting s_M, one can use the formulation (4), but an additional constraint $s_M \geq t_{N_1}$ should be considered in order to avoid affecting segmentation points already determined. Because s_M is univariate and the solution of (12) can be obtained very efficiently for a given s_M, we use grid search for s_M on interval $[t_{N_1}, t_N]$, i.e., several $s \in [t_{N_1}, t_N]$ are generated. For each s value, we solve (12) and denote the optimal solution as $\omega(s)$ and the objective value as $\text{err}(s)$. Then $s_M = \arg\min \text{err}(s)$ is selected as the segmentation point. The summary of this discussion is given as Algorithm 2, named online SAHH. The segmentation detection part of online SAHH refers to [9].

The basic idea of online SAHH is using l_{up} and l_{low} to judge whether a new segment is needed for the new coming data. If $l_{\text{up}} < l_{\text{low}}$, backtrack procedure is used to find the

suitable segmentation point. Then we turn back to the newly found segmentation point or SID, i.e., the last point satisfying $l_{\text{low}} \leq (y(t_i) - p_M)/(t_i - s_M) \leq l_{\text{up}}$. In the backtrack procedure, the times of computing (13) is proportional to $(t_N - t_{N_1})/\Delta$. Hence the computation time is approximately inversely proportional to Δ. Typically, when the sampling interval is 1, $t_i - t_{i-1} = 1$, we set $\Delta = 0.1$ and we can change it according to computation time requirement. $d_{\max}$ defines the error threshold, and has effect on the length of segments. If a high accuracy is required, we should select a small $d_{\max}$. But when the data contain noise, we prefer a large value for $d_{\max}$, which leads to long segments and makes the result insensitive to noise.

Online SAHH is established based on the framework of FSW. For FSW, the computational complexity is $\mathcal{O}(MN)$, which has been given in [9]. In online SAHH, the segmentation points are determined by a backtracking procedure, for which the computation load is proportional to $(t_N - t_{N_1})/\Delta$. Hence, the computational complexity of online SAHH is $\mathcal{O}(MNL) = \mathcal{O}(N^2)$, where L stands for the average length of the segments.

V. Numerical Experiments

In this section, we apply SAHH and online SAHH to testing datasets and then compare them to the following segmentation algorithms: sliding window and bottom-up algorithm (SWAB [6]), FSW, stepwise FSW algorithm (SFSW [9]), l_1 trend filtering method [30], and SwiftReg [4]. SWAB uses less computation time than bottom-up algorithm and has comparable accuracy. FSW is an efficient sliding window algorithm, and SFSW uses backward search to improve the precision of FSW. l_1 trend filtering is designed for trend filtering but can be used for segmentation as well. SwiftReg is an efficient online method for piecewise polynomial approximation. When we set the maximal degree of the polynomial to be 1, SwiftReg provides a PWL signal and can be used for segmentation. Note that SAHH, SWAB, FSW, and SFSW all provide a continuous reconstructed signal but the result of SwiftReg is discontinuous. Though SwiftReg is not suitable to deal with continuous signals, in this section we can use it to evaluate the proposed methods. All the experiments are done in MATLAB R2011a in Core 2-2.83 GHz, 2.96G RAM.

To compare the performance of these algorithms, the number of segments and the approximation precision should be considered. The precision is measured by the relative sum of squared error (RSSE), defined as

$$\text{RSSE} = \frac{\sum_{x\in\mathcal{V}} \left(f(x) - \hat{f}(x)\right)^2}{\sum_{x\in\mathcal{V}} \left(f(x) - E_{\mathcal{V}}(f(x))\right)^2} \tag{14}$$

where $f(x)$ is the underlying function, $E_{\mathcal{V}}(f(x))$ is the average value of $f(x)$ on $\mathcal{V}$, and $\hat{f}(x)$ is the identified function. RSSE can be used to measure both the training error and the validation error. In segmentation problems, we are primarily interested in the error between the original and the reconstructed signal; therefore, we consider the approximation error in $\mathcal{V} = \{t_1, t_2, \ldots, t_N\}$. In the involved algorithms, there are tradeoff parameters for the approximation accuracy and the number of segments. We tune these parameters to make the number of segments in each algorithm similar and then compare the RSSEs.

