Pattern Analysis and Applications (2020) 23:295–308
https://doi.org/10.1007/s10044-019-00779-2

THEORETICAL ADVANCES

ML-SLSTSVM: a new structural least square twin support vector machine for multi-label learning

Meisam Azad-Manjiri[1] · Ali Amiri[1] · Alireza Saleh Sedghpour[2]

Received: 5 August 2017 / Accepted: 7 January 2019 / Published online: 12 February 2019
© Springer-Verlag London Ltd., part of Springer Nature 2019

Abstract
Multi-label learning (MLL) is a special supervised learning task, where any single instance possibly belongs to several classes simultaneously. Nowadays, MLL methods are increasingly required by modern applications, such as protein function classification, speech recognition and textual data classification. In this paper, a structural least square twin support vector machine (SLSTSVM) classifier for multi-label learning is presented. This proposed ML-SLSTSVM focuses on the cluster-based structural information of the corresponding class in each optimization problem, which is vital for designing a good classifier in different real-world problems. This method is extended to a nonlinear version by the kernel trick. Experimental results demonstrate that proposed method is superior in generalization performance to other classifiers.

Keywords Multi-label learning · Support vector machine · Twin SVM · Structural SVM · Lest square SVM

1 Introduction

Multi-label learning (MLL) is a form of supervised learning where the learning algorithm is required to learn from a set of instances, and each instance can belong to multiple classes and so after be able to predict a set of class labels for a new instance [1]. In the past decade, such a learning issue has received a lot of attention because of many real-world applications, e.g., protein function classification [2, 3], speech recognition [4, 5], music categorization [6, 7] and image annotation [8, 9]. Due to the complex nature of this type of data, their learning is more difficult than two- or multi-class data. MLL methods can generally be divided into the following [10].

- Problem transformation methods: These methods transform the MLL problem into one or more single-label classification problems, which can be solved using existing single-label learning methods. An essential property of problem transformation methods is that they are algorithm independent. Binary relevance [10], label powerset [11] and calibrated label ranking [12] are some of the algorithms which use problem transformation methods for MLL.
- Algorithm adaptation methods: These methods handle directly the MLL problems by adapting popular learning techniques to deal with multi-label data. Support vector machine [13–15], nearest neighbor [16], neural network [17, 18] and decision tree [19, 20] are some of the machine learning techniques which have been extended for MLL. The proposed method is one of the algorithm adaptation methods that extend SVM to learn multi-label data.

Despite the works done in MLL, there is still a need to develop more efficient methods in terms of precision and other metrics in Sect. 5.2. Because the structural information of data may contain useful prior domain knowledge for training a classifier, in this paper, we first present an improvement on MLTSVM [15], called structural twin support vector machine for multi-label learning (ML-STSVM). In this method, we, respectively, embed the data structures of classes into the optimization problems of MLTSVM based on the same clustering technology of S-TWSVM [21]. Least square SVM [22] solves linear system instead of QP problem. This property improves both the generalization capability and running speed. For this reason, we present least

✉ Meisam Azad-Manjiri
m.azadm@znu.ac.ir

[1] Department of Computer Engineering, University of Zanjan, Zanjan, Iran

[2] Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran

Table 1 Symbols and notations used in this paper

Symbol/notation	Definition
$X \in \mathbb{R}^d$	Instance space of d-dimensional features
$\mathcal{Y} = \{\lambda_1, \lambda_2, \ldots, \lambda_L\}$	Finite set of labels
$D = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$	Training dataset (consisting n training instances)
$L = \lvert\mathcal{Y}\rvert$	Number of labels
n	Number of instances
d	Number of features
I_k	Index of instances that are associated with the kth label
$I_{\bar{k}}$	Index of instances that are not associated with the kth label
$c \in \mathbb{R}$	Penalty for the classification error
$\lambda \in \mathbb{R}$	Regularization parameter
$\eta \in \mathbb{R}$	Parameter that regulates the relative importance of the structural information
$\xi \in \mathbb{R}$	Slack variable
$e \in \mathbb{R}^{arb}$	Column vector of ones in real space of arbitrary dimension
$K(.,.)$	Kernel function

square version of ML-STSVM (using the proposed idea in [22, 23]) called ML-SLSTSVM. In the algorithm, we modified the primal QPPs of TSVM in least square sense and solved them with equality constraints instead of inequalities of TSVM.

To thoroughly evaluate the performance of the proposed approach, comparative studies over both synthetic and real-world multi-label datasets have been conducted. Experimental results show that ML-SLSTSVM achieves superior performance against several state-of-the-art multi-label learning algorithms. Finally, the main contribution of ML-SLSTSVM is defining a ranking-based SVM that extends the MLTSVM method [15] with considering structural information of training samples to improve the generalization capability and using the least square idea [22] to improve both generalization capability and running speed.

This paper is organized as follows. In Sect. 2, the setting of a MLL and related works are introduced. Some preliminaries are given in Sect. 3. The proposed algorithm is described in detail in Sect. 4. In Sect. 5, some simulation results are discussed. Finally, the conclusions are drawn in Sect. 6.

2 Related works

Let $X \in \mathbb{R}^d$ be an instance space of d-dimensional features and $\mathcal{Y} = \{\lambda_1, \lambda_2, \ldots, \lambda_L\}$ be a finite set of labels or classes. Each instance $x \in X$ has multiple class labels Y in $\mathcal{Y}$. Given a set of training examples consisting of n instances,$D = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, where each instance is independent and identically distributed (i.i.d.) drawn from an unknown distribution $\mathcal{D}$. $x_i \in X$ and labels $Y_i \subseteq \mathcal{Y}$ are known [11]. The goal of multi-label classification is to learn categories' properties from labeled examples and find a multi-label classifier $g : X \to 2^{\mathcal{Y}}$ that maps an instance x to its label such that specific performance criteria are optimized. Here, $2^{\mathcal{Y}}$ is the powerset of $\mathcal{Y}$, which is the set of all subsets of $\mathcal{Y}$. To facilitate our discussion, we summarize major symbols and notations used in this paper in Table 1. Others are defined following their first appearance as required.

There is another method to learn multi-label data called label ranking. In the label ranking, the goal is to learn to predict a total order, a ranking, of all possible labels for a new training example. In other words, the task is to learn the function $f : X \times \mathcal{Y} \to \mathbb{R}$ that, for a new instance x, assign a real number to each label $y \in \mathcal{Y}$ where more related label with x has greater value of f. In the label ranking, it is necessary to be considered a threshold that based on related labels separated from unrelated ones. It should be noted that in this paper, the proposed method builds a label ranking model to learn multi-label data.

As mentioned in the Introduction, current MLL algorithms have been developed with two strategies: the problem transformation and algorithm adaptation. In the following, we introduce some well-known works on both these strategies and especially focus on SVM-based methods that used for MLL problem.

The binary relevance approach (BR) [10] decomposes multi-label learning task into several independent binary classification problems, one for each label in the set of labels. Finally, this method determines the labels for a new instance by aggregating the predictions from all the classification problems. Since the BR method does not model dependencies between labels, the predictive performance of it is low.

To address this deficiency (to model labels dependency), some new methods have improved. One of these methods is the label powerset (LP) method [11]. In LP, each of the different combinations of labels in a dataset is considered to be a single label. Thus, LP transforms a multi-label problem to a multi-class problem with $2^{|L|}$ different classes, where L is the set of labels in original problem and 2^L is the powerset of all labels in L. The predictive performance of LP is greater than BR, but the main drawback of it is high computational complexity (the number of label combinations grows exponentially with the number of labels).

Ranking by pairwise comparison (RPC) [24] is another type of binary classification approach which uses problem transformation strategy. This method learns $L(L-1)/2$ binary models, one for each pair of labels. To predict a new instance by RPC, all the models are invoked and a ranking is obtained by counting the votes received by each label. This method has a good predictive performance, but the time complexity of it is quadratic.

Another strategy to classify multi-label data is the algorithm adaptation. In the following, we review some works that use it. In [18], an extension of backpropagation algorithm called backpropagation for multi-label learning (BP-MLL) is proposed. It uses multiple binary outputs as the label variables and defines a novel error function that captures the characteristics of multi-label learning. In [20], an extension of tree algorithm C4.5 is presented to handle multi-label problems. It modifies the entropy to consider instances associated with multiple labels. In [19], an algorithm based on Bayesian approach is proposed for multi-label textual data classification. It constructs a probabilistic generative model to model multiple labels associated with each document.

Multi-label k nearest neighbor (MLkNN) [16] is a k nearest-neighbor method adapted for MLL. This method classifies a new instance by voting from the labels found in its neighbors. In [25], the authors have tried to extend the ELM technique for multi-label learning by proposing a thresholding method based on ELM. In [26], an extension of AdaBoost algorithm called BoosTexter is presented for MLL problem. It maintains a set of weights over both training examples and labels to handle multiple labels.

Some works on MLL are general-purpose and we can use theme for learning different types of data [15, 16, 27, 28]. Some other works are specific-purpose and learn the specific type of data (e.g., image, sound and gene expression data) [29–34]. There are many works in the literature that are proposed for images classification. Since deep neural networks and more specifically convolutional neural networks have shown promising results in image classification, most of the MLL works have used them. Multi-label classification for image annotation [8], pedestrian attribute classification [30], facial attribute classification [31] and social image understanding [29, 32] are examples of this type of works.

