Pattern Recognition 46 (2013) 885–898

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Fast multi-label core vector machine

Jianhua Xu*

School of Computer Science and Technology, Nanjing Normal University, Nanjing, Jiangsu 210097, China

ARTICLE INFO

Article history:
Received 28 November 2011
Received in revised form
6 June 2012
Accepted 3 September 2012
Available online 11 September 2012

Keywords:
Support vector machine
Core vector machine
Multi-label classification
Frank–Wolfe method
Linear programming
Quadratic programming

ABSTRACT

The existing multi-label support vector machine (Rank-SVM) has an extremely high computational complexity due to a large number of variables in its quadratic programming. When the Frank–Wolfe (FW) method is applied, a large-scale linear programming still needs to be solved at any iteration. Therefore it is highly desirable to design and implement a new efficient SVM-type multi-label algorithm. Binary core vector machine (CVM), as a variant of traditional SVM, is formulated as a quadratic programming with a unit simplex constraint, in which each linear programming in FW has an analytical solution. In this paper, we combine Rank-SVM with CVM to construct a novel SVM-type multi-label classifier (Rank-CVM) which is described as the same optimization form as binary CVM. At its any iteration of FW, there exist analytical solution and step size, and several useful recursive formulae for proxy solution, gradient vector, and objective function value, all of which reduce computational cost greatly. Experimental study on nine benchmark data sets shows that when Rank-CVM performs as statistically well as its rival Rank-SVM according to five performance measures, our method runs averagely about 13 times faster and has less support vectors than Rank-SVM in the training phase under C/C++ environment.

1. Introduction

Multi-label classification is a special supervised learning issue, in which any single instance could be possibly associated with several classes simultaneously and thus the classes are no longer mutually exclusive [1–3]. Recently, it has been paid more attention to than before due to lots of real-world applications, e.g., text categorization [4–7], scene and video annotation [8–11], bioinformatics [6,12,13], and music emotion categorization [14]. Currently, there are three main strategies to design and implement various discriminative multi-label classification methods: data decomposition, algorithm extension and hybrid strategies. Further, label correlation, i.e., label co-occurrence information, has been exploited in four different levels: individual and partial instances, and pairwise and different labels.

Data decomposition strategy divides a multi-label data set into either one or more single-label (binary or multi-class) subsets, builds a sub-classifier for each subset using an existing classifier, and then assembles all sub-classifiers into an entire multi-label classifier. There are mainly three decomposition tricks: one-versus-rest (OVR), one-versus-one (OVO), and label powerset (LP) [1–3]. It is convenient to implement a data decomposition multi-label method since lots of popular classifiers and their free software are available. This strategy depicts label correlation of individual instance, partial instances and pairwise labels respectively by reusing multi-label instances in the OVR methods implicitly, constructing possible label combinations in the LP methods directly, and considering pairwise label combinations in the OVO methods explicitly.

Algorithm extension strategy generalizes a specific multi-class classification algorithm to consider all training instances and all classes (or labels) of a multi-label training data set at once. Such a strategy could induce some complicated optimization problems, e.g., large-scale quadratic programming in multi-label support vector machine (Rank-SVM) [15,16] and unconstrained optimization in multi-label back-propagation neural networks (BP-MLL) [6]. However, they explicitly characterize as many label correlations of individual instance using pairwise constraints between relevant labels and irrelevant ones as possible.

Hybrid strategy not only extends an existing single-label method but also splits a multi-label data set into a series of subsets implicitly or explicitly. This strategy has been used to design and implement several popular multi-label classifiers, e.g., two *k*NN-based multi-label approaches (ML-*k*NN and IBLR-ML) [9,17], which cascade *k*NN with discrete Bayesian rule and logistic model respectively while the OVR trick is applied implicitly. Generally this strategy weakly characterizes label correlations either explicitly or implicitly.

As mentioned above, algorithm extension strategy takes as many label correlations into account as possible, which is regarded

* Tel.: +86 25 86306209; fax: +86 25 85891990.
E-mail addresses: xujianhua@njnu.eud.cn, xujianhua99@tsinghua.org.cn

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.09.003

as an optimal way to improve multi-label classification performance further [18]. But, its corresponding methods have a high computational cost, which limits their usability for many applications. Therefore, it is highly desirable to design and implement some novel efficient multi-label classifiers.

In this paper, we focus on SVM-type multi-label classification techniques. When Rank-SVM [15,16] is solved by the Frank–Wolfe (FW) algorithm [19,20], its each iteration needs to deal with a large-scale linear programming [15,16]. Binary core vector machine (CVM), as a variant of traditional SVM, is formulated as a quadratic programming with a unit simplex constraint, which is converted into minimum enclosing ball (MEB) problem in [21,22]. An alternative solution is based on FW [23], where any linear programming has an analytical solution. In this paper, we combine Rank-SVM with CVM to construct a novel SVM-type multi-label classifier (Rank-CVM), which has the same quadratic programming form as binary CVM. When FW is applied, we can derive an analytical solution, an analytical step size, and several efficient recursive formulae for gradient vector, objective function and so on. It is shown theoretically that our Rank-CVM has a lower time complexity than Rank-SVM. Experimental results on nine benchmark data sets demonstrate that our method is a competitive candidate for multi-label classification according to five performance measures, compared with four existing techniques: Rank-SVM [15,16], ADTree [24], and BP-MLL [6] and ML-*k*NN [9]. Moreover, our Rank-CVM runs averagely 13 times faster and has less support vectors than its rival Rank-SVM in the training phase using C/C++ language.

