Neurocomputing 117 (2013) 118–127

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

Support vector machines with piecewise linear feature mapping ☆

Xiaolin Huang *, Siamak Mehrkanoon, Johan A.K. Suykens

KU Leuven, Department of Electrical Engineering, ESAT-SCD-SISTA, B-3001 Leuven, Belgium

ARTICLE INFO

Article history:
Received 17 August 2012
Received in revised form
16 November 2012
Accepted 22 January 2013
Communicated by D. Zhang
Available online 27 February 2013

Keywords:
Support vector machine
Piecewise linear classifier
Nonlinear feature mapping

ABSTRACT

As the simplest extension to linear classifiers, piecewise linear (PWL) classifiers have attracted a lot of attention, because of their simplicity and classification capability. In this paper, a PWL feature mapping is introduced by investigating the property of the PWL classification boundary. Then support vector machines (SVM) with PWL feature mappings are proposed, called PWL-SVMs. In this paper, it is shown that some widely used PWL classifiers, such as k-nearest-neighbor, adaptive boosting of linear classifier and intersection kernel support vector machine, can be represented by the proposed feature mapping. That means the proposed PWL-SVMs at least can achieve the performance of the above PWL classifiers. Moreover, PWL-SVMs enjoy good properties of SVM and the performance on numerical experiments illustrates the effectiveness. Then some extensions are discussed and the application of PWL-SVMs can be expected.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Piecewise linear (PWL) classifiers are a kind of classification method which provide PWL boundaries. In some point of view, PWL boundaries are the simplest extension to linear boundaries. On one hand, PWL classifiers enjoy simplicity in processing and storing [1]. On the other hand, the classification capability of PWL classifiers can be expected, since arbitrary nonlinear boundary always can be approximated by a PWL boundary. Therefore, PWL classification methods are needed for small reconnaissance robots, intelligent cameras, embedded and real-time systems [1,2].

As one of the simplest and widely used classifiers, k-nearest-neighbor algorithm (kNN [3]) can be regarded as a PWL classifier [4]. Also, adaptive boosting (Adaboost [5]) provides a PWL classification boundary when one uses linear classifiers as the weak classifiers. Besides, there have been some other PWL classifiers proposed in [6–8]. One way to get PWL boundaries is first to do nonlinear classification and then to approach the obtained boundary by PWL functions, which was studied in [9]. However, nonlinear classification and PWL approximation themselves are complicated. Another way is using integer variables to describe which piece a point belongs to and establish an optimization problem for constructing a PWL classifier. The resulting classifier can be very flexible, but the corresponding optimization problem is hard to solve and the number of pieces is limited, see [8,10,11] for details. One major property of a PWL classifier is that in each piece, the classification boundary is linear. Therefore, one can locally train linear classifiers and get a PWL one. Following this way, [6,12,13] proposed some methods to construct PWL classifiers. The obtained classifiers demonstrate some effectiveness in numerical examples. However, these methods have some crucial limitations. For example, the method of [13] can only deal with separable data sets.

Support vector machine (SVM) proposed in [14] by Vapnik along with other researchers has shown a great performance in classification. In this paper, we introduce PWL feature mappings and establish PWL-SVMs, which provide a PWL boundary with maximum margin between two classes. This method enjoys the advantages of SVMs, such as good generalization and a good theoretical foundation. The proposed PWL feature mapping is constructed from investigating the property of piecewise linear sets. The specific formulation is motivated by the compact representation of PWL function, which was first proposed in [15] and then extended in [16–19].

The rest of this paper is organized as follows: in Section 2, SVMs with piecewise linear feature mappings, i.e., PWL-SVMs, are

☆This work was supported in part by the scholarship of the Flemish Government; Research Council KUL: GOA/11/05 Ambiorics, GOA/10/09 MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC), G.0377.12 (Structured models), IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare; Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007–2011); IBBT; EU: ERNSI; ERC AdG A-DATADRIVE-B, FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940); Contract Research: AMINAL; Other: Helmholtz: viCERP, ACCM, Bauknecht, Hoerbiger. Johan Suykens is a professor at KU Leuven, Belgium.

* Corresponding author. Tel.: +86 10 62785047; fax: +86 10 62786911.
E-mail addresses: huangxl06@mails.tsinghua.edu.cn (X. Huang), siamak.mehrkanoon@esat.kuleuven.be (S. Mehrkanoon), johan.suykens@esat.kuleuven.be (J.A.K. Suykens).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.01.023

proposed. Section 3 discusses the relation between the proposed methods and other PWL classifiers. Then PWL-SVMs are evaluated by numerical experiments in Section 4. Some extensions are studied in Section 5. Section 6 ends the paper with conclusions.

2. SVM with piecewise linear feature mapping

2.1. PWL boundary and PWL equation

In a two-class classification problem, the domain is typically partitioned into two parts by a boundary, e.g., a hyperplane obtained by linear classification methods, according to the input data $x \in \mathbf{R}^n$ and the corresponding label $y \in \{+1,-1\}$. Linear classifiers have been studied for many years, however, their classification capability is too limited and nonlinear classifiers are required. In this paper, the term of classification capability of one classifier means the flexibility of the possible types and shapes of the classification boundary. For example, a linear classifier can only provide a linear boundary, which is an affine set, i.e., a hyperplane, and hence its classification capability is not enough for many applications. In some point of view, the simplest extension to an affine set is a piecewise linear one, which provides a PWL boundary. As the name suggests, a PWL set equals an affine set in each of the subregions of the domain, and the subregions partition the domain.

Consider the two moons data set shown in Fig. 1(a), where points in two classes are marked by green stars and red crosses, respectively. The two classes cannot be separated by a linear boundary and we can use a PWL boundary, shown by black lines, to classify the two sets very well. This PWL set, denoted by $\mathcal{B}$, consists of three segments. Each segment can be defined as a line restricted to a subregion. For example, we can partition the domain into $\Omega_1 = \{x : 0 \le x(2) \le \frac{1}{3}\}, \Omega_2 = \{x : \frac{1}{3} \le x(2) \le \frac{2}{3}\}$, $\Omega_3 = \{x : \frac{2}{3} \le x(2) \le 1\}$, where $x(i)$ stands for the ith component of x. Then $\mathcal{B}$ can be defined as

$$\mathcal{B} = \bigcup_{k=1}^{3} \{x : c_k^T x + d_k = 0, x \in \Omega_k\},$$

where $c_k \in \mathbf{R}^2$, $d_k \in \mathbf{R}$ define the line in each subregion, as shown in Fig. 1(b).

For the convenience of expression, the definition of PWL set is given below.

Definition 1. If a set $\mathcal{B}$ defined in the domain $\Omega \subseteq \mathbf{R}^n$ meets the following two conditions:

(i) The domain Ω can be partitioned into finite polyhedra $\mathring{\Omega}_1, \mathring{\Omega}_2, \ldots, \mathring{\Omega}_K$, i.e., $\Omega = \bigcup_{k=1}^{K} \Omega_k$ and $\mathring{\Omega}_k \cap \mathring{\Omega}_l = \emptyset$, $\forall k \ne l$, where $\mathring{\Omega}_k$ stands for the interiors of Ω_k.
(ii) In each of the subregions, $\mathcal{B}$ equals a linear set, i.e., for each k, there exist $c_k \in \mathbf{R}^n, d_k \in \mathbf{R}$ such that $\mathcal{B} \cap \Omega_k = \{x : c_k^T x + d_k = 0\}$.

Then, $\mathcal{B}$ is a piecewise linear set.

