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

Abstract

As the simplest extension of 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, named as PWL-SVMs. In this paper, it is shown that some widely used 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 archive the performance of other 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.

Keywords: support vector machine, piecewise linear classifier, nonlinear feature mapping

1. Introduction

Piecewise linear (PWL) classifiers are a kind of classification methods which provide PWL boundaries. In some point of view, PWL boundaries are the simplest extension for linear boundaries. On the 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, 7, 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 to which piece a point belongs and establish an optimization problem for constructing a PWL classifier. The resulting classifier has great classification capability, 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 some linear classifiers and construct 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 capability 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

[☆]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.

Email addresses: huangxl06@mails.tsinghua.edu.cn (Xiaolin Huang), siamak.mehrkanoon@esat.kuleuven.be (Siamak Mehrkanoon), johan.suykens@esat.kuleuven.be (Johan A.K. Suykens)

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, 17, 18] and [19].

The rest of this paper is organized as follows: in Section 2, SVMs with piecewise linear feature mappings, i.e., PWL-SVMs, are proposed. Section 3 gives the comparison of the proposed methods with other PWL classifiers. Then PWL-SVMs are evaluated by numerical experiments in Section 4. Some extensions are discussed in Section 5. And Section 6 ends the paper with conclusions.

2. SVM with Piecewise Linear Feature

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 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 for some applications and nonlinear classifiers are required. A linear classification boundary, i.e., a hyperplane, is an affine set. In some point of view, the simplest extension of 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 $\mathscr{B}$, 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 i-th component of x. Then $\mathscr{B}$ can be defined as

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

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 $\mathscr{B}$ defined in the domain $\Omega \subseteq \boldsymbol{R}^n$ meets the following two conditions,*

i) the domain Ω can be partitioned into finite polyhedra $\Omega_1, \Omega_2, \ldots, \Omega_K$, i.e., $\Omega = \bigcup_{k=1}^{K} \Omega_k$ and $\overset{\circ}{\Omega}_{k_1} \bigcap \overset{\circ}{\Omega}_{k_2} = \emptyset, \forall k_1 \neq k_2$, where $\overset{\circ}{\Omega}_k$ stands for the interiors of Ω_k;

ii) in each of the subregions, $\mathscr{B}$ equals a linear set, i.e., for each k, there exist $c_k \in \boldsymbol{R}^n, d_k \in \boldsymbol{R}$ such that $\mathscr{B} \bigcap \Omega_k = \{x : c_k^T x + d_k = 0\}$;

then, $\mathscr{B}$ is a piecewise linear set.

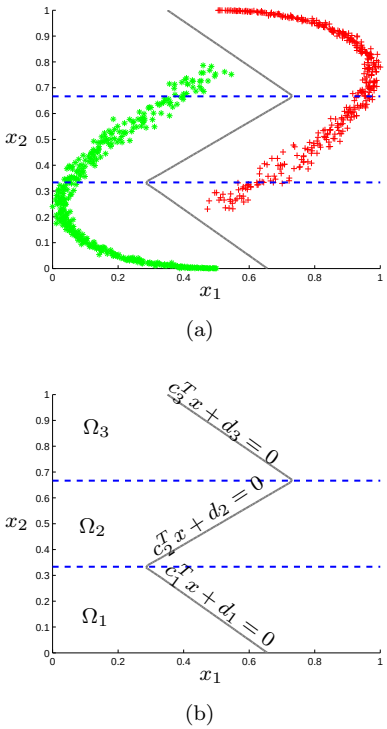

Figure 1: An example of a piecewise linear boundary. (a) Points in two classes are marked by green stars and red crosses, respectively; the classification boundary is shown by black lines; (b) Restricted to each of the subregions, the PWL boundary corresponds to a line.