Extreme multi-label learning (XML) is another problem in the area of MLL, and it is important problem since the boom of big data [35]. The objective in XML is to learn a classifier that can automatically tag a data point with the most relevant subset of labels from a large label set. Many existing MLL methods are not suitable for XML, due to the large label space (as large as in millions). Tree-based methods [36, 37] and embedding-based methods [38, 39] are two general types of methods in this area. It should be noted that our proposed method is general-purpose and does not claim to solve XML problems. In this paper, we extend SVM to handle multi-label data. Therefore, in the following we introduce some SVM-based methods for MLL problems.

Many well-known approaches extend the traditional SVM for MLL problem, e.g., RankSVM [13], RankCVM [14] and MLTSVM [15]. RankSVM is proposed via extending multi-class SVM and uses a novel ranking loss function as its empirical loss. This method solves a quadratic programming problem to rank the label set for new instances. A drawback of RankSVM is high computational complexity due to a large number of variables in its quadratic programming. To address this drawback, the RankCVM was proposed. This method combines RankSVM with CVM to construct a novel SVM-type multi-label classifier which is described as the same optimization form as binary CVM.

We cite [15] as the key reference for the proposed method, since we freely use some of the terminologies and the ideas presented in it. It proposes a novel twin support vector machine to multi-label learning called MLTSVM. This method which is an extension of twin SVM (TWSVM) [40] determines multiple nonparallel hyperplanes to capture the multi-label information embedded in data. MLTSVM solves L quadratic programming problems (L is the number of labels) to obtain L nonparallel proximal hyperplanes, and each problem is similar to binary TWSVM. Note that the optimization is done in a way that kth hyperplane is closer to the instances with the label k, and is as far as possible from the others.

3 Preliminaries

Since the proposed method is based on SVM, in this section we briefly introduce the SVM and some extension of it.

3.1 Support vector machine

Support vector machine (SVM) was originally proposed by Vapnik [41] for binary classification. SVM implements the structural risk minimization (SRM), in contrast to other

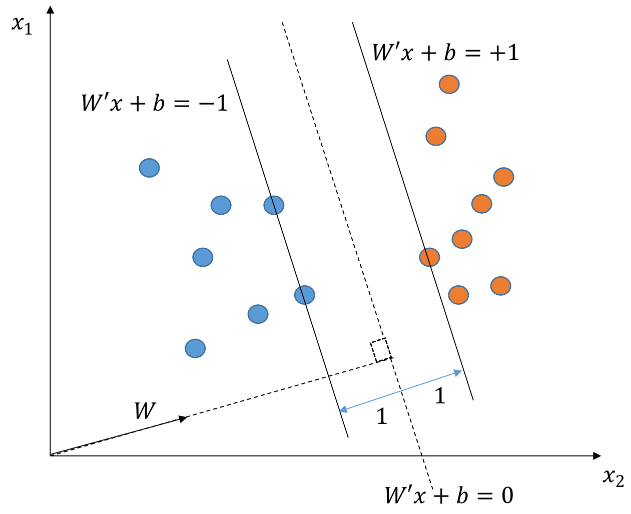

Fig. 1 Linear separation of data points using SVM classifier

machine learning approaches like conventional artificial neural network which aims at minimizing empirical risk (ERM). (SRM minimizes the upper bound of generation, whereas ERM minimizes the error on the training data.) Recently SVMs have been extended to solve the multi-class and MLL problems.

The main idea of the SVM classifier is to obtain a linear decision boundary that maximizes a separation margin. In other words, SVM tries to find a discriminant hyperplane by solving an optimization problem to separate two classes of patterns.

We consider the problem of classifying n points in the d-dimensional real space R^d, represented by the $n \times d$ matrix X, according to membership of each point X_i in the classes $+1$ or -1 as specified by a given $m \times m$ diagonal matrix D with ones or minus ones along its diagonal. For this problem, the SVM discriminant function is $d(x) = w'x + b$ and it obtained by minimizing the following objective function.

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + ce'\xi \quad \text{s.t}: D(Xw + eb) \geq e - \xi \quad \xi \geq 0 \tag{1}$$

where ξ is the n-dimensional vector of slack variables, c is a penalty parameter and e is a vector of ones of appropriate dimensions. Figure 1 shows the linear separation of data points using SVM classifier. As you see in this figure, the discriminant hyperplane lies in a space that is halfway between the two sets of points and has maximum margin.

Training algorithm of SVM performs a linear classification and builds a model that assigns new examples into one class or the other. When data points are not linearly separable, we can use nonlinear SVM, in which data points are mapped to higher-dimensional feature space where the examples become linearly separable. This mapping is complex, and training time increases exponentially with the input data dimension. To tackle with this drawback, we can use the kernel trick [42]. Kernel trick provides a way to work in a feature-rich space without explicitly mapping the points to higher dimension. This trick is done by using an appropriate kernel function, which describes the hyperplane in higher dimensions. Therefore, the art of using nonlinear SVMs is to choose the correct kernel function, which is well suited for the underlying problem.

SVM has such advantages as good generalization capacity, less limit in training set size and more robust to noise [43], but there is a main challenge in the traditional SVM and it is the high computational complexity of quadratic programming problem (QPP) [43]. The computational complexity of SVM is $O(n^3)$, where n denotes the count of training data. This drawback restricts the application of SVM to large-scale problem domains.

3.2 Twin support vector machine

The study in [40] proposed a novel binary classifier, twin support vector machine (TWSVM), the speed of which is approximately four times faster than SVM.

This method seeks two nonparallel hyperplanes such that each hyperplane is close to the patterns of one class and far away from the patterns of the other classes simultaneously. The TWSVM hyperplanes have the forms (2) and are obtained by minimizing the objective functions (3) and (4).

$$x'w_1 + b_1 = 0, x'w_2 + b_2 = 0 \tag{2}$$

$$\min_{w1,b1,\xi} \frac{1}{2}(Aw_1 + eb_1)'(Aw_1 + eb_1) + ce'\xi \quad \text{s.t}: -(Bw_1 + eb_1) + \xi \geq e, \xi \geq 0 \tag{3}$$

$$\min_{w2,b2,\xi} \frac{1}{2}(Bw_2 + eb_2)'(Bw_2 + eb_2) + ce'\xi \quad \text{s.t}: (Aw_2 + eb_2) + \xi \geq e, \xi \geq 0 \tag{4}$$

where A and B are the data points belonging to classes + 1 and − 1, respectively, and ξ is vector of slack variables. Figure 2 shows how the linear TWSVM separates training data points. When this method found the hyperplanes, a new data instance is classified based on the distance between it and the hyperplanes.

3.3 Structural twin SVM

Using prior information of training data can help to design more powerful classifiers, and the structural information of

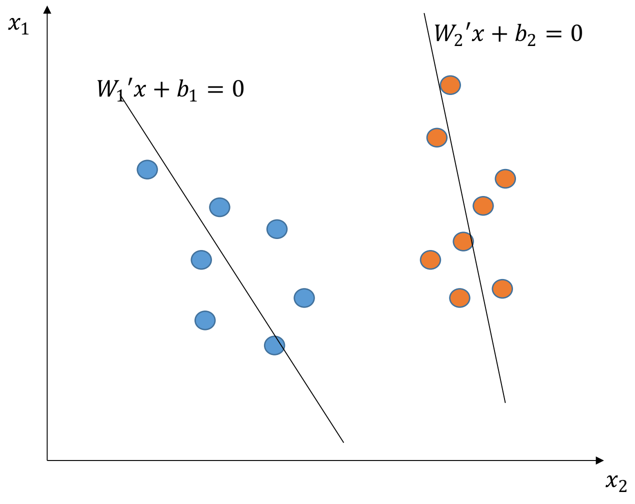

Fig. 2 Separation of training data using linear TWSVM

data points is one of them [44]. For this reason, some SVM-based methods are presented that use covariance matrix of data. STSVM or structural TWSVM is one of these methods, which was introduced in 2013 by Zhiquan [21].

In this method which is an extension of TWSVM, the derived decision hyperplanes tend to lie in the same direction with the data dispersion. The hyperplanes have the form (5) and are obtained from solving the objective functions (6) and (7).

$$f_+(x) = W_+x + b_+ = 0, f_-(x) = W_-x + b_- = 0 \quad (5)$$

$$\begin{aligned} &\min_{w_k,b_k,\xi_j} \frac{1}{2}\sum_{i\in I_k}\left\|\left(AW_+ + e_+b_+\right)\right\|^2 + c_1e_-'\xi \\ &+\frac{1}{2}c_2\left(\|w_+\|^2 + b_+^2\right) + \frac{1}{2}c_3\left(w_+'\Sigma_+w_+\right) \\ &\text{s.t}: -(BW_+ + e_-b_+) + \xi \ge e_-, \quad \xi \ge 0 \end{aligned} \quad (6)$$

$$\begin{aligned} &\min_{w_k,b_k,\xi_j} \frac{1}{2}\sum_{i\in I_k}\left\|\left(BW_- + e_-b_-\right)\right\|^2 + c_4e_+'\xi \\ &+\frac{1}{2}c_5\left(\|w_-\|^2 + b_-^2\right) + \frac{1}{2}c_6\left(w_-'\Sigma_-w_-\right) \\ &\text{s.t}: -(AW_- + e_+b_-) + \xi \ge e_+, \xi \ge 0 \end{aligned} \quad (7)$$

where $c_1, \ldots c_6$ are the pre-specified penalty factors, e_+, e_- are vectors of ones of appropriate dimensions, ξ is the vector of slack variables,$\Sigma_+ = \Sigma_{p1} + \cdots + \Sigma_{pm}$, $\Sigma_- = \Sigma_{n1} + \cdots + \Sigma_{nk}$, Σ_{Pi} and Σ_{Nj} are, respectively, the covariance matrices corresponding to the ith and jth clusters in the two classes, $i=1$, …, m, $j=1$, …, k.

