Classification with Truncated ℓ_1 Distance Kernel

Xiaolin Huang,Johan A. K. Suykens, Shuning Wang, Andreas Maier, and Joachim Hornegger

***Abstract*—This paper focuses on piecewise linear kernels for classification, especially a kernel called *truncated* ℓ_1 *distance (TL1)* kernel. Compared to other nonlinear kernels, such as the radial basis function (RBF) kernel, the main advantage of the TL1 kernel is that non-linearity can be adaptively trained. With this property, for problems which require different non-linearity in different areas, the classification performance of the TL1 kernel can be improved with respect to the RBF kernel. The adaptiveness to non-linearity also makes the TL1 kernel insensitive to its parameter. In many datasets, the TL1 kernel with a pre-given parameter achieves similar or better classification accuracy than the RBF kernel for which the parameter is tuned by cross-validation. The good performance of the TL1 kernel in numerical experiments makes it a promising nonlinear kernel for classification tasks.**

***Index Terms*—support vector machine, piecewise linear, indefinite kernel**

I. Introduction

LINEAR classifiers, which provide hyperplanes as the discrimination rules, are the simplest classifiers and have good performance in many applications. When the problem becomes complicated, nonlinear classifiers, usually induced by nonlinear kernels, are needed. The simplest step going from linear classifiers to nonlinear ones is piecewise linear (PWL) classifiers. As the name is suggesting, a PWL classifier equals a linear classifier in one subregion and all the subregions tessellate the whole domain. Perhaps without being aware of it, one may have already used some PWL classifiers, such as theK-nearest neighbors algorithm [1], classification and regression trees [2] with linear rules, and adaptive boosting [3] from weak linear classifiers.

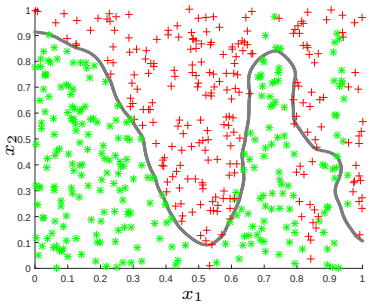
After partitioning the domain into subregions, training a linear classifier in each subregion is in general not hard. Therefore, the main challenge of constructing PWL classifiers is to find boundaries of subregions. There have been already interesting and remarkable results in [4]–[6]. Since a PWL decision boundary can be induced from a PWL equation, PWL classifier training is closely related to PWL function representation theory, which investigates the representation and approximation capability of compact PWL representations [7]–[14]. Those results can be readily extended to classification fields. With the help of a compact PWL function model, training a PWL classifier is formulated as an optimization problem with nonlinear parameter tuning, which is non-convex and related to subregion configuration, and linear parameter training, which is convex and related to linear classifier construction in subregions. This kind of PWL classification methods can be found in [15] [16] and [17].

Nonlinearity of a PWL classifier comes from the existence of subregions. Our aim in this paper is to find a suitable PWL kernel, for which support vectors provide subregions and non-support vectors do not. In this way, it is expected that non-linearity can be adaptively trained by kernel learning methods. In comparison, consider the radial basis function (RBF) kernel $\mathcal{K}(\mathbf{u},\mathbf{v}) = \exp\left(-\|\mathbf{u}-\mathbf{v}\|_{\ell_2}^2/\sigma^2\right)$, which is the most popular nonlinear kernel. In the RBF kernel, a large σ is to generate a smooth classification curve, or equally speaking, the non-linearity is light. If heavy non-linearity is needed, i.e., the ideal classification curve has large curvature, then a small σ performs well. However, for the situation that the ideal classifier has heavy non-linearity on one part but performs linearly on another part, it is hard to find a suitable σ value. As a simple example, we use the classical support vector machine (C-SVM, [18]) to train a classifier from the RBF kernel for a two-dimensional problem the "cosexp". In Fig.1, the sampling points in two classes are displayed by red crosses and green stars, respectively. It can be expected that the ideal classifier changes smoothly on the left part of the domain and has sharp vertices on the right part. To pursue a flat classification boundary, a relatively large σ is preferred. We set $\sigma = 1.0$ and display the obtained decision boundary by solid curves in Fig.1(a). With this setting, the left part is classified quite well but the right part is not. To achieve high accuracy for the right part, we need to decrease σ value. In Fig.1(b), the solid curve stands for classification boundary for $\sigma = 0.2$, which has better performance on the right part but it is not as flat as we expect on the left part. Moreover, since a small σ is used, it needs many support vectors to follow a flat trend.

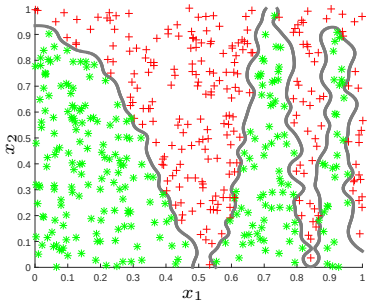
For this kind of problems, which need different non-linearity in different areas, PWL kernels become attractive. With C-SVM or other kernel learning methods, we expect that PWL kernels can adaptively coincide the non-linearity requirement: for the right part of "cosexp" problem, there are many support vectors such that the classification boundary changes dramatically; while, on the left part, there are a few support vectors and the boundary is flat. Unlike the parametric methods based on PWL representation models, learning from PWL kernels combines subregion configuration and local linear classifier training together. Though there is already some research on PWL classifiers, discussion on PWL kernels is rare. One interesting PWL kernel is designed by [19] [20] and is named the *additive kernel*, since it can be written as the sum of several univariate functions. Though the classifier constructed

Manuscript received 2015;

X. Huang, A. Maier, and J. Hornegger are with the Pattern Recognition Lab, the Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany. (emails: huangxl06@mails.tsinghua.edu.cn, anreas.maier@fau.de, joachim.hornegger@fau.de) J.A.K. Suykens is with the Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, B-3001 Leuven, Belgium. (e-mail: johan.suykens@esat.kuleuven.be) S. Wang is with the Department of Automation, Tsinghua University, 10084 Beijing, P.R. China. (email: swang@mail.tsinghua.edu.cn).



(a)



(b)

Fig. 1. Classification boundary for dataset "cosexp", of which the points in one class are marked by green stars and points in the other class by red crosses: (a) the RBF kernel with $\sigma = 1$ has a good and flat boundary on the right part but it does not have good classification performance on the right part; (b) the RBF kernel with $\sigma = 0.2$ distinguishes the right part well but the corresponding boundary on the left part is not as smooth as we expect, due to the fact that a small σ is used.

from the additive kernel is separable, it illustrates promising performance in some computer vision tasks, showing the potential advantages of PWL kernels.

The flexibility on the non-linearity of PWL kernels and the good performance of the additive kernel both motivate us to consider non-separable PWL kernels for nonlinear classification. Specifically, we will use the ℓ_1-norm for distance measurement and apply a truncation technique to construct a PWL kernel. Hence, it is named a *truncated ℓ_1-distance (TL1)* kernel. It has some similarity with the RBF kernel and enjoys piecewise linearity. In numerical experiments, the TL1 kernel will show surprisingly good performance. Due to the adaptiveness to nonlinearity, the TL1 kernel is not sensitive to its parameter. Thus, we can give a parameter value for the TL1 kernle without cross-validation and achieve similar or even better performance than the RBF kernel with the parameter chosen by cross-validation.

