MLTSVM: A novel twin support vector machine to multi-label learning

Wei-Jie Chen [a], Yuan-Hai Shao [a,*], Chun-Na Li [a], Nai-Yang Deng [b]

[a] Zhijiang College, Zhejiang University of Technology, Hangzhou 310024, PR China
[b] College of Science, China Agricultural University, Beijing 100083, PR China

ABSTRACT

Multi-label learning paradigm, which aims at dealing with data associated with potential multiple labels, has attracted a great deal of attention in machine intelligent community. In this paper, we propose a novel multi-label twin support vector machine (MLTSVM) for multi-label classification. MLTSVM determines multiple nonparallel hyperplanes to capture the multi-label information embedded in data, which is a useful promotion of twin support vector machine (TWSVM) for multi-label classification. To speed up the training procedure, an efficient successive overrelaxation (SOR) algorithm is developed for solving the involved quadratic programming problems (QPPs) in MLTSVM. Extensive experimental results on both synthetic and real-world multi-label datasets confirm the feasibility and effectiveness of the proposed MLTSVM.

1. Introduction

Support vector machine (SVM) [1–3], being recognized as one of the most useful kernel-based tool for data classification and regression, is successfully and widely used in a variety of real-world problems [4–7]. To implement the principle of structural risk minimization (SRM), SVM constructs two parallel optimal support hyperplanes that maximize the margin between two classes. However, SVM needs to solve a quadratic programming problem (QPP) [3], whose computational complexity is $\mathcal{O}(m^3)$, where m is the scale of the dataset. The time-consuming training stage of SVM may restrict its application to many real-world problems. Consequently, many improved techniques have been proposed, e.g. SMO [8], LIBSVM [9], LSSVM [10] and CVM [11].

On the other hand, some variants based on the idea of constructing nonparallel hyperplanes have emerged in recent years [12–22]. Specifically, as a great improvement of SVM, twin support vector machine (TWSVM) [12] has been studied extensively. It relaxes the parallel requirement in traditional SVM, and attempts to generate two nonparallel hyperplanes such that each one is closer to its own class and as far as possible from the other. For this purpose, it solves a pair of relatively smaller QPPs, instead of a large one in classical SVM. Therefore, the learning procedure of TWSVM is faster than that of the standard SVM [12]. In addition, TWSVM is excellent at dealing with the "Cross Plane" dataset.

The advantages of TWSVM brought much efforts to its improvements in many machine learning problems. Such as, Kumar [14] extended TWSVM to the least square version, leading to an extremely fast and efficient performance. In the light of ν-SVM, Peng [15] proposed ν-TWSVM by introducing parameters ν to control margin error as well as to reduce the number of support vectors. Shao [13] proposed a twin bounded SVM (TBSVM) by considering regularization term to minimize SRM principle of TWSVM. Inspired by linear discriminate analysis (LDA), Chen [16] presented a new projection twin SVM (PTSVM), which is dedicated to generating a projection for each class by solving an associated SVM-type QPP. Further, Shao [17] proposed a nonparallel hyperplane SVM (NHSVM) by clustering the training instances according to the similarity between classes. To make full use of prior information, Qi [19] proposed a structural twin SVM ($\mathcal{S}$−TWSVM) to acquire potential structural information between classes to build more reasonable classifier. To deal with the positive and unlabeled (PU) examples problem, Shao [23] proposed a novel Laplacian unit-hyperplane classifier (LUHC) by introducing manifold term to exploit both geometrical and discriminant properties of the examples. Yang [24] extended TWSVM to deal with the multi-class classification problem, and proposed a multi-class birth twin SVM (MBTSVM). However, to best of our knowledge, there are no extensions of TWSVM to multi-label problem. In fact, in many practical problems, such as image annotation [25], text categorization [26], and functional genomics [27], each instance could possibly have multiple labels simultaneously and the labels are no longer mutually exclusive. Such classification tasks pose challenges to the above nonparallel hyperplane classifiers, which assume that each instance is only associated with a single label

from a set of disjoint labels. The above problem could be can be transfer to the paradigm of multi-label learning (MLL) naturally [27,28]. In multi-label learning, each instance could be related to a set of labels instead of a single label, and the learning task is to determine a decision function which can predict the proper label set for an unseen instance.

The issue of learning from multi-label data has attracted significant attention in recent years. So far, series of approaches have been developed to tackle the MLL problems [29,28,30,31], most of which can be summarized into the following three groups: algorithm adaptation, problem transformation and ensemble. (1) Algorithm adaptation strategy extends the traditional single-label classifiers to perform multi-label classification directly. There are many well-known approaches, including the multi-label SVM-type methods (e.g., RankSVM [27], RankSVMz [32] and RankCVM [33]), the multi-label neural networks (e.g., BPMLL [26] and MLRBF [34]), the multi-label lazy algorithms (e.g., MLkNN [35] and MLIBLR [36]), and so on. (2) Problem transformation strategy first converts the MLL problem into a set of single-label sub-problems, then constructs a sub-classifier for each sub-problem by an existing single-label learning algorithm, and finally transforms the output back into a multi-label form. The representative approaches include the binary relevance (BR) method (one-against-all) [29], calibrated label ranking (CLR) method (one-against-one) [37], and label power-set (LP) method [28,25]. (3) Ensemble strategy is one of the most effective ways to solve the MLL problems, which combines the multi-label or single-label based classifiers by state-of-the-art ensemble techniques. The famous approaches include the boosting-based method (BoosTexter and Adaboost.MH) [38], Random k-labelsets (RAkEL) [39], Ensemble of classifier chains (ECC) [40], and so on [41–43].

In this paper, inspired by the success of TWSVM [12,13] for single label classification, we propose a novel multi-label twin support vector machine (MLTSVM) for multi-label classification, which can capture the multi-label information embedded in instances via multiple nonparallel hyperplanes. To sum up, our MLTSVM owns the following compelling properties:

- As far as we know, MLTSVM is the first nonparallel hyperplane SVM classifier applied in multi-label learning problems, which is a useful extension of nonparallel SVM.
- In training procedure, the one-against-all strategy is introduced to construct multiple nonparallel hyperplanes, with the purpose of exploiting the multi-label information via solving a series of QPPs. Each resulting hyperplane is closer to its corresponding class but far away from the others.
- In predicting procedure, the decision function is designed according to the distances from an instance to different hyperplanes, and thus the label set for a new instance can be easily assigned. By doing this, not only the ambiguous situations in testing procedure can be overcome, but also the misclassification probability can be reduced.
- To speed up the training procedure, an efficient successive overrelaxation (SOR) algorithm is further introduced to solve the QPPs in our MLTSVM.
- Last but not the least, extensive experimental results on both synthetic and real-world multi-label datasets demonstrate that under five different evaluation metrics, our approach is a competitive candidate for multi-label learning tasks compared with other six existing state-of-the-art multi-label methods.

The remaining parts of the paper are organized as follows. Section 2 briefly dwells on the classical SVM, TWSVM, and RankSVM. Section 3 proposes our MLTSVM, at the same time, the SOR method is designed to solve the optimization problems in MLTSVM. Experimental results on both synthetic and real-world datasets are shown in Sections 4 and 5 gives concluding remarks.

In this paper, upper (lower) bold face letters are used for matrices (column vectors). All vectors will be column vectors unless transformed to row vectors by a prime superscript $(\cdot)'$. A vector of zeros of arbitrary dimension is represented by $\mathbf{0}$. In addition, we denote $\boldsymbol{e}$ as a vector of ones and $\boldsymbol{I}$ as an identity matrix of arbitrary dimensions.

2. Background

2.1. Support vector machine

Consider a binary classification problem in the n-dimensional space $\mathbb{R}^n$. We denote the set of training data as $\mathcal{T} = \{(\boldsymbol{x}_i, y_i) | 1 \leq i \leq m\}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ represents an input instance with the corresponding label $y_i \in \{1, -1\}$. Standard linear SVM [1] aims to construct a pair of parallel hyperplanes that separate the two classes well

$$f_1(\boldsymbol{x}): \boldsymbol{w}'\boldsymbol{x}+b = +1 \quad \text{and} \quad f_2(\boldsymbol{x}): \boldsymbol{w}'\boldsymbol{x}+b = -1, \tag{1}$$

where $\boldsymbol{w}$ and b are the normal vector and the bias term of hyperplanes respectively. To measure its empirical risk, the L_1-norm loss function

$$R^{emp}(f) = \frac{1}{m} \sum_{i=1}^{m} |y_i f(\boldsymbol{x}_i) - 1| \tag{2}$$

is used, where $|\cdot|$ denotes absolute value operation. By introducing the RKHS regularization term $\frac{1}{2}\|\boldsymbol{w}\|^2$, the primal problem of SVM can be expressed as

$$\min_{\boldsymbol{w},b,\xi_i} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + c \sum_{i=1}^{m} \xi_i$$
$$\text{s.t.} \quad y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1 - \xi_i,$$
$$\xi_i \geq 0, \quad i = 1, 2, \ldots, m, \tag{3}$$

where $\|\cdot\|$ stands for the L_2-norm, ξ_i is the slack variable to indicate the misclassification error, and $c > 0$ is the penalty parameter. An intuitive geometric interpretation for SVM is shown in Fig. 1(a). Note that the minimization of the regularization term $\frac{1}{2}\|\boldsymbol{w}\|^2$ is equivalent to the maximization of the margin between two parallel hyperplanes (1).

In practice, rather than solving (3) directly, we solve its dual problem to get the appropriate hard or soft margin classifier. Using the Lagrangian technique, the dual version of (3) is derived as

$$\max_{\alpha} \quad -\frac{1}{2} \sum_{i,j=1}^{m} \alpha_i y_i \boldsymbol{x}_i \boldsymbol{x}_j y_j \alpha_j + \sum_{i=1}^{m} \alpha_i$$
$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0,$$
$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, m. \tag{4}$$

As we can see, the QPP of (4) has m variables, and therefore, if we use standard QP solvers, its complexity is $\mathcal{O}(m^3)$ [3]. When we obtain the optimal solution to (4), the solution of the primal problem (3) can be determined by

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \quad \text{and} \quad b = \frac{1}{|N_{sv}|} \sum_{i \in N_{sv}} (\boldsymbol{w}'\boldsymbol{x}_i - y_i), \tag{5}$$

where N_{sv} is the set of support vectors. Once the solution $(\boldsymbol{w}, b)$ is obtained, a new data instance $\boldsymbol{x}$ is classified as "$+1$" or "-1" according to whether the decision function, $f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}'\boldsymbol{x}+b)$, yields one or zero respectively.
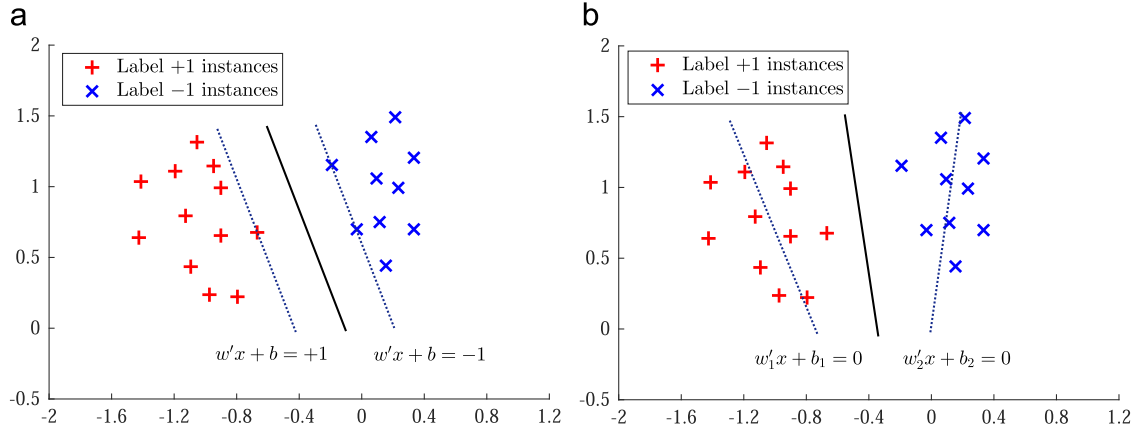
Fig. 1. An intuitive geometric interpretation: (a) SVM and (b) TWSVM.

2.2. Twin support vector machine

TWSVM [12] is originally proposed for binary single-label classification problem. It relaxes the requirement that the hyperplanes should be parallel in classical SVM [1], and aims to seek a pair of nonparallel proximal hyperplanes

$$f_1(\boldsymbol{x}): \boldsymbol{w}'_1\boldsymbol{x}+b_1=0 \quad \text{and} \quad f_2(\boldsymbol{x}): \boldsymbol{w}'_2\boldsymbol{x}+b_2=0, \tag{6}$$

such that each is closer to its own class and is as far as possible from the other, where $\boldsymbol{w}_1, \boldsymbol{w}_2$ and b_1, b_2 are the normal vectors and bias terms of the above two hyperplanes, respectively.