3.4 Least square twin SVM

Least square twin support vector machines (LSTSVM) is another extension of SVM that attempts to solve two modified primal problems of TSVM, instead of two dual problems usually solved [23]. This method requires just the solution of two systems of linear equations for both linear and nonlinear cases, while TWSVM requires solving two quadratic programming problems (QPPs). The training time of LSTSVM is much faster than TWSVM, and this advantage allows to easily classify large datasets. The LSTSVM model can be formulated as:

$$\begin{aligned} &\min_{w_1,b_1} \frac{1}{2}\left\|\left(Aw_1 + eb_1\right)\right\|^2 + \frac{c}{2}\xi'\xi \\ &\text{s.t}: -(Bw_1 + eb_1) + \xi = e \end{aligned} \quad (8)$$

$$\begin{aligned} &\min_{w_2,b_2} \frac{1}{2}\left\|\left(Bw_2 + eb_2\right)\right\|^2 + \frac{c}{2}\xi'\xi \\ &\text{s.t}: (Aw_2 + eb_2) + \xi = e \end{aligned} \quad (9)$$

3.5 Multi-label twin support vector machine

As mentioned in the previous section, in [15] it is presented a SVM-based method called MLTSVM. This method obtains as many proximal hyperplanes as there are labels on the dataset. Each hyperplane has the form (10), where w_k and b_k are the normal vector and the bias term of the kth proximal hyperplane, respectively. These hyperplanes are obtained by minimizing the objective function (11) where C_k and λ_k are the pre-specified penalty factors, and ξ is the vector of slack variables.

$$f_k(x): w_k'x_i + b_k = 0, k \in \{1, 2, \ldots, L\} \quad (10)$$

$$\begin{aligned} &\min_{w_k,b_k,\xi_j} F = \frac{1}{2}\sum_{i\in I_k}\left\|\left(w_k'x_i + b_k\right)\right\|^2 + c_k\sum_{j\in I_{\bar{k}}}\xi_j + \frac{1}{2}\lambda_k\left(\|w_k\|^2 + b_k^2\right) \\ &\text{s.t}: -(w_k'x_j + b_k) \ge 1 - \xi_j, \quad \xi_j \ge 0\,\forall j \in I_{\bar{k}} \end{aligned} \quad (11)$$

Using Lagrangian function and Karush–Kuhn–Tucker (KKT) conditions and by introducing the Lagrangian multipliers α, the dual QPP of (11) can be represented as

$$\begin{aligned} &\max_{\alpha_j} \sum_{j\in I_{\bar{k}}}\alpha_j - \frac{1}{2}\sum_{j_1\in I_{\bar{k}}}\sum_{j_2\in I_{\bar{k}}}\alpha_{j1}\alpha_{j2}z_{j1}'\left(\sum_{i\in I_k}z_iz_i' + \lambda_kI\right)^{-1}z_{j2} \\ &\text{s.t}: 0 \le \alpha_j \le c_k, \quad j \in I_{\bar{k}} \end{aligned} \quad (12)$$

After obtaining the solutions αj from the dual problem (12), the kth proximal hyperplane ($k=1$, …, K) can be constructed according to (13).

$$\begin{bmatrix} w_k \\ b_k \end{bmatrix} = -\left(\sum_{i\in I_k} z_i z_i' + \lambda_k I \right)^{-1} \left(\sum_{j\in I_{\bar{k}}} z_j \alpha_j \right) \tag{13}$$

4 Proposed method

In this section, we first introduce a structural version of MLTSVM [15], called structural twin SVM for multi-label learning (ML-STSVM) and then propose the least square version of it called structural least squared twin support vector machine for multi-label learning (ML-SLSTSVM). Similar to MLTSVM, let us use I_k and $I_{\bar{k}}$ to denote two complementary index sets, and if the instance x_i is associated with the kth label, then $i \in I_k$; otherwise, $i \in I_{\bar{k}}$.

4.1 ML-STSVM

4.1.1 Linear ML-STSVM

For the linear case, the ML-STSVM determines a nonparallel hyperplane for each label

$$f_k(x) = W_k' x + b_k = 0; k = 1, \dots, L \tag{14}$$

where L is the number of labels, the weight vector $W_k \in \mathbb{R}^d$ determines the hyperplane's orientation and the bias $b_k \in \mathbb{R}$ determines the hyperplane's offset relative to the system of coordinates. Here, each hyperplane f_i is closer to the instances with the label k and is as far as possible from the others. These nonparallel hyperplanes can be obtained by solving the following optimization problems for each k = 1, 2, …, L.

$$\begin{aligned} \min_{w_k, b_k, \xi_j} F = & \frac{1}{2} \sum_{i\in I_k} \left\| \left(w_k' x_i + b_k \right) \right\|^2 + c_k \sum_{j\in I_{\bar{k}}} \xi_j \\ & + \frac{1}{2}\lambda_k \left(\|w_k\|^2 + b_k^2 \right) + \frac{1}{2}\eta_k \left(w_k' \Sigma_k w_k \right) \\ & \text{s.t} : -(w_k' x_j + b_k) \geq 1 - \xi_j, \xi_j \geq 0 \forall j \in I_{\bar{k}} \end{aligned} \tag{15}$$

where c_k, λ_k and η_k are the pre-specified penalty factors, ξ_j is the slack variable for each training data and $\Sigma_k = \Sigma_{k_1} + \cdots + \Sigma_{k_{nk}}$, Σ_{k_i} is the covariance matrix corresponding to the ith cluster in training data associated with the kth label, i = 1, 2, …, n_k.

The first term in the objective function is the sum of squared distances from the hyperplane to points of kth label. The second term is a ℓ_2 regularization term that favors sparse models (weighted by $\lambda \in \mathbb{R}$) [45], and the third term embodies the structural information of each label.

The Lagrangian corresponding to the problem (15) is given by

$$\begin{aligned} L = & \frac{1}{2} \sum_{i\in I_k} \left\| \left(w_k' x_i + b_k \right) \right\|^2 + c_k \sum_{j\in I_{\bar{k}}} \xi_j + \frac{1}{2}\lambda_k \left(\|w_k\|^2 + b_k^2 \right) \\ & + \frac{1}{2}\eta_k \left(w_k' \Sigma_k w_k \right) + \sum_{j\in I_{\bar{k}}} \alpha_j \left(w_k' x_j + b_k + 1 - \xi_j \right) - \sum_{j\in I_{\bar{k}}} \beta_j \xi_j \end{aligned} \tag{16}$$

where $\alpha = \alpha_1, \dots, \alpha_{|I_{\bar{k}}|}$ and $\beta = \beta_1, \dots, \beta_{|I_{\bar{k}}|}$ are the Lagrange multipliers. According to the KKT conditions (Karush–Kuhn–Tucker), we know that there exist Lagrange multipliers satisfying:

$$\frac{\partial L}{\partial w_k} = 0 \Rightarrow \sum_{i\in I_k} \left(w_k' x_i + b_k \right) x_i + \lambda_k w_k + \eta_k \Sigma_k w_k + \sum_{j\in I_{\bar{k}}} x_j \alpha_j = 0 \tag{17}$$

$$\frac{\partial L}{\partial b_k} = 0 \Rightarrow \sum_{i\in I_k} \left(w_k' x_i + b_k \right) + \lambda_k b_k + \sum_{j\in I_{\bar{k}}} \alpha_j = 0 \tag{18}$$

$$\frac{\partial L}{\partial \xi_j} = 0 \Rightarrow c_k - \alpha_j - \beta_j = 0; j \in I_{\bar{k}} \tag{19}$$

$$-\left(w_k' x_j + b_k \right) \geq 1 - \xi_j; \xi_j \geq 0 \quad \forall j \in I_{\bar{k}} \tag{20}$$

$$\sum_{j\in I_{\bar{k}}} \alpha_j \left(w_k' x_j + b_k + 1 - \xi_j \right) = 0 \tag{21}$$

$$\sum_{j\in I_{\bar{k}}} \beta_j \xi_j = 0 \tag{22}$$

$$\alpha_j \geq 0, \beta_j \geq 0; \quad \forall j \in I_{\bar{k}} \tag{23}$$

According to (19) and (23):

$$0 \leq \alpha_j \leq c_k$$

Obviously, combining (17) and (18) leads to:

$$\sum_{i\in I_k} \begin{bmatrix} x_i \\ 1 \end{bmatrix} \begin{bmatrix} x_i 1 \end{bmatrix} \begin{bmatrix} w_k \\ b_k \end{bmatrix} + \lambda_k \begin{bmatrix} w_k \\ b_k \end{bmatrix} + \eta_k \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_k \\ b_k \end{bmatrix} + \sum_{j\in I_{\bar{k}}} \begin{bmatrix} x_j \\ 1 \end{bmatrix} \alpha_j = 0 \tag{24}$$