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., $\text{sign}\{f(x)\}$. 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 $\mathscr{B}$ can be represented as the solution of a piecewise linear function.*

Proof. Denote the polyhedra defining $\mathscr{B}$ 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,

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

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

According to the definition of piecewise linear set, for any $x_0 \in \mathscr{B}$, there exists a polyhedron Ω_{k_0} such that $x_0 \in \Omega_{k_0}$ and $c_{k_0}^T x_0 + d_{k_0} = 0$. From $x_0 \in \Omega_{k_0}$, we have

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

therefore,

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

Since $\Omega_1, \Omega_2, \ldots, \Omega_K$ compose a partition of the domain, we have $x_0 \notin \mathring{\Omega}_k, \forall k \neq k_0$, that means

$$\max_{1\le i\le I_k}\left\{a_{ki}^T x + b_{ki}\right\} \ge 0, \forall k \neq k_0.$$

No matter the value of $c_k^T x_0 + d_k$, for any k, there is

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

Accordingly,

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

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

Next we prove that $\{x : f(x) = 0\} \subseteq \mathscr{B}$. 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

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

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 $x_0 \in \mathscr{B} \bigcap \Omega_{k_0} \subseteq \mathscr{B}$ and then $\{x : f(x) = 0\} \subseteq \mathscr{B}$.

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

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,

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

Using a min-max formulation, a PWL classifier can be constructed. The problem of determining the parameters can be posed as a non-convex and non-differentiable 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 for $K \le 5$ 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 machines (SVM). In order to construct a desirable SVM, we need another formulation transformed from (1) based on the following lemma.

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

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

there exist M basis functions $\phi_m(x)$ with parameters $w_m \in \boldsymbol{R}$, $p_{mi} \in \boldsymbol{R}^n$ and $q_{mi} \in \boldsymbol{R}$ 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 $\mathscr{B}$ can be represented as the solution of a PWL equation, i.e., $\mathscr{B} = \{x : f(x) = 0\}$, where $f(x)$ takes the following formulation*

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

and

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

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 seen as a multilayer 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 i-th 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

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

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 one can get the following formulation, called PWL-C-SVM,

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

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

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N} y_k y_l \kappa(x_k, y_l)\alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \\ \text{s.t.} \quad & \sum_{k=1}^{N} \alpha_k y_k = 0, \\ & 0 \leq \alpha_k \leq \gamma, k = 1, 2, \ldots, N, \end{aligned} \tag{6}$$

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

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

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

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

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

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

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

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

and use (8) for classification.

Using a PWL feature mapping in SVM, we get a classifier which can give a PWL classification boundary and enjoy 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 machines (LS-SVM, [22, 23, 24]) and get the following PWL-LS-SVM,

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

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 number 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 $M \leq N$. 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 $i_m \in \{1, \ldots, n\}$ denotes 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 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 line. Generally speaking, the parameters in PWL feature mappings determine the subregion structure and the line 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 can set

$$\phi_1(x) = x(1), \quad \phi_2(x) = x(2),$$

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

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 using PWL-C-SVM (5) or PWL-LS-SVM (9), we can find $w_0, \ldots, w_4$ and $w_0 + \sum_{m=1}^4 w_m \phi_m(x) = 0$ defines a PWL boundary. In this example, some prior knowledge is known, then the reasonable parameters for a PWL feature mapping can be efficiently found. In regular case, we can 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 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.

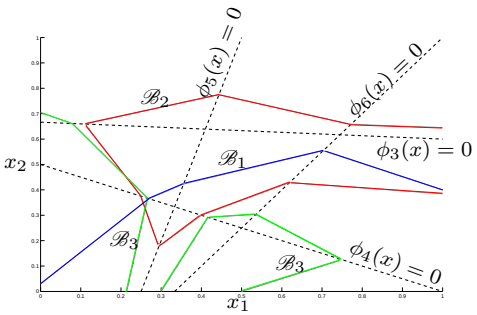

Figure 2: The boundaries of subregions for (14) and related classification boundaries. The four dashed lines correspond to $\phi_3(x) = 0, \phi_4(x) = 0, \phi_5(x) = 0$, and $\phi_6(x) = 0$, respectively. Classification boundary is $\mathscr{B} = \{x : w_0 + \sum_{m=1}^6 w_m \phi_m(x) = 0\}$. $\mathscr{B}_1$ with $w_0 = -1.5, w_1 = 0.5, w_2 = 0.9, w_3 = 0.7, w_4 = 0.8, w_5 = 0.1, w_6 = 0.33$, $\mathscr{B}_2$ 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 $\mathscr{B}_3$ 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 ($\mathscr{B}_1$), non-convex ($\mathscr{B}_2$), and unconnected ($\mathscr{B}_3$).