The rest of this paper is organized as follows. Multi-label classification setting and evaluation are introduced in Section 2 and previous work is summarized in Section 3. Binary SVM and CVM, and Rank-SVM are reviewed briefly in Sections 4 and 5. Our novel method is proposed in Section 6. In Section 7, we summarize FW, and then construct a fast training algorithm for our Rank-CVM. Section 8 is devoted to experiments with nine benchmark data sets. This paper ends with some conclusions in Section 9.

2. Multi-label classification setting and evaluation

Let $X \in \Re^d$ be a d-dimensional input space and $Q=\{1,2,...,q\}$ a finite set of class labels, where q is the number of class labels. Further, assume that each instance $\boldsymbol{x} \in X$ can be associated with a set of relevant labels $L \subseteq 2^Q$. At the same time, the complement of L, i.e., $\overline{L} = Q \backslash L$, is referred to as a set of irrelevant labels of $\boldsymbol{x}$. Given a training data set of size l drawn identically and independently from an unknown probability distribution (i.i.d.) on $X \times 2^Q$, i.e.,

$$\{(\boldsymbol{x}_1,L_1),\ldots,(\boldsymbol{x}_i,L_i),\ldots,(\boldsymbol{x}_l,L_l)\}, \tag{1}$$

the multi-label classification problem is to learn a classifier $f(\boldsymbol{x}):X \to 2^Q$ that generalizes well on both these training instances and unseen ones in the sense of optimizing some expected risk functional with respect to a specific empirical loss function [6,17].

In many traditional q-class single-label classification methods, a widely used trick is to learn q discriminant functions $f_i(\boldsymbol{x}): X \to \Re, i = 1,\ldots,q$ such that $f_k(\boldsymbol{x}) > f_i(\boldsymbol{x})$, $i \neq k$ if $\boldsymbol{x} \in$ class k [25]. For multi-label classification, as an extension of multi-class classification, this idea is adapted as $f_k(\boldsymbol{x}) > f_i(\boldsymbol{x})$, $k \in L$, $i \in \overline{L}$, which implies that any relevant label should be ranked higher than any irrelevant one [15]. In this case, the multi-label prediction can be fulfilled through a proper threshold $t(\boldsymbol{x})$,

$$f(\boldsymbol{x}) = \{k | f_k(\boldsymbol{x}) \geq t(\boldsymbol{x}), k = 1,\ldots,q\}. \tag{2}$$

Now mainly there are three kinds of thresholds: a constant (e.g., 0.0 for $-1/+1$ setting and 0.5 for 0/1 one) [4,8,9], a linear regression model associated with q discriminant function values [6,15], and an additional discriminant function for a virtual or calibrated label [10]. In the first case, $t(\boldsymbol{x})$ is independent of $\boldsymbol{x}$.

So far, more than 10 performance evaluation measures have been introduced [2,3], since it is more complicated to evaluate a multi-label classification algorithm than a single-label one. In this paper, we choose the same five popular and indicative measures: coverage, one error, average precision, ranking loss and Hamming loss, as in [6,9,17]. Assume a test data set of size m to be $\{(\boldsymbol{x}_1,L_1), \ldots,(\boldsymbol{x}_i,L_i),\ldots,(\boldsymbol{x}_m,L_m)\}$. Given some instance $\boldsymbol{x}_i$, its q discriminant function values and predicted set of relevant labels from some multi-label classification algorithm are denoted by $f_k^P(\boldsymbol{x}_i), k = 1, \ldots, q$ and $L_i^P \subseteq 2^Q$ respectively.

The coverage estimates how far we need, on average, to go down the list of labels to cover all relevant labels of the instance:

$$\text{Coverage} = \frac{1}{m}\sum_{i=1}^{m} \left(|C(\boldsymbol{x}_i)|-1\right) \in [0,q-1], \tag{3}$$

where $C(\boldsymbol{x}_i) = \{k | f_k^P(\boldsymbol{x}_i) \geq f_{k'}^P(\boldsymbol{x}_i), k \in Q\}$and $k' = \{k | \min f_k^P(\boldsymbol{x}_i), k \in L_i\}$. The one error evaluates how many times that the top-ranked label is not one of relevant labels:

$$\text{One error} = \frac{1}{m}\sum_{i=1}^{m} \left|\{k \notin L_i | f_k^P(\boldsymbol{x}_i) = \max_{k' \in Q} f_{k'}^P(\boldsymbol{x}_i)\}\right| \in [0,1]. \tag{4}$$