An affine set provides a linear classifier, which can be written as the solution of a linear equation $f(x) = 0$. Hence, in a linear classification problem, one typically pursues a linear function $f(x)$ to classify a point by the sign of the functional value, i.e., $\mathrm{sign}\{f(x)\}$. Similarly, a PWL set provides a PWL boundary and it can be represented as the solution set of a PWL equation, guaranteed by the following theorem:

Theorem 1. *Any piecewise linear set $\mathcal{B}$ can be represented as the solution of a piecewise linear equation.*

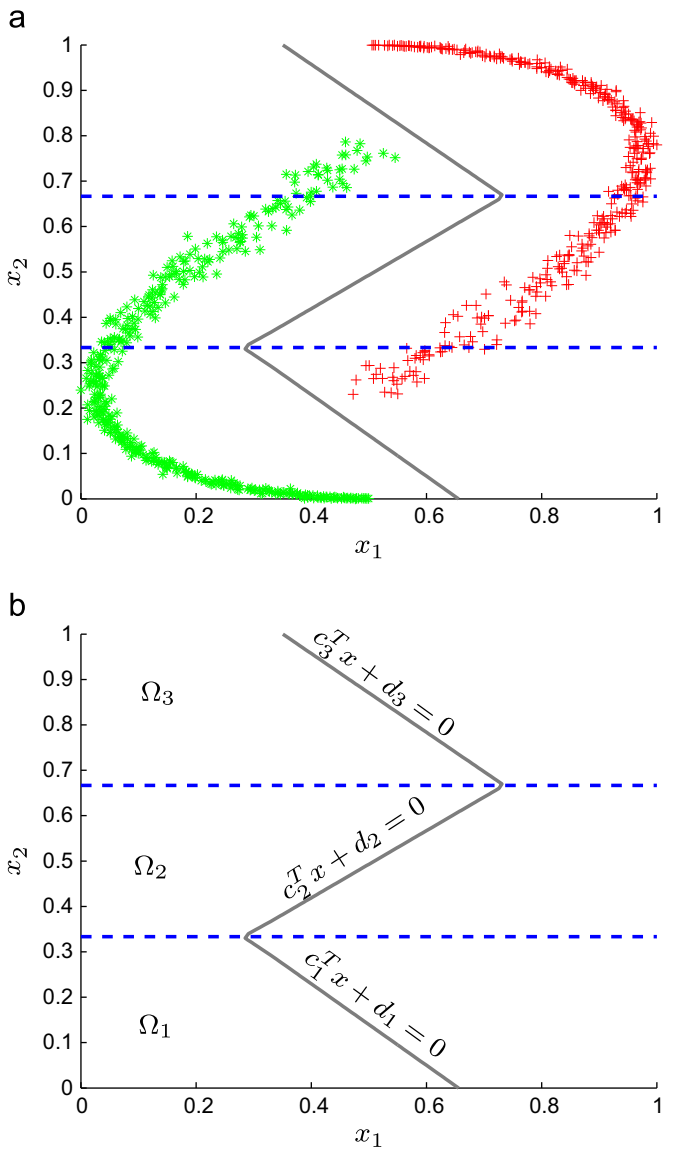


Fig. 1. An example of a piecewise linear boundary. (a) Points in two classes are marked by green stars and red crosses, respectively; the classification boundary is shown by black lines and (b) restricted to each of the subregions, the PWL boundary corresponds to a line. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Proof. Following the notations in Definition 1, we suppose the number of polyhedra defining $\mathcal{B}$ is K and these polyhedra are represented as $\Omega_k = \{x : a_{ki}^T x + b_{ki} \le 0, \forall 1 \le i \le I_k\}$, where I_k is the number of linear inequalities determining Ω_k. Then we construct the following function:

$$f(x) = \min_{1 \le k \le K} \left\{ \max \left\{ |c_k^T x + d_k|, \max_{1 \le i \le I_k} \{a_{ki}^T x + b_{ki}\} \right\} \right\}. \quad (1)$$

Since the max and the absolute function are both continuous and piecewise linear, one can verify that (1) is a continuous PWL function and in the following, we prove that $\mathcal{B} = \{x : f(x) = 0\}$.

First, we show that $\mathcal{B} \subseteq \{x : f(x) = 0\}$. For this, consider an arbitrary point in $\mathcal{B}$, denoted by x_0. According to the definition of piecewise linear set, we can find an index k_0 such that $x_0 \in \Omega_{k_0}$ and $c_{k_0}^T x_0 + d_{k_0} = 0$. Then, we have

$$\max_{1 \le i \le I_{k_0}} \{a_{k_0 i}^T x_0 + b_{k_0 i}\} \le 0,$$

therefore

$$\max\left\{\left|c_{k_0}^T x_0+d_{k_0}\right|, \max_{1\le i\le I_{k_0}}\{a_{k_0 i}^T x_0+b_{k_0 i}\}\right\}=0,$$

where I_{k_0} is the number of constraints defining polyhedra Ω_{k_0}. Because $\Omega_1,\Omega_2,\ldots,\Omega_K$ compose a partition of the domain, we know that $\Omega_k\cap\Omega_l=\emptyset$, $\forall k\neq l$. Therefore, $x_0\notin\Omega_k$, $\forall k\neq k_0$, i.e.

$$\max_{1\le i\le I_k}\{a_{ki}^T x_0+b_{ki}\}\ge 0,\quad \forall k\neq k_0,$$

which follows that

$$\max\left\{\left|c_k^T x_0+d_k\right|, \max_{1\le i\le I_k}\{a_{ki}^T x_0+b_{ki}\}\right\}$$
$$\ge \max\left\{\left|c_{k_0}^T x_0+d_{k_0}\right|, \max_{1\le i\le I_{k_0}}\{a_{k_0 i}^T x_0+b_{k_0 i}\}\right\}.$$

Accordingly

$$f(x_0)=\max\left\{\left|c_{k_0}^T x_0+d_{k_0}\right|, \max_{1\le i\le I_{k_0}}\{a_{k_0 i}^T x_0+b_{k_0 i}\}\right\}=0$$

and hence $\mathcal{B}\subseteq\{x: f(x)=0\}$.

Next we prove that $\{x: f(x)=0\}\subseteq\mathcal{B}$. Suppose $x_0\in\{x: f(x)=0\}$, i.e., $f(x_0)=0$. Then there exists at least one element $k_0\in\{1,2,\ldots,K\}$ such that

$$\max\left\{\left|c_{k_0}^T x_0+d_{k_0}\right|, \max_{1\le i\le I_{k_0}}\{a_{k_0 i}^T x_0+b_{k_0 i}\}\right\}=0.$$

Hence, $c_{k_0}^T x_0+d_{k_0}=0$ and $a_{k_0 i}^T x_0+b_{k_0 i}\le 0$, $\forall 1\le i\le I_{k_0}$, i.e., $x_0\in\Omega_{k_0}$. From this fact, we can conclude that $x_0\in\mathcal{B}\cap\Omega_{k_0}\subseteq\mathcal{B}$ and then $\{x: f(x)=0\}\subseteq\mathcal{B}$.

Summarizing the above discussions, we know $\mathcal{B}=\{x: f(x)=0\}$, i.e., any piecewise linear set can be represented as the solution set of a continuous PWL equation. □

According to the identities $|t|=\max\{t,-t\}$, $\forall t\in\mathbf{R}$ and $\max\{t_1,\max\{t_2,t_3\}\}=\max\{t_1,t_2,t_3\}$, $\forall t_1,t_2,t_3\in\mathbf{R}$, we rewrite (1) as the following min–max formulation:

$$\min_{1\le k\le K}\{\max\{c_k^T x+d_k,-c_k^T x-d_k,\quad a_{k1}^T x+b_{k1},\ldots,a_{kI_k}^T x+b_{kI_k}\}\}.$$

Using a min–max formulation, a PWL classifier can be constructed. The problem of determining the parameters can be posed as a nonconvex and nondifferentiable optimization problem of minimizing the loss function of misclassified points. However, since the related optimization problem is hard to solve, the number of subregions is limited to a small number. For example, in [8], only the cases with $K\le 5$ were considered in numerical experiment.

In order to obtain parameters efficiently and achieve a good generalization, in this paper, we apply the technique of support vector machine (SVM). In order to construct a desirable SVM, we need another formulation transformed from (1) based on the following:

Lemma 1 (*Theorem 1, Wang and Sun [18]*). *For function* $f(x):\mathbf{R}^n\to\mathbf{R}$

$$f(x)=\max_{1\le k\le K}\left\{\min_{1\le i\le I_k}\{a_{ki}^T x+b_{ki}\}\right\},$$

there exist M basis functions $\phi_m(x)$ *with parameters* $w_m\in\mathbf{R}$, $p_{mi}\in\mathbf{R}^n$ *and* $q_{mi}\in\mathbf{R}$ *such that*

$$f(x)=\sum_{m=1}^{M} w_m\phi_m(x),$$

where

$$\phi_m(x)=\max\{p_{m0}^T x+q_{m0},p_{m1}^T x+q_{m1},\ldots,p_{mn}^T x+q_{mn}\}.$$

According to Lemma 1, along with Theorem 1 and the identity $\min_k\max_i\{t_{ik}\}=-\max_k\min_i\{-t_{ik}\}$, we get another formulation of PWL classification functions. This result is presented by the following theorem, which makes SVM applicable for constructing PWL classifiers.