TABLE I
Performance of Different M_0, δ, and ε

M_0	δ	ε	M	RSSE	Time (s)
500	10^{-4}	10^{-4}	25	0.012	23.34
500	10^{-6}	10^{-8}	26	0.013	23.56
250	10^{-4}	10^{-4}	18	0.019	7.43
250	10^{-6}	10^{-8}	19	0.018	8.94
100	10^{-4}	10^{-4}	15	0.027	3.92
100	10^{-6}	10^{-8}	14	0.028	6.94

First, global methods, including SWAB, l_1 trend filtering, and SAHH, are compared. SWAB is actually an online approach using only the data in a buffer. In this experiment, we set the size of the buffer large enough to contain all the data; then it can be regarded as a global method. In order to evaluate the performance of segmentation algorithms for noise-corrupted data, we consider the following three synthetic datasets, each of which contains 1000 time points.

Dataset 1: ([4]): $y(t) = \sin^2(t)$, $t \in [1, 20]$.

Dataset 2: ([4]): $y(t) = \sin(10\ln(t))$, $t \in [1, 20]$.

Dataset 3: Synthetic data provided in [30], $t \in [1, 1000]$.

Before giving comparison, we focus on Dataset 1 and discuss the typical values of the parameters for SAHH. As mentioned in Section III-B, there are five user-defined parameters: δ (the threshold for detecting nonzero components); ε (the tolerance for training segmentation points); γ (the cost of error); M_0 (the number of initial segmentation points); and R_μ (the tradeoff between accuracy and sparseness). Among them, γ is tuned by cross validation and R_μ is set according to different targets. For other parameters, a typical setting is $\delta = 10^{-4}$, $\varepsilon = 10^{-4}$, and $M_0 = \lceil \frac{N}{4} \rceil$. In the following, we consider several groups of values of δ, ε, and M_0 to evaluate the parameter sensitivity of SAHH. The numbers of segments, the RSSEs, and the computation time are reported in Table I, where the results are obtained with $\gamma = 10^6$ and $R_\mu = 1$.

From the results, one can see that the performance is not sensitive to the value of δ and ε, and we set them to be 10^{-4}. M_0 determines the initial number of segmentation points; in this paper we always select $M_0 = \lceil \frac{N}{4} \rceil$. Then the performance of different γ and R_μ values is considered in the following. We select several groups of γ and R_μ and report the RSSEs and computation time in Table II. γ is related to the emphasis put on accuracy on sampling data. Hence, the result corresponding to a large γ can fit the sampling data well but is sensitive to noise. To see this point, Gaussian noise following $\mathcal{N}(0, \sigma^2)$ is added, and one can see that the performance with $\gamma = 10^6$ is good for the noisefree case but the performance with $\gamma = 10^4$ is better when there are noise. To handle different cases, we apply grid search based on ten-fold cross validation to find a suitable γ.

The effect of tuning R_μ, which is set to make a tradeoff between accuracy and compression rate, can be seen in Table II. For users' convenience, in the following experiments, the values of R_μ are reported. Similar to R_μ in SAHH, in

TABLE II
PERFORMANCE OF DIFFERENT γ, R_μ WITH DIFFERENT NOISE LEVELS

σ	γ	R_μ	M	RSSE	Time (s)
0.0	10^6	1	18	0.019	7.43
	10^6	10	31	0.005	13.84
	10^4	1	23	0.016	10.10
	10^4	10	33	0.008	14.98
0.2	10^6	1	20	0.039	8.37
	10^6	10	35	0.023	11.24
	10^4	1	22	0.021	9.51
	10^4	10	24	0.023	9.36

TABLE III
PERFORMANCE OF GLOBAL SEGMENTATION ALGORITHMS ON SYNTHETIC DATASETS

		SWAB		l_1-TF		SAHH		
Data	σ	M	RSSE	M	RSSE	R_μ	M	RSSE
Dataset 1	0	25	0.023	17	0.084	1.0	19	0.013
	0.05	23	0.059	17	0.083	0.5	19	0.014
	0.1	19	0.179	21	0.107	0.5	19	0.023
	0.2	25	0.383	19	0.231	0.5	16	0.032
Dataset 2	0	19	0.014	23	0.037	1.0	19	0.019
	0.05	21	0.031	24	0.154	0.3	20	0.017
	0.1	23	0.031	22	0.159	0.3	18	0.019
	0.2	25	0.172	18	0.164	0.3	20	0.021
Dataset 3	0	7	0.008	17	0.004	1.0	14	0.003
	5	12	0.058	15	0.006	1.0	14	0.004
	10	19	0.175	16	0.009	1.0	12	0.011
	20	44	0.844	16	0.016	1.0	14	0.022