The average precision calculates the average fraction of labels ranked above a specific relevant label $k \in L_i$, which actually are in L_i, i.e.,

$$\text{Average precision} = \frac{1}{m}\sum_{i=1}^{m} \left(\frac{1}{|L_i|} \sum_{k \in L_i} \frac{\left|\{k' \in L_i | f_{k'}^P(\boldsymbol{x}_i) \geq f_k^P(\boldsymbol{x}_i)\}\right|}{\left|\{k' \in Q | f_{k'}^P(\boldsymbol{x}_i) \geq f_k^P(\boldsymbol{x}_i)\}\right|} \right) \in [0,1]. \tag{5}$$

The ranking loss computes the average fraction of labels pairs (a relevant label versus an irrelevant one) that are not correctly ordered for the instance:

$$\text{Ranking loss} = \frac{1}{m}\sum_{i=1}^{m} \left(\frac{1}{|L_i||\overline{L}_i|} \left|\{(k,k') \in (L_i \times \overline{L}_i) | f_k^P(\boldsymbol{x}_i) \leq f_{k'}^P(\boldsymbol{x}_i)\}\right| \right) \in [0,1]. \tag{6}$$

The Hamming loss estimates the percentage of labels, whose relevance is predicted incorrectly:

$$\text{Hamming loss} = \frac{1}{m}\sum_{i=1}^{m} \frac{|L_i \Delta L_i^P|}{q} \in [0,1], \tag{7}$$

where Δ denotes the symmetric difference between two sets. It is desirable that a multi-label algorithm should achieve a larger value for the average precision, and smaller values for the other four measures.

3. Previous work

In the past 10 years, since multi-label classification has received a lot of attention in machine learning, pattern recognition and statistics, a variety of methods have been proposed. In this section, according to three strategies mentioned in Section 1, we categorize previous discriminative multi-label methods into three groups: data decomposition, algorithm extension and hybrid methods. It is worth noting that the last two groups are roughly merged into algorithm adaptation or dependent methods in [1–3].

3.1. Data decomposition methods

Data decomposition methods combine data decomposition tricks with existing single-label classification methods, whose two key techniques are decomposition tricks and integration ways.

The one-versus-rest (OVR) or binary relevance (BR) decomposition trick splits a q-class multi-label data set into q binary subsets [4,8], in which the ith subset consists of positive instances with the ith label and negative ones with the all other labels. Usually, q sub-classifiers are assembled into an entire multi-label algorithm using a constant threshold. So far, many OVR multi-label methods have been verified to work well using various binary classifiers, e.g., SVM [1,4,8,15], C4.5 [1,17], and kNN [1,17]. With a probabilistic output setting, the classification performance can be improved further via a thresholding strategy [26].

The one-versus-one (OVO) decomposition trick divides a q-class multi-label data set into $q(q-1)/2$ binary subsets in a pairwise way, where the subset ij $(i<j)$ only involves instances with the ith and jth labels. Note that for some subsets there is a mixed class whose instances belong to both positive and negative classes simultaneously. In [27], these special subsets are handled by two binary SVM classifiers using the OVR strategy, and a vote threshold is used to detect relevant labels. But in [28], the mixed classes are discarded simply. Further, a calibrated label is estimated by some OVR-based method, whose votes are defined as a threshold $t(\boldsymbol{x})$ in (2). To reduce the test computational cost, three two-stage architectures are designed in [29].

The label powerset (LP) decomposition trick considers each possible label combination of more than one class as a new single class, and then converts a multi-label training set into a standard multi-class one [1,8]. This could produce a large number of new classes, many of which consist of very few instances. Such a shortcoming is avoided by LP-based ensemble classifiers in terms of a small subset of labels in [30].

3.2. Algorithm extension methods

Algorithm extension methods generalize existing multi-class classifiers to consider all instances and all classes in a multi-label data set at once.

In [31], a C4.5-type multi-label algorithm was proposed, through modifying the formula of entropy calculation and permitting multiple labels at the leaves of the tree. BoosTexer [5] is derived from the well-known Adaboost algorithm, which includes two slightly different versions: AdaBoost.MH and AdaBoost.MR. The former is to predict the set of relevant labels of an instance, while the latter to range labels of an instance in descending order. ADTree is constructed by integrating alternative decision tree with Adaboost.MH [24].

Multi-label support vector machine (Rank-SVM) was proposed in [15,16] via extending multi-class SVM [32] and accepting an approximate ranking loss as its empirical loss, resulting into an extremely complicated quadratic programming. Additionally Rank-SVM needs to learn a threshold function $t(\boldsymbol{x})$ in (2) using linear regression. In [10], a virtual label is added to find a natural zero to determine relevant labels simply, which further increases the number of variables to be solved. Although such two Rank-SVM forms are solved by FW [19,20], their training procedures are still computationally expensive. Multi-label back-propagation neural networks (BP-MLL) [6] also define a new empirical loss function based on ranking loss to reflect label correlation of individual instance, which results in a large scale unconstrained optimization problem. Additionally a linear threshold function is needed just as in Rank-SVM.

3.3. Hybrid methods

Hybrid methods extend some specific single-label classification algorithms while one or two data decomposition tricks are utilized implicitly or explicitly, in order to integrate some merits of the aforementioned two groups of multi-label methods.