Besides the TL1 kernel, some other PWL kernels will be designed and discussed as well. One major problem is that except of the simplest ones, like the additive kernel, PWL kernels do not satisfy Mercer's condition, since the corresponding kernel matrices are not positive semi-definite (PSD). In theory, lack of PSD makes classical analysis on Reproducing Kernel Hilbert Space (RKHS), see, e.g. [21]–[23], not applicable. In practice, we need to carefully investigate the training algorithms, since many models for PSD kernel learning may loose its effectiveness when an indefinite kernel is used. People have noticed that linear combination of PSD kernels may result in indefinite kernels and the hyperbolic tangent kernel $\mathcal{K}(\mathbf{u}, \mathbf{v}) = \tanh\{\gamma \mathbf{u}^T \mathbf{v} + r\}$ becomes non-PSD when $r < 0$, which tends to perform better [24]. Even when a PSD kernel is used, the kernel matrix may have negative eigenvalues caused by noise on matrix components [25]. Those observations motivate researchers to consider indefinite learning. There have been interesting attempts for both theoretical analysis and training algorithms. The readers are referred to [26]–[29] for a theoretical discussion and [30]–[32] for algorithms. We in this paper focus on suitable algorithms for the TL1 kernel and leave the discussion on its learning behavior for further study.

This paper is organized as follows: in Section II, the TL1 kernel and other PWL kernels are investigated. The training algorithms are discussed in Section III. Section IV evaluates the proposed TL1 kernel by numerical experiments. Specifically, the TL1 kernel will be compared with the RBF kernel. Section V ends the paper with conclusions.

II. Truncated ℓ_1 Distance Kernel

A. Piecewise linear classifiers

In a binary classification problem, we are given a set of training data $\mathcal{Z} = \{\mathbf{x}_i, y_i\}_{i=1}^m$ with $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i = \{-1, +1\}$. Learning from the training data, we partition the input space into two parts: points in one part are considered to be class *I* and the others belong to class *II*. The simplest decision boundary is a hyperplane, which is given by a linear equation $f(\mathbf{x}) = 0$ where $f(\mathbf{x})$ is an affine function. The first step of extending a hyperplane to non-linear surfaces is to use different hyperplanes in different subregions, which results in a piecewise linear classifier. This PWL classifier can be given as the solution of $f(\mathbf{x}) = 0$ with a PWL function $f(\mathbf{x})$. If we further require the decision boundary being continuous, $f(\mathbf{x})$ needs to be a continuous PWL function.

According to its definition, a PWL function can be given by the subregions and the linear functions therein. However, for the convenience of analysis and application, researchers prefer to have compact representations for continuous PWL functions. These compact representations are built based on the fact that the space of continuous PWL functions is closed under (finite) composition. In other words, if g and h are continuous PWL, so is $g \circ h$. This property holds for linear functions and is not necessarily true for other nonlinear functions.

With the composition closure property, many PWL functions can be constructed from the basic "maximum" operator $\mathbb{R}^2 \to \mathbb{R} : \max\{a, b\}$. For example, [8] established the *hinging hyperplane (HH)* model to represent a continuous PWL function:

$$f(\mathbf{x}) = \mathbf{a}_0^T \mathbf{x} + b_0 + \sum_{i=1}^{m} \alpha_i \max\left\{\mathbf{a}_i^T \mathbf{x} + b_i, 0\right\}, \qquad (1)$$

which has been proved to have general approximation capability. The depth of the nested maximum operator measures the complexity of a PWL model. HH is one of the simplest model that is one-level deep. In [10], it has been proved that with a

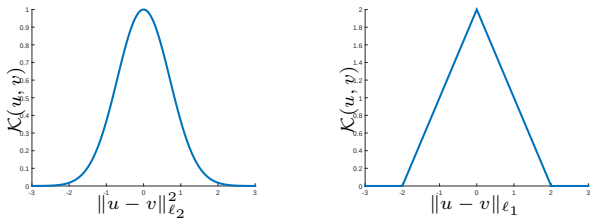
proper formulation, any continuous piecewise linear function defined on $\mathbb{R}^n$ can be represented by at most $\lceil \log(n)/\log(2) \rceil$-level depth, where $\lceil a \rceil$ stands for the smallest integer larger than or equal to a. HH and other PWL representation models illustrate good performance in function regression and system identification tasks. It has also been applied to construct PWL classifiers in [16]. The main drawback of (1) is that the subregion is given independently from samples and is hard to tune, since in (1) the subregions are determined by nonlinear parameters $\mathbf{a}_i$ and b_i.

B. TL1 kernel

In order to adaptively find subregions from the training data, we prefer to use kernel learning methods, which require a suitable PWL kernel. For this aim, the RBF kernel, the currently most popular nonlinear kernel, is first investigated. A RBF kernel

$$\mathcal{K}(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|_{\ell_2}^2}{\sigma^2}\right)$$

is a composition of two operators: i) $\|\mathbf{u}-\mathbf{v}\|_{\ell_2}$ measures the distance between $\mathbf{u}$ and $\mathbf{v}$; ii) $\exp(-a^2/\sigma^2)$ normalizes the distance to a bounded value. Following this idea, we use two piecewise linear functions to establish a PWL kernel: the ℓ_1-norm to measure the distance and a triangle function to normalize the distance. The shape of the triangle function is plotted in Fig.2, which illustrates the similarity of the RBF kernel and the proposed one.



(a) (b)

Fig. 2. (a) exponent in the RBF kernel; (b) triangle operator for the TL1 kernel.

The formulation of this kernel is given by

$$\mathcal{K}(\mathbf{u}, \mathbf{v}) = \max\{\rho - \|\mathbf{u}-\mathbf{v}\|_{\ell_1}, 0\}. \tag{2}$$

Since $\max\{a, 0\}$ is a truncation operator, (2) is named *truncated ℓ_1 distance (TL1)* kernel. Obviously, the truncation operator and the ℓ_1-norm distance are both piecewise linear, their composition hence gives a PWL function. Notice that both the truncation operator and the ℓ_1-norm distance can be represented by one-level maximum operator, the TL1 kernel (2) is a two-level deep piecewise model.

In (2), there is a kernel parameter ρ, which is the truncation threshold: when the ℓ_1 distance between $\mathbf{u}$ and $\mathbf{v}$ is larger than ρ, the corresponding kernel value is set zero. It is similar to σ in the RBF kernel, which controls the kernel width. Throughout this paper, we assume that each component of the training data is in the range of $[0, 1]$. If not, one can easily normalize the training data into $[0, 1]^n$. Then the maximum value of $\|\mathbf{u}-\mathbf{v}\|_{\ell_1}$ is n, the dimension of the input space, and the minimum value is zero. Thus a reasonable value of ρ is between $[0, n]$. ρ also controls the sparsity of the kernel matrix. Since the kernel vanishes when the ℓ_1-norm distance excesses ρ, it belongs to compactly supported kernels, some of which have been investigated and tested by [11] [12]. In fact, the formulation for the TL1 kernel has appeared in [13] and [11] (however, as a counterexample: in a space higher than one dimension, (2) is not a PSD kernel). The lack of positive definiteness prevented researchers to seriously investigate the TL1 kernel and evaluate its performance.