3. Comparison with other PWL Classifiers

As mentioned previously, a piecewise linear boundary is the simplest extension of a linear classification boundary. PWL boundaries enjoy low memory requirement, a little processing effort and hence are suitable for many applications. In fact, there has been some research on PWL classification. We would like to investigate the relationship between PWL-SVM with other PWL classifiers, including the k-nearest neighbor algorithm, adaptive boosting method for linear classifiers, and the intersection kernel SVM. In this section, the classification capability is discussed and the classification performance on numerical experiments is given 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 section, 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 2-norm is used to measure the distance, then $\min_{k:y_k=+1}\{d(x,x_k)\}$ is a continuous piecewise quadratic function, that can be seen from the fact that

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

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

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

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

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

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 of 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 $h_m(x) = a_m^T x + b_m$ and the weights of the classifiers by η_m. Then the Adaboost classifier is

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

Since $\sum_{m=1}^{M} \eta_m \text{sign}\{a_m^T x + b_m\}$ is not a continuous function, the classification boundary of Adaboost cannot be written as $\sum_{m=1}^{M} \eta_m \text{sign}\{a_m^T x + b_m\} = 0$. 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 satisfies that $\lim_{\delta\to 0} s_\delta(t) = \text{sign}(t)$. Then, the boundary obtained by Adaboost can be written as

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

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, the 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

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

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

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

From 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).

4. Numerical Experiments

In Section 3, we analyze the classification capability of several popular PWL classifiers and this section evaluates the classification performance of PWL-SVMs by numerical experiments. On the 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 function γ is tuned by 10-fold cross validation.

Table 1: Classification accuracy on test sets

Data Name	n	$N_{\text{train}}/N_{\text{test}}$	LSSVM	C-SVM	kNN	Adaboost	Ik-SVM	PWL C-SVM	PWL LS-SVM	Type	$\frac{M}{n}$
Clowns	2	500/500	**0.737**	0.688	0.683	0.728	0.692	0.719	0.723	(11)	10
Checker	2	500/500	**0.920**	0.918	0.908	0.516	0.488	0.874	0.866	(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	**0.940**	0.911	(11)	10
Mixture	2	500/500	0.832	0.830	0.794	0.816	0.778	**0.834**	0.826	(11)	10
Pima	8	384/384	**0.768**	0.732	0.667	0.755	0.717	0.742	0.766	(11)	10
Breast	10	350/349	0.949	0.940	0.603	0.951	0.940	0.957	**0.960**	(11)	10
Monk1	6	124/432	0.803	0.769	**0.828**	0.692	0.722	0.750	0.736	(11)	50
Monk2	6	169/132	0.833	**0.854**	0.815	0.604	0.470	0.765	0.769	(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	0.759	0.706	(11)	20
Trans.	4	374/374	**0.783**	0.703	0.757	0.778	0.685	0.751	0.759	(11)	10
Haberman	3	153/153	0.758	0.752	0.673	**0.765**	0.686	**0.765**	0.758	(11)	10
Ionosphere	33	176/175	**0.933**	0.895	0.857	0.867	0.905	0.857	0.829	(11)	10
Parkinsons	23	98/97	0.983	0.983	0.845	**1.000**	0.948	**1.000**	**1.000**	(11)	10
Magic	10	2000/17021	**0.854**	0.839	0.747	0.829	0.771	0.837	0.837	(11)	10

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 [27]. In order to have a fair comparison, the number of used linear classifiers is set to be the same as M, i.e., the number of features of PWL-SVMs. In Ik-SVM, we use a hinge-loss, as used in PWL-C-SVM (5), 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.

In numerical experiments, we first consider 5 synthetic data sets generated by the dataset function in SVM-KM toolbox [28]. Then some real data downloaded from UCI Repository of Machine Learning Dataset ([29]) 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., $\frac{M}{n}$.