After the OVR trick is utilized implicitly, traditional k-nearest neighbor (kNN) method are generalized to construct two slightly different multi-label methods: ML-kNN [9] and IBLR-ML [17]. Such two methods include a training phase combining leave-one-out (LOO) procedure with kNN, in which ML-kNN estimates class prior and conditional probabilities of discrete binary Bayesian rule for each label independently, and IBLR-ML calculates posterior probability associated with weighted sums of k-nearest neighbor labels of all classes using logistic regression. Additionally, the OVR trick is utilized explicitly in multi-label naïve Bayes (ML-NB) [33], classifier chain [34], extended SVM [35], and RBF neural networks (ML-RBF) [36].

As mentioned above, the OVO decomposition trick could result in a mixed class for some subsets. It seems natural and reasonable to locate this mixed class between positive and negative classes. Through extending traditional binary SVM, two parallel hyperplanes are used to deal with three class subsets in [37,38], while the mixed class is forced to reside in the marginal region of binary SVM in [39].

4. Binary support vector machine and core vector machine

In this section, we briefly review binary support vector machine (SVM) [32] and its variant core vector machine (CVM) [21,22]. Here, let a binary i.i.d. training set of size l be

$$\{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_i,y_i),\ldots,(\boldsymbol{x}_l,y_l)\}, \tag{8}$$

where $\boldsymbol{x}_i \in X$ and $y_i \in \{+1,-1\}$ denote the ith training instance vector and its binary label.

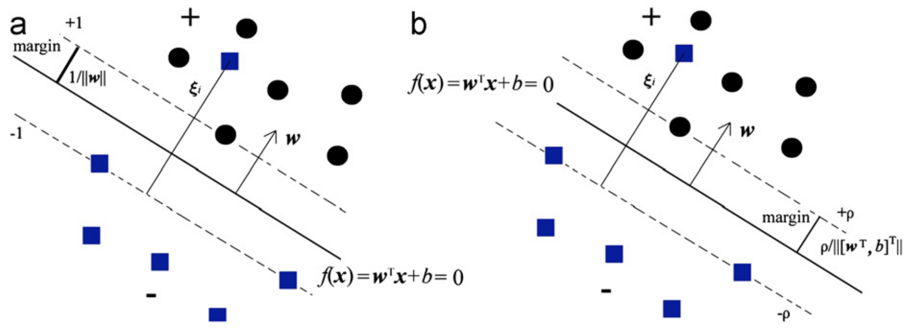

Fig. 1. Schematic illustration of SVM (a) and CVM (b).

In the original space, a linear discriminant function is defined as

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}+b, \tag{9}$$

where $\boldsymbol{w}$ and b are the weight vector and bias term respectively, as shown in Fig. 1. For the linearly separable case in SVM, the hyperplane $f(\boldsymbol{x})=0$ separates positive instances from negative ones with a margin $1/||\boldsymbol{w}||_2$ which is maximizing through minimizing $||\boldsymbol{w}||_2^2=\boldsymbol{w}^T\boldsymbol{w}$, where $||\cdot||_2$indicates 2-norm of vector. The primary problem of SVM for the nonlinearly separable case is formulated as

$$\min \frac{1}{2}\ \boldsymbol{w}^T\boldsymbol{w}+C\sum_{i=1}^{l}\xi_i,$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x_i}+b) \geq 1-\xi_i, \xi_i \geq 0,\ i=1,\ldots,l, \tag{10}$$

where $C>0$ denotes a regularization constant to control the tradeoff between classification errors and model complexity, and $\xi_i \geq 0$ are slack variables to indicate misclassification errors. The dual version of (10) is

$$\min \frac{1}{2}\boldsymbol{\alpha}^T(\boldsymbol{K}\odot\boldsymbol{y}\boldsymbol{y}^T)\boldsymbol{\alpha}-\boldsymbol{u}^T\boldsymbol{\alpha},$$
$$\text{s.t. } \boldsymbol{y}^T\boldsymbol{\alpha}=0, \boldsymbol{0}\leq\boldsymbol{\alpha}\leq C\boldsymbol{u}, \tag{11}$$

where the symbol $\odot$ represents the Hadamard product, $\boldsymbol{\alpha}=[\alpha_1,...,\alpha_l]^T$ is the l Lagrangian multipliers to be solved, $\boldsymbol{K}=[\boldsymbol{x}_i^T\boldsymbol{x}_j|i,j=1,\ \ldots,l]$ denotes the symmetric kernel matrix, $\boldsymbol{u}=[1,...,1]^T$ and $\boldsymbol{0}=[0,...,0]^T$ are two column vectors with l ones and zeros respectively, and $\boldsymbol{y}=[y_1,...,y_l]^T$ indicates the binary label vector.