If we define $z_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix}$, $s_k = \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix}$ and $\theta_k = \begin{bmatrix} w_k \\ b_k \end{bmatrix}$, then Eq. (24) can be rewritten as:

$$\sum_{i\in I_k} z_i z_i' \theta_k + \lambda_k \theta_k + \eta_k s_k \theta_k + \sum_{j\in I_{\bar{k}}} z_j \alpha_j = 0 \tag{25}$$

$$\Rightarrow \left(\sum_{i\in I_k} z_i z_i' + \lambda_k I + \eta_k s_k \right) \theta_k = -\sum_{j\in I_{\bar{k}}} z_j \alpha_j \tag{26}$$

$$\Rightarrow \theta_k = \begin{bmatrix} w_k \\ b_k \end{bmatrix} = -\left(\sum_{i\in I_k} z_i z_i' + \lambda_k I + \eta_k s_k \right)^{-1} \left(\sum_{j\in I_{\bar{k}}} z_j \alpha_j \right) \tag{27}$$

Substituting (27) and (19) into (16), we obtain the dual formulation of (15):

$$\max_{\alpha_j} \sum_{j\in I_{\bar{k}}} \alpha_j - \frac{1}{2}\sum_{j_1\in I_{\bar{k}}}\sum_{j_2\in I_{\bar{k}}} \alpha_{j1}\alpha_{j2}z'_{j1}\left(\sum_{i\in I_k} z_i z'_i + \lambda_k I + \eta_k s_k\right)^{-1} z_{j2}$$
$$\text{s.t}: 0 \le \alpha_j \le c_k, \quad j \in I_{\bar{k}} \tag{28}$$

We can rewrite this problem in the following unified matrix form:

$$\max_{\alpha} e'\alpha - \frac{1}{2}\alpha' Q\alpha$$
$$\text{s.t}: 0 \le \alpha \le c_k e \tag{29}$$

where $\alpha = \alpha_1, \dots, \alpha_{|I_{\bar{k}}|}$ and Q is defined by

$$Q = G_k\left(H'_k H_k + \lambda_k I + \eta_k s_k\right)^{-1} G'_k \tag{30}$$

$$G_k = \left[X_{I_{\bar{k}}}\ e\right], H_k = \left[X_{I_k}\ e\right] \tag{31}$$

In Eq. (31), $X_{I_k} = \{X_i | i \in I_k\}$ (instances associated with the kth label) and $X_{I_{\bar{k}}} = \{X_i | i \in I_{\bar{k}}\}$ (the other instances).

After optimizing the dual problem (29) for each label $k=1, \dots, L$ and obtaining α, we can substitute it in Eq. (27) and construct the decision strategies (14).

Now, we can use these decision strategies to determine the labels of test data. For this purpose, we must determine a threshold T_k ($k=1, \dots, L$) for each label and then assign the label k to a new instance x if the distance between x and $f_k(x)$ is less than or equal to T_k.

$$d_k(x) = \text{distance}\left(x, f_k(x)\right) = \left(W'_k x + b_k\right)/\|W_k\| \tag{32}$$

To determine thresholds T_k, we used the proposed idea in [15]. For this, we set $T_k = T$ for all $k=1$ to L, where $T = \min\left(1/\|W_k\|\right), k=1, \dots, L$.

4.1.2 Nonlinear ML-STSVM

In this subsection, we extend the linear ML-STSVM to the nonlinear case by considering the following kernel generated surfaces:

$$f_k(x) = \boldsymbol{U}'_{\boldsymbol{k}}\mathrm{K}(\boldsymbol{X}, x) + b_k = 0; \quad k = 1, \dots, L \tag{33}$$

where $\boldsymbol{X}$ denotes all the training instances, $\boldsymbol{U}_{\boldsymbol{k}}$ is the weight vector in the kernel space and $K(.,.)$ is a proper chosen kernel. The primal QPPs of the nonlinear ML-STSVM corresponding to the surface (33) is given in (35).

$$\min_{U_k, b_k, \xi_j} \frac{1}{2}\sum_{i\in I_k}\left\|\left(\boldsymbol{U}'_k K(\boldsymbol{X}, x_i) + b_k\right)\right\|^2 + c_k\sum_{j\in I_{\bar{k}}}\xi_j +$$
$$+\frac{1}{2}\lambda_k\left(\|\boldsymbol{U}_k\|^2 + b_k^2\right) + \frac{1}{2}\eta_k\left(\boldsymbol{U}'_k \Sigma_k^{\Phi}\boldsymbol{U}_k\right) \tag{34}$$
$$\text{s.t}: -\left(\mathbf{U}'_k\mathrm{K}(\mathbf{X}, x_j) + b_k\right) \ge 1 - \xi_j, \quad \xi_j \ge 0\ \forall j \in \mathrm{I}_{\bar{k}}$$

where $\Sigma_k^{\Phi} = \Sigma_{k_1}^{\Phi} + \dots + \Sigma_{k_{nk}}^{\Phi}$, $\Sigma_{k_i}^{\Phi}$ is the covariance matrix corresponding to the ith cluster in training data associated with the kth label, $i=1, 2, \dots, n_k$ by the kernel Ward's linkage clustering [21]. Similar to previous subsection, we can use Lagrangian multipliers and derive the dual formulation of the QPP (21) and the solution as follows.

$$\max_{\alpha_j} \sum_{j\in I_{\bar{k}}} \alpha_j - \frac{1}{2}\sum_{j_1\in I_{\bar{k}}}\sum_{j_2\in I_{\bar{k}}} \alpha_{j1}\alpha_{j2}z'_{j1}\left(\sum_{i\in I_k} z_i z'_i + \lambda_k I + \eta_k s_k\right)^{-1} z_{j2}$$
$$\text{s.t}: 0 \le \alpha_j \le c_k, \quad j \in I_{\bar{k}} \tag{35}$$

where $z_i = \begin{bmatrix}\mathrm{K}(\boldsymbol{X}, x_i)\\ 1\end{bmatrix}$ and $s_k = \begin{bmatrix}\Sigma_k^{\Phi} & 0\\ 0 & 0\end{bmatrix}$.

$$\theta_k = \begin{bmatrix}\boldsymbol{U}_k\\ b_k\end{bmatrix} = -\left(\sum_{i\in I_k} z_i z'_i + \lambda_k I + \eta_k s_k\right)^{-1}\left(\sum_{j\in I_{\bar{k}}} \boldsymbol{z_j\alpha_j}\right) \tag{36}$$

The unified matrix form of (35) can be written as (37).

$$\max_{\alpha} e'\alpha - \frac{1}{2}\alpha' Q\alpha$$
$$\text{s.t}: 0 \le \alpha \le c_k e \tag{37}$$

where $\alpha = \alpha_1, \dots, \alpha_{|I_{\bar{k}}|}$ and Q is defined by

$$Q = G_k\left(H'_k H_k + \lambda_k I + \eta_k s_k\right)^{-1} G'_k \tag{38}$$

$$G_k = \left[\mathrm{K}\left(\boldsymbol{X}, X_{I_{\bar{k}}}\right)e\right], H_k = \left[\mathrm{K}\left(\boldsymbol{X}, X_{I_k}\right)e\right] \tag{39}$$

After optimizing the dual problem (36) for each label $k=1, \dots, L$ and obtaining α, we can substitute it in (37) and construct the decision strategies in (32). Similar to linear case, we can use the hyperplanes to determine the labels of test data based on the distance between them and the hyperplanes.

4.2 ML-SLSTSVM

In this subsection, we introduce a least square version of ML-STSVM called structural least squared twin support vector machine for multi-label learning (ML-SLSTSVM). Following the idea of PSVM [46], the decision functions of ML-SLSTSVM are obtained extremely fast and simple by solving the primal problem directly.