each of the considered algorithms, there is one user-defined parameter for the tradeoff between the accuracy and the number of segments. In order to have a fair comparison, we tune these parameters to have similar numbers of segments and then compare the accuracy. In this experiment, noise following $\mathcal{N}(0, \sigma^2)$ with different noise levels is added. The performance of SWAB, l_1 trend filtering, and SAHH is reported in Table III, from which one can see the numbers of segments and the corresponding RSSEs for different datasets and noise levels. Note that the output of l_1 trend filtering is the approximate value of each sampling point. Then the corresponding segmentation results can be obtained by calculating the second-order differences. In the same way of processing the result of (5) in SAHH, only the points with second-order difference larger than δ are regarded as segmentation points.

According to Table III, SWAB performs very well when there is no noise. However, with increase in noise, the performance of SWAB decreases, because SWAB is based on linear interpolation. In contrast, l_1 trend filtering and SAHH are less sensitive to noise. Usually, l_1 trend filtering needs more segments than SAHH to achieve the same accuracy. The obtained segmentation points may be centralized in l_1 trend filtering because the distances between the segmentation points are not considered. The sampling data, the results of SWAB, l_1 trend filtering, and SAHH for Dataset 3 with $\sigma = 10$ are illustrated in Fig. 4. In Fig. 4(c), it seems that

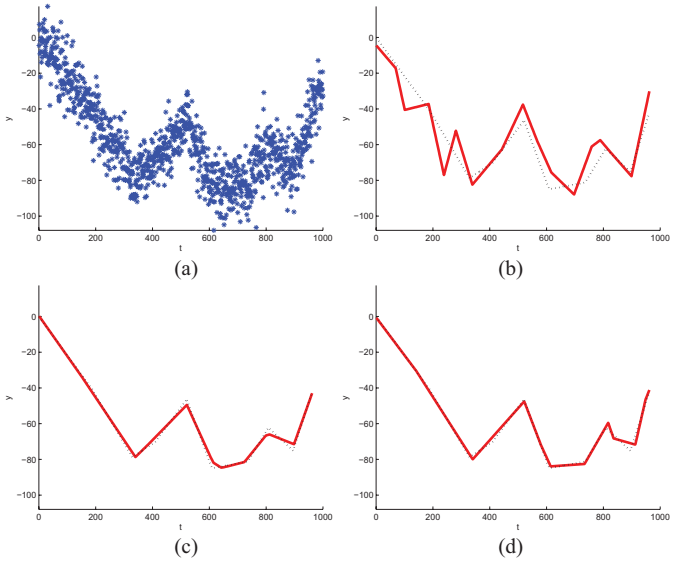


Fig. 4. Segmentation results of Dataset 3. (a) Sampling points. (b) SWAB with 18 segments (red solid line) and the signal (dashed line). (c) l_1 trend filtering with 15 segments (red solid line) and the signal (dashed line). (d) SAHH with 11 segments (red solid line) and the signal (dashed line).

only eight segments are found. However, there are several segmentation points located around $t = 610, 800$, and there are 16 segmentation points, i.e., 15 segments, in Fig. 4(c).

Next, we conduct experiments on the real-world datasets. From [31], datasetA, datasetB, and EDA_signal are downloaded. The three datasets have 2000, 2500, and 67225 sampling points. For datasetA and datasetB, we use all the data for segmentation. For EDA_signal, only the first 2000 points are used. The next dataset is S&P 500 index for 2000 trading days starting from March 25, 1999, which was used in [30] to evaluate the performance of l_1 trend filtering. In the experiments on synthetic data, the noise is added artificially and the real-world data contain some noise. We also investigate the performance for sparse sampling cases. For that, we use $t_1, t_{1+\text{Space}}, t_{1+2\text{Space}}, \ldots$ to do segmentation and use all the data to measure the accuracy, where Space stands for the distance between the two adjacent time points. We also consider different numbers of segments and the corresponding accuracy. The performance of SWAB, l_1 trend filtering, and SAHH is reported in Table IV, from which one can see the numbers of segments and the corresponding RSSEs for different datasets and different Space values.