In CVM, since the minimal discriminant function values for positive and negative instances are denoted by $\pm\rho$ as shown in Fig. 1(b), the margin is defined as $\rho/||[\boldsymbol{w}^T,b]^T||_2$which depends on both $\boldsymbol{w}$ and b for the linearly separable case. This margin is maximized via minimizing a linear combination $\boldsymbol{w}^T\boldsymbol{w}+b^2-\nu\rho$, where $\nu>0$ indicates a positive constant. The primary form of CVM for the nonlinearly separable case is built as

$$\min \frac{1}{2}(\boldsymbol{w}^T\boldsymbol{w}+b^2)-\nu\rho+\frac{1}{2}C\sum_{i=1}^{l}\xi_i^2,$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x_i}+b)\geq\rho-\xi_i, \rho\geq 0,\ i=1,\ldots,l. \tag{12}$$

Note $\xi_i \geq 0$ are redundant due to its squared form in the objective function. The dual version of (12) becomes

$$\min \frac{1}{2}\boldsymbol{\alpha}^T\left(\boldsymbol{K}\odot\boldsymbol{y}\boldsymbol{y}^T+\boldsymbol{y}\boldsymbol{y}^T+\frac{1}{C}\boldsymbol{I}\right)\boldsymbol{\alpha},$$
$$\text{s.t. } \boldsymbol{u}^T\boldsymbol{\alpha}\geq\nu, \boldsymbol{\alpha}\geq\boldsymbol{0}, \tag{13}$$

where $\boldsymbol{I}$ represents the unit matrix of size $l\times l$. In [40], it has been shown that the inequality $\boldsymbol{u}^T\boldsymbol{\alpha}\geq\nu$ in (13) can be replaced by an equality $\boldsymbol{u}^T\boldsymbol{\alpha}=\nu$. Via dividing the ν, the above problem (13) is further simplified into a quadratic programming with a unit simplex constraint:

$$\min \frac{1}{2}\boldsymbol{\alpha}^T\left(\boldsymbol{K}\odot\boldsymbol{y}\boldsymbol{y}^T+\boldsymbol{y}\boldsymbol{y}^T+\frac{1}{C}\boldsymbol{I}\right)\boldsymbol{\alpha},$$
$$\text{s.t. } \boldsymbol{u}^T\boldsymbol{\alpha}=1, \boldsymbol{\alpha}\geq\boldsymbol{0}. \tag{14}$$

Now SVM (11) is widely solved by sequential minimization optimization (SMO) [41]. But, for CVM (14), due to its unit simplex constraint, there exist two special solution ways: the minimum enclosing ball (MEB) technique [21,22] and the Frank–Wolfe (FW) algorithm [23]. The latter has an analytical solution for each linear programming in FW.

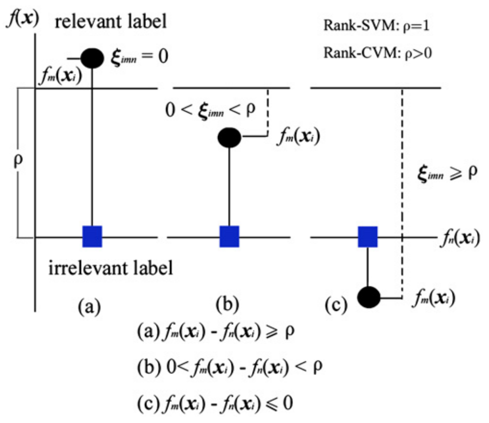

Fig. 2. Three possible relative relationships between a pair of labels in Rank-SVM ($\rho=1$) and our Rank-CVM ($\rho>0$).

5. Multi-label support vector machine

In this section, we review some previous work on multi-label support vector machine (Rank-SVM) [15,16] first and then provide more detailed description. Now, in the original input space, q linear discriminant functions are defined as

$$f_k(\boldsymbol{x})=\boldsymbol{w}_k^T\boldsymbol{x}+b_k,\ k=1,\ldots,q, \tag{15}$$

where $\boldsymbol{w}_k$ and b_k denote the weight vector and bias term of the kth class respectively. As mentioned in Section 2, it is desirable that any relevant label should be ranked higher than any irrelevant. In case such an ideal situation does not happen, a slack variable is introduced just as in binary SVM. Therefore, in Rank-SVM, the relative relationship between any relevant label and any irrelevant one for some training instance $\boldsymbol{x}_i$, is described using the following pairwise constraint:

$$f_m(\boldsymbol{x}_i)-f_n(\boldsymbol{x}_i)=(\boldsymbol{w}_m-\boldsymbol{w}_n)^T\boldsymbol{x}_i+(b_m-b_n)\geq 1-\xi_{imn},\ \ (m,n)\in(L_i\times\overline{L}_i), \tag{16}$$

where the slack variable $\xi_{imn}\geq 0$ is added to force the difference to be equal to or more than 1. As shown in Fig. 2 for $\rho=1$, there exist three possible cases, (a) $f_m(\boldsymbol{x}_i)-f_n(\boldsymbol{x}_i)\geq 1$ and $\xi_{imn}=0$, which is a perfect situation; (b) $0<f_m(\boldsymbol{x}_i)-f_n(\boldsymbol{x}_i)<1$ and $0<\xi_{imn}<1$; and (c) $f_m(\boldsymbol{x}_i)-f_n(\boldsymbol{x}_i)\leq 0$ and $\xi_{imn}\geq 1$.