4.2.1 Linear ML-SLSTSVM

Here we modify the primal problem (15) of linear ML-STSVM in least square sense as (40), with the inequality constraints replaced with equality constraints.

$$\begin{aligned}\min_{w_k,b_k,\xi_j} F = &\frac{1}{2}\sum_{i\in I_k}\left\|\left(w_k'x_i+b_k\right)\right\|^2 + c_k\sum_{j\in I_{\bar k}}\left\|\xi_j\right\|^2 + \frac{1}{2}\lambda_k\left(\left\|w_k\right\|^2+b_k^2\right)\\ &+\frac{1}{2}\eta_k\left(w_k'\Sigma_k w_k\right)\\ &\text{s.t}: -\left(w_k'x_j+b_k\right) = 1-\xi_j \quad \forall j\in I_{\bar k}\end{aligned} \tag{40}$$

where the variables are similar to those defined for (15). On substituting the equality constraint of (41) into the objective function of it, we can get the following unconstrained optimization problem.

$$\begin{aligned}L = &\frac{1}{2}\sum_{i\in I_k}\left\|\left(w_k'x_i+b_k\right)\right\|^2 + c_k\sum_{j\in I_{\bar k}}\left\|w_k'x_j+b_k+1\right\|^2\\ &+\frac{1}{2}\lambda_k\left(\left\|w_k\right\|^2+b_k^2\right)+\frac{1}{2}\eta_k\left(w_k'\Sigma_k w_k\right)\end{aligned} \tag{41}$$

Setting the gradient of (41) with respect to w_k and b_k to zero gives:

$$\begin{aligned}\frac{\partial L}{\partial w_k} = 0 \Rightarrow &\sum_{i\in I_k}\left(w_k'x_i+b_k\right)x_i + c_k\sum_{j\in I_{\bar k}}\left(w_k'x_i+b_k+1\right)x_j\\ &+\lambda_k w_k+\eta_k\Sigma_k w_k = 0\end{aligned} \tag{42}$$

$$\frac{\partial L}{\partial b_k} = 0 \rightarrow \sum_{i\in I_k}\left(w_k'x_i+b_k\right) + c_k\sum_{j\in I_{\bar k}}\left(w_k'x_i+b_k+1\right)+\lambda_k b_k = 0 \tag{43}$$

Combining (42) and (43) leads to:

$$\begin{aligned}&\sum_{i\in I_k}\begin{bmatrix}x_i\\1\end{bmatrix}\begin{bmatrix}x_i & 1\end{bmatrix}\begin{bmatrix}w_k\\b_k\end{bmatrix} + c_k\sum_{j\in I_{\bar k}}\begin{bmatrix}x_j\\1\end{bmatrix}\begin{bmatrix}x_j & 1\end{bmatrix}\begin{bmatrix}w_k\\b_k\end{bmatrix}\\ &+c_k\sum_{j\in I_{\bar k}}\begin{bmatrix}x_j\\1\end{bmatrix}+\lambda_k\begin{bmatrix}w_k\\b_k\end{bmatrix}+\eta_k\begin{bmatrix}\Sigma_k & 0\\0 & 0\end{bmatrix}\begin{bmatrix}w_k\\b_k\end{bmatrix} = 0\end{aligned} \tag{44}$$

Defining $z_i = \begin{bmatrix}x_i\\1\end{bmatrix}$, $s_k = \begin{bmatrix}\Sigma_k & 0\\0 & 0\end{bmatrix}$ and $\theta_k = \begin{bmatrix}w_k\\b_k\end{bmatrix}$, then Eq. (44) can be rewritten as:

$$\sum_{i\in I_k} z_iz_i'\theta_k + c_k\sum_{j\in I_{\bar k}} z_jz_j'\theta_k+\lambda_k\theta_k+\eta_k s_k\theta_k = -c_k\sum_{j\in I_{\bar k}} z_j' \tag{45}$$

$$\Rightarrow\left(\sum_{i\in I_k} z_iz_i' + c_k\sum_{j\in I_{\bar k}} z_jz_j'+\lambda_k I+\eta_k s_k\right)\theta_k = -c_k\sum_{j\in I_{\bar k}} z_j' \tag{46}$$

Finally, the solution of (40) can be obtained by (47)

$$\Rightarrow\theta_k = \begin{bmatrix}w_k\\b_k\end{bmatrix} = -c_k\left(\sum_{i\in I_k} z_iz_i' + c_k\sum_{j\in I_{\bar k}} z_jz_j'+\lambda_k I+\eta_k s_k\right)^{-1}\left(\sum_{j\in I_{\bar k}} z_j'\right) \tag{47}$$

Equation (47) can be rewritten in the following unified matrix form:

$$\begin{bmatrix}w_k\\b_k\end{bmatrix} = -c_k\left(H_k'H_k+c_kG_k'G_k+\lambda_k I+\eta_k s_k\right)^{-1}G_k'e \tag{48}$$

where G_k and H_k are:

$$G_k = \begin{bmatrix}X_{I_{\bar k}} & e\end{bmatrix}, H_k = \begin{bmatrix}X_{I_k} & e\end{bmatrix} \tag{49}$$

We can obtain the decision strategies as (14) by solving Eq. (48) and use them to determine the labels of test data.

4.2.2 Nonlinear ML-SLSTSVM

In this subsection, we propose nonlinear ML-SLSTSVM by introducing kernel function $K\left(x,x'\right) = \left(\boldsymbol{\Phi}(x).\boldsymbol{\Phi}\left(x'\right)\right)$ and the corresponding transformation: $\boldsymbol{X} = \boldsymbol{\Phi}(x)$ where $\boldsymbol{X}\in\boldsymbol{H}$ and $\boldsymbol{H}$ is the Hilbert space. In order to extend our algorithm to nonlinear cases, we consider the kernel generated surfaces (33), instead of planes.

Then the optimization problem of nonlinear ML-SLSTSVM is constructed as follows:

$$\begin{aligned}\min_{w_k,b_k,\xi_j} F = &\frac{1}{2}\sum_{i\in I_k}\left\|\left(\boldsymbol{U}_k'K\left(\boldsymbol{X},x_i\right)+b_k\right)\right\|^2\\ &+c_k\sum_{j\in I_{\bar k}}\left\|\xi_j\right\|^2+\frac{1}{2}\lambda_k\left(\left\|\boldsymbol{U}_k\right\|^2+b_k^2\right)\\ &+\frac{1}{2}\eta_k\left(\boldsymbol{U}_k'\Sigma_k^{\Phi}\boldsymbol{U}_k\right)\\ &\text{s.t}: -\left(\boldsymbol{U}_k'K\left(\boldsymbol{X},x_j\right)+b_k\right) = 1-\xi_j \quad \forall j\in I_{\bar k}\end{aligned} \tag{50}$$

Similar to linear case, we can obtain the unknown variables on the decision boundaries (33) ($\boldsymbol{U}_k$ and b_k) by substituting the equality constraint of (50) into the objective function of it and then differentiating obtained function with respect to $\boldsymbol{U}_k$ and b_k.

$$\begin{bmatrix}\boldsymbol{U}_k\\b_k\end{bmatrix} = -c_k\left(H_k'H_k+c_kG_k'G_k+\lambda_k I+\eta_k s_k\right)^{-1}G_k'e \tag{51}$$

where $G_k = \left[K\left(\boldsymbol{X},X_{I_{\bar k}}\right)\ e\right]$, $H_k = \left[K\left(\boldsymbol{X},X_{I_k}\right)\ e\right]$ and $s_k = \begin{bmatrix}\Sigma_k^{\Phi} & 0\\0 & 0\end{bmatrix}$.

After solving Eq. (51), we can obtain the normal vector and the bias term of the kth proximal surface (33). Therefore, we can construct a similar prediction strategy as the linear case to predict the labels of unseen instances.

5 Experiments and results

In this section at first, the used datasets are introduced, and then, evaluation criteria for measuring the efficiency of MLL algorithms are expressed. Next, the parameter setting is described, and finally, the experimental results and parameter sensitivity analysis are given. All experiments have been implemented in MATLAB R2010b on a personal computer (PC) with an Intel (R) Core-i5 processor (2.3 GHz) and 4 GB random-access memory (RAM).

5.1 Benchmark datasets

To evaluate the proposed ML-SLSTSVM algorithm, in this section we have used several synthetic and real-world datasets. All synthetic data are generated by Mldatagen according to some predefined parameters such as the used strategies (hyperspheres or hypercubes), number of relevant, irrelevant and redundant features, number of instances and number of labels. We add 5% noises to the labels of each instance to make the learning tasks more challengeable. Also, the real datasets Emotion, Birds, Yeast, Flags and Medical are widely used for evaluating multi-label learning methods. These datasets represent a wide range of domains (include music, text, image and biology), sizes (from 194 to 2417), features (from 19 to 1449) and labels (from 19 to 45).

All real-world datasets (summarized in Table 3) were downloaded from the MULAN multi-label dataset repositories [47].[1] Note that all samples are normalized such that the continuous features are located in the range [0,1] before learning. Tables 2 and 3 show the used synthetic and real datasets in more details.

5.2 Evaluation criteria

There are different criteria to evaluate the performance of multi-label data classifiers. For this reason, in this paper, we used five standard criteria, which are explained in more detail below. In all criteria, n, L, y_i and $\bar{y}_i$ denote, respectively, the number of training data, the number of labels, the set of labels relevant to the ith instance and the set of labels that are irrelevant to it. In addition, the function $f_y(x)$ is a real-valued function ($f : X \times \mathcal{Y} \to \mathbb{R}$) that returns the confidence of being proper label of x and $\mathrm{rank}_f(x, y)$ returns the rank of y in $\mathcal{Y}$ based on the descending order induced from $f(x)$.