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 non-sparable data and is easily corrupted by noise, hence the accuracies of kNN for some data sets, e.g., Brest, Spect, Harberman, are not very good. Comparatively, Ik-SVM enjoys the good properties of SVM and gives nice results for Brest, Spect, Harberman. However, due to the lack of classification capability, the classification results of Ik-SVM for Pima, Monk2 are poor. The proposed PWL feature mapping has great classification capability and SVM is applicable to find the parameters, hence one can see from the results that PWL-SVMs generally outperform other 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-SVM 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, where M is the number of features. Since N_s is usually larger than M, the store space of PWL-SVM is less than that of SVM with RBF kernel. Consider 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 the storing space for $1013 \times (10+1)$ real numbers is required. And for PWL-C-SVM, we need only 100 real numbers. Moreover, when applying PWL-SVM to classify new coming data, we only need to do additive operation, multiplication, and maximum operation, which are very quick and can be implemented by hardware.

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 is first proposed in [30] 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,

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

Using a lasso technique, one can find a good PWL boundary with a small number of pieces. Let us consider data set Cosexp. 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.3, 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.3(b), it also can be seen that the boundary consist of 7 segments and only 8 points need to be stored for reconstructing the classification boundary.

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

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

Using PWL-LS-SVM-Lasso and PWL-C-SVM-Lasso, we 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 2, where the number of nonzero coefficients are 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, [31] 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 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. Moreover, PWL-SVMs enjoy the advantages of SVM and outperform other PWL classifiers in numerical study, which implies PWL-SVMs promising tools for many classification tasks.

References

[1] D. Webb. *Efficient piecewise linear classifiers and applications*. PhD 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):1949–1962, 2006.

[3] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[4] K. Fukunaga. Statistical pattern recognition. In *Handbook of Pattern Recognition & Computer Vision* (eds. C.H. Chen, L.F. Pau, and P.S.P. Wang), pages 33–60, World Scientific Publishing Co., Inc., 1993.

[5] Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory*, pages 23–37. Springer, 1995.

[6] J. Sklansky and L. Michelotti. Locally trained piecewise linear classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):101–111, 1980.

[7] H. Tenmoto, M. Kudo, and M. Shimbo. Piecewise linear classifiers with an appropriate number of hyperplanes. *Pattern Recognition*, 31(11):1627–1634, 1998.

[8] A.M. Bagirov. Max–min separability. *Optimization Methods and Software*, 20(2-3):277–296, 2005.

[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, and D. Webb. An efficient algorithm for the incremental construction of a piecewise linear classifier. *Information Systems*, 36(4):782–790, 2011.

[11] A.M. Bagirov, J. Ugon, D. Webb, and B. Karasözen. Classification through incremental max–min separability. *Pattern Analysis & Applications*, 14(2):165–174, 2011.

[12] K. Gai and C. Zhang. Learning discriminative piecewise linear models with boundary points. In *the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

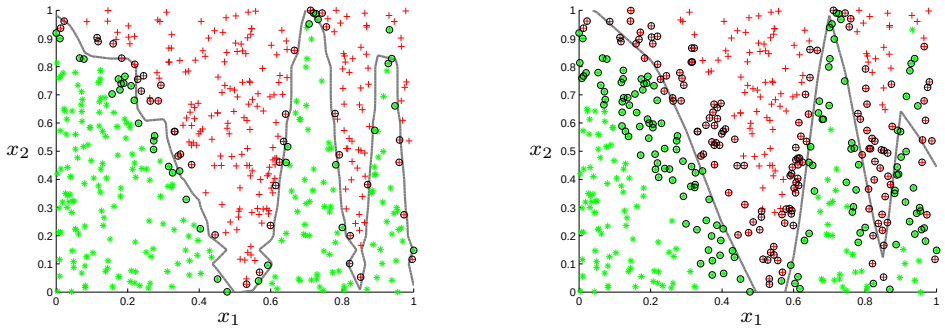

Figure 3: The classification results of data set Cosexp: points in class $+1$ are marked by green stars, points in class -1 are marked by red cross, support vectors are marked by black circle, and classification boundaries are shown by black lines. (a) PWL-C-SVM; (b) PWL-C-SVM-Lasso.