Without loss of generality, we assume the matrix $\boldsymbol{X}_1 \in \mathbb{R}^{m_1 \times n}$ as the labeled data belonging to "+1" class, and $\boldsymbol{X}_2 \in \mathbb{R}^{m_2 \times n}$ as the labeled data belonging to "−1" class, where $m_1 + m_2 = m$. To obtain the above two proximal hyperplanes (6), the optimization problems for TWSVM can be formulated as

$$\min_{\boldsymbol{w}_1, b_1, \boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{X}_1\boldsymbol{w}_1+\boldsymbol{e}_1 b_1\|^2+c_1\boldsymbol{e}'_2\boldsymbol{\xi}$$
$$\text{s.t.} \quad -(\boldsymbol{X}_2\boldsymbol{w}_1+\boldsymbol{e}_2 b_1)+\boldsymbol{\xi} \geq \boldsymbol{e}_2, \quad \boldsymbol{\xi} \geq \boldsymbol{0}, \tag{7}$$

and

$$\min_{\boldsymbol{w}_2, b_2, \boldsymbol{\eta}} \quad \frac{1}{2}\|\boldsymbol{X}_2\boldsymbol{w}_2+\boldsymbol{e}_2 b_2\|^2+c_2\boldsymbol{e}'_1\boldsymbol{\eta}$$
$$\text{s.t.} \quad (\boldsymbol{X}_1\boldsymbol{w}_2+\boldsymbol{e}_1 b_2)+\boldsymbol{\eta} \geq \boldsymbol{e}_1, \quad \boldsymbol{\eta} \geq \boldsymbol{0}, \tag{8}$$

where $c_1, c_2 > 0$ are the penalty parameters, and $\boldsymbol{\xi}, \boldsymbol{\eta}$ are the slack vectors. An intuitive geometric interpretation for the TWSVM is shown in Fig. 1(b). We now give a detail explanation. The first term in the objective function of (7) is to make "+1" labeled instances proximate to the hyperplane $\boldsymbol{w}'_1\boldsymbol{x}+b_1=0$, while the second term and constraints force "−1" labeled instances bounded in the hyperplane $\boldsymbol{w}'_1\boldsymbol{x}+b_1=-1$. We have the similar explanations for problem (8). To get the solutions of problems (7) and (8), we derive their dual problems as

$$\max_{\boldsymbol{\alpha}} \quad \boldsymbol{e}'_2\boldsymbol{\alpha}-\frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{G}(\boldsymbol{H}'\boldsymbol{H})^{-1}\boldsymbol{G}'\boldsymbol{\alpha}$$
$$\text{s.t.} \quad \boldsymbol{0} \leq \boldsymbol{\alpha} \leq c_1\boldsymbol{e}_2, \tag{9}$$

and

$$\max_{\boldsymbol{\beta}} \quad \boldsymbol{e}'_1\boldsymbol{\beta}-\frac{1}{2}\boldsymbol{\beta}'\boldsymbol{H}(\boldsymbol{G}'\boldsymbol{G})^{-1}\boldsymbol{H}'\boldsymbol{\beta}$$
$$\text{s.t.} \quad \boldsymbol{0} \leq \boldsymbol{\beta} \leq c_1\boldsymbol{e}_1, \tag{10}$$

where $\boldsymbol{H} = [\boldsymbol{A}\boldsymbol{e}_1]$, $\boldsymbol{G} = [\boldsymbol{B}\boldsymbol{e}_2]$ and $\boldsymbol{J} = [\boldsymbol{X}\boldsymbol{e}]$. By observing (9) and (10), we see that TWSVM solves a pair of smaller sized QPPs rather than a large one in the standard SVM. In fact, as mentioned in [12,13], the complexity of solving problems (9) and (10) approximates to $\mathcal{O}(\frac{m^3}{4})$. Once the solutions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of problems (9) and (10) are obtained, the nonparallel proximal hyperplanes (6) can be constructed by

$$\begin{bmatrix} \boldsymbol{w}_1 \\ b_1 \end{bmatrix} = -(\boldsymbol{H}'\boldsymbol{H})^{-1}\boldsymbol{G}'\boldsymbol{\alpha} \quad \text{and} \quad \begin{bmatrix} \boldsymbol{w}_2 \\ b_2 \end{bmatrix} = -(\boldsymbol{G}'\boldsymbol{G})^{-1}\boldsymbol{H}'\boldsymbol{\beta} \tag{11}$$

A new unseen instance $\boldsymbol{x}$ is assigned to label "+1" or "−1", depending on which of the proximal hyperplanes (6) it lies closer to.

2.3. Multi-label SVM (Rank-SVM)

Consider a K possible multi-label classification problem [28] in the n-dimensional space $\mathbb{R}^n$. Let $\mathcal{X} \subset \mathbb{R}^n$ be an input domain of instances and $\mathcal{Y}$ represent a finite domain of K class labels. The task of multi-label learning is to construct the decision function $h(\cdot): \mathcal{X} \to 2^{\mathcal{Y}}$, which maps from a training set

$$\mathcal{T} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) | 1 \leq i \leq m\}, \tag{12}$$

to the power set of $\mathcal{Y}$, where $\boldsymbol{x}_i \in \mathcal{X}$ denotes an input instance with the associated label set $\boldsymbol{y}_i \subseteq \mathcal{Y}$. For convenience, we write $\boldsymbol{y}_i = [y_{i1}, ..., y_{ik}, ..., y_{iK}]'$, where

$$y_{ik} = \begin{cases} +1 & \text{if } \boldsymbol{x}_i \text{ belongs to } k\text{th class} \\ -1 & \text{otherwise} \end{cases}, \quad 1 \leq k \leq K. \tag{13}$$

For an unseen instance $\boldsymbol{x} \in \mathcal{X}$, the classifier will predict $h(\boldsymbol{x}) \subseteq \mathcal{Y}$ as its associated label set.[1] The decision function h is always required to generalize well on unseen instances in the sense of optimizing some expected risk functional with respect to a specific empirical loss function [28,33].

To extend traditional SVM to the multi-label learning case, Elisseeff [27] originally proposed a multi-label SVM (RankSVM). For a K possible multi-label problem, it attempts to find K hyperplanes

$$f_k(\boldsymbol{x}): \boldsymbol{w}'_k\boldsymbol{x}+b_k=0, \quad k=1, ..., K, \tag{14}$$

one for each label, such that any relevant label should be ranked higher than any irrelevant one, where $\boldsymbol{w}_k$ and b_k are the normal vector and the bias term of the kth label hyperplane, respectively. For example, an instance $\boldsymbol{x}_i$, which belongs to the pth label rather than the q-th label, should satisfy $f_p(\boldsymbol{x}_i) > f_q(\boldsymbol{x}_i)$. To characterize

---

[1] Instead of outputting multi-label directly, most of multi-label learning systems usually return a real-valued function $f_y(\boldsymbol{x})$ to characterize the confidence of y being the proper label of instance $\boldsymbol{x}$. For a successful multi-label learning system, it will tend to output larger values for relevant labels than irrelevant labels, i.e., $f_{y' \in \boldsymbol{y}}(\boldsymbol{x}) > f_{y' \notin \boldsymbol{y}}(\boldsymbol{x})$.

this, the approximate ranking loss function

$$R^{emp}(f) = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{|\mathbf{y}_i| |\overline{\mathbf{y}_i}|} \sum_{(p,q) \in (\mathbf{y}_i \times \overline{\mathbf{y}_i})} |f_p(\mathbf{x}_i) - f_q(\mathbf{x}_i) - 1| \right), \quad (15)$$

is introduced, where $\mathbf{y}_i$ is the relevant label set of the instance $\mathbf{x}_i$, and $\overline{\mathbf{y}_i}$ (the complement of $\mathbf{y}_i$) is referred to its irrelevant label set, while $|\mathbf{y}_i|$ and $|\overline{\mathbf{y}_i}|$ denote their cardinalities. Then, the primal problem of RankSVM can be depicted as

$$\min_{\mathbf{w}, b, \xi_{ipq}} \quad \frac{1}{2} \sum_{k=1}^{K} \| \mathbf{w}_k \|^2 + c \sum_{i=1}^{m} \left(\frac{1}{l_i} \sum_{(p,q) \in (\mathbf{y}_i \times \overline{\mathbf{y}_i})} \xi_{ipq} \right)$$
$$\text{s.t.} \quad (\mathbf{w}_p - \mathbf{w}_q)' \mathbf{x}_i + (b_p - b_q) \geq 1 - \xi_{ipq},$$
$$\xi_{ipq} \geq 0, \quad i = 1, 2, \ldots, m, \quad (16)$$

where $c > 0$ is the penalty parameter, $l_i = |\mathbf{y}_i| |\overline{\mathbf{y}_i}|$, and ξ_{ipq} is the slack variable to indicate ranking error. The first term in the objective function of (16) is to minimize the sum of norms of hyperplanes (14), which controls the model complexity to avoid over-fitting. The second term and constraints make the value of relevant label hyperplane $f_p(\mathbf{x}_i)$ on instance $\mathbf{x}_i$ be at least 1 larger than the value of irrelevant label hyperplane $f_q(\mathbf{x}_i)$.

By introducing the Lagrangian multipliers α, the dual QPP of (16) can be represented as

$$\max_{\alpha} \quad -\frac{1}{2} \sum_{k=1}^{K} \sum_{i,j=1}^{m} \beta_{ki} \beta_{kj} (\mathbf{x}_i' \mathbf{x}_j) + \sum_{i=1}^{m} \sum_{(p,q) \in (\mathbf{y}_i \times \overline{\mathbf{y}_i})} \alpha_{ipq}$$
$$\text{s.t.} \quad \sum_{i=1}^{m} \beta_{ki} = 0, \quad 0 \leq \alpha_{ipq} \leq \frac{c}{l_i}, \quad k = 1, \ldots, K, \quad (17)$$

where

$$\beta_{ki} = \sum_{(p,q) \in (\mathbf{y}_i \times \overline{\mathbf{y}_i})} \left(h_{ipq}^{k} \alpha_{ipq} \right) \quad \text{and} \quad h_{ipq}^{k} = \begin{cases} +1 & \text{if } p = k \\ -1 & \text{if } q = k \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

For the ith instance, we define $\boldsymbol{\alpha}_i = [a_{ipq} | (p,q) \in (\mathbf{y}_i \times \overline{\mathbf{y}_i})] \in \mathbb{R}^{l_i}$, $\mathbf{h}_{ki} = \left[h_{ipq}^{k} | (p,q) \in (\mathbf{y}_i \times \overline{\mathbf{y}_i}) \right] \in \mathbb{R}^{l_i}$, $\mathbf{H}_i = [\mathbf{h}_{1i}, \ldots, \mathbf{h}_{Ki}]' \in \mathbb{R}^{K \times l_i}$, and simplify (18) as $\beta_{ki} = \mathbf{h}_{ki}' \boldsymbol{\alpha}_i$. Then, the dual problem (17) can be reformulated as

$$\max_{\alpha} \quad -\frac{1}{2} \sum_{i,j=1}^{m} \boldsymbol{\alpha}_i' (\mathbf{H}_i' \mathbf{H}_j) \boldsymbol{\alpha}_j (\mathbf{x}_i' \mathbf{x}_j) + \sum_{i=1}^{m} \mathbf{e}' \boldsymbol{\alpha}_i$$
$$\text{s.t.} \quad \sum_{i=1}^{m} \mathbf{H}_i \boldsymbol{\alpha}_i = 0, \quad 0 \leq \boldsymbol{\alpha}_i \leq \frac{c}{l_i} \mathbf{e}. \quad (19)$$

As pointed in [33], the complexity of solving the problem (19) is $\mathcal{O}(K l_t^2)$, where $l_t = \sum_i^m l_i$ is the number of variables α. Once the

solutions of (16) are obtained via solving its dual problem (19), the multi-label prediction of RankSVM can be fulfilled by

$$f(\mathbf{x}) = \{k | f_k(\mathbf{x}) \geq t(\mathbf{x}), \quad k = 1, \ldots, K\}, \quad (20)$$

where $t(\mathbf{x})$ is a proper threshold. More details can be seen in [27,33].

3. Multi-label twin support vector machine

In TWSVM [12,13], it assumes that each instance is associated with a single-label from a set of disjoint labels, which makes it incapable of dealing with the case that each instance is associated with multi-labels. In this section, we extend TWSVM to multi-label problem, and give the derivation of our multi-label twin support vector machine (MLTSVM) for both linear and nonlinear cases.