Theorem 2. *Any piecewise linear set* $\mathcal{B}$ *can be represented as the solution of a PWL equation, i.e.,* $\mathcal{B}=\{x: f(x)=0\}$, *where* $f(x)$ *takes the following formulation:*

$$f(x)=\sum_{m=1}^{M} w_m\phi_m(x), \tag{2}$$

and

$$\phi_m(x)=\max\{p_{m0}^T x+q_{m0},p_{m1}^T x+q_{m1},\ldots,p_{mn}^T x+q_{mn}\}. \tag{3}$$

2.2. SVM with PWL feature mapping

Representing a PWL classifier as a linear combination of basis functions makes it possible to use the SVM technique to determine the linear coefficients of (2). SVM with PWL feature mapping can also be regarded as a multi-layer perception (MLP) with hidden layers. The relation between the feature mapping of SVM and the hidden layer of MLP has been described in [20]. Using a PWL function (3) as the feature mapping, we can establish a SVM which provides a PWL classification boundary. Denote $\tilde{p}_{mi}=p_{mi}-p_{m0}$, $\tilde{q}_{mi}=q_{mi}-q_{m0}$. Formulation (3) can be equivalently transformed into

$$\phi_m(x)=p_{m0}^T x+q_{m0}+\max\{0,\tilde{p}_{m1}^T x+\tilde{q}_{m1},\ldots,\tilde{p}_{mn}^T x+\tilde{q}_{mn}\}.$$

Denoting the ith component of x as $x(i)$, we use the following PWL feature mapping in n-dimensional space:

$$\phi(x)=[\phi_1(x),\phi_2(x),\ldots,\phi_M(x)]^T,$$

where

$$\phi_m(x)=\begin{cases} x(m), & m=1,\ldots,n,\\ \max\{0,p_{m1}^T x+q_{m1},\ldots,p_{mn}^T x+q_{mn}\}, & m=n+1,\ldots,M.\end{cases} \tag{4}$$

We can construct a series of SVMs with PWL feature mappings, named as PWL-SVMs. For example, a PWL feature mapping can be applied in C-SVM [21] and we get the following formulation, called PWL-C-SVM:

$$\begin{aligned}\min_{w,w_0,e}\quad & \frac{1}{2}\sum_{m=1}^{M} w_m^2+\gamma\sum_{k=1}^{N} e_k\\ \text{s.t.}\quad & y_k\left[w_0+\sum_{m=1}^{M} w_m\phi_m(x_k)\right]\ge 1-e_k,\ \forall k,\\ & e_k\ge 0,\quad k=1,2,\ldots,N,\end{aligned} \tag{5}$$

where $x_k\in\mathbf{R}^n$, $k=1,2,\ldots,N$ are the input data, $y_k\in\{+1,-1\}$ are the corresponding labels, and feature mapping $\phi(x)=[\phi_1(x),\phi_2(x),\ldots,\phi_M(x)]^T$ takes the formulation of (4). To avoid confusion with the parameters in a feature mapping, in this paper, we use the notation w_0 to denote the bias term in SVMs. The dual problem is

$$\max_{\alpha}\quad -\frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N} y_k y_l\kappa(x_k,y_l)\alpha_k\alpha_l+\sum_{k=1}^{N}\alpha_k$$

$$\text{s.t.} \quad \sum_{k=1}^{N} \alpha_k y_k = 0,$$
$$0 \le \alpha_k \le \gamma, \quad k = 1,2,\ldots,N, \tag{6}$$

where $\alpha \in \mathbf{R}^N$ is the dual variable and the kernel is

$$\kappa(x_k,x_l) = \phi(x_k)^T \phi(x_l) = \sum_{m=1}^{M} \phi_m(x_k)\phi_m(x_l). \tag{7}$$

From the primal problem (5), we get a PWL classifier

$$\text{sign}\{w^T\phi(x)+w_0\}. \tag{8}$$

The number of the variables in (5) is $M+N+1$, while in the dual problem (6) that number is N and hence we prefer to solve (6) to get the following classifier

$$\text{sign}\left\{\sum_{k=1}^{N} \alpha_k y_k \kappa(x,x_k)+w_0\right\}.$$

To construct the classifier using the above dual form, α_k, x_k, w_0 should be stored and for (8) we only need to remember w_m,w_0. Therefore, we solve (6) to obtain dual variables and then calculate w by

$$w_m = \sum_{k=1}^{N} \alpha_k y_k \phi_m(x_k)$$

and use (8) for classification.

Using a PWL feature mapping in SVM, we get a classifier which gives a PWL classification boundary and enjoys the advantages of SVMs. In [13], researchers constructed a PWL classifier using SVM and obtained good results. However, their method can only handle separable cases and some crucial problems are remaining, including "how to introduce reasonable soft margins?" and "how to extend to nonseparable data set?" as mentioned in [13]. Using a PWL feature mapping, we successfully construct a PWL-SVM, which provides a PWL classification boundary and can deal with any kind of data.

Similarly, we can use a PWL feature mapping in least squares support vector machine (LS-SVM [22–24]) and get the following PWL-LS-SVM:

$$\min_{w,w_0,e} \frac{1}{2}\sum_{m=1}^{M} w_m^2 + \gamma\frac{1}{2}\sum_{k=1}^{N} e_k^2$$
$$\text{s.t.} y_k\left[w_0 + \sum_{m=1}^{M} w_m\phi_m(x_k)\right] = 1-e_k, \quad k = 1,2,\ldots,N. \tag{9}$$

The dual problem of (9) is a linear equation of α,w_0, i.e.

$$\begin{bmatrix} 0 & y^T \\ y & \mathcal{K}+\frac{\mathbf{I}}{\gamma} \end{bmatrix}\begin{bmatrix} w_0 \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}, \tag{10}$$

where $\mathbf{I} \in \mathbf{R}^{N\times N}$ is an identity matrix, $\mathbf{1} \in \mathbf{R}^N$ denotes the vector with components equal to one, $\alpha \in \mathbf{R}^N$ is the dual variable, $\mathcal{K}_{kl} = y_k y_l \kappa(x_k,x_l)$ and the kernel is the same as (7). The numbers of variables involved in the primal problem (9) and the dual problem (10) are $M+1$ and $N+1$, respectively. Therefore, we prefer to solve (9) to construct the classifier when $M \le N$. Otherwise, i.e., when $M > N$, we solve (10) to obtain the dual variables α_k, then calculate the coefficients w_m and use the primal formulation as the classifier.

2.3. Classification capability of PWL feature mappings

SVM with a PWL feature mapping gives a PWL boundary and enjoys the good properties of SVM. The classification capability of PWL-SVMs is related to the specific formulation of $\phi_m(x)$. The simplest one is

$$\phi_m(x) = \max\{0, x(i_m) - q_m\}, \tag{11}$$

where $q_m \in \mathbf{R}$ and $i_m \in \{1,\ldots,n\}$ denote the component used in $\phi_m(x)$. Using (11) as feature mapping, an additive PWL classifier can be constructed. The change points of the PWL classification boundaries should be located at the boundaries of subregions and the boundaries provided by (11) are restricted to be lines parallel to one of the axes.

Since the boundaries of the subregions defined by (11) are not flexible enough, some desirable classifiers cannot be obtained. To enlarge the classification capability, we should extend (11) to

$$\phi_m(x) = \max\{0, p_m^T x - q_m\}, \tag{12}$$

where $p_m \in \mathbf{R}^n$,$q_m \in \mathbf{R}$. This formulation is called a hinging hyperplane (HH [16]) and the boundaries of subregions provided by HH are lines throughout the domain, which are more flexible than that of (11). To obtain a PWL classifier with more powerful classification capability, we can add more linear functions in the following way

$$\phi_m(x) = \max\{0, p_{m1}^T x - q_{m1}, p_{m2}^T x - q_{m2}, \ldots\}.$$

The classification capability is extended along with the increase of the linear functions used in max{}. As proved in Theorem 2, an arbitrary PWL boundary in n-dimensional space can be realized using n linear functions, i.e.

$$\phi_m(x) = \max\{0, p_{m1}^T x - q_{m1}, p_{m2}^T x - q_{m2}, \ldots, p_{mn}^T x - q_{mn}\}. \tag{13}$$

Following the notation in [18], we call (13) a generalized hinging hyperplane (GHH) feature mapping.