Recall the formulation of the TL1 kernel (2). When a support vector $\mathbf{x}_i$ is given, $\mathcal{K}(\mathbf{x}_i, \mathbf{x})$ partitions the domain into several subregions, in each of which it performs linearly. To give an intuitive impression of the TL1 kernel, its shape in two-dimensional space is shown in Fig.3, where the support vector is $\mathbf{x}_i = [0.5, 0.5]^T$ and $\rho = 0.5$. From (2) also Fig.3, we know that in n-dimensional space, the TL1 kernel $\mathcal{K}(\mathbf{x}_i, \mathbf{x})$ brings 2^n simplices, where $\mathcal{K}(\mathbf{x}_i, \mathbf{x})$ is linear. The vertices of those simplices are $\mathbf{x}_i$ and n other points taken as follows: the j-th point takes value $\mathbf{x}_i + \rho\mathbf{e}_j$ or $\mathbf{x}_i - \rho\mathbf{e}_j$, where $\mathbf{e}_j$ denotes a n-dimensional vector with the j-th component being one and others zero.

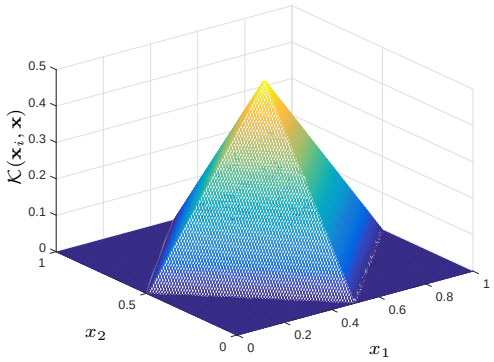
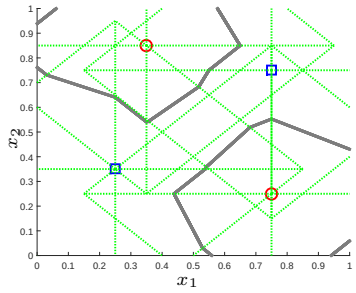


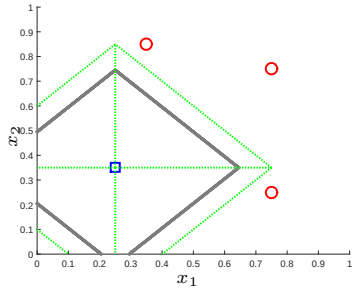
Fig. 3. Shape of the TL1 kernel in two dimensional space. The support vector in this figure is $[0.5, 0.5]^T$ and the kernel parameter is $\rho = 0.5$.

In Fig.4(a), four support vectors (blue squares and red circles stand for two classes) are given and the corresponding potential subregion boundaries are shown by green dotted lines. The subregion structure can result in a very flexible decision boundary. In two-dimensional space, the boundary consists of a series of segments. One possible boundary is plotted by solid line in Fig.4(a). This boundary is linear in each subregion and has tuning points on the junction of subregions. When the problem is simple, e.g., in Fig.4(b), the top-right support vector has the same label as the two nearby ones, then training procedure may give a sparse result that there is only one support vector, i.e., the left-bottom point. Then the subregion structure and the potential classification boundary become simple, as shown in Fig.4(b).

The above explanation shows the flexibility of the classifier induced by the TL1 kernel. Considering the "cosexp" dataset shown in Fig.1, we now use the TL1 kernel and LP-SVM (see Section III.D) to train a classifier. The classifier and the support



(a)



(b)

Fig. 4. Support vectors, marked by red circles and blue squares for two classes, the potential subregion boundaries, shown by dotted lines, and a randomly generated classification curve, shown by black solid line. (a) when there are four support vectors, the classifier is very flexible and can easily solve this XOR problem; (b) if we change the label of the right-top point, the result could be more sparse. If there is only one support vector, the left-bottom point, the subregion structure is simple and the corresponding classifier performs linearly in a relatively large region.

vectors are displayed in Fig.5, from which one can find the mechanism of non-linearity adjustment: on the left part, there are a few support vectors and the classification curve is flat; on the right part, the classifier is heavily non-linear due to the existence of many support vectors.

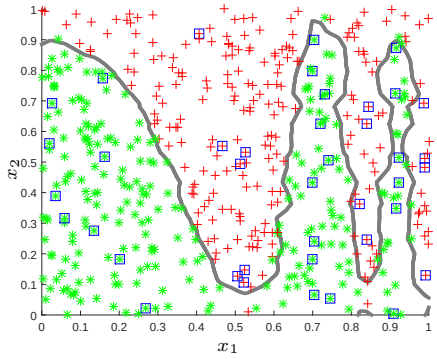


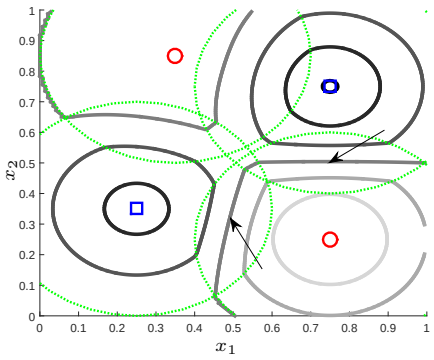
Fig. 5. Classification result for "cosexp" using the TL1 kernel. On the left part, the classification boundary is flat and on the right part it has heavy non-linearity. Non-linearity is adjusted by the support vectors, marked by squares: on the left part, there are a few support vectors; on the right part, the density of support vectors is high.

The triangle operator is a common way to obtain locally supported kernels or basis functions. It is closely related to 1-order B-splines [33]. In nonparametric modeling, researchers have considered the following kernel,

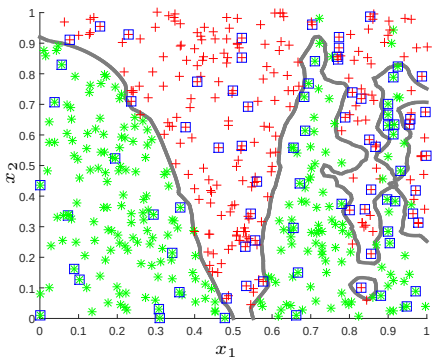
$$\mathcal{K}(\mathbf{u},\mathbf{v}) = \max\{\rho - \|\mathbf{u}-\mathbf{v}\|_{\ell_2}^p, 0\}, \tag{3}$$

where $0 < p \leq 2$. Particularly, (3) is called *Epanechnikov kernel* when $p = 2$ and *triangular kernel* when $p = 1$; see [34]–[36] for the discussion on classification. The difference between the TL1 and the triangular kernel is that different norms are used to measure the distance, which results in different subregion structure and different behavior in subregions. As discussed previously, the TL1 kernel has polygonal subregions and performs linearly in subregion. For the triangular kernel, one subregion is the intersection of several balls with support vectors as the centers. Moreover, in each subregion, the classifier still has non-linearity, which is determined by the distance to support vectors effective in this subregion.

In Fig.6(a), the subregion structure is plotted, together with several counters of the classification function, which are the potential classification boundaries due to different bias terms. Similarly to the TL1 kernel, the classification curve for the triangular kernel also has non-differentiable points on the subregion boundary. But only when the effect of support vectors counteract each other, the classification curve becomes flat, as the segments marked by arrows. In this viewpoint, the triangular kernel is similar to the RBF kernel that they need many support vectors to fit a flat classification boundary, as illustrated by Fig.6(b).



(a)



(b)

Fig. 6. (a) Subregion structure and potential classification boundaries for the triangular kernel; the legends follow those in Fig.4; (b) classification result for "cosexp" using the triangular kernel; the legends follow those in Fig.5.