According to the definition of ranking loss (6), the first two cases do not induce any ranking loss value, while the last one does result into a ranking loss "1". However, regardless of any case, its ranking loss value is less than or equal to the slack variable ξ_{imn}. Therefore, an approximate ranking loss function can be defined as

$$\text{Approximate ranking loss}=\frac{1}{l}\sum_{i=1}^{l}\frac{1}{|L_i||\overline{L}_i|}\sum_{(m,n)\in(L_i\times\overline{L}_i)}\xi_{imn}, \tag{17}$$

which is essentially the upper bound of the actual ranking loss (6) and is utilized as an empirical loss term in Rank-SVM after $1/l$ is omitted. The original optimization problem of Rank-SVM is depicted as

$$\min \frac{1}{2}\sum_{k=1}^{q}\boldsymbol{w}_k^T\boldsymbol{w}_k+C\sum_{i=1}^{l}\frac{1}{|L_i||\overline{L}_i|}\sum_{(m,n)\in(L_i\times\overline{L}_i)}\xi_{imn},$$
$$\text{s.t. } (\boldsymbol{w}_m-\boldsymbol{w}_n)^T\boldsymbol{x}_i+(b_m-b_n)\geq 1-\xi_{imn},$$
$$\xi_{imn}\geq 0, (m,n)\in(L_i\times\overline{L}_i), i=1,\ldots,l. \tag{18}$$

The model regularization term, i.e., the first one in the above objective function, is defined according to the corresponding term in binary SVM. Using the standard Lagrangian technique, the dual version of (18) is derived as follows [15,16]:

$$\min \frac{1}{2}\sum_{k=1}^{q}\sum_{i,j=1}^{l}\beta_{ki}\beta_{kj}(\boldsymbol{x}_i^T\boldsymbol{x}_j)-\sum_{i=1}^{l}\sum_{(m,n)\in(L_i\times\overline{L}_i)}\alpha_{imn},$$

$$\text{s.t.} \sum_{i=1}^{l}\sum_{(m,n)\in(L_i\times\overline{L}_i)}c_{imn}^k\alpha_{imn}=0,\ k=1,\ldots,q,$$

$$0\le\alpha_{imn}\le C_i=\frac{1}{|L_i||\overline{L}_i|}C,(m,n)\in(L_i\times\overline{L}_i),\ i=1,\ldots,l, \tag{19}$$

with

$$c_{imn}^k=\begin{cases}0 & \text{if } m\ne k \text{ and } n\ne k,\\ +1 & \text{if } m=k,\\ -1 & \text{if } n=k,\end{cases} \tag{20}$$

and

$$\beta_{ki}=\sum_{(m,n)\in(L_i\times\overline{L}_i)}c_{imn}^k\alpha_{imn}. \tag{21}$$

In [15,16], the above component-based form is described only. Now we transform (19) and (21) into concise vector and matrix representations. For the ith instance, we construct a column vector $\boldsymbol{\alpha}_i$ with α_{imn}, a row vector $\boldsymbol{h}_{ki}$ with c_{imn}^k and a unit vector $\boldsymbol{u}_i$, whose length is $l_i=|L_i||\overline{L}_i|$, and a matrix $\boldsymbol{H}_i$ of size $q\times l_i$ using $\boldsymbol{h}_{ki}$

$$\boldsymbol{\alpha}_i=\left[\alpha_{imn}|(m,n)\in(L_i\times\overline{L}_i)\right]^T=\left[\alpha_i^j|j=1,\ldots,l_i\right]^T,$$

$$\boldsymbol{h}_{ki}=\left[c_{imn}^k|(m,n)\in(L_i\times\overline{L}_i)\right]=\left[h_{ki}^j|j=1,\ldots,l_i\right],$$

$$\boldsymbol{u}_i=[1,\ldots,1]^T,$$

$$\boldsymbol{H}_i=\left[\boldsymbol{h}_{1i}^T,\ldots,\boldsymbol{h}_{qi}^T\right]^T. \tag{22}$$

According to (22), the formulae (21) and (19) can be simplified as

$$\beta_{ki}=\boldsymbol{h}_{ki}\boldsymbol{\alpha}_i, \tag{23}$$

and

$$\min \frac{1}{2}\sum_{i,j=1}^{m}\boldsymbol{\alpha}_i^T(\boldsymbol{H}_i^T\boldsymbol{H}_j)\boldsymbol{\alpha}_j(\boldsymbol{x}_i^T\boldsymbol{x}_j)-\sum_{i=1}^{l}\boldsymbol{u}_i^T\boldsymbol{\alpha}_i,$$

$$\text{s.t.} \sum_{i=1}^{l}\boldsymbol{H}_i\boldsymbol{\alpha}_i=\boldsymbol{0},\boldsymbol{0}\le\boldsymbol{\alpha}_i\le C_i\boldsymbol{u}_i, \tag{24}$$