3.1. Linear MLTSVM

For K possible multi-label classification problem with the training set (12), the basic idea of our linear MLTSVM is to seek K proximal hyperplanes

$$f_k(\mathbf{x}) : \mathbf{w}_k' \mathbf{x} + b_k = 0, \quad k = 1, \ldots, K, \quad (21)$$

such that the kth hyperplane is closer to the instances with the label k, and is as far as possible from the others, where $\mathbf{w}_k$ and b_k are the normal vector and the bias term, respectively, of the kth proximal hyperplane.

To obtain the kth proximal hyperplane, for convenience, we use $\mathcal{I}_k$ and $\overline{\mathcal{I}_k}$ to denote two complementary index sets, and if the instance $\mathbf{x}_i$ is associated with the kth label, then $i \in \mathcal{I}_k$, otherwise $i \in \overline{\mathcal{I}_k}$. Additionally, we have $\mathcal{I} = \mathcal{I}_k \bigcup \overline{\mathcal{I}_k}$, where $k = 1, 2, \ldots, K$. Similar to TWSVM [12], the empirical risk is implemented by the following loss function:

$$R^{emp}(f_k) = \sum_{i \in \mathcal{I}_k} \| f_k(\mathbf{x}_i) \|^2 + c_k \sum_{j \in \overline{\mathcal{I}_k}} \max(1 + f_k(\mathbf{x}_j), 0), \quad (22)$$

where $c_k > 0$ is the empirical risk penalty parameter that determines the trade-off between the loss terms in (22).

By introducing the RKHS regularization term $\frac{1}{2}(\| \mathbf{w}_k \|^2 + b_k^2)$, the primal problem of MLTSVM for the kth hyperplane can be expressed as

$$\min_{\mathbf{w}_k, b_k, \xi_j} \quad \frac{1}{2} \sum_{i \in \mathcal{I}_k} \| \mathbf{w}_k' \mathbf{x}_i + b_k \|^2 + c_k \sum_{j \in \overline{\mathcal{I}_k}} \xi_j + \frac{1}{2} \lambda_k (\| \mathbf{w}_k \|^2 + b_k^2)$$
$$\text{s.t.} \quad -(\mathbf{w}_k' \mathbf{x}_j + b_k) \geq 1 - \xi_j, \xi_j \geq 0, j \in \overline{\mathcal{I}_k}, \quad (23)$$
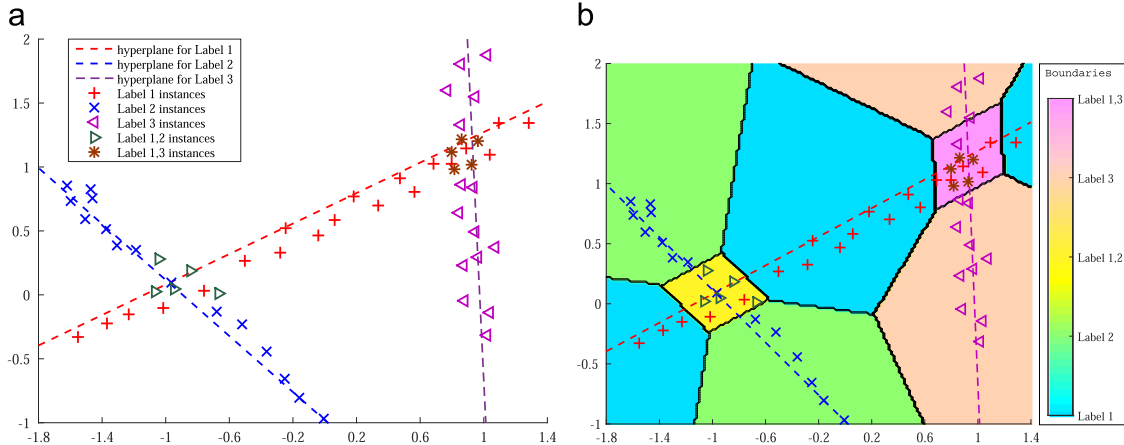

Fig. 2. An intuitive geometric interpretation for MLTSVM: (a) nonparallel proximal hyperplanes and (b) decision boundaries.

where $\lambda_k > 0$ is the regularization parameter and $\xi_{j \in \overline{\mathcal{I}_k}}$ are the slack variables to indicate mislabeled errors.

Before optimizing the problem (23), we give its geometric interpretation. The first term in the objective function minimizes the squared sum of $f_k(x_i)$ for $x_i \in \mathcal{I}_k$, which makes the kth labeled instances $\boldsymbol{x}_{i \in \mathcal{I}_k}$ be as close as possible to the kth proximal hyperplane $f_k(\boldsymbol{x})$. Optimizing the second term together with the constraints requires the instances $\boldsymbol{x}_{j \in \overline{\mathcal{I}_k}}$ be at a distance larger than one from the hyperplane $f_k(\boldsymbol{x})$. Otherwise, slack variables $\xi_{j \in \overline{\mathcal{I}_k}}$ are introduced to measure the errors. The last term in the objective function of (23) is the norm of the RKHS term, which controls the model complexity of MLTSVM to avoid over-fitting.

It is worthy noting that, in the training procedure of our MLTSVM, the one-against-all strategy is introduced to construct multiple nonparallel hyperplanes, with the purpose of exploiting the multi-label information via solving a series of QPPs. Fig. 2 (a) gives a intuitive description of the above optimization technique on the two-dimensional space. As we can see, each hyperplane is generated to proximate to its corresponding instances while is far away from the other instances to some extent. Furthermore, by comparing the primal problems of TWSVM and MLTSVM, we can see that there is a major difference between them. In fact, on one hand, each instance appears in objective function of TWSVM only once, which implies that each instance is just closer to one hyperplane. On the other hand, the instance may appear in the objective functions of MLTSVM multiple times, which means that the instance in MLTSVM may lie near to multiple associated hyperplanes.

We now turn to the solution to problem (23). For this purpose, we derive its dual problem. By introducing the non-negative Lagrangian multipliers $\alpha_{j \in \overline{\mathcal{I}_k}}$ and $\beta_{j \in \overline{\mathcal{I}_k}}$, the Lagrangian problem of of (23) is given by

$$L(\boldsymbol{w}_k, b_k, \xi_j, \alpha_j, \beta_j) = \frac{1}{2} \sum_{i \in \mathcal{I}_k} \| \boldsymbol{w}_k' \boldsymbol{x}_i + b_k \|^2 + c_k \sum_{j \in \overline{\mathcal{I}_k}} \xi_j + \frac{1}{2} \lambda_k (\| \boldsymbol{w}_k \|^2$$
$$+ b_k^2) + \sum_{j \in \overline{\mathcal{I}_k}} \alpha_j (\boldsymbol{w}_k' \boldsymbol{x}_j + b_k + 1 - \xi_j) - \sum_{j \in \overline{\mathcal{I}_k}} \beta_j \xi_j. \tag{24}$$

Setting the derivatives of the Lagrangian function (24) with respect to $\boldsymbol{w}_k$, b_k, ξ_j, α_j and β_j to zero yields the following Karush–Kuhn–Tucker (KKT) [3] necessary and sufficient optimality conditions

$$\nabla_{\boldsymbol{w}_k} L = \sum_{i \in \mathcal{I}_k} (\boldsymbol{w}_k' \boldsymbol{x}_i + b_k) \boldsymbol{x}_i + \lambda_k \boldsymbol{w}_k + \sum_{j \in \overline{\mathcal{I}_k}} \alpha_j \boldsymbol{x}_j = \boldsymbol{0}, \tag{25}$$

$$\nabla_{b_k} L = \sum_{i \in \mathcal{I}_k} (\boldsymbol{w}_k' \boldsymbol{x}_i + b_k) + \lambda_k b_k + \sum_{j \in \overline{\mathcal{I}_k}} \alpha_j = 0, \tag{26}$$

$$\nabla_{\xi_j} L = c_k - \alpha_j - \beta_j = 0, \quad j \in \overline{\mathcal{I}_k}, \tag{27}$$

$$-(\boldsymbol{w}_k' \boldsymbol{x}_j + b_k) \geq 1 - \xi_j, \xi_j \geq 0, \tag{28}$$

$$\sum_{j \in \overline{\mathcal{I}_k}} \alpha_j (\boldsymbol{w}_k' \boldsymbol{x}_j + b_k + 1 - \xi_j) = 0, \quad \alpha_j \geq 0, \tag{29}$$

$$\sum_{j \in \overline{\mathcal{I}_k}} \beta_j \xi_j = 0, \beta_j \geq 0. \tag{30}$$

Obviously, combining (25) and (26) leads to

$$\sum_{i \in \mathcal{I}_k} \begin{pmatrix} \boldsymbol{x}_i \\ 1 \end{pmatrix} (\boldsymbol{x}_i 1) \begin{pmatrix} \boldsymbol{w}_k \\ b_k \end{pmatrix} + \lambda_k \begin{pmatrix} \boldsymbol{w}_k \\ b_k \end{pmatrix} + \sum_{j \in \overline{\mathcal{I}_k}} \begin{pmatrix} \boldsymbol{x}_j \\ 1 \end{pmatrix} \alpha_j = \boldsymbol{0}. \tag{31}$$

If we define $\boldsymbol{z}_l = (\boldsymbol{x}_l', 1)'$ for each $l \in \mathcal{I}$, then Eq. (31) can be rewritten as

$$\left(\sum_{i \in \mathcal{I}_k} \boldsymbol{z}_i \boldsymbol{z}_i' + \lambda_k \boldsymbol{I} \right) \begin{pmatrix} \boldsymbol{w}_k \\ b_k \end{pmatrix} + \sum_{j \in \overline{\mathcal{I}_k}} \boldsymbol{z}_j \alpha_j = \boldsymbol{0}. \tag{32}$$

This gives

$$\begin{pmatrix} \boldsymbol{w}_k \\ b_k \end{pmatrix} = - \sum_{j \in \overline{\mathcal{I}_k}} \boldsymbol{z}_j \alpha_j \left(\sum_{i \in \mathcal{I}_k} \boldsymbol{z}_i \boldsymbol{z}_i' + \lambda_k \right)^{-1}. \tag{33}$$

Since $\beta_j \geq 0$, we conclude from (27) that $0 \leq \alpha_j \leq c_k$. Then, replacing (33) into the Lagrangian function (24) with the above KKT conditions (25)–(30), we obtain the dual of problem (23) as

$$\max_{\alpha_j} \quad \sum_{j \in \overline{\mathcal{I}_k}} \alpha_j - \frac{1}{2} \sum_{j_1 \in \mathcal{I}_k} \sum_{j_2 \in \mathcal{I}_k} \alpha_{j_1} \alpha_{j_2} \boldsymbol{z}_{j_1}' \left(\sum_{i \in \mathcal{I}_k} \boldsymbol{z}_i \boldsymbol{z}_i' + \lambda_k \boldsymbol{I} \right)^{-1} \boldsymbol{z}_{j_2}$$
$$\text{s.t.} \quad 0 \leq \alpha_j \leq c_k, j \in \overline{\mathcal{I}_k}. \tag{34}$$

Once the solutions α_j of the dual problem (34) are obtained, the kth proximal hyperplane (21) can be constructed according to (33), where $k = 1, \dots, K$.

In the following, we describe the prediction strategy for a new unseen instance $\overline{\boldsymbol{x}} \in \mathbb{R}^n$. As mentioned above, our MLTSVM is a proximal classifier, and if the new instance $\overline{\boldsymbol{x}}$ is closer enough to some proximal hyperplanes, it will be assigned to the corresponding labels. That is to say, if the distance from $\overline{\boldsymbol{x}}$ to the kth proximal hyperplane (21)

$$d_k(\overline{\boldsymbol{x}}) = \frac{|\boldsymbol{w}_k' \overline{\boldsymbol{x}} + b_k|}{\| \boldsymbol{w}_k \|}, \quad k = 1, \dots, K, \tag{35}$$

is small enough, i.e., $d_k(\overline{\boldsymbol{x}}) \leq \Delta_k$ with a positive threshold Δ_k, then we could assign $\overline{\boldsymbol{x}}$ to the k-th label.