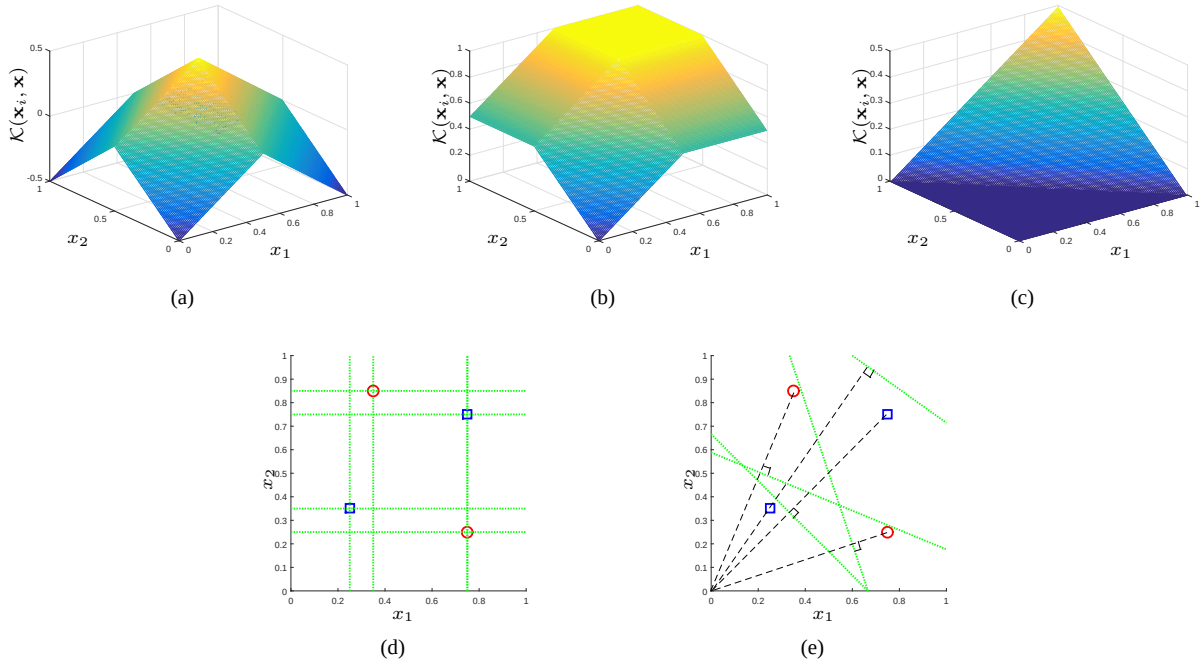


Fig. 7. Shapes of PWL kernels (the support vector is $[0.5, 0.5]^T$) and the potential subregion boundaries, shown by dotted line, generated by four support vectors, shown by red circles and blue squares. (a) shape of the additive kernel; (b) shape of the L1 kernel; (c) shape of the TLinear kernel; (d) the additive kernel and the L1 kernel share the same subregion structure; (e) the normal vectors for subregion boundaries of the the TLinear kernel are determined by support vectors.

C. Other piecewise linear kernels

Using a composition of two maximum operators, we got the TL1 kernel in the last subsection. Following the same way of composition of basic PWL operators, one can design other piecewise linear kernels. For example, with the ℓ_1-norm, an ℓ_1 *distance (L1)* kernel is established. It takes the formulation as below,

$$\mathcal{K}(\mathbf{u}, \mathbf{v}) = -\|\mathbf{u} - \mathbf{v}\|_{\ell_1}.$$

This kernel is equal to the TL1 kernel with a large enough ρ. The shape of the L1 kernel is shown in Fig.7(a) and its subregion structure shown in Fig.7(d). Compared to the TL1 kernel, the L1 kernel uses only one-level maximum operator and its structure is simpler. We will also see later in the numerical example that the L1 kernel is too simple to handle complicated problems.

Besides based on the ℓ_1-norm distance, piecewise linearity can also be introduced from the linear kernel

$$\mathcal{K}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T\mathbf{v} = \sum_{i=1}^{n} u_i v_i.$$

In many piecewise linear models, the minimum operator is always regarded as a PWL approximation of the multiplication, see, e.g., adaptive hinging hyperplane [37]. Following this idea, we replace $u_i v_i$ by $\min\{u_i, v_i\}$, and reformulate the linear kernel to a PWL one:

$$\mathcal{K}(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{n} \min\{u_i, v_i\},$$

which is actually the *additive kernel* given by [19] and has been successfully applied in computer vision [20]. In Fig.7(b), we display the shape of the additive kernel. Its subregions structure is the same as the L1 kernel shown by Fig.7(d). As claimed by [20], the runtime and memory complexity of the additive kernel is independent of the number of support vectors, which is one advantage of linear classifiers over nonlinear ones. Since the classifiers induced from the additive kernel are separable, i.e., they can be written as the sum of several univariate PWL functions, its performance on problems with strongly coupled variables is not good.

Another way to design a PWL kernel based on the linear kernel is to use truncation on $\mathbf{u}^T\mathbf{v}$, resulting in the following *truncated linear (TLinear)* kernel,

$$\mathcal{K}(\mathbf{u}, \mathbf{v}) = \max\{\mathbf{u}^T\mathbf{v} - \rho, 0\}.$$

Similarly to the TL1 kernel, ρ in the TLinear kernel also takes value from $[0, n]$. The subregion boundary corresponding to a support vector $\mathbf{x}_i$ is given by a linear equation $\mathbf{x}_i^T\mathbf{x} - \rho = 0$. Unlike other PWL kernels, for the TLinear kernel, a support vector and the corresponding subregion boundary is not locally linked. Instead, the support vector determines the normal vector of the subregion boundary, as shown in Fig.7(e).

In this section, we designed several PWL kernels using one or two-level nested maximum operator. With more nested levels of the maximum operators, PWL kernels will have more complicated subregion structures and then have more flexibility. Its training can draw the experience of deep convolutional network, see, e.g., [38] where the maximum operator is used as the basic computational unit. In this paper, we first focus on these one or two-level PWL kernels. If the performance is promising, it is then interesting to go deep to nested PWL kernels and train them as convolutional networks.

III. Indefinite Learning Algorithms

A. Positive semi-definiteness

In the preceding section, we gave several PWL kernels and discussed their subregion structures. Due to the flexibility of the classification boundary and the adaptivity to non-linearity, the classification performance of PWL kernels, especially the TL1 kernel, could be expected. However, PWL kernels may lead to an indefinite kernel matrix. In this case, they do not satisfy Mercer's condition and the training algorithms should be carefully discussed. To investigate the definiteness, we calculate the eigenvalues of the matrix induced by PWL kernels for "cosexp" dataset and plot the largest and smallest ones in Fig.8. As shown in this figure, the additive kernel and the L1 kernel are PSD (the definiteness of the additive kernel has been discussed in [20]; that of the L1 kernel can be analyzed similarly). However, the TL1 and the TLinear kernel are not PSD when dimension of the input data is higher than 1.