Table 2: Classification accuracy on test sets and the dimension of the feature mappings

	Clowns	Checker	Gaussian	Cosexp	Mixture	Pima	Breast	Monk1
PWL-C-SVM	0.719 (17)	0.874 (93)	0.960 (16)	0.940 (20)	0.834 (17)	0.742 (80)	0.957 (95)	0.750 (149)
PWL-C-SVM-Lasso	0.719 (11)	0.812 (44)	0.964 (8)	0.913 (12)	0.832 (12)	0.797 (23)	0.954 (24)	0.718 (29)
PWL-LS-SVM	0.723 (40)	0.866 (98)	0.956 (20)	0.911 (20)	0.826 (20)	0.766 (80)	0.960 (100)	0.736 (295)
PWL-LS-SVM-Lasso	0.723 (11)	0.810 (50)	0.958 (10)	0.924 (12)	0.822 (14)	0.792 (24)	0.957 (33)	0.732 (57)
	Monk2	Monk3	Spect	Trans.	Haberman	Ionosphere	Parkinsons	Magic
PWL-C-SVM	0.765 (300)	0.972 (20)	0.759 (405)	0.751 (35)	0.765 (59)	0.857 (330)	1.000 (82)	0.837 (93)
PWL-C-SVM-Lasso	0.743 (196)	0.972 (14)	0.743 (171)	0.765 (13)	0.765 (13)	0.867 (87)	1.000 (20)	0.842 (48)
PWL-LS-SVM	0.769 (300)	0.972 (60)	0.706 (400)	0.759 (40)	0.758 (60)	0.829 (330)	1.000 (228)	0.837 (100)
PWL-LS-SVM-Lasso	0.595 (204)	0.972 (15)	0.711 (308)	0.762 (12)	0.765 (13)	0.867 (101)	1.000 (71)	0.838 (51)

[13] Y. Li, B. Liu, X. Yang, Zhao. Fu, and H. Li. Multiconlitron: a general piecewise linear classifier. *IEEE Transactions on Neural Networks*, (99):276–289, 2011.

[14] V. Vapnik. *Statistical learning theory*. Wiley, New York, 1998.

[15] L.O. Chua and S.M. Kang. Section-wise piecewise-linear functions: canonical representation, properties, and applications. *Proceedings of IEEE*, 65(6):915–929, 1977.

[16] L. Breiman. Hinging hyperplanes for regression, classification and function approximation. *IEEE Transactions on Information Theory*, 39(3):999–1013, 1993.

[17] J.M. Tarela and M.V. Martinez. Region configurations for realizability of lattice piecewise-linear models. *Mathematical and Computer Modelling*, 30(11-12):17–27, 1999.

[18] S. Wang and X. Sun. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 12(51):4425–4431, 2005.

[19] S. Wang, X. Huang, and K. M. Junaid. Configuration of continuous piecewise-linear neural networks. *IEEE Transactions on Neural Networks*, 19(8):1431–1445, 2008.

[20] J.A.K. Suykens and J. Vandewalle. Training multilayer perceptron classifiers based on a modified support vector method. *IEEE Transactions on Neural Networks*, 10(4):907–911, 1999.

[21] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[22] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[23] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least squares support vector machines*. World Scientific, Singapore, 2002.

[24] J.A.K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle. Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, 48(1-4):85–105, 2002.

[25] A. Barla, F. Odone, and A. Verri. Histogram intersection kernel for image classification. In *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pages: 513–516, 2003.

[26] S. Maji, A.C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[27] D. Kroon. Classic adaboost classifier. Department of Electrical Engineering Mathematics and Computer Science (EEMCS), University of Twente, Netherlands, 2010.

[28] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. SVM and kernel methods matlab toolbox. Perception Systmes et Information, INSA de Rouen, Rouen, France, 2005.

[29] A. Frank and A. Asuncion. UCI machine learning repository. [http://archive.ics.uci.edu/ml]. Irvine, University of California, School of Information and Computer Science, 2010.

[30] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[31] X. Huang, M. Matijaš, and J.A.K. Suykens. Hinging hyperplanes for time-series segmentation. *IEEE Transactions on Neural Networks and Learning Systems*, 24(8): 1279–1291, 2013.