The results in Table IV show the advantages of SAHH in segmentation problems. We illustrate the segmentation results visually: the results for datasetB with Space $= 1$ are given in Fig. 5, where the sampling points and the segmentation results of SWAB, l_1 trend filtering, and SAHH are shown, respectively.

Though SAHH is established for continuous signals, it can also be used for discontinuous signals. The testing dataset is downloaded from [32], which was used to evaluate a segmentation algorithm based on grid search; for more details see [33]. We use the population involved in the surveys in different years as the time series. Putting these populations

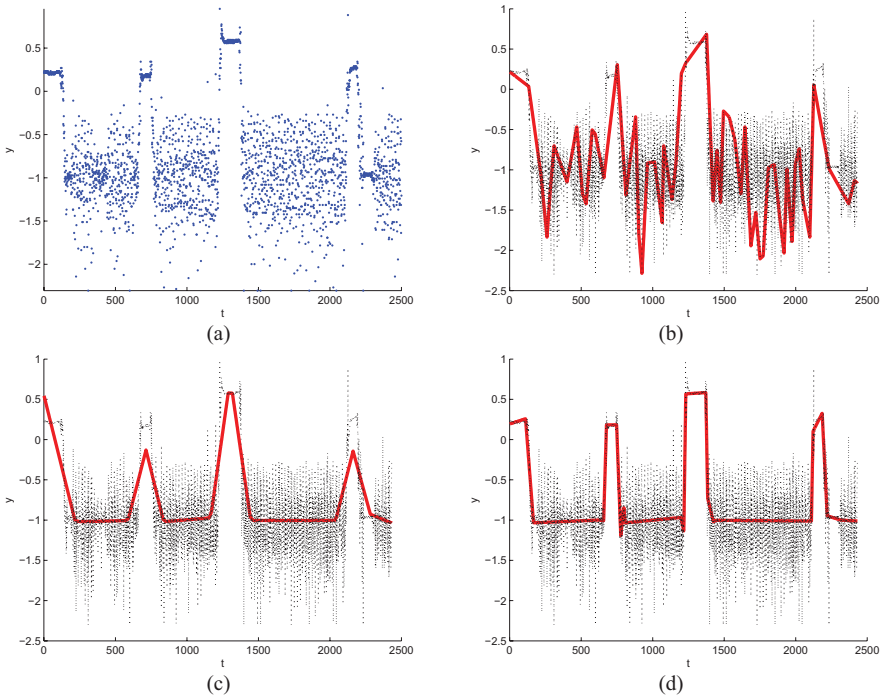


Fig. 5. Segmentation results for datasetB. (a) Sampling points. (b) SWAB with 55 segments (red solid line) and the sampled signal (dashed line). (c) l_1 trend filtering with 22 segments (red solid line) and the sampled signal (dashed line). (d) SAHH with 23 segments (red solid line) and the sampled signal (dashed line).

TABLE IV
PERFORMANCE OF GLOBAL SEGMENTATION ALGORITHMS

		SWAB		l_1-TF		SAHH		
Data	Space	M	RSSE	M	RSSE	R_μ	M	RSSE
datasetA	1	9	0.498	9	0.375	0.01	6	0.119
	1	14	0.138	16	0.108	0.5	14	0.053
	5	7	0.521	8	0.235	0.5	6	0.195
	5	23	0.301	20	0.164	10	20	0.081
datasetB	1	66	0.773	12	0.616	0.05	10	0.563
	1	97	0.719	24	0.363	1.0	23	0.297
	5	17	0.938	15	0.612	3.0	16	0.472
	5	29	0.872	30	0.355	10	28	0.331
EDA_signal	1	16	0.018	17	0.060	0.01	18	0.010
	1	37	0.007	55	0.004	0.5	46	0.003
	5	20	0.022	24	0.070	20	23	0.008
	5	44	0.004	77	0.005	100	43	0.004
S&P 500	1	9	0.076	6	0.080	0.01	4	0.070
	1	22	0.070	16	0.043	0.5	14	0.040
	5	5	0.123	6	0.072	0.1	7	0.069
	5	30	0.045	21	0.046	10	20	0.030