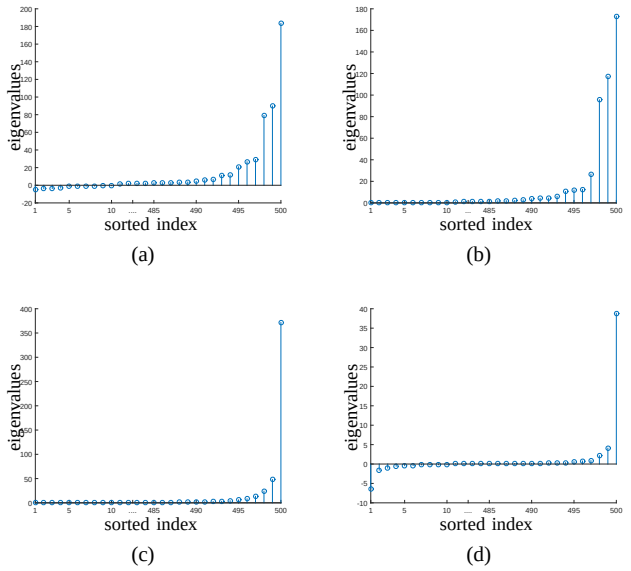


Fig. 8. Eigenvalues of the kernel matrix for "cosexp" (the indices are sorted by the eigenvalues): (a) the TL1 kernel; (b) the additive kernel; (c) the L1 kernel; (d) the TLinear kernel.

For PSD kernels, there are many learning algorithms applicable. For indefinite kernels, researchers need to carefully investigate the training algorithms. One important issue is that the functional space induced from an indefinite kernel is not a Reproducing Kernel Hilbert Space (RKHS). Instead, it is a Reproducing Kreĭn Spaces (RKKS) and the readers are referred to [29] and references therein for detailed discussion. As proved in [26], a general representer theorem is still true and a regularized risk can be defined in RKKS. Specifically, a classifier, which is generated by an indefinite kernel and stabilizes one regularized risk, also has the following expansion:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i \mathcal{K}(\mathbf{x}_i, \mathbf{x}) + \alpha_0.$$

Therefore, when the training data are given, learning for TL1 and the TLinear kernel still can be formulated as a finite-dimensional optimization problem on α_i.

B. Indefinite learning methods

According to the representer theorems for RKHS and RKKS, learning with PSD or indefinite kernels can be formulated as the following optimization problem:

$$\min_{f \in \mathcal{S}_{\mathcal{K},\mathcal{Z}}, \alpha_0 \in \mathbb{R}} \quad \frac{1}{2}\Omega(f) + C\sum_{i=1}^{m} L(1 - y_i(f(\mathbf{x}_i) + \alpha_0)), \quad (4)$$

where $C > 0$ is the trade-off coefficient between the regularization term and the loss function, and

$$\mathcal{S}_{\mathcal{K},\mathcal{Z}} = \left\{ f : f(\mathbf{x}) = \sum\nolimits_{i=1}^{m} \alpha_i y_i \mathcal{K}(\mathbf{x}, \mathbf{x}_i) \right\}$$

is a finite-dimensional space spanned by kernel $\mathcal{K}$ and training data $\mathcal{Z}$.

If $\mathcal{K}$ satisfies Mercer's condition, there is a nonlinear feature mapping $\phi(\mathbf{x})$, of which the outputs could be in infinite dimensional space, such that $\mathcal{K}(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u})^T \phi(\mathbf{v})$. In this case, one can choose a norm induced by $\mathcal{K}$ [39] as the regularization term, choose the hinge function as the loss, and then formulate (4) as the following support vector machine (C-SVM, [18]):

$$\min_{\mathbf{w},\alpha_0} \quad \frac{1}{2}\|\mathbf{w}\|_{\ell_2}^2 + C\sum_{i=1}^{m} \max\left\{1 - y_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + \alpha_0\right), 0\right\}. \quad (5)$$

Its dual problem is

$$\begin{aligned} \min_{\alpha_i} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{m} \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} y_i \alpha_i = 0, \\ & 0 \le \alpha_i \le C, \forall i = 1, 2, \ldots, m. \end{aligned} \quad (6)$$

There have been many algorithms to solve (6). For instance, when the sampling data size is not very huge, the sequential minimization optimization (SMO) developed by [40] [41] can efficiently find the optimal dual variables for (6).

However, if indefinite kernels, such as the TL1 kernel and the TLinear kernel, are used, we cannot find the corresponding feature mapping, i.e., there is no kernel trick, then (5) and (6) are not a pair of dual-primal problems. Moreover, (6) is no longer convex. An alternative way is to find an approximate PSD kernel $\tilde{\mathcal{K}}$ for an indefinite one $\mathcal{K}$, and then solve (6) for $\tilde{\mathcal{K}}$. In [25], it was proposed to set all negative eigenvalues to zero to obtain $\tilde{\mathcal{K}}$. Similarly, one can flip signs of the negative eigenvalues, as discussed in [42]. An optimization problem was introduced by [30] to find the nearest positive semi-definite kernel by minimizing the Frobenius distance of the two kernel matrices. Since training and classification are based on two different kernels, the above methods are efficient only when $\mathcal{K}$ and $\tilde{\mathcal{K}}$ are similar. To overcome the inconsistency, an eigen-decomposition support vector machine was established by [29]. This method was reported to outperform the other PSD approximate methods.

The above methods which transfer the non-PSD kernels to PSD ones have been reviewed and compared by [43]. That review also discusses another type of indefinite learning methods which directly use non-PSD kernels in the methods that

are insensitive to metric violations. For example, in the dual problem (6), we can directly use a non-PSD $\mathcal{K}$ as suggested by [24]. Then there are two crucial problems arising. First, there is no feature mapping ϕ such that $\mathcal{K}(\mathbf{u},\mathbf{v}) = \phi(\mathbf{u})^T\phi(\mathbf{v})$, from which it follows that C-SVM with a non-PSD kernel cannot be explained as maximizing the margin in a feature space. Instead, it is to minimize the distance between two convex hulls in pseudo-Euclidean space. The reader is referred to [26] and [44] for this important interpretation. The second problem is that a non-PSD kernel makes (6) non-convex, and many algorithms cannot be used if they rely on global optimality. Fortunately, descent algorithms, such as SMO, are still applicable and then a stationary point can be achieved efficiently.

Comparing the above two categories of indefinite learning methods, one can find that transferring a non-PSD kernel to an approximate PSD one takes significantly longer time than directly training it by local minimization methods. Thus, for the aim of practical use, we choose SMO to solve (6) for the proposed PWL kernels. This is also a fair comparison with the RBF kernel, which is commonly trained by (6) with SMO. Notice that when $\mathcal{K}$ in (6) is not PSD, the result of SMO differs for different starting points. Since sparsity is always our aim, we choose the zero vector as the initial solution in this paper.

The approach of directly using a non-PSD kernel is also applicable to the least squares support vector machine (LS-SVM, [45][46]), another popular kernel learning method. In the dual space, LS-SVM is to solve the following linear system:

$$\left[\begin{array}{cc} 0 & \mathbf{y}^T \\ \mathbf{y} & \mathcal{H}+\frac{1}{C}\mathcal{I} \end{array}\right][\alpha_0,\alpha_1,\ldots,\alpha_m]^T = \left[\begin{array}{c} 0 \\ \mathbf{1} \end{array}\right], \qquad (7)$$

where $\mathcal{I}$ is an identity matrix, $\mathbf{1}$ is an all ones vector with the proper dimension, and $\mathcal{H}$ is given by

$$\mathcal{H}_{ij} = y_i y_j \mathcal{K}(\mathbf{x}_i,\mathbf{x}_j).$$

Similarly to the discussion about C-SVM, we directly use PWL kernels in (7) as well. Notice that if a non-PSD kernel is used in LS-SVM, the classical feature space interpretation and primal-dual relationship are not valid anymore. Different to C-SVM (6), where a non-PSD kernel makes that problem non-convex, using a non-PSD kernel in LS-SVM (7) is still easy to solve.