Further, using (22), we define a solution vector $\boldsymbol{\alpha}$ and a unit vector $\boldsymbol{u}$, whose length is $l_t=\sum_{i=1}^{l}l_i$, and a new matrix $\boldsymbol{H}$ of size $q\times l_t$ and a kernel matrix $\boldsymbol{K}$ of size $l_t\times l_t$

$$\boldsymbol{\alpha}=\left[\boldsymbol{\alpha}_1^T,\ldots,\boldsymbol{\alpha}_i^T,\ldots,\boldsymbol{\alpha}_l^T\right]^T,$$

$$\boldsymbol{u}=\left[\boldsymbol{u}_1^T,\ldots,\boldsymbol{u}_l^T\right]^T,$$

$$\boldsymbol{H}=[\boldsymbol{H}_1,\ldots,\boldsymbol{H}_i,\ldots,\boldsymbol{H}_l],$$

$$\boldsymbol{K}=\left[(\boldsymbol{x}_i^T\boldsymbol{x}_j)(\boldsymbol{u}_i\boldsymbol{u}_j^T)|i,j=1,\ldots,l\right]. \tag{25}$$

In terms of (25), the above quadratic programming (24) can be rewritten as a more concise form:

$$\min W(\boldsymbol{\alpha})=\frac{1}{2}\boldsymbol{\alpha}^T\left((\boldsymbol{H}^T\boldsymbol{H})\odot\boldsymbol{K}\right)\boldsymbol{\alpha}-\boldsymbol{u}^T\boldsymbol{\alpha},$$

$$\text{s.t. } \boldsymbol{H}\boldsymbol{\alpha}=\boldsymbol{0},\boldsymbol{0}\le\boldsymbol{\alpha}\le\boldsymbol{C}, \tag{26}$$

where the column vector $\boldsymbol{C}=\left[C_1\boldsymbol{u}_1^T \quad \ldots \quad C_l\boldsymbol{u}_l^T\right]^T$. Due to l_t variables to be solved, this quadratic programming (26) has an extremely high computational complexity. Now, the above Rank-SVM (19) or (26) is solved by the FW method [19], where any iteration needs to solve a large-scale linear programming [16].

Similarly in binary SVM and CVM [21,22,32], the dot product between two vectors in (25) can be replaced by various kernels [32], and q nonlinear discriminant functions with kernel are rewritten as

$$f_k(\boldsymbol{x})=\sum_{i=1}^{l}\beta_{ki}K(\boldsymbol{x},\boldsymbol{x}_i)+b_k,k=1,\ldots,q, \tag{27}$$

where $K(\boldsymbol{x},\boldsymbol{x}_i)$ denotes some kernel function [32]. To determine the threshold function $t(\boldsymbol{x})$ in (2), each training instance $\boldsymbol{x}_i$ is converted into a q-dimensional vector $[f_1(\boldsymbol{x}_i),\ldots,f_q(\boldsymbol{x}_i)]^T$ using (27), and then an optimal threshold $t^*(\boldsymbol{x}_i)$ is determined via minimizing the Hamming loss (7). A linear regression threshold function is trained in Rank-SVM, i.e., $t(\boldsymbol{x})=\sum_{k=1}^{q}s_kf_k(\boldsymbol{x})+s_0$, where $s_i(i=0,1,\ldots,q)$ represent the regression coefficients. Then this threshold form is used to predict the relevant labels in the test phase.

6. Multi-label core vector machine

In this section, we combines Rank-SVM with binary CVM to construct a novel multi-label SVM-type classifier (Rank-CVM simply), whose any iteration of FW has an analytical solution.

Now it is still desired that any relevant label should be ranked higher than any irrelevant one. According to the constraints in binary CVM and Rank-SVM, we define the following pairwise constraints for Rank-CVM:

$$f_m(\boldsymbol{x}_i)-f_n(\boldsymbol{x}_i)=(\boldsymbol{w}_m-\boldsymbol{w}_n)^T\boldsymbol{x}_i+(b_m-b_n)\ge\rho-\xi_{imn},(m,n)\in(L_i\times\overline{L}_i), \tag{28}$$

where the slack variable ξ_{imn} is used to force the difference to be more than $\rho>0$. As shown in Fig. 2 for $\rho>0$, there still are three possible situations: (a) $f_m(\boldsymbol{x}_i)-f_n(\boldsymbol{x}_i)\ge\rho$ and $\xi_{imn}=0$; (b) $0<f_m(\boldsymbol{x}_i)-f_n(\boldsymbol{x}_i)<\rho$ and $0<\xi_{imn}<\rho$; and (c) $f_m(\boldsymbol{x}_i)-f_n(\boldsymbol{x}_i)\le 0$ and $\xi_{imn}\ge\rho$. According to the ranking loss (6), it could be found out that the ranking loss value is always less than or equal to ξ_{imn}/ρ. Therefore, an approximate ranking loss function for Rank-CVM can be defined as follows:

$$\text{Approximate ranking loss}=\frac{1}{l\rho^2}\sum_{i=1}^{l}\frac{1}{|L_i||\overline{L}_i|}\sum_{(m,n)\in(L_i\times\overline{L}_i)}\xi_{imn}^2. \tag{29}$$