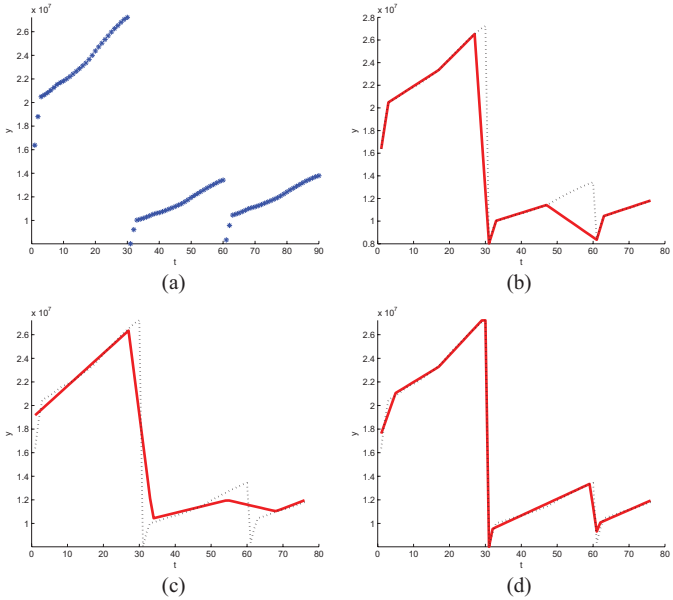


Fig. 6. Example of a discontinuous signal. (a) Sampling points. (b) SWAB with nine segments (red solid line) and the signal (dashed line). (c) l_1 trend filtering with ten segments (red solid line) and the signal (dashed line). (d) SAHH with nine segments (red solid line) and the signal (dashed line).

together gives a discontinuous signal on which we evaluate the performance of the segmentation algorithms. The results of segmentation are shown in Fig. 6, from which one can see that SAHH can handle the discontinuous signals well.

Finally, we evaluate the performance of online SAHH. Except for EDA_signal which was introduced earlier, we created two weather time series from freely available data [34]. The wind speed and the temperature datasets are univariate time series created by querying the Zagreb Airport weather station for data from January 1, 2009, to November 8, 2011. Load1, Load2, and Load3 are the hourly aggregated electric loads, which can be freely downloaded at [35]. The number of sampling data for these datasets are listed in Table V.

The methods involved in this experiment include FSW, SFSW, and SwiftReg. SWAB with a small buffer can serve as an online method as well. As reported in [4], SwiftReg can

TABLE V
PERFORMANCE OF ONLINE SEGMENTATION ALGORITHMS

		FSW				SFSW				SwiftReg			
Data	Size	$d_{\max}$	Time	M	RSSE	$d_{\max}$	Time	M	RSSE	$d_{\max}$	Time	M	RSSE
Wind speed	57 713	6	31.6	3028	0.1400	6	1586	2807	0.1053	12	2508	3484	0.0917
Temperature	57 713	5	19.2	984	0.1125	5	1465	952	0.0724	9	2528	1412	0.0745
Load1	33 600	200	29.9	5876	0.0752	200	2067	5718	0.0571	400	1438	4536	0.0544
Load2	24 960	1000	15.2	2273	0.1368	1000	913.0	2224	0.0912	2000	1078	1840	0.1086
Load3	9504	1500	9.20	1548	0.0104	1500	257.1	1515	0.0078	1500	410.1	1432	0.0136
EDA_signal	67 225	10	118	312	0.0270	10	1973	305	0.0077	30	3355	335	0.0122
		Online SAHH ($\Delta = 0.05$)				Online SAHH ($\Delta = 0.1$)				Online SAHH ($\Delta = 0.5$)			
Data	Size	$d_{\max}$	Time	M	RSSE	$d_{\max}$	Time	M	RSSE	$d_{\max}$	Time	M	RSSE
Wind speed	57 713	6	1146	3356	0.0326	6	715.2	3303	0.0316	6	428.0	3367	0.1188
Temperature	57 713	5	607.9	1287	0.0674	5	423.7	1293	0.0669	5	299.7	1292	0.0672
Load1	33 600	200	1675	5859	0.0269	200	1001	5816	0.0273	200	467.2	5870	0.0291
Load2	24 960	1000	665.8	2054	0.0604	1000	437.0	2336	0.0644	1000	207.7	2049	0.1064
Load3	9504	1500	121.5	1577	0.0081	1500	273.2	1580	0.0054	1500	456.7	1587	0.0054
!EDA_signal	67 225	10	411.3	324	0.0023	10	331.3	325	0.0023	10	252.1	331	0.0028