One drawback of (7) is that its results lack of sparsity. As explained previously, PWL kernels are motivated by the fact that its linearity in different areas can be adaptively tuned. We expect that for the flat part of the ideal classifier, a PWL classifier can adaptively perform linearly, which requires sparsity on α. To pursue sparsity, the ℓ_1-norm and the hinge loss are chosen in the framework of (4), resulting the linear programming support vector machine (LP-SVM, [47]):

$$\begin{aligned}\min_{\alpha_i} \quad & \frac{1}{2}\sum_{i=1}^{m}|\alpha_i| + C\sum\nolimits_{i=1}^{m}\max\Big\{0, \\ & 1 - y_i\Big(\sum\nolimits_{j=1}^{m} y_j\alpha_j\mathcal{K}(\mathbf{x}_i,\mathbf{x}_j) + \alpha_0\Big)\Big\}. \qquad (8)\end{aligned}$$

The analysis of LP-SVM on learning behavior and the corresponding algorithms can be found in [48]–[50]. The existing discussion on LP-SVM is mainly for RKHS. For RKKS, the analysis for LP-SVM will be of interest.

IV. Numerical Experiments

In the previous sections, we discussed several PWL kernels, including the TL1, the TLinear, the L1, the additive kernel, and three training algorithms, including C-SVM, LS-SVM, and LP-SVM. In this section, we will first investigate the suitable algorithms for different PWL kernels. After that, the best PWL kernel will be compared to the RBF kernel. Most of the data are downloaded from UCI Repository of Machine Learning Datasets [51]. The rest come from libsvm Datasets [52]. For some problems, both the training and test sets are provided. Otherwise, we randomly pick half of the data as the training set and use the remaining for test. The classification accuracy reported below is the average test accuracy of 10 trials. We also care about the number of support vectors, which means that the absolute value of α_i is larger than 10^{-3}. For the TLinear, the TL1, and the RBF kernel, there are kernel parameters. Besides, there are regularization coefficients in the learning algorithms. All those parameters are tuned by 10-fold cross-validation based on misclassification. As discussed before, the reasonable range of ρ in both TLinear and the TL1 kernel is between 0 and n, the value candidate set is hence $\{0.1n, 0.3n, 0.5n, 0.7n, 0.9n\}$. For C-SVM, LS-SVM, and LP-SVM, the regularization coefficients are chosen from $\{0.01, 0.1, 0.5, 1, 2, 10\}$.

We first use some small datasets to test the mentioned PWL kernels with different learning algorithms. The results are reported in Table I, which shows that simply introducing piecewise linearity to the linear kernel can significantly improve the classification performance. Among the tested PWL kernels, the TL1 kernel performs the best, which is mainly due to that the TL1 kernel provides more flexibility than the others. The flexibility can be roughly observed from the subregion structure. As shown by Fig.7(d), the L1 and the additive kernel share the same potential subregion boundaries, so they have similar classification performance.

TABLE I
Performance of Piecewise Linear Kernels: Accuracy on Test Data and Number of Support Vectors

dataset	kernel	C-SVM	LS-SVM	LP-SVM
Spect.	Linear.	74.33% (53)	73.26% (80)	70.59% (3)
	Add.	77.01% (43)	73.26% (80)	77.01% (10)
	L1	72.19% (54)	73.80% (80)	79.14% (16)
	TLinear	66.31% (46)	64.61% (80)	67.38% (15)
	TL1	84.49% (68)	84.49% (80)	86.63% (28)
Iono	Linear	88.15% (52)	84.04% (175)	87.02% (18)
	Add.	91.52% (69)	91.97% (170)	91.40% (13)
	L1	90.96% (86)	91.91% (170)	91.91% (18)
	TLinear	87.98% (76)	86.80% (173)	88.71% (21)
	TL1	92.98% (97)	91.97% (170)	95.06% (31)
Monk1	Linear	66.90% (90)	66.20% (124)	67.13% (6)
	Add.	73.61% (95)	74.07% (124)	75.00% (9)
	L1	75.00% (79)	74.07% (124)	75.00% (7)
	TLinear	66.20% (97)	67.54% (124)	76.85% (28)
	TL1	79.63% (51)	85.19% (124)	83.33% (36)

TABLE II
COMPARISON BETWEEN TL1 KERNEL AND RBF KERNEL: ACCURACY ON TEST DATA AND NUMBER OF SUPPORT VECTORS

dataset	m	RBF C-SVM	RBF LS-SVM	TL1 C-SVM	TL1 LS-SVM	TL1 LP-SVM
DBWords-Sub	32	84.81% (31)	85.01% (32)	85.75% (30)	85.01% (32)	84.38% (20)
Fertility	50	88.56% (49)	88.45% (50)	88.84% (19)	89.43% (50)	88.25% (3)
Planning	91	70.76% (66)	71.48% (91)	71.09% (54)	72.03% (91)	72.23% (31)
Sonar	104	83.24% (72)	83.33% (104)	83.04% (74)	83.00% (103)	82.11% (51)
Statlog	135	82.96% (71)	82.76% (135)	83.09% (76)	83.53% (135)	83.32% (10)
Monk2	169	86.34% (102)	87.59% (169)	86.57% (130)	86.57% (169)	86.57% (69)
Monk3	122	93.59% (42)	93.74% (122)	97.22% (37)	97.22% (122)	96.30% (20)
Climate	270	94.28% (113)	93.25% (270)	93.82% (67)	92.07% (270)	93.66% (46)
Liver	292	72.59% (289)	71.38% (292)	72.66% (290)	72.02% (292)	73.13% (80)
Austr.	345	84.75% (133)	85.08% (345)	86.07% (117)	85.90% (345)	85.62% (65)
Breast	349	96.58% (46)	96.30% (349)	96.87% (83)	96.87% (349)	96.38% (16)
Trans.	374	76.53% (193)	77.60% (374)	77.87% (193)	76.91% (374)	75.53% (9)

Since the TL1 kernel outperforms other PWL kernels in general, the following experiments will focus on the TL1 kernel and compare it with the RBF kernel. As the most widely used nonlinear kernel in classification, there have been efficient and popular toolboxes for the RBF kernel, e.g., libsvm [52] for C-SVM (6) and LS-SVMlab [53] for LS-SVM (7). The kernel parameter σ in the RBF kernel is crucial to its performance and it can be tuned by cross-validation. In this paper, the automatic parameter selection provided by [54] is used to find suitable parameters for libsvm and the default tuning process is used for LS-SVMlab. In Table II, the classification accuracy on test data and the number of support vectors are reported, and the highest accuracy for each dataset is underlined.

Comparing the RBF and the TL1 kernel with the same training algorithm, we find that the TL1 kernel gives at least a comparable result for each dataset, which confirms the universality of the TL1 kernel. Moreover, on some datasets, like "Monk3" and "Austr.", the performance is improved with respect to the RBF kernel. For the TL1 kernel, the three algorithms all output good results. Especially, learning the TL1 kernel by LP-SVM is attractive, which uses relatively less support vectors and achieves similar accuracy, but its computation time is relatively long. Notice that the TL1 kernel is non-PSD, then (6) is not a convex optimization problem. Nevertheless, we can use SMO to efficiently obtain a stationary point and the performance is satisfying.