Next, we give a proper way to choose Δ_k. Notice that, for the primal QPP (23) of MLTSVM, it demands the instances $\boldsymbol{x}_{j \in \overline{\mathcal{I}_k}}$ to have at least one distance from the k-th hyperplane. That is to say, when $d_k(\overline{\boldsymbol{x}}) > 1$ (i.e., $d_k(\overline{\boldsymbol{x}})$ is bigger than the one distance threshold), then $\overline{\boldsymbol{x}}$ is more likely irrelevant to the k-th label. On the other hand, when $d_k(\overline{\boldsymbol{x}}) \leq 1$, it is more likely relevant to the k-th label. In order to keep the consistency, we introduce the normalized measurement and set $\Delta_k = \frac{1}{\| \boldsymbol{w}_k \|}$ instead of 1 as the threshold for the kth label class. For all the classes, a simple choice is that we set the new threshold $\Delta_{sel} = \min_{k=1,\dots,K}(\Delta_k) = \min_{k=1,\dots,K}(\frac{1}{\| \boldsymbol{w}_k \|})$. Then, we have the following prediction procedure:

1. Compute the distances of instance $\overline{\boldsymbol{x}}$ to each proximal hyperplane according to (35), and set the threshold $\Delta_{sel} = \min_{k=1,\dots,} K(\frac{1}{\| \boldsymbol{w}_k \|})$.
2. If $d_k(\overline{\boldsymbol{x}}) \leq \Delta_{sel}$, we associate instance $\overline{\boldsymbol{x}}$ with the kth label.
3. If there is no $d_k(\overline{\boldsymbol{x}})$ meets the condition in step 2, then we assign $\overline{\boldsymbol{x}}$ as label $= \mathbf{arg}\min_{k=1,\dots,K} (d_k(\overline{\boldsymbol{x}}))$.

To get a clear picture of the advantages of the above procedure, we draw the decision boundaries obtained by using our MLTSVM on the aforementioned two-dimensional example in Fig. 2(b). It can be seen from Fig. 2(b) that, by introducing the threshold Δ_{sel}, the misclassification possibility in label overlapping place is reduced and the intrinsic distribution of the data could be captured well by MLTSVM.

3.2. Nonlinear MLTSVM

For the nonlinear case, we introduce the kernel trick to realize the learning process by mapping the linearly nonseparable instances in the input space into a kernel space, which will make them more likely linearly separable. For this purpose, we consider

the following K kernel-generated proximal surfaces instead of linear hyperplanes:

$$f_k(\boldsymbol{x}) : \boldsymbol{w}'_k \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}) + b_k = 0, \quad k = 1, \ldots, K, \tag{36}$$

where $\boldsymbol{X}$ denotes all the training instances, $\boldsymbol{w}_k$ and b_k represent the normal vector and the bias term, respectively, in the kernel space, and $\mathcal{K}(\cdot, \cdot)$ is a proper chosen kernel.[2]

Similar to the linear case, the kth kernel-based proximal surface is determined by the following primal problem:

$$\min_{\boldsymbol{w}_k, b_k, \xi_j} \quad \frac{1}{2} \sum_{i \in \mathcal{I}_k} \| \boldsymbol{w}'_k \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_i) + b_k \|^2 + c_k \sum_{j \in \overline{\mathcal{I}_k}} \xi_j + \frac{1}{2} \lambda_k (\| \boldsymbol{w}_k \|^2 + b_k^2)$$
$$\text{s.t.} \quad -(\boldsymbol{w}'_k \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_j) + b_k) \geq 1 - \xi_j, \xi_j \geq 0, j \in \overline{\mathcal{I}_k}, \tag{37}$$

where $c_k > 0$ is the empirical risk penalty parameter, $\lambda_k > 0$ is the regularization parameter, and $\xi_{j \in \overline{\mathcal{I}_k}}$ are the slack variables.

Let us now derive the dual problem of (37). Its Lagrangian function is given by

$$L(\boldsymbol{w}_k, b_k, \xi_j, \alpha_j, \beta_j) = \frac{1}{2} \sum_{i \in \mathcal{I}_k} \| \boldsymbol{w}'_k \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_i) + b_k \|^2 + c_k$$
$$\sum_{j \in \overline{\mathcal{I}_k}} \xi_j + \frac{1}{2} \lambda_k (\| \boldsymbol{w}_k \|^2 + b_k^2) + \sum_{j \in \overline{\mathcal{I}_k}} \alpha_j (\boldsymbol{w}'_k \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_j) + b_k + 1 - \xi_j) - \sum_{j \in \overline{\mathcal{I}_k}} \beta_j \xi_j, \tag{38}$$

where $\alpha_{j \in \overline{\mathcal{I}_k}}$ and $\beta_{j \in \overline{\mathcal{I}_k}}$ are the non-negative Lagrangian multipliers. The KKT conditions are given by

$$\nabla_{\boldsymbol{w}_k} L = \sum_{i \in \mathcal{I}_k} (\boldsymbol{w}'_k \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_i) + b_k) \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_i) + \lambda_k \boldsymbol{w}_k + \sum_{j \in \overline{\mathcal{I}_k}} \alpha_j \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_j) = \boldsymbol{0}, \tag{39}$$

$$\nabla_{b_k} L = \sum_{i \in \mathcal{I}_k} (\boldsymbol{w}'_k \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_i) + b_k) + \lambda_k b_k + \sum_{j \in \overline{\mathcal{I}_k}} \alpha_j = 0, \tag{40}$$

$$\nabla_{\xi_j} L = c_k - \alpha_j - \beta_j = 0, \quad j \in \overline{\mathcal{I}_k}, \tag{41}$$

$$-(\boldsymbol{w}'_k \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_j) + b_k) \geq 1 - \xi_j, \quad \xi_j \geq 0, \tag{42}$$

$$\sum_{j \in \overline{\mathcal{I}_k}} \alpha_j (\boldsymbol{w}'_k \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_j) + b_k + 1 - \xi_j) = 0, \quad \alpha_j \geq 0, \tag{43}$$

$$\sum_{j \in \overline{\mathcal{I}_k}} \beta_j \xi_j = 0, \quad \beta_j \geq 0. \tag{44}$$

Obviously, combining (39) and (40) leads to

$$\sum_{i \in \mathcal{I}_k} \begin{pmatrix} \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_i) \\ 1 \end{pmatrix} (\mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_i) 1) \begin{pmatrix} \boldsymbol{w}_k \\ b_k \end{pmatrix} + \lambda_k \begin{pmatrix} \boldsymbol{w}_k \\ b_k \end{pmatrix} + \sum_{j \in \overline{\mathcal{I}_k}} \begin{pmatrix} \mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_j) \\ 1 \end{pmatrix} \alpha_j = \boldsymbol{0}. \tag{45}$$

If we define $\overline{\boldsymbol{z}}_l = (\mathcal{K}(\boldsymbol{X}, \boldsymbol{x}_l)', 1)'$ for each instance $l \in \mathcal{I}$, then Eq. (45) can be rewritten as

$$\left(\sum_{i \in \mathcal{I}_k} \overline{\boldsymbol{z}}_i \overline{\boldsymbol{z}}'_i + \lambda_k \boldsymbol{I} \right) \begin{pmatrix} \boldsymbol{w}_k \\ b_k \end{pmatrix} + \sum_{j \in \overline{\mathcal{I}_k}} \overline{\boldsymbol{z}}_j \alpha_j = \boldsymbol{0}. \tag{46}$$

This gives

$$\begin{pmatrix} \boldsymbol{w}_k \\ b_k \end{pmatrix} = - \sum_{j \in \overline{\mathcal{I}_k}} \overline{\boldsymbol{z}}_j \alpha_j \left(\sum_{i \in \mathcal{I}_k} \overline{\boldsymbol{z}}_i \overline{\boldsymbol{z}}'_i + \lambda_k \boldsymbol{I} \right)^{-1}. \tag{47}$$

Since $\beta_j \geq 0$, from (41) we conclude $0 \leq \alpha_j \leq c_k$. Then, putting (47) into the Lagrangian function (38) with the above KKT conditions

(39)–(44), we obtain the dual of problem (37) as

$$\max_{\alpha_j} \sum_{j \in \overline{\mathcal{I}_k}} \alpha_j - \frac{1}{2} \sum_{j_1 \in \overline{\mathcal{I}_k}} \sum_{j_2 \in \overline{\mathcal{I}_k}} \alpha_{j_1} \alpha_{j_2} \overline{\boldsymbol{z}}'_{j_1} \left(\sum_{i \in \mathcal{I}_k} \overline{\boldsymbol{z}}_i \overline{\boldsymbol{z}}'_i + \lambda_k \boldsymbol{I} \right)^{-1} \overline{\boldsymbol{z}}_{j_2}$$
$$\text{s.t.} \quad 0 \leq \alpha_j \leq c_k, j \in \overline{\mathcal{I}_k}. \tag{48}$$

After optimizing the above dual QPP (48), we obtain the normal vector and the bias term of the kth proximal surface (36) according to (47). For an unseen instance $\overline{\boldsymbol{x}} \in \mathbb{R}^n$, we can construct a similar prediction strategy as the linear case.

3.3. Implementation

Now, we discuss the implementation of our proposed method. In our MLTSVM, the most computational cost occurs when solving the dual QPPs (34) and (48). For convenience, we define the set $\boldsymbol{X}_{\mathcal{I}_k} = \{\boldsymbol{x}_{i \in \mathcal{I}_k}\}$ as the instances associated with the kth label, and $\boldsymbol{X}_{\overline{\mathcal{I}_k}} = \{\boldsymbol{x}_{j \in \overline{\mathcal{I}_k}}\}$ as the rest instances. It is easy to see that these problems can be rewritten in the following unified matrix form:

$$\max_{\boldsymbol{\alpha}} \quad \boldsymbol{e}' \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}' \boldsymbol{Q} \boldsymbol{\alpha}$$
$$\text{s.t.} \quad \boldsymbol{0} \leq \boldsymbol{\alpha} \leq c_k \boldsymbol{e}, \tag{49}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_l] \in \mathbb{R}^l$, l is the scale of $\overline{\mathcal{I}_k}$, and $\boldsymbol{Q}$ is defined by

$$\boldsymbol{Q} = \boldsymbol{G}'_k (\boldsymbol{H}'_k \boldsymbol{H}_k + \lambda_k \boldsymbol{I})^{-1} \boldsymbol{G}_k, \tag{50}$$

$$\boldsymbol{H}_k = [\boldsymbol{X}_{\mathcal{I}_k} \boldsymbol{e}], \quad \boldsymbol{G}_k = [\boldsymbol{X}_{\overline{\mathcal{I}_k}} \boldsymbol{e}] \tag{51}$$

for linear case, or

$$\boldsymbol{Q} = \overline{\boldsymbol{G}}'_k (\overline{\boldsymbol{H}}'_k \overline{\boldsymbol{H}}_k + \lambda_k \boldsymbol{I})^{-1} \overline{\boldsymbol{G}}_k, \tag{52}$$

$$\overline{\boldsymbol{H}}_k = [\mathcal{K}(\boldsymbol{X}, \boldsymbol{X}_{\mathcal{I}_k}) \boldsymbol{e}], \quad \overline{\boldsymbol{G}}_k = [\mathcal{K}(\boldsymbol{X}, \boldsymbol{X}_{\overline{\mathcal{I}_k}}) \boldsymbol{e}] \tag{53}$$

for nonlinear case. It is time-consuming to solve problem (49) by some standard QP packages. Luckily, for such special simplex constraints, this QP problem can be solved effectively by an optimization technique called successive overrelaxation (SOR) [13,17,44]. Mangasarian [44] pointed out that the SOR can linearly converge to an optimum, and process large-scale datasets without needing to reside in memory.

Algorithm 1. The successive overrelaxation algorithm for MLTSVM.

Input:
The penalty parameter c_k, the relaxation factor $\omega \in (0, 2)$, and the matrix $\boldsymbol{Q}$ is defined by (50) or (52).
Output:
The optimal solution $\boldsymbol{\alpha}$ for the problem (49).
1: Initialize iterator $i = 0$ and start with any $\boldsymbol{\alpha}^0 \in \mathbb{R}^l$.
2: Split $\boldsymbol{Q} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{L}'$, where $\boldsymbol{L}$ is the strictly lower triangular matrix and $\boldsymbol{D}$ is the diagonal matrix.
3: **while** ($\| \boldsymbol{\alpha}^{i+1} - \boldsymbol{\alpha}^i \| < 10^{-6}$) **do**
4: Compute $\boldsymbol{\alpha}^{i+1} = \boldsymbol{\alpha}^i + \omega \Delta_{\boldsymbol{\alpha}}$ where $\Delta_{\boldsymbol{\alpha}}$ is given by $\Delta_{\boldsymbol{\alpha}} = -\boldsymbol{D}^{-1} (\boldsymbol{Q} \boldsymbol{\alpha}^i - \boldsymbol{e} + \boldsymbol{L}' (\boldsymbol{\alpha}^{i+1} - \boldsymbol{\alpha}^i))$.
5: Project $\boldsymbol{\alpha}^{i+1}$ to the feasible range $[0, c_k]$.
6: **end while**

The whole procedure is summarized in Algorithm 3.3. In each iteration, having $\boldsymbol{\alpha}^i$, compute $\boldsymbol{\alpha}^{i+1}$ as follows:

$$\boldsymbol{\alpha}^{i+1} = \boldsymbol{\alpha}^i + \omega \Delta_{\boldsymbol{\alpha}},$$
$$\Delta_{\boldsymbol{\alpha}} = -\boldsymbol{D}^{-1} (\boldsymbol{Q} \boldsymbol{\alpha}^i - \boldsymbol{e} + \boldsymbol{L}' (\boldsymbol{\alpha}^{i+1} - \boldsymbol{\alpha}^i)), \tag{54}$$

where $\omega \in (0, 2)$ is the relaxation factor, the nonzero elements of $\boldsymbol{L}$

---

[2] Such as the RBF kernel $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \| \boldsymbol{x}_i - \boldsymbol{x}_j \|^2}$ with parameter $\gamma > 0$. Note that the surfaces (36) will be degraded into the hyperplanes (21) when the linear kernel is selected.