Table 2 Synthetic dataset statistics

Dataset	#Sample	Features			#Label
		Relevant	Irrelevant	Redundant	
Hyperspheres (HS1)	400	15	5	0	5
Hyperspheres (HS2)	600	35	10	5	10
Hypercubes (HC1)	400	15	5	0	5
Hypercubes (HC2)	600	35	10	5	10

Table 3 Real dataset statistics

Dataset	Domain	#Sample	#Feature	#Label
Emotions	Music	593	72	6
Birds	Audio	645	260	19
Yeast	Biology	2417	103	14
Flags	Images	194	19	17
Medical	Text	978	1449	45

5.2.1 Hamming loss

This criterion indicates the fraction of labels that are incorrectly predicted to the total number of labels.

$$\mathrm{Hloss} = \frac{1}{n \times L} \sum_{i=1}^{n} \sum_{j=1}^{l} \left(h_j(x_i) \neq y_j\right) \tag{52}$$

5.2.2 Ranking loss

This metric is used for ranking-based algorithms and measures the average fraction of label pairs that are reversely ordered. For example, an irrelevant label is ranked higher than a relevant label.

$$\mathrm{Rloss} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{|\mathbf{y_i}||\bar{\mathbf{y}}_\mathbf{i}|} \left| \left\{ (y'y'') | f_{y' \in \mathbf{y_i}}(x_i) \leq f_{y'' \in \bar{\mathbf{y}}_\mathbf{i}}(x_i) \right\} \right| \right) \tag{53}$$

5.2.3 Coverage

This measure evaluates how many steps are needed, on average, to move down the ranked label list so as to cover all the true labels of an instance.

$$\mathrm{Coverage} = \frac{1}{n} \sum_{i=1}^{n} \max_{y \in \mathbf{y_i}} \mathrm{rank}_f(x_i, y) - 1 \tag{54}$$

[1] http://mulan.sourceforge.net/datasets-mlc.html.

5.2.4 One error

The one error evaluates the fraction of examples whose the label with the best rank computed by the classification algorithm is not in the relevant label set.

$$\text{Oerror} = (1/n)\sum_{i=1}^{n} H(x_i)$$
$$H(x_i) = \begin{cases} 0 & \text{if } \operatorname{argmax} f_y(x_i) \in \mathbf{y_i} \\ 1 & \text{otherwise} \end{cases} \tag{55}$$

5.2.5 Average precision

Average precision evaluates the average fraction of relevant labels ranked higher than a particular label $y \in \mathbf{y_i}$.

$$\text{A.percision} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{|\mathbf{y_i}|}\sum_{y\in\mathbf{y_i}}\frac{\left|\left\{\left(y' \in \mathbf{y_i}\right)|\text{rank}_f\left(x_i, y'\right) \le \text{rank}_f\left(x_i, y\right)\right\}\right|}{\text{rank}_f\left(x_i, y\right)}\right) \tag{56}$$

The smaller the values for all previous measures except the average precision, the better the performance is.

5.3 Parameters setting

It is clear that the performance of SVM-based classification methods depends on the choices of parameters [48]. In these simulations, cross-validation was used to find the best parameters for each dataset. Due to the better performance of the kernel version, in our experiments we only consider the Gaussian kernel function $K(x_i, x_j) = \exp\left(-\left\|x_i - x_j\right\|^2/\gamma^2\right)$ as it yields great generalization performance. To reduce the parameter selection complexity, we set $C_i = C, \eta_i = \eta$ and $\lambda_i = \lambda$ for all $i = 1, 2, \ldots, L$. The value of the parameters (C, η and λ) is selected from the set $(2^{-10}, 2^{-9}, \ldots, 2^{9}, 2^{10})$ by cross-validation and γ is fixed to be 1.

5.4 Experimental results

In order to evaluate the proposed ML-SLSTSVM, we investigate its efficiency using the above criteria on the presented datasets. In our experiments, we focus on comparison between our ML-SLSTSVM and several state-of-the-art multi-label learning methods including calibrated label ranking (CLR) [11], MLkNN [16], RankSVM [13] and MLTSVM [15]. Since we have used the same datasets and algorithms as in [15] to evaluate the performance of our proposed approach, we used the results reported on it.

Table 4 shows the mean and the deviation of tenfold cross-validation learning results of each classifier on the synthetic and real datasets, where the best result on each dataset is shown in boldface. To compare these methods statistically, we choose nonlinear ML-SLSTSVM as a baseline, and compare whether our method is statistically better than one of the remaining four methods.

We performed statistical significance tests using a Wilcoxon rank-sum test [49] with $\alpha = 0.05$ to ensure the statistical significance of the results comparison in terms of evaluation metrics. The result of the test is presented at the end of each comparison. If the differences in the results are statistically significant, a "$\ll$" symbol is shown in the tables. In these tables, the symbols $\uparrow$ and $\downarrow$ after the title of each evaluation metric define the expected behavior of it, with $\downarrow$ meaning that the lowest values are the best ones and $\uparrow$ meaning that the highest values are the best ones.

The results show that ML-SLSTSVM outperforms the other approaches in all five metrics. We can also see that the performance of proposed algorithm is statistically significantly better than other compared algorithms in terms of all evaluation metrics. For example, on the Flags dataset, our method achieves 2.8%, 4.3%, 7.7%, 2.4% and 6.3% relative improvement in terms of the five evaluation criteria over MLTSVM that obtains the second-best results on this dataset.

Table 5 presents the average CPU time of our method and other compared methods. We used the Wilcoxon signed-rank test to ensure the statistical significance of the results comparison in terms of CPU time. In this test, α was set to 0.05. The results of the test are presented at the bottom of Table 5. From the table, we can see that in terms of computational cost, the nonlinear ML-SLSTSVM method takes slightly more time than the MLkNN and MLTSVM methods; however, the difference is not significant.

5.5 Sensitivity analysis

In the ML-SLSTSVM model, there are a few parameters. The first parameter is the C, which is used as a penalty for the classification error and it controls the trade-off between allowing training errors and forcing rigid margins. The second parameter is the η, which is the parameter that regulates the relative importance of the structural information within the clusters. Finally, the third is the λ, which is the regularization parameter.

In order to examine the sensitivity of these parameters in classification accuracy of ML-SLSTSVM, we conducted a set of experiments by varying one parameter and fixing the other two to the values elected by cross-validation. The results on ranking loss and average precision are illustrated in Fig. 3. (The x-axis of all charts represents the log2 of parameters value.) ML-SLSTSVM had similar situations on

Table 4 Predictive performance of each comparing algorithm (mean + SD) on the real-world and synthetic datasets

Dataset	CLR	MLkNN	RankSVM	MLTSVM (linear)	ML-SLSTSVM (linear)	ML-SLSTSVM (nonlinear)
Hamming loss ↓						
Emotion	.257 ± .039	.209 ± .028	.228 ± .036	.206 ± .025	.195 ± .027	**.178 ± .021**
Flags	.271 ± .063	.286 ± .062	.281 ± .072	.266 ± .059	.269 ± .056	**.256 ± .051**
Birds	.071 ± .013	.057 ± .011	.064 ± .023	**.049 ± .018**	.065 ± .026	.055 ± .015
Yeast	.224 ± .012	.209 ± .017	.219 ± .027	.201 ± .015	.205 ± .014	**.187 ± .016**
Medical	.162 ± .041	.170 ± .052	.107 ± .017	.116 ± .015	.112 ± .013	**.105 ± .013**
HS1	.281 ± .037	.254 ± .028	.272 ± .034	.250 ± .029	.273 ± .026	**.224 ± .022**
HS2	.195 ± .020	.192 ± .010	.191 ± .012	.181 ± .014	.216 ± .013	**.174 ± .019**
HC1	.081 ± .020	.053 ± .018	.062 ± .015	.055 ± .016	.050 ± .015	**.044 ± .014**
HC2	.131 ± .192	.122 ± .127	.153 ± .021	.117 ± .018	.080 ± .012	**.071 ± .014**
W-test	≪	≪	≪	≪	≪	
Average precision ↑						
Emotion	.772 ± .035	.762 ± .049	.773 ± .042	.781 ± .029	.809 ± .033	**.824 ± .028**
Flags	.827 ± .073	.821 ± .067	.786 ± .083	.831 ± .081	.819 ± .077	**.842 ± .070**
Birds	.533 ± .081	.503 ± .070	.494 ± .091	.541 ± .105	.606 ± .099	**.631 ± .087**
Yeast	.743 ± .016	.758 ± .017	.737 ± .039	.764 ± .019	.761 ± .022	**.774 ± .017**
Medical	.802 ± .048	.795 ± .053	.894 ± .041	.812 ± .045	.900 ± .017	.910 ± .014
HS1	.675 ± .037	.668 ± .038	.672 ± .040	**.712 ± .046**	.652 ± .039	.705 ± .041
HS2	.517 ± .046	.489 ± .051	.506 ± .038	.514 ± .041	.643 ± .035	**.656 ± .031**
HC1	.955 ± .019	.957 ± .014	.948 ± .019	.960 ± .017	.955 ± .016	**.966 ± .014**
HC2	.743 ± .040	.755 ± .049	.716 ± .056	.761 ± .041	.827 ± .076	**.849 ± .040**
W-test	≪	≪	≪	≪	≪	
Coverage ↓						
Emotion	1.820 ± .205	1.805 ± .198	1.977 ± .257	1.794 ± .317	1.723 ± .302	**1.665 ± .260**
Flags	3.609 ± .468	3.758 ± .506	3.861 ± .614	**3.729 ± .508**	3.867 ± .590	3.733 ± .503
Birds	3.313 ± .710	3.024 ± .705	4.172 ± 1.05	3.218 ± .688	2.526 ± .709	**2.336 ± .668**
Yeast	6.757 ± .238	6.268 ± .313	7.129 ± .452	6.312 ± .357	6.262 ± .390	**6.138 ± .317**
Medical	1.932 ± .401	2.592 ± .480	2.012 ± .552	1.980 ± .461	1.380 ± .128	**1.352 ± .119**
HS1	1.633 ± .179	1.943 ± .190	1.429 ± .181	1.682 ± .175	1.557 ± .188	**1.319 ± .163**
HS2	4.241 ± .346	4.136 ± .377	4.736 ± .363	4.211 ± .360	4.746 ± .378	**2.384 ± .290**
HC1	.575 ± .149	.442 ± .140	.611 ± .154	.545 ± .141	.509 ± .138	**.470 ± .134**
HC2	2.612 ± .302	2.631 ± .298	2.854 ± .271	2.62 ± .280	2.307 ± .282	**2.094 ± .250**
W-test	≪	≪	≪	≪	≪	
Ranking loss ↓						
Emotion	.178 ± .030	.173 ± .041	.158 ± .041	.163 ± .029	.151 ± .038	**.139 ± .032**
Flags	.236 ± .062	.214 ± .060	.226 ± .078	.206 ± .072	.212 ± .075	**.201 ± .062**
Birds	.137 ± .034	.141 ± .032	**.122 ± .053**	.125 ± .045	.153 ± .044	.129 ± .036
Yeast	.180 ± .013	.179 ± .016	.169 ± .019	.175 ± .009	.175 ± .014	**.160 ± .010**
Medical	.029 ± .001	.040 ± .002	.0195 ± .003	.0190 ± .002	.018 ± .001	**.017 ± .002**
HS1	.313 ± .046	.352 ± .030	.310 ± .046	.307 ± .040	.309 ± .041	**.249 ± .038**
HS2	.354 ± .044	.359 ± .030	.359 ± .038	.344 ± .041	.262 ± .030	**.223 ± .038**
HC1	.057 ± .029	.051 ± .015	.056 ± .019	.055 ± .024	.055 ± .018	**.046 ± .016**
HC2	.173 ± .015	.169 ± .023	.168 ± .026	.153 ± .020	.123 ± .012	**.120 ± .017**
W-test	≪	≪	≪	≪	≪	
One error ↓						
Emotion	.327 ± .068	.289 ± .101	.332 ± .078	.304 ± .065	.264 ± .081	**.241 ± .073**
Flags	.220 ± .145	.227 ± .136	.297 ± .142	.219 ± .179	.186 ± .162	**.179 ± .155**
Birds	.739 ± .053	.763 ± .069	.767 ± .097	.718 ± .074	.691 ± .083	**.684 ± .069**