2.4. Parameters in PWL feature mappings

Like other nonlinear feature mappings or kernels, PWL feature mappings have some nonlinear parameters, which have a big effect on the classification performance but are hard to tune optimally. To obtain reasonable parameters for PWL feature mappings, we investigate the geometrical meaning of the parameters. From the definition of a PWL set, the domain is partitioned into subregions Ω_k, in each of which the PWL set equals a hyperplane. Generally speaking, the parameters in PWL feature mappings determine the subregion structure and the hyperplane in each subregion is obtained by SVMs technique.

Let us consider again the two moons set shown in Fig. 1(a). The boundary consists of three segments which are located in subregion Ω_k as illustrated in Fig. 1(b). To construct the desirable classifier, we set

$$\phi_1(x) = x(1), \quad \phi_2(x) = x(2),$$
$$\phi_3(x) = \max\{0, x(2) - \tfrac{1}{3}\}, \quad \phi_4(x) = \max\{0, x(2) - \tfrac{2}{3}\},$$

i.e., we use feature mappings (11) with $i_1 = 1, q_1 = 0$, $i_2 = 2, q_2 = 0$, $i_3 = 2, q_3 = \frac{1}{3}$, and $i_4 = 2, q_4 = \frac{2}{3}$. Then solving PWL-C-SVM or PWL-LS-SVM leads to $w_0,\ldots,w_4$, which define a PWL boundary $w_0 + \sum_{m=1}^{4} w_m\phi_m(x) = 0$. In this example, some prior knowledge is known, then the reasonable parameters for a PWL feature mapping can be efficiently found. In black box problems, we set the parameters of (11) by equidistantly dividing the domain in each axis into several segments.

For other PWL feature mappings, we use random parameters. The boundaries of segments provided by (12) are hyperplanes $p_m^T x + q_m = 0$. To get $p_m \in \mathbf{R}^n$ and $q_m \in \mathbf{R}$, we first generate n points in the domain with uniform distribution, then we select $p_m(1)$ from $\{1,-1\}$ with equal probability, and calculate q_m and other components of p_m such that the generated points are located in $p_m^T x + q_m = 0$. The parameters obtained by this way can provide

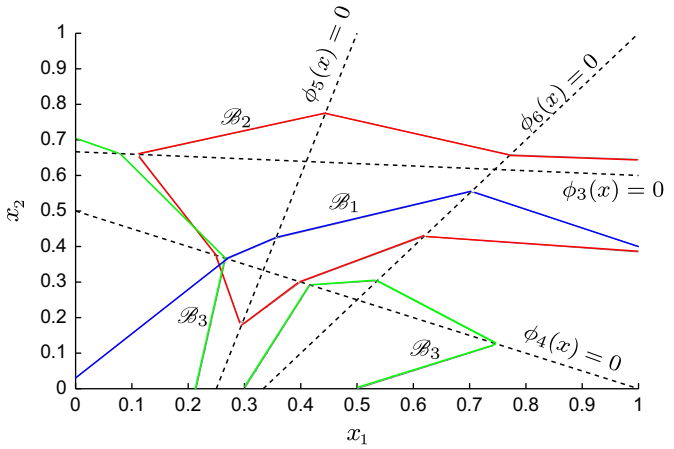


Fig. 2. The boundaries of subregions for (14) and related classification boundaries. The four dashed lines correspond to $\phi_3(x)=0$, $\phi_4(x)=0$, $\phi_5(x)=0$, and $\phi_6(x)=0$, respectively. Classification boundary is $\mathcal{B}=\{x: w_0+\sum_{m=1}^{6} w_m\phi_m(x)=0\}$. $\mathcal{B}_1$ with $w_0=-1.5$, $w_1=0.5$, $w_2=0.9$, $w_3=0.7$, $w_4=0.8$, $w_5=0.1$, $w_6=0.33$, $\mathcal{B}_2$ with $w_0=-0.5$, $w_1=-0.5$, $w_2=1$, $w_3=-1$, $w_4=0.8$, $w_5=-0.25$, $w_6=0.33$, and $\mathcal{B}_3$ with $w_0=-0.5$, $w_1=1$, $w_2=-2$, $w_3=1$, $w_4=4$, $w_5=-0.75$, $w_6=0.67$ are illustrated by red, blue, and green lines, showing the flexibility of classification boundary, which can be convex ($\mathcal{B}_1$), nonconvex ($\mathcal{B}_2$), and unconnected ($\mathcal{B}_3$). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

flexible classification boundaries. For the sake of easy comprehension, Fig. 2 shows the boundaries of subregions related to the following randomly generated PWL feature mapping:

$$\begin{aligned}
&\phi_1(x)=x(1), \quad \phi_2(x)=x(2),\\
&\phi_3(x)=\max\{0,x(1)+15x(2)-10\},\\
&\phi_4(x)=\max\{0,x(1)+2x(2)-1\},\\
&\phi_5(x)=\max\{0,-x(1)+0.25x(2)+0.25\},\\
&\phi_6(x)=\max\{0,-x(1)+0.67x(2)+0.33\}.
\end{aligned} \tag{14}$$

Three possible classification boundaries corresponding to different groups of w_m are shown. One can see the flexibility of the potential classifiers, from which the optimal one can be picked out by SVMs. In most cases, the classification performance is satisfactory, otherwise, we can generate another group of parameters. Similarly, the parameters of (13) can be generated and the resulting subregions provide flexible classification boundaries.

3. Relation to other PWL classifiers

As mentioned previously, a piecewise linear boundary is the simplest extension to a linear classification boundary. PWL boundaries enjoy low memory requirement, a little processing effort and hence are suitable for many applications. There have been some researches on PWL classification. We would like to investigate the relationship between PWL-SVM and other PWL classifiers, including k-nearest-neighbor algorithm, adaptive boosting method for linear classifiers, and intersection kernel SVM. In this section, the classification capability is discussed and the classification performance on numerical experiments is reported in the next section.

3.1. k-Nearest-neighbor

In a k-nearest-neighbor (kNN) classifier, a data point x is classified according to the k-nearest input points. In [4], it has been shown that kNN provides a PWL boundary. In this subsection, we show the specific formulation of boundaries for $k=1$ and boundaries for $k>1$ can be analyzed similarly. In kNN ($k=1$), we conclude that x belongs to class $+1$ if $d_+(x)<d_-(x)$ and x belongs to class -1 if $d_-(x)<d_+(x)$, where $d_+(x)=\min_{k:y_k=+1}\{d(x,x_k)\}$ and $d_-(x)=\min_{k:y_k=-1}\{d(x,x_k)\}$. Obviously, the classification boundary of kNN is given by $d_+(x)=d_-(x)$, i.e.

$$\min_{k:y_k=+1}\{d(x,x_k)\}=\min_{k:y_k=-1}\{d(x,x_k)\}. \tag{15}$$

Usually, the l_2 norm is used to measure the distance, then $\min_{k:y_k=+1}\{d(x,x_k)\}$ is a continuous piecewise quadratic function, which can be seen from the fact that

$$\min_{k:y_k=+1}\{d(x,x_k)\}=d(x,x_{k_1}), \quad \forall x\in\Omega_{k_1}^{+},$$

where

$$\Omega_{k_1}^{+}=\{x: d(x,k_1)\le d(x,k),\ \forall k: y_k=+1\}.$$

Subregion $\Omega_{k_1}^{+}$ is a polyhedron, since

$$\begin{aligned}
d(x,k_1)-d(x,k)&=\sum_{i=1}^{n}((x(i)-x_{k_1}(i))^2-(x(i)-x_k(i))^2)\\
&=\sum_{i=1}^{n}(2(x_k(i)-x_{k_1}(i))x(i)+x_{k_1}(i)^2-x_k(i)^2)\le 0
\end{aligned}$$

is a linear inequality with respect to x. Similarly, $d_-(x)$ is also a piecewise quadratic function, of which the subregions are denoted by Ω_k^{-}. Then one can see that (15) provides a PWL boundary, because

$$\begin{aligned}
&\min_{k:y_k=+1}\{d(x,x_k)\}-\min_{k:y_k=-1}\{d(x,x_k)\}=d(x,x_{k_1})-d(x,x_{k_2})\\
&\quad=\sum_{i=1}^{n}(2x(i)(x_{k_2}(i)-x_{k_1}(i))+x_{k_1}(i)^2-x_{k_2}(i)^2)\\
&\quad\le 0, \quad \forall x\in\Omega_{k_1}^{+}\bigcap\Omega_{k_2}^{-}.
\end{aligned}$$

Hence, the boundary of kNN given by (15) can be realized by a PWL feature mapping (4), according to Theorem 2.