provide a result with similar accuracy but the computation time is significantly smaller than SWAB. The result of SwiftReg is not continuous, but in the pure view of approximation, it has some advantages over SWAB. Hence, in this experiment, we consider SwiftReg (with maximum degree 1 and the criterion of deviation of the predicted value to the measured value) instead of SWAB. For online approaches, we are interested in computation time. As analyzed previously, the computation time of the online SAHH is approximately inversely proportional to Δ, but a small Δ results in better accuracy. A typical setting is $\Delta = 0.1$, and in this experiment the performance for $\Delta = 0.05, 0.1, 0.5$ is evaluated. Besides Δ, there is another user-defined parameter $d_{\max}$, which is also used in other three algorithms to define the error threshold for each segment. We tune $d_{\max}$ for each algorithm to get the results with similar numbers of segments. Then we compare the RSSEs and the computation time (in milliseconds) in Table V, where the corresponding $d_{\max}$ values are reported as well.

From Table V, one can see that FSW is faster than the other methods but its accuracy is lower. In fact, the basic segmentation parts of the FSW, SFSW, and online SAHH are similar, and one can regard SFSW and online SAHH as the extensions to FSW. Compared to FSW, SFSW generally uses less segments and returns similar or better results, but without significant difference. Comparatively, the computation time of online SAHH is better than that of SFSW and its accuracy is significantly higher. One attractiveness of SwiftSeg is that it offers a low computational complexity $\mathcal{O}(N)$. Basically, SwiftSeg traverses each time and uses updating formulation to solve a least squares problem for each point. By contrast, in online SAHH, only a very simple calculation for $l_{\text{up}}, l_{\text{low}}$ is needed for each point. The problem of online SAHH is that the backtrack procedure may turn back to a very early point, which makes online SAHH to have a higher complexity $\mathcal{O}(N^2)$. In practice, that extreme case is rare. Therefore, though there is a risk that the online SAHH needs more time than SwiftSeg, in most applications the computation time of the online SAHH is less than that in SwiftSeg, as reported in Table V.

VI. CONCLUSION

Representing segmentation problems by HH is advantageous compared to interpolation-based methods for three reasons. First, instead of interpolation, which is very sensitive to noise, regression can be utilized. Second, advanced data mining techniques are applicable. Third, the segmentation points can be tuned according to the derivative information. Based on these advantages, we establish an LS-SVM, which takes HH as the feature map, with lasso for segmentation problems (SAHH), as well as an online version of that segmentation algorithm (online SAHH). SAHH has a better accuracy, and returns a comparable number of segments using the similar compression rate as SWAB and l_1 trend filtering. Online SAHH has much higher runtime than FSW, but lower runtime than SFSW, which, like online SAHH, can be considered as an extension to the FSW approach. In terms of RSSE, SAHH has the best accuracy, which makes it a viable choice for the time-series segmentation applications in which there is no strong emphasis on runtime compared to accuracy; e.g., for segmentation of 10 000 data points, the allowed runtime is several seconds or more.

The increase of the amount of data in real-time systems asks for algorithms that increase efficiency in data management without high loss of information. We believe that for real-time systems in which segmentation is the underlying important optimization tasks, such as smart grid and surveillance systems, SAHH is a good option for the time-series segmentation.

One possible direction for further study is using the segmentation results for forecasting. However, the segmentation

results cannot be directly extended to do the forecasting because the reconstructed signal, which is PWL, becomes a linear function outside the domain of interest and is hence not suitable to forecast a nonlinear signal. Instead of using the obtained signal, we can analyze the segmentation points and find some useful information. For example, one can use the period information of the segmentation points and forecast the future segmentation points. An interesting attempt was given in [36], where the authors first do segmentation and then do forecasting using the knowledge captured from the segmentation points. Since there may be other factors besides time in a signal, multivariate approach can be considered. Extending the result of this paper to multivariate time series is one of the promising research directions, see [37], [38] for some works. HH takes the form of $\max\{0, l_m(x)\}$. In this paper, we restricted $l_m(x)$ to be a 1-D function and segmentation algorithms for univariate time series were obtained. By extending $l_m(x)$ to high-dimensional space, we can describe the segmentation boundary by $l_m(x) = 0$ and new segmentation methods for multivariate problems can be expected.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for insightful comments.

REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Proc. Int. Conf. Found. Data Org. Algorithms*, vol. 730. 1993, pp. 69–84.