Table 4 (continued)

Dataset	CLR	MLkNN	RankSVM	MLTSVM (linear)	ML-SLSTSVM (linear)	ML-SLSTSVM (nonlinear)
Yeast	.249 ± .029	.243 ± .015	.266 ± .029	.238 ± .024	.233 ± .320	**.222 ± .021**
Medical	.192 ± .023	.250 ± .031	**.138 ± .028**	.160 ± .014	.152 ± .019	.140 ± .012
HS1	.445 ± .061	.437 ± .067	.465 ± .053	**.428 ± .058**	.534 ± .068	.503 ± .060
HS2	.596 ± .066	.612 ± .070	.608 ± .064	.590 ± .052	.392 ± .059	**.379 ± .061**
HC1	.017 ± .031	.013 ± .022	.021 ± .029	.014 ± .020	.020 ± .024	**.008 ± .020**
HC2	.297 ± .042	.314 ± .039	.307 ± .059	.295 ± .045	.126 ± .040	**.117 ± .031**
W-test	≪	≪	≪	≪	≪	

Table 5 Average CPU time (mean + SD)

Dataset	CLR	MLkNN	RankSVM	MLTSVM	ML-SLSTSVM (linear)	ML-SLSTSVM (nonlinear)
Emotion	2.192 ± .187	.930 ± .069	2.429 ± .230	.893 ± .073	.083 ± .008	.629 ± .030
Flags	.132 ± .094	.059 ± .007	.349 ± .045	.078 ± .012	.044 ± .006	.087 ± .013
Birds	1.076 ± .368	1.772 ± .095	2.836 ± .353	1.445 ± .162	.303 ± .026	2.501 ± .188
Yeast	61.09 ± 3.82	19.126 ± .75	108.19 ± 9.2	20.64 ± 1.28	1.785 ± .031	37.123 ± 4.21
Medical	68 ± 9.29	25 ± 3.67	143 ± 12.2	27 ± 4.91	9.123 ± 1.20	30 ± 5.4
W-test	==	==	≪	==	==	

Hamming loss, one error and coverage. Due to the limitation of space, the results have not been presented here.

According to the first row in Fig. 3, parameter C is the most sensitive parameter among all three parameters in our study; hence, selecting a suitable value of C can significantly improve the performance of the algorithm. As shown in Fig. 3, with increasing the value of C, performance (both ranking loss and average precision) first improved and then worsened. The best value for this parameter is 2^6 for Flags dataset and close to 2^{-1} for the other datasets.

As depicted in the second row in Fig. 3, large values for the parameter λ reduces the performance, i.e., ranking loss and average precision. More precisely, increasing the value of λ leads to a slight increase in the performance at first which then diminishes as the λ becomes larger. The decline is considerably higher in Birds dataset in comparison with others. The best value for λ is 2^4 for Flags and it is close to 2^{-2} for the other datasets. This means that the regularization term (minimization of hyperplane parameters w and b) is important to obtain the best classifier for Flags dataset and is relatively ineffective for the others.

Finally, as shown in the third row of Fig. 3, the different values of parameter η don't have significant effect on ML-SLSTSVM, except in Birds and Flags datasets in which the average precision can increase up to 7% by a proper selection of the parameter. This means that the structural information of the both datasets contains useful prior domain knowledge for training the classifier. In addition, since the ranking loss values are small, changing the value of η does not have much effect on it and the ranking loss chart is fairly smooth (except Flags dataset). The best value for the parameter η is close to 2^{-6} for Emotion, Birds and Yeast datasets and 2^2 and 2^5 for Medical and Flags datasets, respectively.

6 Conclusion

For the MLL problem, a new algorithm, termed as ML-SLSTSVM, is proposed in this paper. This algorithm is a ranking-based SVM that extends the MLTSVM method [15] with considering structural information of training samples and using least square idea. ML-SLSTSVM seeks a proximal hyperplane for each label where the kth hyperplane is closer to the instances with the label k, and is as far as possible from the others. We only need to solve systems of linear equations for both linear and nonlinear cases rather than to solve systems of QPPs in the MLTSVM. Experiments on nine synthetic and real-world multi-label datasets show that in term of evaluation metrics mentioned in subsection 5.2, ML-SLSTSVM outperforms some well-established multi-label learning algorithms.