3.2. Adaptive boosting

kNN is a simple extension to linear classification but it performs poorly for noise corrupted or overlapped data. Another widely used method for extending a linear classifier is adaptive boosting (Adaboost [5]). If we apply linear classification as the weak classifier in Adaboost, the resulting classification boundary is piecewise linear as well. Denote the linear classifiers used in Adaboost by $a_m^T x+b_m$ and the weights of the classifiers by η_m. Then the Adaboost classifier is

$$\operatorname{sign}\left\{\sum_{m=1}^{M}\eta_m\operatorname{sign}\{a_m^T x+b_m\}\right\}.$$

Since $\sum_{m=1}^{M}\eta_m\operatorname{sign}\{a_m^T x+b_m\}$ is not a continuous function, the classification boundary of Adaboost cannot be written as $\sum_{m=1}^{M}\eta_m\operatorname{sign}\{a_m^T x+b_m\}=0$. In order to formulate the boundary, we define the following function:

$$s_\delta(t)=-1+\frac{1}{\delta}\max\{t+\delta,0\}-\frac{1}{\delta}\max\{t-\delta,0\},$$

which has the property that $\lim_{\delta\to 0}s_\delta(t)=\operatorname{sign}(t)$. The boundary obtained by Adaboost then can be written as

$$\begin{aligned}
&\lim_{\delta\to 0}\sum_{m=1}^{M}\eta_m s_\delta(a_m^T x+b_m)\\
&\quad=\lim_{\delta\to 0}\left\{\sum_{m=1}^{M}\eta_m\left\{-1+\frac{1}{\delta}\max\{a_m^T x+b_m+\delta,0\}\right.\right.\\
&\quad\left.\left.-\frac{1}{\delta}\max\{a_m^T x+b_m-\delta,0\}\right\}\right\}=0.
\end{aligned}$$

Therefore, using (12) or a more complicated formulation, e.g., (13), PWL-SVMs can approach the boundary of Adaboost with arbitrary precision.

3.3. Intersection kernel SVM

In order to construct a SVM with a PWL boundary, intersection kernel SVM (Ik-SVM) was proposed in [25,26] and has attracted some attention. The intersection kernel takes the following form:

$$\kappa(x_1,x_2)=\sum_{i=1}^{n}\min\{x_1(i),x_2(i)\}. \tag{16}$$

By solving SVM with kernel (16), we get the dual variable α and bias w_0, then the classification function is

$$\operatorname{sign}\left\{\sum_{k=1}^{N}\alpha_k y_k \kappa(x,x_k)+w_0\right\}.$$

Therefore, the boundary obtained by Ik-SVM is the solution of the following equation:

$$w_0+\sum_{k=1}^{N}\alpha_l y_k\sum_{i=1}^{n}\min\{x(i),x_k(i)\}=0.$$

According to the identity $\min\{x(i),x_k(i)\}=-\max\{0,x(i)\ -x_k(i)\}-x(i)$, we know that the boundary of Ik-SVM can be obtained by PWL-SVMs with feature mapping (11). In another word, any Ik-SVM can be written as a PWL-SVM with particular parameters. From this fact, we can conclude that the classification capability of PWL-SVMs is better than that of Ik-SVM.

4. Numerical experiments

In Section 3, we analyze the classification capability of several existing PWL classifiers and this section evaluates the classification performance of PWL-SVMs by numerical experiments. On one hand, using (13) as the feature mapping, PWL-SVM has more classification capability than that of (11) and (12). On the other hand, we prefer a simple feature mapping, which has benefits for storing and online application. Therefore, we first use (11) as the feature mapping. If the classification precision is not satisfactory, then (12) or (13) is used as the feature mapping, where the parameters are generated as described in Section 2.4. In PWL-C-SVM (5) and PWL-LS-SVM (9), the cost of loss γ is tuned by 10-fold cross validation.

The number of features, i.e., M, is a user defined parameter. Generally, larger M results in a more flexible classifier but more computation time is needed. Consider the data set Cosexp provided by [27]. Five hundred sampling points are randomly selected as the training set and 500 new points are generated for

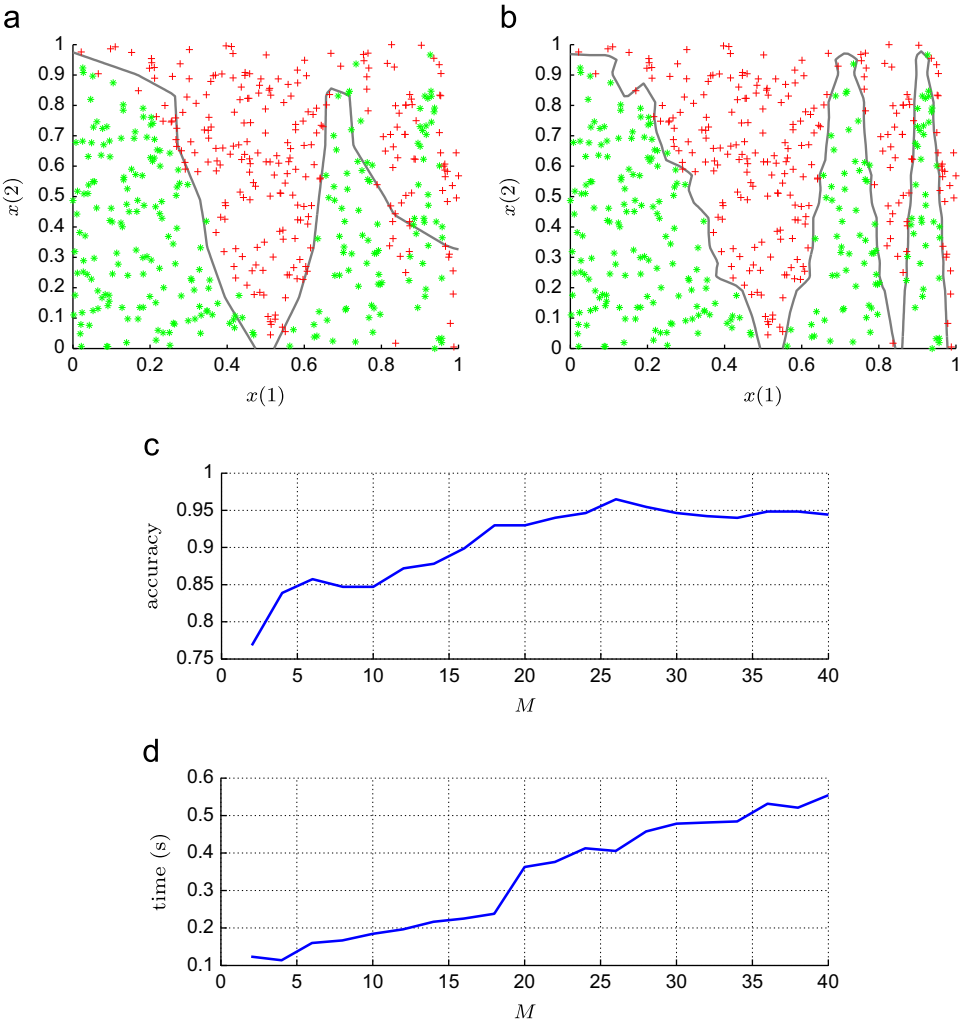


Fig. 3. The classification results of the data set Cosexp for different values of M: points in class $+1$ are marked by green stars, points in class -1 are marked by red cross, and classification boundaries are shown by black lines. (a) $M=12$; (b) $M=40$; (c) the accuracy on testing set for different M; and (d) the training time (in seconds) for different M. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

testing. We apply PWL-C-SVM with feature mapping (11) and vary M from 2 to 40. When $M=2$, the PWL feature mapping reduces to the linear mapping and the classification accuracy on testing data is 76.83%. Along with the increase of M, the potential classifier becomes more flexible, which can be seen from the comparison between Fig. 3(a) and (b) corresponding to $M=12$ and $M=40$, respectively. Large M brings high accuracy with a long training time, as illustrated by Fig. 3(c) and (d). According to this and other examples, for data with dimension n, we typically set the number of features to be $M=10n$. By this setting, the domain in each dimension is partitioned into 10 segments and the corresponding feature mappings usually provide enough flexibility. If the result with $M=10n$ is not satisfactory, we use a larger M. Another way to set M is using l_1-regularization to trade off between model complexity and accuracy, which are discussed in Section 5.