[2] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowl. Inf. Syst.*, vol. 3, no. 3, pp. 263–286, 2001.

[3] K. Chan and A. Fu, "Efficient time series matching by wavelets," in *Proc. 15th Int. Conf. Data Eng.*, 1999, pp. 126–133.

[4] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick, "Online segmentation of time series based on polynomial least-squares approximations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2232–2245, Dec. 2010.

[5] H. Shatkay and S. Zdonik, "Approximate queries and representations for large data sequences," in *Proc. 12th Int. Conf. Data Eng.*, 1996, pp. 536–545.

[6] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proc. IEEE Int. Conf. Data Mining*, Nov.–Dec. 2001, pp. 289–296.

[7] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting time series: A survey and novel approach," in *Data Mining in Time Series Databases*, M. Last, A. Kandel and H. Bunke, Eds. Singapore: World Scientific, 2004, pp. 1–23.

[8] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, and W. Truppel, "Online amnesic approximation of streaming time series," in *Proc. 20th Int. Conf. Data Eng.*, 2004, pp. 339–349.

[9] X. Liu, Z. Lin, and H. Wang, "Novel online methods for time series segmentation," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 12, pp. 1616–1626, Dec. 2008.

[10] V. Tseng, C. Chen, P. Huang, and T. Hong, "Cluster-based genetic segmentation of time series with DWT," *Pattern Recognit. Lett.*, vol. 30, no. 13, pp. 1190–1197, 2009.

[11] J. Guerrero, J. Garca, and J. Molina, "Piecewise linear representation segmentation in noisy domains with large number of measurements: The air traffic control domain," *Int. J. Artif. Intell. Tools*, vol. 20, no. 2, pp. 367–399, 2011.

[12] E. Keogh and M. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback," in *Proc. 4th Int. Conf. Knowl. Discovery Data Mining*, 1998, pp. 239–278.

[13] G. Bryant and S. Duncan, "A solution to the segmentation problem based on dynamic programming," in *Proc. 3rd IEEE Conf. Control Appl.*, Aug. 1994, pp. 1391–1396.

[14] L. Chua and S. Kang, "Section-wise piecewise-linear functions: Canonical representation, properties, and applications," *Proc. IEEE*, vol. 65, no. 6, pp. 915–929, Jun. 1977.

[15] L. Breiman, "Hinging hyperplanes for regression, classification and function approximation," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 999–1013, May 1993.

[16] J. Lin, H. Xu, and R. Unbehauen, "A generalization of canonical piecewise-linear functions," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 41, no. 4, pp. 345–347, Apr. 1994.

[17] P. Julián, A. Desages, and O. Agamennoni, "High-level canonical piecewise linear representation using a simplicial partition," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 46, no. 4, pp. 463–480, Apr. 1999.

[18] S. Wang and X. Sun, "Generalization of hinging hyperplanes," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4425–4431, Dec. 2005.

[19] S. Wang, X. Huang, and K. M. Junaid, "Configuration of continuous piecewise-linear neural networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 8, pp. 1431–1445, Aug. 2008.

[20] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[21] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal Stat. Soc., Ser. B Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.

[22] P. Pucar and J. Sjöberg, "On the hinge-finding algorithm for hinging hyperplanes," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 3310–3319, May 1998.

[23] V. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.

[24] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.

[25] L. Duan, D. Xu, and I.W. Tsang, "Domain adaptation from multiple sources: A domain-dependent regularization approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 3, pp. 504–518, Mar. 2012.

[26] S. Mehrkanoon, T. Falck, and J. A. K. Suykens, "Approximate solutions to ordinary differential equations using least squares support vector machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 9, pp. 1356–1367, Sep. 2012.

[27] A. Miranian and M. Abdollahzade, "Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 207–218, Feb. 2013.

[28] L. Breiman, J. Friedman, C. J. Stone, and R. A. Ohlshen, *Classification and Regression Trees*. London, U.K.: Chapman & Hall, 1984.

[29] K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle, and J. A. K. Suykens, "LS-SVMlab toolbox user's guide version 1.8," ESAT-SISTA, K. U. Leuven, Leuven, Belgium, Internal Rep. 10-146, 2010.

[30] S. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "l_1 trend filtering," *SIAM Rev.*, vol. 51, no. 2, pp. 339–359, 2009.