Table 1
Statistics for the synthetic multi-label datasets used in our experiments.

Dataset	Instances	Features			Labels	Cardinality	Density
		Relevant	Irrelevant	Redundant			
Hyperspheres (HS1)	400	15	5	0	5	1.268	0.254
Hyperspheres (HS2)	600	35	10	5	10	1.408	0.141
Hyperspheres (HS3)	1000	50	20	15	20	1.433	0.072
Hyperspheres (HS4)	2000	100	30	25	40	1.805	0.045
Hypercubes (HC1)	400	15	5	0	5	1.225	0.245
Hypercubes (HC2)	600	35	10	5	10	1.133	0.113
Hypercubes (HC3)	1000	50	20	15	20	1.109	0.055
Hypercubes (HC4)	2000	100	30	25	40	1.344	0.034

constitute the strictly lower triangular part of the symmetric matrix $\boldsymbol{Q}$, and the nonzero elements of $\boldsymbol{D}$ constitute the diagonal of $\boldsymbol{Q}$. The SOR is an iterative procedure that employs the Gauss–Seidel (GS) iterations with the relaxation factor[3] ω to accelerate the solving of QPP (49). In practice [44,45], α_j^{i+1} is computed from $(\alpha_1^{i+1}, ..., \alpha_{j-1}^{i+1}, \alpha_j^i, ..., \alpha_l^i)$, and the lower triangular matrix $\boldsymbol{L}$ in (54) can be seen as a substitution operator, i.e., using $(\alpha_1^{i+1}, ..., \alpha_{j-1}^{i+1})$ to replace the components $(\alpha_1^i, ..., \alpha_{j-1}^i)$ in the latest $\boldsymbol{\alpha}^i$. Then, the iterative formula in (54) can be rewritten as

$$\alpha_j^{i+1} = \alpha_j^i - \omega D_{jj}^{-1} \left(\sum_{k=1}^{j-1} Q_{jk}\alpha_k^{i+1} + \sum_{k=j}^{l} Q_{jk}\alpha_k^i - 1 \right). \tag{55}$$

From (55), we can see that there is only one variable α_j^{i+1} needs to be updated in each iteration, which requires $(l+2)$ products.[4] This implies that the complexity of each updating iteration is $\mathcal{O}(l)$. Thus, the complexity of solving QPP (49) is reduced to $\sharp Iter \times \mathcal{O}(l)$, where $\sharp Iter$ is the number of SOR iterations.

4. Numerical experiments

4.1. Experimental setup

To evaluate the performance of our MLTSVM,[5] in this section, we investigate its performance on both synthetic and real-world datasets. In our implementation, we focus on the comparison between MLTSVM and six start-of-the-art multi-label classification methods, including BoosTexter [38], CLR [37], MLkNN [35], BPMLL [26], RAkEL [39] and RankSVM [27]. For a test set $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) | 1 \le i \le p\}$, similar to [29,30,35,28,32,33], we choose the following five popular and indicative metrics to perform comparison, including the Hloss (hamming loss), Avepre (average precision), Cov (coverage), Rloss (ranking loss), and Oerr (one-error). Here, we use "↓" to denote the metric that the larger values, the better performance, while "↑" means the smaller, the better.

1. Hamming loss: Evaluates how many times an instance-label pair is misclassified between the predicted label set $h(\boldsymbol{x})$ and the ground-truth label set $\boldsymbol{y}$

$$Hloss\downarrow = \frac{1}{p} \sum_{i=1}^{p} \frac{1}{K} |h(\boldsymbol{x}_i)\Delta \boldsymbol{y}_i| \in [0, 1], \tag{56}$$

where Δ stands for the symmetric difference of two sets.

2. Average precision: Evaluates the average fraction of labels ranked above a particular label $y \in \boldsymbol{y}_i$:

$$Avepre\uparrow = \frac{1}{p} \sum_{i=1}^{p} \left(\frac{1}{|\boldsymbol{y}_i|} \sum_{y \in \boldsymbol{y}_i} \frac{|\{y' \in \boldsymbol{y}_i \mid rank_f(\boldsymbol{x}_i, y') \le rank_f(\boldsymbol{x}_i, y)\}|}{rank_f(\boldsymbol{x}_i, y)} \right) \in [0, 1]. \tag{57}$$

3. Coverage: Evaluates how far we need, on average, to go down the ranked list of labels in order to cover all the possible labels of the instance:

$$Cov\downarrow = \frac{1}{p} \sum_{i=1}^{p} \left(\max_{y \in \boldsymbol{y}_i} rank_f(\boldsymbol{x}_i, y) - 1 \right) \in [0, K-1]. \tag{58}$$

4. Ranking loss: Evaluates the average fraction of label pairs that are reversely ordered. Let $\overline{\boldsymbol{y}}$ be the complementary set of $\boldsymbol{y}$ in $\mathcal{Y}$

$$Rloss\downarrow = \frac{1}{p} \sum_{i=1}^{p} \left(\frac{1}{|\boldsymbol{y}_i||\overline{\boldsymbol{y}_i}|} \left| \{(y', y'') | f_{y' \in \boldsymbol{y}_i}(\boldsymbol{x}_i) \le f_{y'' \in \overline{\boldsymbol{y}_i}}(\boldsymbol{x}_i)\} \right| \right) \in [0, 1]. \tag{59}$$

5. One-error: Evaluates how many times the top-ranked label is not in the set of proper labels of the instance

$$Oerr\downarrow = \frac{1}{p} \sum_{i=1}^{p} H(\boldsymbol{x}_i) \in [0, 1], H(\boldsymbol{x}_i) = \begin{cases} 0 & \text{if } \arg\max f_y(\boldsymbol{x}_i) \in \boldsymbol{y}_i \\ 1 & \text{otherwise} \end{cases}. \tag{60}$$

All the experiments are implemented on a personal computer (PC) with an Intel Core-i5 processor (2.3 GHz) and 4 GB random-access memory (RAM). With regard to parameter selection, we employ the 10-fold cross-validation technique.[6] For the sake of brevity, we set all penalties $c_k = c$ and $\lambda_k = \lambda$ for our MLTSVM. For BoosTexter, the number of boosting rounds is set to be 500. For MLkNN, the parameter range $\{6, 8, ..., 20\}$ is considered for the number of neighbors in kNN during the model selection, and the Laplacian estimator s is fixed to be 1. For BPMLL, the number of hidden neurons is $\{5\%, 10\%, ..., 25\%\}$ of the number of input neurons, the training epochs is set to be 100 and the regularization constant is fixed to be 0.1. For CLR, RAkEL, RankSVM and MLTSVM,

---

[3] In this paper, as suggested in [44,45], we set the factor $\omega = 0.2$ for our experiments.

[4] In each iteration, SOR requires $(l+2)$ products: the term $\sum_{k=1}^{j-1} Q_{jk}\alpha_k^{i+1}$ needs $(j-1)$ products, the term $\sum_{k=j}^{l} Q_{jk}\alpha_k^i$ needs $(l-j+1)$ product, and the multiplied ω and D_{jj}^{-1} need 2 products. Additionally, we can cache the inversion of diagonal matrix $\boldsymbol{D}$ (i.e., D_{jj}^{-1}) before iteration.

[5] Code is available at http://www.optimal-group.org/Resource/MLTSVM.html

[6] In detail [3], each dataset is partitioned into 10 subsets with similar sizes and distributions. Then, the union of nine subsets is used as the training set while the remaining subset is used as the test set. The experiment is repeated 10 times such that every subset is used once as a test set.
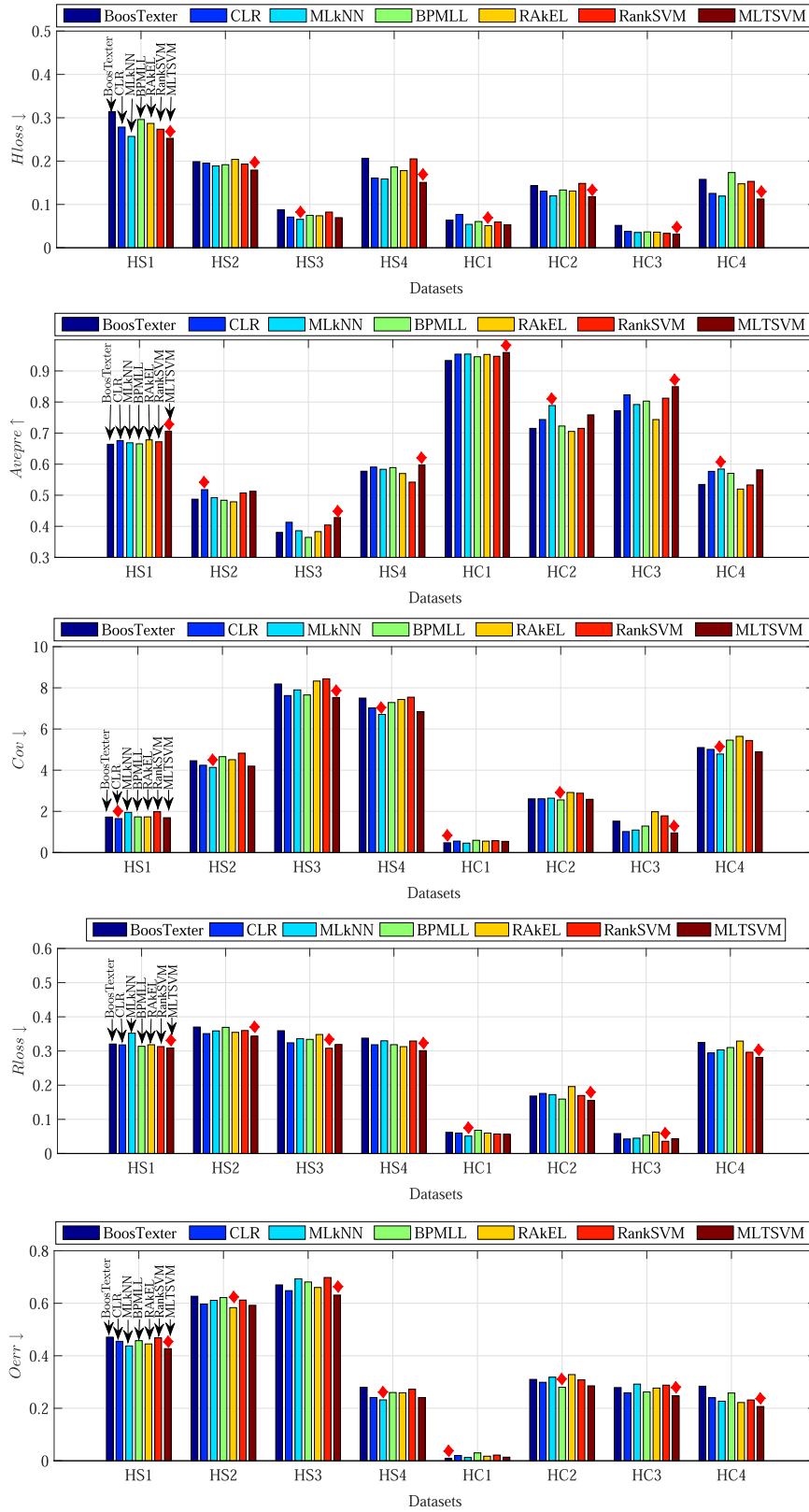
Fig. 3. Learning results of the multi-label learning approaches on benchmark synthetic datasets, in terms of *Hloss*↓, *Avepre*↑, *Cov*↓, *Rloss*↓ and *Oerr*↓ metric.

the penalty and the kernel parameter are selected from the set $\{2^{-6}, 2^{-5}, ..., 2^{6}\}$ and $\{2^{-4}, 2^{-3}, ..., 2^{4}\}$, respectively. Once the optimal parameters are selected, they are employed to learn the final decision function.