To evaluate the performance of PWL-SVM, we consider other three PWL classifiers, i.e., kNN ($k=1$), Adaboost, and Ik-SVM. To realize Adaboost, we use toolbox [28]. In order to have a fair comparison, the number of used linear classifiers is set to be M. In Ik-SVM, we use C-SVM to train the parameters, and γ is determined by 10-fold cross validation as well. Besides the three PWL classifiers, we also compare the performance with other nonlinear SVMs, including C-SVM with RBF kernel and LS-SVM with RBF kernel, of which the parameters are tuned by grid search and 10-fold cross validation. C-SVM and LS-SVM are realized by Bioinformatics Toolbox embedded in Matlab and LS-SVMlab downloaded from [29], respectively. All the experiments are done in Matlab R2011a in Core 2—2.83 GHz, with 2.96 G RAM.

In numerical experiments, we first consider five synthetic data sets generated by the dataset function in SVM-KM toolbox [27]. Then some real data downloaded from UCI Repository of Machine Learning Dataset [30] are tested. The name, along with the dimension n, the number of training data N_{train}, and the number of testing data N_{test} of each used set are listed in Table 1. In some of the data sets, there are training and testing data. For others, we randomly partition the data set into two parts, one of which is used for training (containing half of the data) and the other one is for testing. The classification accuracies on the testing data are reported in Table 1. For PWL-SVMs, we also show the type of feature mapping and the number of features for each dimension, i.e., M/n.

In Section 3, boundaries provided by PWL classifiers are analyzed: kNN can be realized by feature mapping (13), Adaboost can be realized by (12), and Ik-SVM can be realized by (11). Therefore, in theory kNN has a more classification capability than Adaboost and Ik-SVM. However, kNN performs poorly in nonseparable data and is easily corrupted by noise, hence the accuracies of kNN for some data sets, e.g., Breast, Spect, Harberman, are not very good. Comparatively, Ik-SVM enjoys the good properties of SVM and gives nice results for Breast, Spect, Harberman. However, due to the lack of flexibility, the classification results of Ik-SVM for Checker, Pima, and Monk2 are poor. The proposed PWL feature mapping has great classification capability, which means that the potential classification boundary is very flexible as shown in Section 2.4. Moreover, SVM is applicable to find the parameters, hence one can see from the results that PWL-SVMs generally outperform the discussed PWL classifiers.

In Table 1, we also compare the performance of PWL feature mappings and RBF kernel. One can see that the performance of PWL-SVMs is comparable to that of SVMs with RBF kernel. Though the accuracy of RBF kernel is better than that of PWL feature mappings in general, the difference is not significant. Compared to RBF kernel, the advantage of PWL feature mapping is the simplicity of a PWL classification boundary. For example, to remember a SVM with RBF kernel, we should store approximately $N_s(1+n)$ real numbers, where N_s stands for the number of support vectors and n is dimension of the space. Comparatively, for a SVM with (11), we only need to store M real numbers, for a SVM with (12), we need to store $M(n+1)$ real numbers and for a SVM with (13), we need to store $M(n^2+1)$ real numbers. Since N_s is usually larger than M, the store space of PWL-SVM is less than that of SVM with RBF kernel. Consider the data set Magic. The accuracy of C-SVM with RBF kernel is 0.839, which is slightly better than that of PWL-C-SVM (0.837). There are 1013 support vectors for this C-SVM with RBF kernel, then the storing space for $1013\times(10+1)$ real numbers is required. For PWL-C-SVM, we need only 100 real numbers. Moreover, when applying PWL-SVM to classify new coming data, we need only to do additive operation, multiplication, and maximum operation, which are very quick and can be implemented by hardware.

Besides the memory usage, we are also interested in the computing time of the concerned methods. Among these methods, kNN is the fastest since this classifier does not need to be trained. For other six methods, we report the training time in Table 2. LS-SVMs involve linear equations and C-SVMs can be formulated as linearly constrained quadratic programming. Hence, the training times of LS-SVMs are less than that of C-SVMs. The difference between LS-SVM with RBF kernel and PWL-LS-SVM is that different nonlinear kernels are used. Consider the calculation of one element of the kernel. For RBF kernel, $\kappa_\sigma(x_k,x_l)=\exp(-\|x_k-x_l\|^2/\sigma^2)$ and for the proposed PWL feature

Table 1
Classification accuracy on test sets.

Data name	n	$N_{\text{train}}/N_{\text{test}}$	LS-SVM	C-SVM	kNN	Adaboost	Ik-SVM	PWL LS-SVM	PWL C-SVM	Type	M/n
Clowns	2	500/500	**0.737**	0.688	0.683	0.728	0.692	0.723	0.719	(11)	10
Checker	2	500/500	**0.920**	0.918	0.908	0.516	0.488	0.866	0.874	(12)	50
Gaussian	2	500/500	**0.970**	**0.970**	0.966	0.962	**0.970**	0.960	0.960	(11)	10
Cosexp	2	500/500	0.934	0.895	0.932	0.886	0.938	0.911	**0.940**	(11)	10
Mixture	2	500/500	0.832	0.830	0.794	0.816	0.778	0.826	**0.834**	(11)	10
Pima	8	384/384	**0.768**	0.732	0.667	0.755	0.717	0.766	0.742	(11)	10
Breast	10	350/349	0.949	0.940	0.603	0.951	0.940	**0.960**	0.957	(11)	10
Monk1	6	124/432	0.803	0.769	**0.828**	0.692	0.722	0.736	0.750	(11)	50
Monk2	6	169/132	0.833	**0.854**	0.815	0.604	0.470	0.769	0.765	(12)	50
Monk3	6	122/432	0.951	0.944	0.824	0.940	**0.972**	**0.972**	**0.972**	(11)	10
Spect	21	80/187	0.818	**0.845**	0.562	0.685	0.717	0.706	0.759	(11)	20
Trans.	4	374/374	**0.783**	0.703	0.757	0.778	0.685	0.759	0.751	(11)	10
Haberman	3	153/153	0.758	0.752	0.673	**0.765**	0.686	0.758	**0.765**	(11)	10
Ionosphere	33	176/175	**0.933**	0.895	0.857	0.867	0.905	0.829	0.857	(11)	10
Parkinsons	23	98/97	0.983	0.983	0.845	**1.000**	0.948	**1.000**	**1.000**	(11)	10
Magic	10	2000/17,021	**0.854**	0.839	0.747	0.829	0.771	0.837	0.837	(11)	10

Table 2
Training time (in seconds).