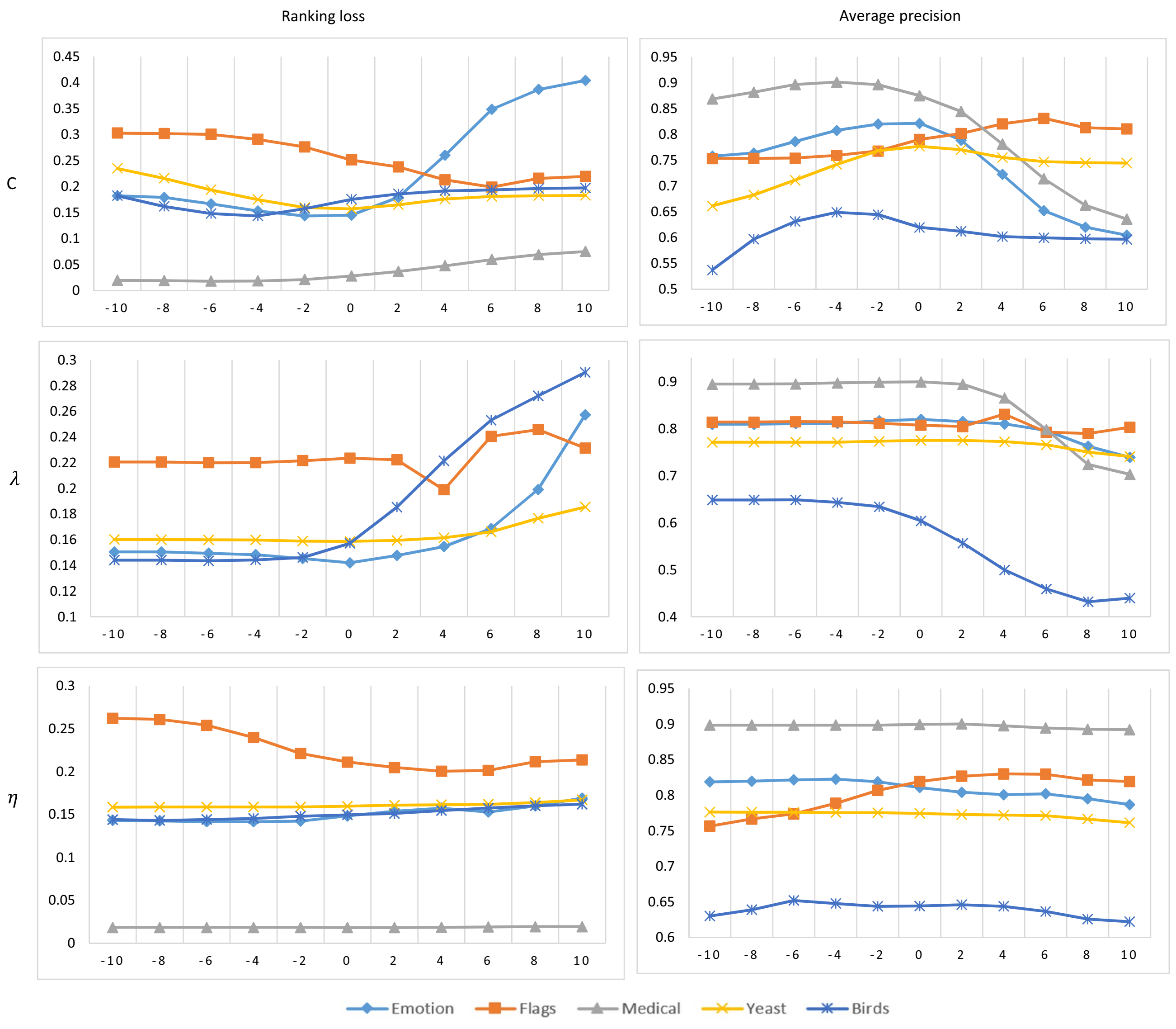

Fig. 3 Effect of parameters including C, λ and η on the performance of ML-SLSTSVM on the real datasets

References

1. Sorower MS (2010) A literature survey on algorithms for multi-label learning, vol 18. Oregon State University, Corvallis
2. Wu J-S, Huang S-J, Zhou Z-H (2014) Genome-wide protein function prediction through multi-instance multi-label learning. IEEE/ACM Trans Comput Biol Bioinform 11(5):891–902
3. Wang X, Zhang W, Zhang Q, Li G-Z (2015) MultiP-SChlo: multi-label protein subchloroplast localization prediction with Chou's pseudo amino acid composition and a novel multi-label classifier. Bioinformatics 31(16):2639–2645
4. Singh-Miller N, Collins M (2009) Learning label embeddings for nearest-neighbor multi-class classification with an application to speech recognition. In: Advances in neural information processing systems 22 (NIPS 2009), pp 1678–1686
5. Xu G, Lee H, Koo M-W, Seo J (2017) Convolutional neural network using a threshold predictor for multi-label speech act classification. In: 2017 IEEE international conference on big data and smart computing (BigComp), 2017, pp 126–130
6. Wu B, Zhong E, Horner A, Yang Q (2014) Music emotion recognition by multi-label multi-layer multi-instance multi-view learning. In: Proceedings of the 22nd ACM international conference on multimedia, 2014, pp 117–126
7. Trohidis K, Tsoumakas G, Kalliris G, Vlahavas IP (2008) Multi-label classification of music into emotions. ISMIR 8:325–330
8. Liu Y, Wen K, Gao Q, Gao X, Nie F (2018) SVM based multi-label learning with missing labels for image annotation, vol 78. Elsevier Ltd., Amsterdam
9. Wu J, Yu Y, Huang C, Yu K (2015) Deep multiple instance learning for image classification and auto-annotation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp 3460–3469
10. Tsoumakas G, Katakis I (2007) Multi-label classification: an overview. Int J Data Warehous Min 3(3):1–13

11. Zhang M-L, Zhou Z-H (2014) A review on multi-label learning algorithms. IEEE Trans Knowl Data Eng 26(8):1819–1837
12. Fürnkranz J, Hüllermeier E, Loza Mencía E, Brinker K (2008) Multilabel classification via calibrated label ranking. Mach Learn 73(2):133–153
13. Elisseeff A, Weston J (2001) A kernel method for multi-labelled classification. In: Advances in neural information processing systems 14 (NIPS 2001), pp 681–687
14. Xu J (2013) Fast multi-label core vector machine. Pattern Recognit 46(3):885–898
15. Chen WJ, Shao YH, Li CN, Deng NY (2016) MLTSVM: a novel twin support vector machine to multi-label learning. Pattern Recognit 52:61–74
16. Zhang ML, Zhou ZH (2007) ML-KNN: a lazy learning approach to multi-label learning. Pattern Recognit 40(7):2038–2048
17. Zhang M-L (2009) Ml-rbf: RBF neural networks for multi-label learning. Neural Process Lett 29(2):61–74
18. Zhang M-L, Zhou Z-H (2006) Multilabel neural networks with applications to functional genomics and text categorization. IEEE Trans Knowl Data Eng 18(10):1338–1351
19. Clare A, King RD (2001) Knowledge discovery in multi-label phenotype data. In: European conference on principles of data mining and knowledge discovery, 2001, pp 42–53
20. Vens C, Struyf J, Schietgat L, Džeroski S, Blockeel H (2008) Decision trees for hierarchical multi-label classification. Mach Learn 73(2):185–214
21. Qi Z, Tian Y, Shi Y (2013) Structural twin support vector machine for classification. Knowl Based Syst 43:74–81
22. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
23. Arun Kumar M, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36(4):7535–7543
24. Hüllermeier E, Fürnkranz J, Cheng W, Brinker K (2008) Label ranking by learning pairwise preferences. Artif Intell 172(16–17):1897–1916
25. Sun X et al (2016) ELM-ML: study on multi-label classification using extreme learning machine. In: Proceedings of ELM-2015, vol 2. Springer, 2016, pp 107–116
26. Schapire RE, Singer Y (2000) BoosTexter: a boosting-based system for text categorization. Mach Learn 39(2–3):135–168
27. Wang Y et al (2017) A multi-label learning method for efficient affective detection. In: 2017 IEEE EMBS international conference on biomedical and health informatics, pp 61–64
28. Reyes O, Morell C, Ventura S (2018) Effective active learning strategy for multi-label learning. Neurocomputing 273:494–508
29. Li Z, Tang J, Mei T (2018) Deep collaborative embedding for social image understanding. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2018.2852750
30. Zhu J, Liao S, Lei Z, Li SZ (2017) Multi-label convolutional neural network based pedestrian attribute classification. Image Vis Comput 58:224–229
31. Zhuang N, Yan Y, Chen S, Wang H, Shen C (2018) Multi-label learning based deep transfer neural network for facial attribute classification. Pattern Recognit 80:225–240
32. Li Z, Tang J (2017) Weakly supervised deep matrix factorization for social image understanding. IEEE Trans Image Process 26(1):276–288
33. Cakir E, Heittola T, Huttunen H, Virtanen T (2015) Multi-label vs. combined single-label sound event detection with deep neural networks. In: 2015 23rd European signal processing conference (EUSIPCO), 2015, pp 2551–2555
34. Barutcuoglu Z, Schapire RE, Troyanskaya OG (2006) Hierarchical multi-label prediction of gene function. Bioinformatics 22(7):830–836
35. Zhang W, Yan J, Wang X, Zha H (2018) Deep extreme multi-label learning. In: Proceedings of the 2018 ACM on international conference on multimedia retrieval. ACM, pp 100–107
36. Prabhu Y, Varma M (2014) Fastxml: a fast, accurate and stable tree-classifier for extreme multi-label learning. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp 263–272
37. Weston J, Makadia A, Yee H (2013) Label partitioning for sublinear ranking. In: International conference on machine learning, 2013, pp 181–189
38. Xu C, Tao D, Xu C (2016) Robust extreme multi-label learning. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp 1275–1284
39. Bhatia K, Jain H, Kar P, Varma M, Jain P (2015) Sparse local embeddings for extreme multi-label classification. In: Advances in neural information processing systems 28 (NIPS 2015), pp 730–738
40. Jayadeva, Khemchandani R, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910
41. Vapnik VN (1999) An overview of statistical learning theory. IEEE Trans Neural Netw 10(5):988–999
42. Hofmann T, Schölkopf B, Smola AJ (2008) Kernel methods in machine learning. Ann Stat 36(3):1171–1220
43. Auria L, Moro R (2008) Support vector machines (SVM) as a technique for solvency analysis. DIW Berlin discussion paper no. 811. Available at SSRN: https://ssrn.com/abstract=1424949 or https://doi.org/10.2139/ssrn.1424949
44. Xue H, Chen S, Yang Q (2011) Structural regularized support vector machine: a framework for structural large margin classifier. IEEE Trans Neural Netw 22(4):573–587
45. Shao Y-H, Zhang C-H, Wang X-B, Deng N-Y (2011) Improvements on twin support vector machines. IEEE Trans Neural Netw 22(6):962–968
46. Fung G, Mangasarian OL (2001) Proximal support vector machine classifiers. In: Proceeding of ACM SIGKDD international conference on knowledge discovery and data mining—KDD'01, pp 77–86
47. Tsoumakas G, Spyromitros-Xioufis E, Vilcek J, Vlahavas I (2011) Mulan: a java library for multi-label learning. J Mach Learn Res 12(Jul):2411–2414
48. Nasiri JA, Charkari NM, Jalili S (2015) Least squares twin multi-class classification support vector machine. Pattern Recognit 48(3):984–992
49. Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics Bull 1(6):80–83

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.