4.2. Results on benchmark synthetic datasets

To compare our MLTSVM with the aforementioned five multi-label classifiers, we utilize eight benchmark synthetic datasets

Table 2
Relative performance between MLTSVM and other five compared classifiers on benchmark synthetic datasets, according to pairwise t-test 5% significance level.

Dataset	W–T–L (Win–Tie–Loss)					
	BoosTexter	CLR	MLkNN	BPMLL	RAkEL	RankSVM
HS1	3–2–0	3–2–0	2–3–0	3–2–0	4–1–0	4–1–0
HS2	4–1–0	2–2–1	2–2–1	4–1–0	3–2–0	2–3–0
HS3	4–1–0	1–4–0	2–3–0	3–2–0	2–3–0	3–1–1
HS4	3–2–0	2–3–0	3–1–1	3–2–0	2–3–0	4–1–0
HC1	2–3–0	1–4–0	2–1–2	2–3–0	2–2–1	2–3–0
HC2	3–2–0	3–2–0	2–3–0	2–2–1	4–1–0	3–2–0
HC3	3–2–0	3–2–0	2–3–0	3–2–0	3–2–0	3–1–1
HC4	4–1–0	2–3–0	1–4–0	3–2–0	2–3–0	4–1–0
Average	3.25–1.75–0	2.125–2.75–0.125	2–2.5–0.5	2.875–2–0.125	2.75–2.125–0.125	3.125–0.625–0.5

Table 3
Statistics for the real-world multi-label datasets used in our experiments.

Dataset	Domain	Instances	Features	Labels	Cardinality	Density
Emotions[a]	music	593	72	6	1.869	0.311
Birds[a]	audio	645	260	19	1.014	0.053
Flags[a]	images	194	19	7	3.392	0.485
Yeast[a]	biology	2417	103	14	4.237	0.303
Plant[b]	biology	948	440	12	1.082	0.090

[a] Datasets are available at http://mulan.sourceforge.net/datasets.html.
[b] Datasets are available at http://computer.njnu.edu.cn/Lab/LABIC/LABIC_Software.html.

of the datasets. The average "W–T–L" summarization listed at the bottom of Table 2 also reveals the feasibility of our approach.

4.3. Results on real-world datasets

In this subsection, we apply our MLTSVM to several real-world datasets, and investigate its performance and computational efficiency by using the aforementioned five multi-label metrics. For comparison, we consider five commonly used real-world datasets, whose statistics are listed in Table 3. These datasets represent a wide range of domains (include music, image, biology, and so on), sizes (from 194 to 2417), features (from 19 to 1449), and labels (from 6 to 45). All reported results are estimated from 10-fold cross validation executions and the pairwise t-test is then used to determine their significance in 5% level.

Tables 4–8 list the mean and the deviation of 10-fold cross-validation multi-label classification results of each classifier on the above real-world datasets, where the best result on each dataset is shown in bold face. From these tables, we can find that our MLTSVM owns the best performance on at least three out of five metrics for all the real-world datasets. Thus, it is clear that our approach performs fairly well, which demonstrates the better performance of MLTSVM compared to the other classifiers. In particular, MLTSVM outperforms all the other algorithms when *Avepre* metric is taken. We also record the ten-run computation time[8] (mean ± std) of each classifier for the above real-world datasets' experiments, which is shown in the right column of Tables 4–8. The comparison results demonstrate that the learning efficiency of our MLTSVM is comparable to the lazy classifier MLkNN, and a lot faster than the remaining five. The possible explanation is that MLTSVM has benefited from the effectivity of SOR algorithm, which results in its fast learning speed.

For comparison purpose, similar as the synthetic dataset, we further introduce the comprehensive metric "W–L–T" to measure the relative performance between our MLTSVM and other five multi-label classifiers, shown in Table 9. ▴/▿ indicates whether MLTSVM is statistically superior/inferior to the compared algorithm, according to pairwise t-test 5% significance level. The results show our approach is very efficacious, since it can more effectively capture the label information by nonparallel hyperplanes. Overall, it slightly outperforms MLkNN and CLR, while is better performance than BoosTexter, BPMLL, RAkEL and RankSVM. The average "W–T–L" summarization of each compared classifier is also listed at the bottom of Table 9, which further shows the advantage of our MLTSVM over the others.

To validate our threshold chosen strategy in prediction process, we further implement our MLTSVM on the above real-world datasets with the different threshold parameter Δ. Here, we denote Δ_{sel} as the threshold chosen by our strategy and the range of parameter Δ is $\{2^m \times \Delta_{sel} | m = -4, -3, ..., 3, 4\}$. Fig. 4 shows the

generated by *Mldatagen*.[7] The synthetics are created according to the predefined parameters: the strategies (hyperspheres or hypercubes), number of features (relevant, irrelevant, and redundant), number of instances, and number of labels. After choosing the strategy to be applied, firstly, *Mldatagen* randomly generates a geometric shape (hypersphere or hypercube) for each label, which is populated with instances randomly generated. Afterwards, each instance is labeled according to the shape it belongs to, which defines the instance multi-label. Furthermore, we introduce 5% noises to pollute the labels of each instance to make the learning tasks be more challenge. Table 1 presents the basic statistics of the generated datasets, where "Instances", "Features", and "Labels" represent the number of instances, features, and labels that the dataset contains, "Cardinality" denotes the average number of labels per example: $\frac{1}{m}\sum_{i=1}^{m}|\mathbf{y}_i|$, and "Density" indicates the normalized label cardinality in the label space: $\frac{1}{|\mathcal{Y}| \times m}\sum_{i=1}^{m}|\mathbf{y}_i|$.

Fig. 3 shows the mean of 10-fold cross-validation learning results of each classifier on the above benchmark synthetic datasets (in terms of "*Hloss*", "*Avepre*", "*Cov*", "*Rloss*" and "*Oerr*" metric), where the best performance is highlighted by symbol "♦". It can be clearly seen that compared with other multi-label classifiers, our MLTSVM exhibits very good performance on most of the synthetic datasets. For example, for the "*Hloss*" metric, our MLTSVM owns six best performance among all classifiers, while both MLkNN and RAkEL obtain one best. For other four metrics, our MLTSVM performs better than other classifiers on all of them, except for "*Cov*" metric, where MLTSVM performs the best on two of datasets and MLkNN obtains three best. This may be explained by the fact that MLTSVM fully utilizes the underlying multi-label information by constructing optimal multi-nonparallel hyperplanes, and thus results in a better generalization ability.

To make a clearer view of the relative performance among classifiers, similar to [28,35,26], we perform a pairwise t-test to compare our MLTSVM to the other classifiers for each metric. The significance level (SL) is set to 0.05. That is, when the t-test value of two compared classifiers is greater than 1.7341, the performance between them is deemed as statistically significantly different. By comparing t-test values, a comprehensive metric "W–T–L" (Win–Tie–Loss) is further introduced to characterize their relative performance, which denotes the number of metrics that MLTSVM is significantly superior/equal/inferior to the compared classifiers. Table 2 displays the relative performance results for our MLTSVM compared with other multi-label classifiers on synthetic datasets. It can be clearly seen that MLTSVM performs rather well on most

---

[7] *Mldatagen* is a benchmark synthetic dataset generator for multi-label learning, which is available at http://sites.labic.icmc.usp.br/mldatagen/

[8] We use the computation time (training and predicting CPU time) to denote the computational efficiency for each classifier.

Table 4
Learning results of each multi-label classifier (mean $\pm$ std) on the *Emotions* data.

Algorithms	Metrics					Times (s)
	Hloss↓	Avepre↑	Cov↓	Rloss↓	Oerr↓	
BoosTexter	0.238 ± 0.042 ∙	0.768 ± 0.051	1.838 ± 0.176	0.198 ± 0.048 ∙	0.314 ± 0.056	4.351 ± 0.187
CLR	0.257 ± 0.039 ∙	0.772 ± 0.035	1.820 ± 0.205	0.178 ± 0.030	0.327 ± 0.068 ∙	2.192 ± 0.218
MLkNN	0.209 ± 0.028	0.762 ± 0.049 ∙	1.805 ± 0.198	0.173 ± 0.041	**0.289 ± 0.101**	0.930 ± 0.069
BPMLL	0.214 ± 0.019	0.754 ± 0.034 ∙	1.967 ± 0.210 ∙	0.166 ± 0.047	0.309 ± 0.075	2.248 ± 0.171
RAkEL	0.220 ± 0.023	0.759 ± 0.035 ∙	2.042 ± 0.223 ∙	0.206 ± 0.033 ∙	0.341 ± 0.069 ∙	1.305 ± 0.102
RankSVM	0.228 ± 0.036	0.773 ± 0.042	1.977 ± 0.257 ∙	**0.158 ± 0.041**	0.332 ± 0.078 ∙	2.429 ± 0.230
MLTSVM	**0.206 ± 0.025**	**0.781 ± 0.029**	**1.794 ± 0.317**	0.163 ± 0.029	0.304 ± 0.065	0.893 ± 0.073

Table 5
Learning results of each multi-label classifier (mean $\pm$ std) on the *Birds* data.

Algorithms	Metrics					Times (s)
	Hloss↓	Avepre↑	Cov↓	Rloss↓	Oerr↓	
BoosTexter	0.094 ± 0.023 ∙	0.496 ± 0.042 ∙	3.507 ± 0.637 ∙	0.148 ± 0.030	0.741 ± 0.073	21.264 ± 0.218
CLR	0.071 ± 0.013 ∙	0.533 ± 0.081	3.313 ± 0.710	0.137 ± 0.034	0.739 ± 0.053	1.076 ± 0.368
MLkNN	0.057 ± 0.011	0.503 ± 0.070 ∙	**3.024 ± 0.705** ▽	0.141 ± 0.032	0.763 ± 0.069 ∙	1.772 ± 0.095
BPMLL	0.114 ± 0.059 ∙	0.387 ± 0.039 ∙	6.371 ± 1.392 ∙	0.288 ± 0.072 ∙	0.731 ± 0.036	6.740 ± 0.508
RAkEL	0.053 ± 0.009	0.532 ± 0.091	3.965 ± 0.905 ∙	0.174 ± 0.041 ∙	0.735 ± 0.085	16.844 ± 1.329
RankSVM	0.064 ± 0.023	0.494 ± 0.091 ∙	4.172 ± 1.052 ∙	**0.122 ± 0.053**	0.767 ± 0.097 ∙	3.836 ± 0.353
MLTSVM	**0.049 ± 0.018**	**0.541 ± 0.105**	3.218 ± 0.688	0.125 ± 0.045	**0.718 ± 0.074**	1.445 ± 0.162

Table 6
Learning results of each multi-label classifier (mean $\pm$ std) on the *Flags* data.

Algorithms	Metrics					Times (s)
	Hloss↓	Avepre↑	Cov↓	Rloss↓	Oerr↓	
BoosTexter	0.276 ± 0.059	0.803 ± 0.104 ∙	3.764 ± 0.588	0.224 ± 0.096	0.247 ± 0.150 ∙	0.929 ± 0.104
CLR	0.271 ± 0.063	0.827 ± 0.073	**3.609 ± 0.468** ▽	0.236 ± 0.062 ∙	0.220 ± 0.145	0.132 ± 0.094
MLkNN	0.286 ± 0.062 ∙	0.821 ± 0.067	3.758 ± 0.506 ∙	0.214 ± 0.060	0.227 ± 0.136	0.059 ± 0.007
BPMLL	0.290 ± 0.063 ∙	0.781 ± 0.065 ∙	3.826 ± 0.512	0.248 ± 0.080 ∙	0.264 ± 0.135 ∙	0.549 ± 0.068
RAkEL	**0.257 ± 0.055**	0.809 ± 0.071	3.929 ± 0.475 ∙	0.236 ± 0.081 ∙	0.231 ± 0.127	0.194 ± 0.082
RankSVM	0.281 ± 0.072 ∙	0.786 ± 0.083 ∙	3.861 ± 0.614	0.226 ± 0.078	0.297 ± 0.142 ∙	0.349 ± 0.045
MLTSVM	0.266 ± 0.059	**0.831 ± 0.081**	3.729 ± 0.508	**0.206 ± 0.072**	**0.219 ± 0.179**	0.078 ± 0.012

Table 7
Learning results of each multi-label classifier (mean $\pm$ std) on the *Yeast* data.