Method	Clowns	Checker	Gaussian	Cosexp
LS-SVM	0.0100	0.0419	0.0440	0.0432
C-SVM	0.0451	0.0847	0.0266	0.1279
Adaboost	0.6398	3.0040	0.6005	0.6373
Ik-SVM	0.5066	3.6868	1.9578	2.0849
PWL-LSSVM	0.0044	0.0345	0.0404	0.0365
PWL-C-SVM	0.0842	0.0933	0.2483	0.3763
	Mixture	Pima	Breast	Monk1
LS-SVM	0.0419	0.0293	0.0209	0.0063
C-SVM	0.0468	0.0657	0.0317	0.0201
Adaboost	0.6341	8.7040	4.6273	26.074
Ik-SVM	3.8048	1.7676	2.6454	0.1017
PWL-LSSVM	0.0374	0.0286	0.0186	0.0024
PWL-C-SVM	0.2605	0.2388	0.2558	0.1659
	Monk2	Monk3	Spect	Trans.
LS-SVM	0.0082	0.0077	0.0021	0.0333
C-SVM	0.0390	0.0251	0.0127	0.0826
Adaboost	24.815	5.0844	1.3083	2.5626
Ik-SVM	0.1552	0.0890	0.0420	2.0620
PWL-LSSVM	0.0051	0.0016	0.0009	0.0204
PWL-C-SVM	0.2887	0.0540	0.0283	0.2157
	Haberman	Ionosphere	Parkinsons	Magic
LS-SVM	0.0051	0.0070	0.0047	0.8715
C-SVM	0.0346	0.0354	0.0173	8.1670
Adaboost	1.3620	11.429	0.2903	15.258
Ik-SVM	0.2610	0.2291	0.0648	13.613
PWL-LSSVM	0.0025	0.0069	0.0013	0.7446
PWL-C-SVM	0.0594	0.2464	0.0491	5.5917

mappings, the kernel formulation is given by (7). RBF kernel needs exponential computation and for (7), we need to calculate the sum of maximal values. Generally, the computation time of RBF kernel and (7) is similar when M is moderate, which can be seen from the training times of LS-SVM and PWL-LS-SVM. One property of RBF kernel is that when $\|x_k - x_l\|$ is large, $\kappa_\sigma(x_k, x_l)$ is very small. However, kernel (7) and the kernel used in Ik-SVM lose this property, which makes C-SVMs with these two kernels need more training time.

From the reported training time, one can infer the computing time for parameter tuning as well. For SVMs with RBF kernel, γ and σ are tuned by grid search and 10-fold cross validation. For Adaboost, there is no parameter to be tuned. Hence, though the training time of Adaboost is longer than the other considered methods, the time of constructing Adaboost is similar to the time of constructing C-SVM, which needs parameter tuning phase and training phase. For Ik-SVM, we only need to find γ and the tuning time is less than C-SVM with RBF kernel. Similarly, if M is pre-arranged as in the experiment, only γ needs to be tuned for PWL-SVMs. We can also choose M by cross validation. Then the ratio of tuning times for PWL-SVMs and SVMs with RBF kernel will be approximately the same as that of the training times reported in Table 2.

5. Extensions

In this paper, a new kind of nonlinear feature mapping which can provide piecewise linear classification boundary is proposed. Like SVMs with other nonlinear feature mappings, PWL-SVMs can be extended in different directions. The following are some examples.

First, we can use l_1-regularization, which was first proposed in [31] and called lasso, to reduce the number of features. Applying lasso to PWL-C-SVM (5) leads the following convex problem, named as PWL-C-SVM-Lasso

$$\begin{aligned} \min_{w,w_0,e} \quad & \frac{1}{2}\sum_{m=1}^{M} w_m^2 + \gamma \frac{1}{2}\sum_{k=1}^{N} e_k + \mu \sum_{m=1}^{M} |w_m| \\ \text{s.t.} \quad & y_k\left[w_0 + \sum_{m=1}^{M} w_m \phi_m(x_k)\right] \geq 1 - e_k, \quad \forall k, \\ & e_k \geq 0, \quad k = 1,2,\ldots,N. \end{aligned} \tag{17}$$

Using lasso technique, one can find a good PWL boundary with a small number of pieces. Let us consider data set Cosexp again. We use feature mapping (11) with 10 segments in each axis and solve PWL-C-SVM (5) to get the classifier. Then we use the same feature mapping and solve (17) with $\mu = 0.2\gamma$. The classification results are shown in Fig. 4, where one can see the classification boundary (black line) and the support vectors (black circle). In this case, the number of nonzero coefficients of features in PWL-C-SVM is reduced from 20 into 12 via lasso. From Fig. 4(d), it can also be seen that the boundary consists of seven segments and only eight points need to be stored for reconstructing the classification boundary.

Similarly, we apply lasso in PWL-LS-SVM, resulting in PWL-LS-SVM-Lasso below

$$\begin{aligned} \min_{w,w_0,e} \quad & \frac{1}{2}\sum_{m=1}^{M} w_m^2 + \gamma \frac{1}{2}\sum_{i=1}^{N} e_k^2 + \mu \sum_{m=1}^{M} |w_m|, \\ \text{s.t.} \quad & y_k\left[w_0 + \sum_{m=1}^{M} w_m \phi_m(x_k)\right] = 1 - e_k, \\ & k = 1,2,\ldots,N. \end{aligned} \tag{18}$$

PWL-LS-SVM-Lasso and PWL-C-SVM-Lasso can get satisfactory classification results with a small number of features. In numerical experiments, we use the same γ as used in the experiments in Section 4 and then set $\mu = 0.2\gamma$. The accuracy on testing data is reported in Table 3, where the numbers of nonzero coefficients are

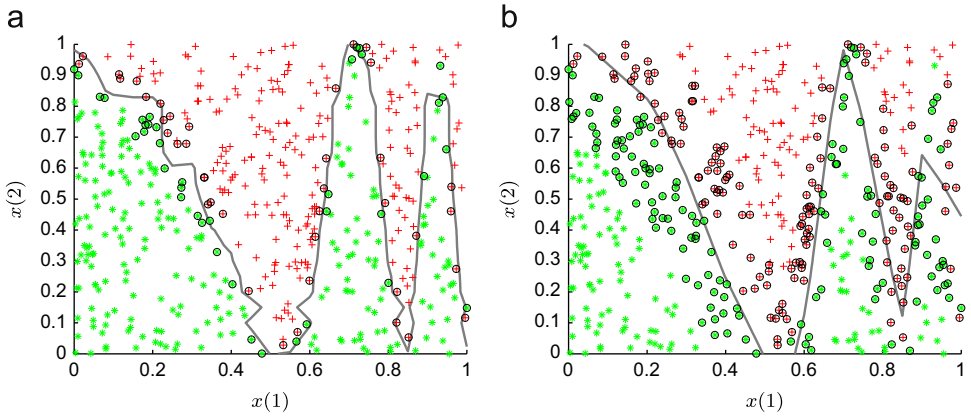


Fig. 4. The classification results of data set Cosexp: points in class $+1$ are marked by green stars, points in class -1 are marked by red cross, support vectors are marked by black circle, and classification boundaries are shown by black lines. (a) PWL-C-SVM and (b) PWL-C-SVM-Lasso. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Table 3
Classification accuracy on test sets and the dimension of the feature mappings.

Method	Clowns	Checker	Gaussian	Cosexp	Mixture	Pima	Breast	Monk1
PWL-C-SVM	0.719 (18)	0.874 (94)	0.960 (17)	0.940 (21)	0.834 (18)	0.742 (81)	0.957 (96)	0.750 (150)
PWL-C-SVM-Lasso	0.719 (12)	0.812 (45)	0.964 (9)	0.913 (13)	0.832 (13)	0.797 (24)	0.954 (25)	0.718 (30)
PWL-LS-SVM	0.723 (41)	0.866 (99)	0.956 (21)	0.911 (21)	0.826 (21)	0.766 (81)	0.960 (101)	0.736 (296)
PWL-LS-SVM-Lasso	0.723 (12)	0.810 (51)	0.958 (11)	0.924 (13)	0.822 (15)	0.792 (25)	0.957 (34)	0.732 (58)
	Monk2	Monk3	Spect	Trans.	Haberman	Ionosphere	Parkinsons	Magic
PWL-C-SVM	0.765 (301)	0.972 (21)	0.759 (406)	0.751 (36)	0.765 (60)	0.857 (331)	1.000 (83)	0.837 (94)
PWL-C-SVM-Lasso	0.743 (197)	0.972 (15)	0.743 (172)	0.765 (14)	0.765 (14)	0.867 (88)	1.000 (21)	0.842 (49)
PWL-LS-SVM	0.769 (301)	0.972 (61)	0.706 (401)	0.759 (41)	0.758 (61)	0.829 (331)	1.000 (229)	0.837 (101)
PWL-LS-SVM-Lasso	0.595 (205)	0.972 (16)	0.711 (309)	0.762 (13)	0.765 (14)	0.867 (102)	1.000 (72)	0.838 (52)

given in brackets. From the results one can see the effectiveness of using lasso in PWL-SVMs.