The motivation of using a PWL kernel is to adaptively fit different non-linearity in different areas. This also implies that the classification performance is not sensitive to the value of ρ, then one parameter value is suitable for different non-linearity. Thus, for different problems, we can use a pre-given parameter without tuning process. In order to verify this property, classification accuracy versus parameter value is plotted in Fig.9.

In the considered three datasets, the classification accuracy of the TL1 kernel is similar for a quite large range of ρ values. One can observe that for different datasets, the performance for ρ between $0.6n$ to $0.9n$ is generally stable and setting $\rho = 0.7n$ always leads to a good result. In contrast, the best σ of the RBF kernel for different problems differs a lot: for

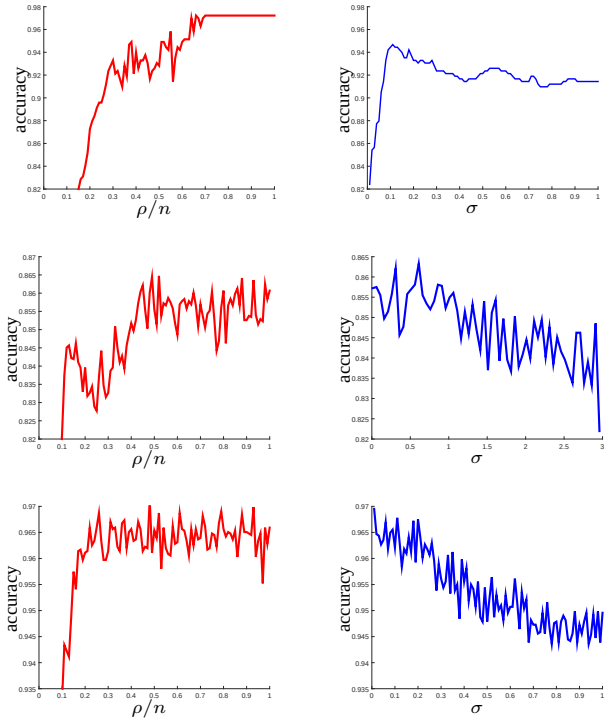


Fig. 9. Classification accuracy on test data versus kernel parameters (left: the TL1 kernel; right: the RBF kernel) for different datasets (top: "Monk3"; middle: "Austr."; bottom: "Breast"). For the considered three datasets, setting $\rho = 0.7n$ for the TL1 kernel can give satisfactory results; while, for the RBF kernel, the suitable value for σ varies a lot for different problems.

"Monk3", the best σ is around 0.15; for "Austr.", it is about 0.7; and for "Breast", it is about 0.02. The stability of ρ in the TL1 kernel allows us to pre-chose it without cross-validation. This is particularly useful when the data size is large. In the following experiments, we fix $\rho = 0.7n$ for the TL1 kernel and compare it with the RBF kernel with parameter tuning process. The previous experiments show that LP-SVM is probably the best training method for the TL1 kernel, considering both the accuracy and sparsity. However, we can hardly use LP-SVM for larger datasets, since it needs much more time than C-SVM/LS-SVM. For the aim of practical applications, we in the

following use C-SVM for the TL1 kernel. The classification performance of the TL1 kernel with $\rho = 0.7n$ is reported in Table III, together with the results of libsvm for the RBF kernel, of which the kernel parameter is tuned by cross-validation.

TABLE III
PERFORMANCE OF TL1 KERNEL WITH SUGGESTED PARAMETER: ACCURACY ON TEST DATA AND NUMBER OF SUPPORT VECTORS

dataset	m	RBF by C-SVM (σ by CV)	TL1 by C-SVM ($\rho = 0.7n$)
Qsar	528	86.92% (209)	86.05% (300)
Splice	1000	89.83% (651)	92.74% (229)
Guide3	1243	84.15% (482)	97.56% (487)
Madelon	2000	58.83% (1852)	61.33% (1845)
Spamb.	2300	93.32% (464)	94.05% (453)
ML-prove	3059	72.48% (1604)	79.08% (1684)
Guide1	3089	96.84% (308)	97.12% (397)
Wilt	4339	85.80% (130)	86.80% (475)
Phish.	5528	95.92% (2136)	93.83% (1657)
Magic	9510	86.48% (3124)	86.04% (4342)
RNA	59535	96.66% (7072)	95.74% (13656)

As reported in Table III, the TL1 kernel shows pretty good performance. For dataset "Splice", "ML-prove", and "Guide3", the classification accuracy is improved with respect to the RBF kernel. For other sets, the performance of the TL1 kernel and the RBF kernel is similar. Notice that in this experiment, the parameter of the RBF kernel is tuned by cross-validation, while, that of the TL1 kernel is set to be $\rho = 0.7n$. With one parameter less to tune, the computation time of the TL1 kernel is largely saved.

Summarizing all the experiments, we can conclude that the TL1 kernel is a promising nonlinear kernel for classification. For small scale problems, LP-SVM is a good choice, which has good performance on both sparsity and accuracy. For larger datasets, C-SVM (solved by SMO) becomes attractive for the great computational efficiency. The TL1 kernel shows great stability to its kernel parameter such that simply setting $\rho = 0.7n$ without cross-validation outputs satisfactory results.

V. CONCLUSION AND FURTHER STUDY

We in this paper discussed a piecewise linear kernel, called truncated ℓ_1 distance kernel, and evaluated it on numerical experiments. Though the TL1 kernel is not PSD, the numerical experiments show that three popular kernel learning methods, including C-SVM, LS-SVM, and LP-SVM, can be directly applied to train classifiers for the TL1 kernel. Compared to the RBF kernel, the TL1 kernel illustrates similar classification performance on most of the considered datasets and for some problems, the accuracy could be significantly improved. Those experiments verify the flexibility of subregion structure and adaptiveness to non-linearity for the TL1 kernel. Moreover, adaptiveness to non-linearity makes the TL1 kernel not sensitive to its kernel parameter. Therefore, without parameter tuning, we can use a pre-given parameter for the TL1 kernel and achieve similar performance as the RBF kernel with parameter tuning.

All the experiments indicate that the TL1 kernel is a promising nonlinear kernel for classification. It can be implemented in popular classification toolboxes, such as libsvm and LS-SVMlab, by simply changing the kernel evaluation function. The good performance motivates us to investigate the properties of the space associated the TL1 kernel in the future. Also, it is interesting to extend the TL1 kernel using nested PWL operators and then train a classifier from convolutional networks.

ACKNOWLEDGMENT

Xiaolin Huang is currently supported by the Humboldt research fellowship for postdoctoral researchers program. Johan Suykens acknowledges support of ERC AdG A-DATADRIVE-B (290923), KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; FWO: G.0377.12, G.088114N, SBO POM (100031); IUAP P7/19 DYSCO. S. Wang is supported by the National Natural Science Foundation of China (Project No. 61134012, 61473165).