Algorithms	Metrics					Times (s)
	Hloss↓	Avepre↑	Cov↓	Rloss↓	Oerr↓	
BoosTexter	0.235 ± 0.063 ∙	0.746 ± 0.055	6.984 ± 0.437 ∙	0.187 ± 0.028	0.271 ± 0.036 ∙	263.705 ± 0.218
CLR	0.224 ± 0.012 ∙	0.743 ± 0.016 ∙	6.757 ± 0.238 ∙	0.180 ± 0.013	0.249 ± 0.029	61.085 ± 3.824
MLkNN	0.209 ± 0.017	0.758 ± 0.017	**6.268 ± 0.313**	0.179 ± 0.016	0.243 ± 0.015	19.126 ± 0.751
BPMLL	0.226 ± 0.013 ∙	0.739 ± 0.014 ∙	6.587 ± 0.266	0.181 ± 0.012	0.257 ± 0.025	41.332 ± 2.259
RAkEL	0.233 ± 0.013 ∙	0.749 ± 0.021	7.718 ± 0.295 ∙	0.225 ± 0.018 ∙	0.282 ± 0.041 ∙	133.707 ± 12.97
RankSVM	0.219 ± 0.027	0.737 ± 0.039 ∙	7.129 ± 0.452 ∙	**0.169 ± 0.019**	0.266 ± 0.029	108.186 ± 9.187
MLTSVM	**0.201 ± 0.015**	**0.764 ± 0.019**	6.312 ± 0.357	0.175 ± 0.009	**0.238 ± 0.024**	20.640 ± 1.283

Table 8
Learning results of each multi-label classifier (mean $\pm$ std) on the *Plant* data.

Algorithms	Metrics					Times (s)
	Hloss↓	Avepre↑	Cov↓	Rloss↓	Oerr↓	
BoosTexter	0.147 ± 0.018 ∙	0.502 ± 0.046	2.940 ± 0.482 ∙	0.302 ± 0.057 ∙	0.725 ± 0.073 ∙	87.048 ± 3.463
CLR	0.115 ± 0.015	0.497 ± 0.057 ∙	2.753 ± 0.560	0.234 ± 0.053	0.711 ± 0.076 ∙	53.272 ± 1.357
MLkNN	0.096 ± 0.006	0.522 ± 0.044	2.731 ± 0.422	0.259 ± 0.040 ∙	0.684 ± 0.065	6.449 ± 0.285
BPMLL	0.102 ± 0.055	0.428 ± 0.020 ∙	3.189 ± 0.318 ∙	0.293 ± 0.028 ∙	0.797 ± 0.031 ∙	39.082 ± 1.536
RAkEL	0.126 ± 0.007 ∙	0.450 ± 0.042 ∙	2.749 ± 0.503	0.286 ± 0.050 ∙	0.691 ± 0.052	63.451 ± 2.747
RankSVM	0.131 ± 0.023 ∙	0.531 ± 0.051	2.894 ± 0.513 ∙	**0.218 ± 0.047** ▽	0.675 ± 0.062	37.516 ± 0.492
MLTSVM	**0.093 ± 0.008**	**0.549 ± 0.039**	**2.724 ± 0.475**	0.242 ± 0.031	**0.667 ± 0.048**	6.028 ± 0.316

learning results of MLTSVM with various threshold Δ in terms of five multi-label metrics, where the best performance is highlighted by black circle. It can be seen that MLTSVM arrives at the best performance on most of the metrics when the threshold value is near or equal to Δ_{sel}, expect for the "Avepre" metric on the "Emotions" dataset. It also can be seen that MLTSVM presents the similar trend on the most datasets, but with slight differences. For example, for the "Bird" dataset, MLTSVM owns three best metrics when Δ is equal to Δ_{sel}. In general, although the threshold value Δ_{sel} is not global optimal, the performance is acceptable. By further considering the tuning of the optimal threshold Δ requires lots of time, in practice, we recommend setting $\Delta = \Delta_{sel}$ unless there is a special request on the performance.

Table 9
Relative performance between MLTSVM and other five compared classifiers on real-world multi-label datasets, according to pairwise t-test 5% significance level.

Dataset	W–T–L (Win–Tie–Loss)					
	BoosTexter	CLR	MLkNN	BPMLL	RAkEL	RankSVM
Emotions	2–3–0	2–3–0	1–4–0	2–3–0	4–1–0	2–3–0
Birds	3–2–0	1–4–0	2–2–1	4–1–0	2–3–0	3–2–0
Flags	2–3–0	1–3–1	2–3–0	3–2–0	2–3–0	3–2–0
Yeast	3–2–0	3–2–0	0–5–0	3–2–0	4–1–0	2–3–0
Plant	4–1–0	2–3–0	1–4–0	4–1–0	3–2–0	2–2–1
Average	2.8–2.2–0	1.8–3–0.2	1.2–3.6–0.2	3.2–1.8–0	3–2–0	2.4–2.4–0.2

4.4. Application to semantic scene recognition

Semantic scene recognition is the task of automatically acquiring semantic information of images, and has wide applications in many real-world information systems [35,4]. However, many scene images may have several labels simultaneously, which is a severe challenge to many machine learning methods. Thus, in this subsection, we will further apply our MLTSVM to this multi-label learning task.
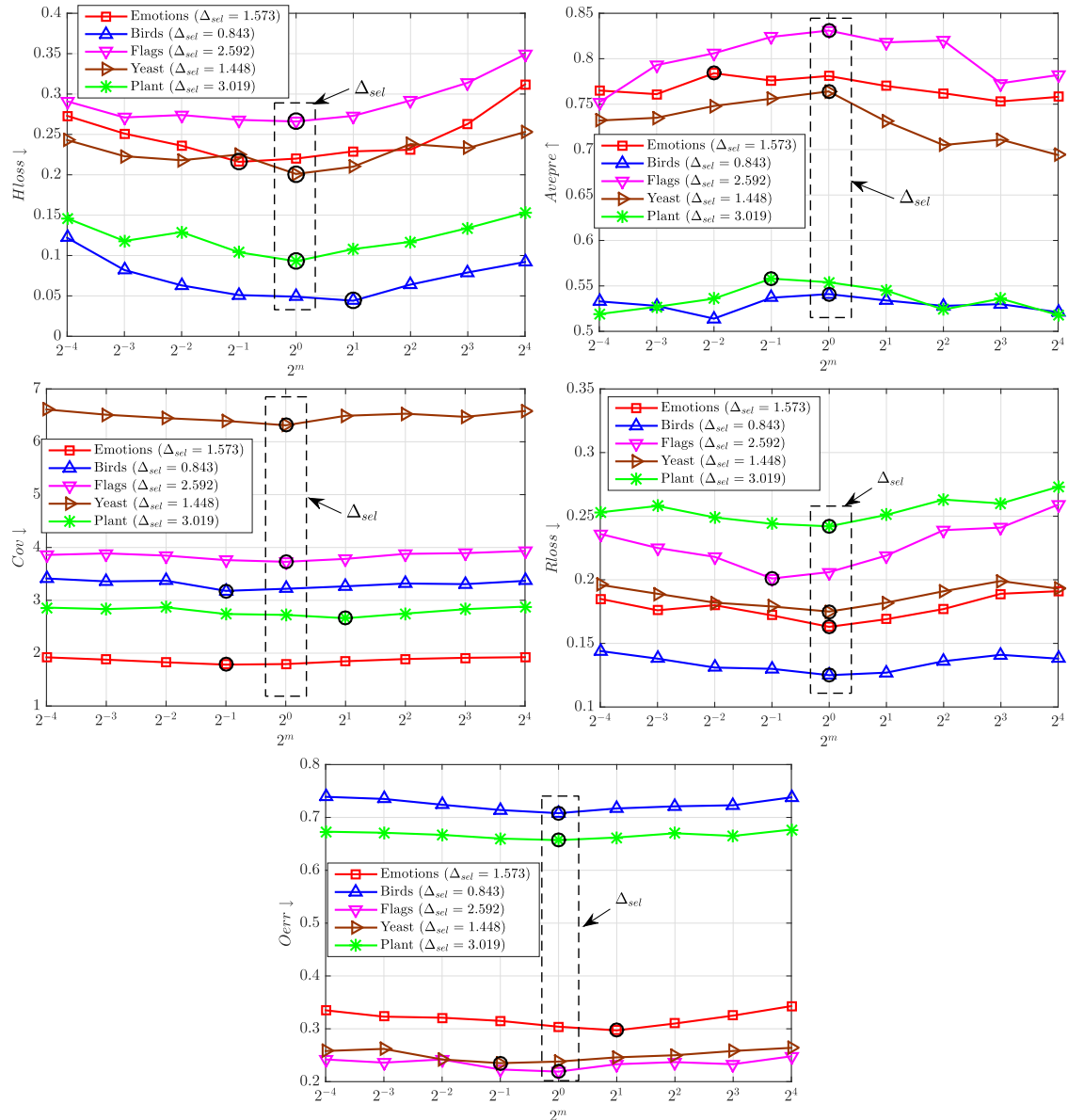

Fig. 4. Learning results of the MLTSVM classifier with various threshold $\Delta = 2^m \times \Delta_{sel}$ on five real-world datasets, in terms of Hloss↓, Avepre↑, Cov↓, Rloss↓ and Oerr↓ metric, where Δ_{sel} denotes the threshold value chosen by our strategy.

Table 10
Characteristics of the multi-label semantic scene dataset.

Label set	#Images	Label set	#Images
tree	543	people+mountain	16
people	523	grass+buildings	3
grass	318	grass+mountain	13
buildings	264	buildings+mountain	3
mountain	179	tree+people+grass	1
tree+people	53	tree+people+buildings	4
tree+grass	88	tree+people+mountain	1
tree+buildings	64	tree+grass+buildings	5
tree+mountain	85	tree+grass+mountain	8
people+grass	9	tree+buildings+mountain	2
people+buildings	62	people+grass+buildings	1

Fig. 5. The conversion procedure of pixel scene image to feature vector.

Table 11
Learning results of each multi-label classifier (mean ± std) on *scene* dataset.

Metrics	Algorithms						
	BoosTexter	CLR	MLkNN	BPMLL	RAkEL	RankSVM	MLTSVM
$Hloss\downarrow$	0.158 ± 0.024 ⋆	0.148 ± 0.018 ⋆	0.098 ± 0.026	0.234 ± 0.057 ⋆	0.122 ± 0.026	0.138 ± 0.072 ⋆	**0.091 ± 0.015**
$Avepre\uparrow$	0.827 ± 0.036	0.805 ± 0.028 ⋆	0.817 ± 0.017 ⋆	0.731 ± 0.045 ⋆	0.838 ± 0.021	0.807 ± 0.037 ⋆	**0.849 ± 0.021**
$Cov\downarrow$	0.583 ± 0.060 ⋆	0.608 ± 0.074 ⋆	0.482 ± 0.070	0.495 ± 0.167	0.573 ± 0.052 ⋆	0.494 ± 0.074	**0.478 ± 0.049**
$Rloss\downarrow$	0.133 ± 0.019 ⋆	0.115 ± 0.014 ⋆	0.119 ± 0.023 ⋆	0.088 ± 0.032	0.127 ± 0.029 ⋆	**0.082 ± 0.034**	0.094 ± 0.021
$Oerr\downarrow$	0.274 ± 0.067	0.271 ± 0.054	0.259 ± 0.025	0.357 ± 0.066 ⋆	0.286 ± 0.042 ⋆	0.294 ± 0.036 ⋆	**0.255 ± 0.028**
W–L–T	3–2–0	4–1–0	2–3–0	3–2–0	3–2–0	3–2–0	–
Times (s)	175.605 ± 7.359	73.159 ± 4.174	24.362 ± 0.811	59.487 ± 3.493	146.204 ± 9.24	93.482 ± 8.290	21.513 ± 1.268

In our experiment, we select 2245 images from the popular semantic scene Corel dataset[9] with five possible categories/classes, namely "tree", "people", "grass", "buildings" and "mountain". They are artificially used to label the images, and details are given in Table 10. It can be seen that each image is associated with an average of 1.2 class labels. Further, the number of images belonging to more than one class comprises over 18.6% of the dataset, and multiple combined classes (more than three) are extremely rare.

In the following, we adopt the similar strategy used in [35,25] to convert each pixel image into a feature vector. The main process is illustrated in Fig. 5, for which we give the following explanation: firstly, we convert the color space of each image from RGB to Luv,

which will result in a better approximate perceptual uniformity such that perceived color differences correspond closely to Euclidean distances in this color space. Then, we divide each Luv space image into 49 blocks using a 7×7 grid. Afterwards, the first and second moments (mean and variance) of L, u, v band are computed in each block, corresponding to a low-resolution image and to computationally inexpensive texture features, respectively. Finally, each image is transformed into a 294-dimensional feature vector.[10]

The 10-fold cross-validation results (mean ± std) of each classifier on semantic scene dataset are reported in Table 11. We have highlighted the best result. We can see that MLTSVM gets the best performance on four out of five metrics except *Rloss*, and

---

[9] Dataset is available at https://sites.google.com/site/dctresearch/Home/content-based-image-retrieval

[10] 294-dimensional feature vector consists of 49 blocks, and each block has the mean and the variance of L, u, v band features, i.e., 3×2 features.