Lasso is realized by an additional convex term of the objective function. Similarly, we can consider some convex constraints which maintain the convexity. For example, if we have the prior knowledge that points of class $+1$ come from a convex set, then we can let $w_m \geq 0$ which results in a convex PWL function $f(x)$ and $\{x : f(x) \geq 0\}$ is a convex PWL set, i.e., polyhedron.

PWL-SVMs can also be used in nonlinear regression, which results in continuous PWL functions. For example, in time series segmentation problem, researchers try to find segments for a time series and use linear function in each segment to describe the original signal. For this problem, [32] applies HH feature mapping (12) and lasso technique in LS-SVM to approach one-dimensional signals.

6. Conclusion

In this paper, piecewise linear feature mapping is proposed. In theory, any PWL classification boundary can be realized by a PWL feature mapping and the relationship between a PWL feature mapping and some widely used PWL classifiers is discussed. Then we combine PWL feature mappings and SVM technique to establish an efficient PWL classification method. Due to the different types of SVMs, alternative PWL classifiers can be constructed, including PWL-C-SVM, PWL-LS-SVM and the ones using lasso. These methods give PWL classification boundaries, which need a little storage space and are suitable for online application. The potential classification boundary provided by PWL-SVMs is very flexible and PWL-SVMs enjoy the advantages of SVM, such as good generalization and solid foundation in statistical inference. The analysis and numerical experiments both imply PWL-SVMs promising tools for many classification tasks.

Acknowledgment

The authors would like to thank the reviewers for their helpful comments on this paper.

References

[1] D. Webb, Efficient Piecewise Linear Classifiers and Applications, Ph.D. Thesis, The Graduate School of Information Technology and Mathematical Sciences, University of Ballarat, 2010.
[2] A. Kostin, A simple and fast multi-class piecewise linear pattern classifier, Pattern Recognition 39 (11) (2006) 1949–1962.
[3] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[4] K. Fukunaga, Statistical pattern recognition, in: C.H. Chen, L.F. Pau, P.S.P. Wang (Eds.), Handbook of Pattern Recognition & Computer Vision, World Scientific Publishing Co., Inc., 1993, pp. 33–60.
[5] Y. Freund, R. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: Computational Learning Theory, Springer, 1995, pp. 23–37.
[6] J. Sklansky, L. Michelotti, Locally trained piecewise linear classifiers, IEEE Trans. Pattern Anal. Mach. Intell. (2) (1980) 101–111.
[7] H. Tenmoto, M. Kudo, M. Shimbo, Piecewise linear classifiers with an appropriate number of hyperplanes, Pattern Recognition 31 (11) (1998) 1627–1634.
[8] A.M. Bagirov, Max–min separability, Optim. Methods Software 20 (2–3) (2005) 277–296.
[9] R. Kamei, Experiments in Piecewise Approximation of Class Boundary Using Support Vector Machines, Master Thesis, Electrical and Computer Engineering and Computer Science, The College of Engineering, Kansai University, 2003.
[10] A.M. Bagirov, J. Ugon, D. Webb, An efficient algorithm for the incremental construction of a piecewise linear classifier, Inf. Syst. 36 (4) (2011) 782–790.
[11] A.M. Bagirov, J. Ugon, D. Webb, B. Karasözen, Classification through incremental max–min separability, Pattern Anal. Appl. 14 (2) (2011) 165–174.

[12] K. Gai, C. Zhang, Learning discriminative piecewise linear models with boundary points, in: The 24th AAAI Conference on Artificial Intelligence, 2010.
[13] Y. Li, B. Liu, X. Yang, Z. Fu, H. Li, Multiconlitron: a general piecewise linear classifier, IEEE Trans. Neural Networks (99) (2011) pp. 276–289.
[14] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[15] L.O. Chua, S.M. Kang, Section-wise piecewise-linear functions: canonical representation, properties, and applications, Proc. IEEE 65 (6) (1977) 915–929.
[16] L. Breiman, Hinging hyperplanes for regression, classification and function approximation, IEEE Trans. Inf. Theory 39 (3) (1993) 999–1013.
[17] J.M. Tarela, M.V. Martinez, Region configurations for realizability of lattice piecewise-linear models, Math. Comput. Modelling 30 (11–12) (1999) 17–27.
[18] S. Wang, X. Sun, Generalization of hinging hyperplanes, IEEE Trans. Inf. Theory 12 (51) (2005) 4425–4431.
[19] S. Wang, X. Huang, K.M. Junaid, Configuration of continuous piecewise-linear neural networks, IEEE Trans. Neural Networks 19 (8) (2008) 1431–1445.
[20] J.A.K. Suykens, J. Vandewalle, Training multilayer perceptron classifiers based on a modified support vector method, IEEE Trans. Neural Networks 10 (4) (1999) 907–911.
[21] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[22] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[23] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, Singapore, 2002.
[24] J.A.K. Suykens, J. De Brabanter, L. Lukas, J. Vandewalle, Weighted least squares support vector machines: robustness and sparse approximation, Neurocomputing 48 (1–4) (2002) 85–105.
[25] A. Barla, F. Odone, A. Verri, Histogram intersection kernel for image classification, in: Proceedings of IEEE International Conference on Image Processing, vol. 3, 2003, pp. 513–516.
[26] S. Maji, A.C. Berg, J. Malik, Classification using intersection kernel support vector machines is efficient, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[27] S. Canu, Y. Grandvalet, V. Guigue, A. Rakotomamonjy, SVM and kernel methods Matlab toolbox, in: Perception Systmes et Information, INSA de Rouen, Rouen, France, 2005.
[28] D. Kroon, Classic Adaboost Classifier, Department of Electrical Engineering Mathematics and Computer Science (EEMCS), University of Twente, The Netherlands, 2010.
[29] K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle, J.A.K. Suykens, LS-SVMlab Toolbox User's Guide Version 1.8, Internal Report 10-146, ESAT-SISTA, K.U.Leuven, Leuven, Belgium, 2010.
[30] A. Frank, A. Asuncion, UCI Machine Learning Repository, School of Information and Computer Science, University of California, Irvine, 2010 〈http://archive.ics.uci.edu/ml〉.
[31] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Ser. B Methodol. 58 (1) (1996) 267–288.
[32] X. Huang, M. Matijaš, J.A.K. Suykens, Hinging hyperplanes for time-series segmentation, submitted for publication.



Xiaolin Huang 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. In 2012, he received the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China. Since then, he has been working as a post-doctoral researcher in ESAT-SCD-SISTA, KU Leuven, Leuven, Belgium.

His current research areas include optimization, classification, and identification for nonlinear systems via continuous piecewise linear analysis.



Siamak Mehrkanoon received the B.S. degree in pure mathematics in 2005 and the M.S. degree in applied mathematics from the Iran University of Science and Technology, Tehran, Iran, in 2007. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium. His current research interests include machine learning, system identification, pattern recognition, and numerical algorithms.



Johan A.K. Suykens 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 the Katholieke Universiteit Leuven, in 1989 and 1995, respectively. In 1996 he has been a visiting post-doctoral researcher at the University of California, Berkeley. He has been a post-doctoral researcher with the Fund for Scientific Research FWO Flanders and is currently a Professor (Hoogleraar) with KU Leuven. He is the author of the books Artificial Neural Networks for Modeling and Control of Nonlinear 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. He is a Senior IEEE member and 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 received an IEEE Signal Processing Society 1999 Best Paper (Senior) Award and several Best Paper Awards at International Conferences. He is a recipient of the International Neural Networks Society INNS 2000 Young Investigator Award for significant contributions in the field of 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 2004 and the International Symposium on Nonlinear Theory and its Applications 2005, as an organizer of the International Symposium on Synchronization in Complex Networks 2007 and a co-organizer of the NIPS 2010 Workshop on Tensors, Kernels and Machine Learning. He has been recently awarded an ERC Advanced Grant 2011.