REFERENCES

[1] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth & Brooks/Cole Advanced Books & Software, CA, 1984.
[3] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
[4] J. Sklansky and L. Michelotti, "Locally trained piecewise linear classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 2, pp. 101–111, 1980.
[5] A. M. Bagirov, J. Ugon, and D. Webb, "An efficient algorithm for the incremental construction of a piecewise linear classifier," *Information Systems*, vol. 36, no. 4, pp. 782–790, 2011.
[6] K. Gai and C. Zhang, "Learning discriminative piecewise linear models with boundary points," in proceedings of *the 24th AAAI Conference on Artificial Intelligence*, pp. 444–450, 2010.
[7] L. O. Chua and S. M. Kang, "Section-wise piecewise-linear functions: Canonical representation, properties, and applications," *Proceedings of the IEEE*, vol. 65, no. 6, pp. 915–929, 1977.
[8] L. Breiman, "Hinging hyperplanes for regression, classification, and function approximation," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 999–1013, 1993.
[9] P. Julián, "The complete canonical piecewise-linear representation: functional form for minimal degenerate intersections," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 3, pp. 387–396, 2003.
[10] S. Wang and X. Sun, "Generalization of hinging hyperplanes," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4425–4431, 2005.
[11] M. G. Genton, "Classes of kernels for machine learning: a statistics perspective," *Journal of Machine Learning Research*, vol. 2, pp. 299–312, 2002.
[12] B. Hamers, J.A.K. Suykens, and B. De Moor, "Compactly supported RBF kernels for sparsifying the Gram matrix in LS-SVM regression models," in proceedings of *the International Conference on Artificial Neural Networks (ICANN 2002)*, pp. 720–726, 2002
[13] N. Cressie, *Statistics for Spatial Data*. Wiely, New York, 1993.
[14] S. Wang, X. Huang, and K. M. Junaid, "Configuration of continuous piecewise-linear neural networks," *IEEE Transactions on Neural Networks*, vol. 19, no. 8, pp. 1431–1445, 2008.
[15] Y. Li, B. Liu, X. Yang, Y. Fu, and H. Li, "Multiconlitron: a general piecewise linear classifier," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 276–289, 2011.
[16] X. Huang, S. Mehrkanoon, and J.A.K. Suykens, "Support vector machines with piecewise linear feature mapping," *Neurocomputing*, vol. 117, pp. 118–127, 2013.

[17] Y. Li and Q. Leng, "Alternating multiconlitron: a novel framework for piecewise linear classification," *Pattern Recognition*, vol. 48, no. 3, pp. 968–975, 2015.

[18] V. Vapnik, *Statistical Learning Theory*. Wiley, New York, 1998.

[19] 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 (CVPR)*, 2008.

[20] ——, "Efficient classification for additive kernel svms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 66–77, 2013.

[21] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2002.

[22] F. Cucker and D. X. Zhou, *Learning Theory: an Approximation Theory Viewpoint*. Cambridge University Press, 2007, vol. 24.

[23] I. Steinwart and A. Christmann, *Support Vector Machines*. Springer Science & Business Media, 2008.

[24] H.-T. Lin and C.-J. Lin, "A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods." Technical report, Department of Computer Science, National Taiwan University, 2003. http://www.csie.ntu.edu.tw/∼cjlin/papers/tanh.pdf

[25] E. Pekalska, P. Paclik, and R. P. Duin, "A generalized kernel approach to dissimilarity-based classification," *Journal of Machine Learning Research*, vol. 2, pp. 175–211, 2002.

[26] C. S. Ong, X. Mary, S. Canu, and A. J. Smola, "Learning with non-positive kernels," in proceeding of *the 21st International Conference on Machine Learning (ICML)*, pp. 639–646, 2004,

[27] Y. Ying, C. Campbell, and M. Girolami, "Analysis of SVM with indefinite kernels," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds. pp. 2205–2213, 2009.

[28] S. Gu and Y. Guo, "Learning SVM classifiers with indefinite kernels." in proceedings of *the 26th AAAI Conference on Artificial Intelligence*, pp. 942–948, 2012.

[29] G. Loosli, S. Canu, and C. S. Ong, "Learning SVM in Kreĭn spaces." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, doi:10.1109/TPAMI.2015.2477830, 2015.

[30] R. Luss and A. d'Aspremont, "Support vector machine classification with indefinite kernels," in *Advances in Neural Information Processing Systems 20*, J.C. Platt, D. Koller, Y. Singer and S.T. Roweis, Eds. pp. 953–960, 2008.

[31] J. Chen and J. Ye, "Training SVM with indefinite kernels," in proceedings of *the 25th International Conference on Machine Learning (ICML)*. pp. 136–143, 2008.

[32] I. Alabdulmohsin, X. Gao, and X. Z. Zhang, "Support vector machines with indefinite kernels," in proceedings of *the 6th Asian Conference on Machine Learning*, pp. 32–47, 2014.

[33] P. H. Eilers and B. D. Marx, "Flexible smoothing with B-splines and penalties," *Statistical Science*, pp. 89–102, 1996.

[34] F. Fleuret and H. Sahbi, "Scale-invariance of support vector machines based on the triangular kernel," in proccedings of *the 3rd International Workshop on Statistical and Computational Theories of Vision*, pp. 1–13, 2003.

[35] H. Sahbi and F. Fleuret, "Kernel methods and scale invariance using the triangular kernel," Research Report, RR-5143, 2004.

[36] L. V. Utkin, Y. A. Zhuk, and A. I. Chekh, "A robust one-class classification model with interval-valued data based on belief functions and minimax strategy," in *Machine Learning and Data Mining in Pattern Recognition*, pp. 107–118, Springer, 2014.

[37] J. Xu, X. Huang, and S. Wang, "Adaptive hinging hyperplanes and its applications in dynamic system identification," *Automatica*, vol. 45, no. 10, pp. 2325–2332, 2009.

[38] F. Anselmi, L. Rosasco, C. Tan, and T. Poggio, "Deep convolutional networks are hierarchical kernel machines," *arXiv preprint arXiv:1508.01084*, 2015.

[39] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, pp. 337–404, 1950.

[40] J. Platt, "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods - Support Vector Learning*, MIT press, pp. 185-208, 1999.

[41] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training support vector machines," *Journal of Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.

[42] V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann, "Optimal cluster preserving embedding of nonmetric proximity data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1540–1551, 2003.

[43] F. Schleif and P. Tino, "Indefinite proximity learning: a review," *Neural Computation*, vol. 27, pp. 2039–2096, 2015.

[44] B. Haasdonk, "Feature space interpretation of SVMs with indefinite kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 482–492, 2005.

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

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

[47] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-norm support vector machines," *Advances in Neural Information Processing Systems 16*, S. Thrun, L.K. Saul and B. Schölkopf, Eds. pp. 49–56, 2004.

[48] G. M. Fung and O. L. Mangasarian, "A feature selection Newton method for support vector machine classification," *Computational Optimization and Applications*, vol. 28, no. 2, pp. 185–202, 2004.

[49] Q. Wu and D.-X. Zhou, "SVM soft margin classifiers: linear programming versus quadratic programming," *Neural Computation*, vol. 17, no. 5, pp. 1160–1187, 2005.

[50] O. L. Mangasarian, "Exact 1-norm support vector machines via unconstrained convex differentiable minimization," *Journal of Machine Learning Research*, vol. 7, pp. 1517–1530, 2006.

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

[52] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, p. 27, 2011.

[53] K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle, and J.A.K. Suykens, "LS-SVMlab toolbox user's guide, version 1.8" *ESAT-SISTA Technical Report 10-146*, 2010.

[54] K. Kampa, "Automatic parameter selection". Integrated Brain Imaging Center, UW Medical Center, Seattle, UW. http://www.csie.ntu.edu.tw/∼cjlin/libsvm/faq.html#/Q9:_MATLAB_interface