	Image 1	Image 2	Image 3
groundtruth	buildings, mountain	tree, people, grass	people, buildings
BoosTexter	buildings, grass	tree, people, grass, mountain	people, mountain
CLR	grass, mountain	tree, people	buildings
MLkNN	mountain	tree, people, grass	tree, buildings
BPMLL	buildings, grass, mountain	grass, buildings	people, mountain
RAkEL	buildings, tree	tree, grass	tree, buildings, mountain
RankSVM	buildings	people. grass	tree, buildings
MLTSVM	buildings, mountain	tree, people, grass	people, buildings

	Image 4	Image 5	Image 6
groundtruth	tree, buildings, grass	people	tree, buildings, grass, mountain
BoosTexter	buildings, grass	people, buildings	tree, grass, mountain
CLR	buildings	tree	grass, mountain
MLkNN	grass, mountain	tree, people	people, buildings, grass
BPMLL	buildings, grass	people	tree, grass
RAkEL	tree, buildings	tree, people	grass, mountain
RankSVM	buildings, grass, mountain	tree, buildings	buildings, grass
MLTSVM	buildings, grass	people	tree, grass, mountain

	Image 7	Image 8	Image 9
groundtruth	tree, buildings	people, mountain	buildings, grass
BoosTexter	grass, buildings	people, buildings	buildings
CLR	buildings	tree, mountain	grass, mountain
MLkNN	grass, buildings	mountain	buildings, grass
BPMLL	tree, grass, buildings	people, mountain	tree, buildings
RAkEL	tree, mountain	tree, buildings	tree, buildings, grass
RankSVM	grass, mountain	tree, mountain	buildings
MLTSVM	tree, buildings	people, mountain	buildings, grass

Fig. 6. Some image classification results. The first-line labels below each image are the groundtruth. The remaining-line labels are sequentially generated by BoosTexter, CLR, MLkNN, BPMLL, RAkEL, RankSVM and MLTSVM, respectively.

outperforms BoosTexter, CLR, BPMLL and RankSVM on most metrics. It is not surprising that both BPMLL and RankSVM outperform all the other classifiers with respect to *Rloss*. The possible explanation is that the objectives of them are to minimize the ranking loss criterion. Fig. 6 shows some images that MLTSVM performs better than other classifiers. As for computation time, our MLTSVM performs comparable to MLkNN, and is much less than the others. Furthermore, "W–L–T" summarization based on *t*-test is also listed in Table 11. It can be clearly seen that our approach obtains the better performance than the others.

In terms of generalization, the above experiments on both synthetic and real-world multi-label datasets have indicated that the proposed MLTSVM achieves the comparable or better classification results and faster learning speed, as compared with the other six existing state-of-the-art multi-label methods. It is worth pointing out that our MLTSVM could capture the underlying multi-label information by constructing optimal multi-nonparallel hyperplanes for each class.

5. Conclusions

The issue of learning from multi-label data has attracted significant attention in recent years. However, such classification tasks pose challenges to the traditional nonparallel hyperplane classifiers, which assumes that each instance is only associated with a single label from a set of disjoint labels. To address the above issues, a novel nonparallel hyperplane classifier termed as MLTSVM is proposed in this paper. MLTSVM exploits the potential multi-label information embedded in data by constructing multiple nonparallel hyperplanes, while an efficient SOR algorithm is applied to solve the involved QPPs. Furthermore, its decision function based on distances from instances to hyperplanes overcomes the ambiguity in testing procedure. The feasibility of our MLTSVM is supported by a series of experiments on synthetic datasets as well as on real-world datasets.

In the future, we will conduct a more efficient parameters selection strategy for our MLTSVM. Moreover, modifying MLTSVM

to feature selection and semi-supervised learning would also be interesting.

Conflict of interest

No conflicts of interest.

Acknowledgment

References

[1] V. Vapnik, Statistical Learning Theory, Wiley, New York, USA, 1998.
[2] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Disc. 2 (2) (1998) 121–167.
[3] N. Deng, Y. Tian, C. Zhang, Support Vector Machines: Theory, Algorithms and Extensions, CRC Press, Philadelphia, USA, 2012.
[4] H. Yin, X. Jiao, Y. Chai, B. Fang, Scene classification based on single-layer sae and svm, Expert Syst. Appl. 42 (7) (2015) 3368–3380.
[5] B. Zhang, J. Su, X. Xu, A class-incremental learning method for multi-class support vector machines in text classification, in: International Conference on Machine Learning and Cybernetics, Dalian, China, 2006, pp. 2581–2585.
[6] A. Subasi, Classification of emg signals using pso optimized svm for diagnosis of neuromuscular disorders, Comput. Biol. Med. 43 (5) (2013) 576–586.
[7] D. Zhang, H. Xu, Z. Su, Y. Xu, Chinese comments sentiment classification based on word2vec and svmperf, Expert Syst. Appl. 42 (4) (2015) 1857–1863.
[8] J. Platt, Fast training of support vector machines using sequential minimal optimization, in: Advances in Kernel Methods, MIT Press, Cambridge, MA, 1998, pp. 185–200.
[9] C. Chang, C. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 1–27.
[10] T. van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. de Moor, J. Vandewalle, Benchmarking least squares support vector machine classifiers, Mach. Learn. 54 (1) (2004) 5–32.
[11] I.W. Tsang, J.T. Kwok, P. Cheung, Core vector machines: fast svm training on very large data sets, J. Mach. Learn. Res. 6 (2005) 363–392.
[12] Jayadeva, R. Khemchandani, S. Chandra, Twin support vector machines for pattern classification, IEEE Trans. Pattern Anal. 29 (5) (2007) 905–910.
[13] Y. Shao, C. Zhang, X. Wang, N. Deng, Improvements on twin support vector machines, IEEE Trans. Neural Netw. 22 (6) (2011) 962–968.
[14] M. Arun Kumar, M. Gopal, Least squares twin support vector machines for pattern classification, Expert Syst. Appl. 36 (4) (2009) 7535–7543.
[15] X. Peng, TPMSVM: a novel twin parametric-margin support vector machine for pattern recognition, Pattern Recognit. 44 (10-11) (2011) 2678–2692.
[16] X. Chen, J. Yang, Q. Ye, J. Liang, Recursive projection twin support vector machine via within-class variance minimization, Pattern Recognit. 44 (10–11) (2011) 2643–2655.
[17] Y. Shao, W. Chen, N. Deng, Nonparallel hyperplane support vector machine for binary classification problems, Inf. Sci. 263 (2014) 22–35.
[18] Z. Qi, Y. Tian, Y. Shi, Laplacian twin support vector machine for semi-supervised classification, Neural Netw. 35 (2012) 46–53.
[19] Z. Qi, Y. Tian, Y. Shi, Structural twin support vector machine for classification, Knowl. Based Syst. 43 (2013) 74–81.
[20] W. Chen, Y. Shao, D. Xu, Y. Fu, Manifold proximal support vector machine for semi-supervised classification, Appl. Intell. 40 (4) (2014) 623–638.
[21] W. Chen, Y. Shao, N. Deng, Z. Feng, Laplacian least squares twin support vector machine for semi-supervised classification, Neurocomputing 145 (2014) 465–476.
[22] Y. Shao, W. Chen, J. Zhang, Z. Wang, N. Deng, An efficient weighted lagrangian twin support vector machine for imbalanced data classification, Pattern Recognit. 47 (9) (2014) 3158–3167.
[23] Y. Shao, W. Chen, L. Liu, N. Deng, Laplacian unit-hyperplane learning from positive and unlabeled examples, Inf. Sci. 314 (2015) 152–168.
[24] Z. Yang, Y. Shao, X. Zhang, Multiple birth support vector machine for multi-class classification, Neural Comput. Appl. 22 (1) (2013) 153–161.
[25] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (9) (2004) 1757–1771.
[26] M. Zhang, Z. Zhou, Multilabel neural networks with applications to functional genomics and text categorization, IEEE Trans. Knowl. Data Eng. 18 (10) (2006) 1338–1351.
[27] W.J. Elisseeff A, A kernel method for multi-labelled classification, in: Proceedings of the 14th Conference on Neural Information Processing Systems (NIPS2001), Vancouver, British Columbia, Canada, 2001, pp. 681–687.
[28] M. Zhang, Z. Zhou, A review on multi-label learning algorithms, IEEE Trans. Knowl. Data Eng. 26 (8) (2014) 1819–1837.
[29] G. Tsoumakas, I. Katakis, Multi-label classification, Int. J. Data Wareh. Min. 3 (3) (2007) 1–13.
[30] G. Madjarov, D. Kocev, D. Gjorgjevikj, S. Dzeroski, An extensive experimental comparison of methods for multi-label learning, Pattern Recognit. 45 (9) (2012) 3084–3104.
[31] B. Zhang, X. Xu, J. Su, An ensemble method for multi-class and multi-label text categorization, in: International Conference on Intelligent Systems and Knowledge Engineering, Chengdu, China, 2007, pp. 1–4.
[32] J. Xu, Random block coordinate descent method for multi-label support vector machine with a zero label, Expert Syst. Appl. 41 (7) (2014) 3418–3428.
[33] J. Xu, Fast multi-label core vector machine, Pattern Recognit. 46 (3) (2013) 885–898.
[34] M. Zhang, ML-RBF: RBF neural networks for multi-label learning, Neural Process. Lett. 29 (2) (2009) 61–74.
[35] M. Zhang, Z. Zhou, ML-KNN: A lazy learning approach to multi-label learning, Pattern Recognit. 40 (7) (2007) 2038–2048.
[36] W. Cheng, E. Hllermeier, Combining instance-based learning and logistic regression for multilabel classification, Mach. Learn. 76 (2–3) (2009) 211–225.
[37] J. Frnkranz, E. Hllermeier, E.L. Menca, K. Brinker, Multilabel classification via calibrated label ranking, Mach. Learn. 73 (2) (2008) 133–153.
[38] R. Schapire, Y. Singer, BoosTexter: a boosting-based system for text categorization, Mach. Learn. 39 (2/3) (2000) 135–168.
[39] G. Tsoumakas, I. Katakis, L. Vlahavas, Random k-labelsets for multilabel classification, IEEE Trans. Knowl. Data Eng. 23 (7) (2011) 1079–1089.
[40] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Mach. Learn. 85 (3) (2011) 333–359.
[41] P. Li, H. Li, M. Wu, Multi-label ensemble based on variable pairwise constraint projection, Inf. Sci. 222 (2013) 269–281.
[42] L. Rokach, A. Schclar, E. Itach, Ensemble methods for multi-label classification, Expert Syst. Appl. 41 (16) (2014) 7507–7523.
[43] M.A. Tahir, J. Kittler, A. Bouridane, Multilabel classification using heterogeneous ensemble of multi-label classifiers, Pattern Recognit. Lett. 33 (5) (2012) 513–523.
[44] O.L. Mangasarian, D.R. Musicant, Successive overrelaxation for support vector machines, IEEE Trans. Neural Netw. 10 (5) (1999) 1032–1037.
[45] Z. Qi, Y. Tian, Y. Shi, Successive overrelaxation for Laplacian support vector machine, IEEE Trans. Neural Netw. Learn. Syst. 26 (4) (2015) 674–683.

Wei-Jie Chen received his B.S. degree and Ph.D. degree both in the College of Information from Zhejiang University of Technology, China, in 2006 and 2011. He is now an associate professor at the Zhijiang College, Zhejiang University of Technology. His research interests include pattern recognition, intelligence computation and manifold learning. He has published over 20 refereed papers.

Yuan-Hai Shao received his B.S. degree in the College of Mathematics from Jilin University, and received Ph.D. degree in the College of Science from China Agricultural University, China, in 2006 and 2011, respectively. Currently, he is an associate professor at the Zhijiang College, Zhejiang University of Technology. His research interests include optimization methods, machine learning and data mining. He has published over 30 refereed papers.

Chun-Na Li received her Master's degree and Ph.D degree in the Department of Mathematics from Harbin Institute of Technology, China, in 2009 and 2012, respectively. Currently, she is a lecturer at the Zhijiang College, Zhejiang University of Technology. Her research interests include optimization methods, machine learning and data mining.

Nai-Yang Deng received the M.Sc. degrees in the Department of Mathematics from Peking University, China, in 1967. Now, he is a professor in the College of Science, China Agricultural University, he is an honorary director of China Operations Research Society, Managing Editor Journal of Operational Research, International Operations Research Abstracts Editor. His research interests mainly including operational research, optimization, machine learning and data mining. He has published over 100 refereed papers.