[31] G. Troester. (2011). *Dynamic Time Warping* [Online]. Available: http://www.ife.ee.ethz.ch/education/WS1_HS2011_ex04.zip

[32] National Cancer Institute. (2009). *Joinpoint Regression Program, Version 3.5.3* [Online]. Available: http://srab.cancer.gov/joinpoint

[33] H. Kim, M. Fay, E. Feuer, and D. Midthune, "Permutation tests for joinpoint regression with applications to cancer rates," *Stat. Med.*, vol. 19, no. 3, pp. 335–351, 2000.

[34] *Weather Underground*. (2007). [Online]. Available: http://www.wunderground.com

[35] *ENTSO-E*. (2008) [Online]. Available: http://www.entsoe.net

[36] J. L. Wu and P. C. Chang, "A trend-based segmentation method and the support vector regression for financial time series forecasting," *Math. Problems Eng.*, vol. 2012, p. 615152, Mar. 2012.

[37] N. Dobigeon, J. Y. Tourneret, and J. D. Scargle, "Joint segmentation of multivariate astronomical time series: Bayesian sampling with a hierarchical model," *IEEE Trans. Signal Process.*, vol. 55, no. 2, pp. 414–423, Feb. 2007.

[38] D. A. J. Blythe, P. von Bünau, F. C. Meinecke, and K. R. Müller, "Feature extraction for change-point detection using stationary subspace analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 631–643, Apr. 2012.



Xiaolin Huang (S'10–M'12) received the B.S. degree in control science and engineering and the B.S. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, China, in 2006, and the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China, in 2012.

He has been a Postdoctoral Researcher with ESAT-SCD-SISTA, KU Leuven, Leuven, Belgium, since 2012. His current research interests include optimization, classification, and identification for nonlinear systems via piecewise linear analysis.



Marin Matijaš was born in Split, Croatia, on December 18, 1984. He received the M.Eng. degree in power systems from the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia, in 2008, where he recently submitted his Ph.D. thesis for grading.

He was a Visiting Researcher under supervision of Prof. Johan A. K. Suykens at ESAT-SCD-SISTA, KU Leuven, Leuven, Belguim, from 2011 to 2012. He is currently developing and maintaining pricing and forecasting models at an electricity supplier HEP Opskrba d.o.o. His current research interests include machine learning, organized markets, and software agents design.



Johan A. K. Suykens (SM'05) was born in Willebroek Belgium, on May 18, 1966. He received the M.S. degree in electro-mechanical engineering and the Ph.D. degree in applied sciences from Katholieke Universiteit Leuven (KU Leuven), Belgium, in 1989 and 1995, respectively.

He was a Visiting Postdoctoral Researcher with the University of California, Berkeley, CA, USA, in 1996. He has been a Postdoctoral Researcher with the Fund for Scientific Research FWO Flanders and is currently a Professor (Hoogleraar) at KU Leuven. He is author of the books *Artificial Neural Networks for Modelling and Control of Non-Linear Systems* (Kluwer Academic Publishers) and *Least Squares Support Vector Machines* (World Scientific), co-author of the book *Cellular Neural Networks, Multi-Scroll Chaos and Synchronization* (World Scientific) and editor of the books *Nonlinear Modeling: Advanced Black-Box Techniques* (Kluwer Academic Publishers) and *Advances in Learning Theory: Methods, Models and Applications* (IOS Press). In 1998, he organized an International Workshop on Nonlinear Modeling with Time-Series Prediction Competition.

Dr. Suykens has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS (1997–1999 and 2004–2007) and for the IEEE TRANSACTIONS ON NEURAL NETWORKS (1998–2009). He was the recipient of the IEEE Signal Processing Society 1999 Best Paper (Senior) Award and several Best Paper Awards at International Conferences. He is the recipient of the International Neural Networks Society INNS 2000 Young Investigator Award for significant contributions in neural networks. He has served as a Director and Organizer of the NATO Advanced Study Institute on Learning Theory and Practice (Leuven 2002), as a program Co-Chair for the International Joint Conference on Neural Networks in 2004 and the International Symposium on Nonlinear Theory and its Applications in 2005, as an organizer of the International Symposium on Synchronization in Complex Networks in 2007, and a co-organizer of the NIPS 2010 Workshop on Tensors, Kernels and Machine Learning. He was awarded the ERC Advanced Grant in 2011.