This is used as an empirical loss term when$1/l\rho^2$ is removed in our Rank-CVM. On the other hand, we constitute a model regularization term $(1/2)\sum_{k=1}^{q}(\boldsymbol{w}_k^T\boldsymbol{w}_k+b_k^2)-\nu\rho$ for Rank-CVM in terms of binary CVM and Rank-SVM principles. Now we formulate the primary form of our Rank-CVM as

$$\min \frac{1}{2}\sum_{k=1}^{q}(\boldsymbol{w}_k^T\boldsymbol{w}_k+b_k^2)-\nu\rho+\frac{1}{2}C\sum_{i=1}^{l}\frac{1}{|L_i||\overline{L}_i|}\sum_{(m,n)\in(L_i\times\overline{L}_i)}\xi_{imn}^2,$$

$$\text{s.t. } (\boldsymbol{w}_m\text{-}\boldsymbol{w}_n)^T\boldsymbol{x}_i+(b_m\text{-}b_n)=\sum_{k=1}^{q}c_{imn}^k(\boldsymbol{w}_k^T\boldsymbol{x}_i+b_k)\ge\rho-\xi_{imn},$$

$$\rho>0,(m,n)\in(L_i\times\overline{L}_i),i=1,\ldots,l. \tag{30}$$

Note that $\xi_{imn}\ge 0$ hold naturally due to the squared form of slack variables in the objective function (30), and the concise constraints are reformulated according to the definition of c_{imn}^kin (20). Again, C represents the regularization constant to control the tradeoff between the model complexity and the ranking loss.

The dual problem of (30) can be derived using the standard Lagrangian technique. Let $\alpha_{imn}\ge 0$ and $\eta\ge 0$ be the Lagrangian multipliers for the inequality constraints in (30). The Lagrangian for the primal form (30) becomes

$$L=\frac{1}{2}\sum_{k=1}^{q}(\boldsymbol{w}_k^T\boldsymbol{w}_k+b_k^2)-\nu\rho+\frac{1}{2}\sum_{i=1}^{l}C_i\sum_{(m,n)\in(L_i\times\overline{L}_i)}\xi_{imn}^2$$

$$-\sum_{i=1}^{l}\sum_{(m,n)\in(L_i\times\overline{L}_i)}\alpha_{imn}\left(\sum_{k=1}^{q}c_{imn}^k\left(\boldsymbol{w}_k{}^T\boldsymbol{x}_i+b_k\right)-\rho+\xi_{imn}\right)-\eta\rho, \tag{31}$$

where $C_i = C/|L_i||\bar{L}_i|$. The Karush–Kuhn–Tucker (KKT) conditions for this primary problem require the following relations to be true

$$\frac{\partial L}{\partial \boldsymbol{w}_k} = 0 \Rightarrow \boldsymbol{w}_k = \sum_{i=1}^{l}\left(\sum_{(m,n)\in(L_i\times\bar{L}_i)} c_{imn}^{k}\alpha_{imn}\right)\boldsymbol{x}_i = \sum_{i=1}^{l}\beta_{ki}\boldsymbol{x}_i, \tag{32}$$

$$\frac{\partial L}{\partial b_k} = 0 \Rightarrow b_k = \sum_{i=1}^{l}\sum_{(m,n)\in(L_i\times\bar{L}_i)} c_{imn}^{k}\alpha_{imn} = \sum_{i=1}^{l}\beta_{ki}, \tag{33}$$

$$\frac{\partial L}{\partial \rho} = 0 \Rightarrow \sum_{i=1}^{l}\sum_{(m,n)\in(L_i\times\bar{L}_i)} \alpha_{imn} = v+\eta, \tag{34}$$

$$\frac{\partial L}{\partial \xi_{imn}} = 0 \Rightarrow \alpha_{imn} = C_i\xi_{imn}. \tag{35}$$

The last relation (35) shows that $\xi_{imn}\ge 0$ hold naturally due to $\alpha_{imn}\ge 0$. By introducing the above KKT conditions (32)-(35) into the Lagrangian (31), the dual form is

$$\min W(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{k=1}^{q}\sum_{i,j=1}^{l}\beta_{ki}\beta_{kj}(\boldsymbol{x}_i^T\boldsymbol{x}_j) + \frac{1}{2}\sum_{k=1}^{q}\sum_{i,j=1}^{l}\beta_{ki}\beta_{kj} + \frac{1}{2}\sum_{i=1}^{l}\left(\frac{1}{C_i}\sum_{(m,n)\in(L_i\times\bar{L}_i)}\alpha_{imn}^2\right),$$

$$\text{s.t.} \sum_{i=1}^{l}\sum_{(m,n)\in(L_i\times\bar{L}_i)}\alpha_{imn}\ge v;\ \alpha_{imn}\ge 0, (m,n)\in(L_i\times\bar{L}_i),\ i=1,\ldots,l. \tag{36}$$

According to those notions in (22) and (25), and the derivation of binary CVM, this formula (36) can further be simplified into a quadratic programming with a unit simplex constraint too

$$\begin{aligned}\min W(\boldsymbol{\alpha}) &= \frac{1}{2}\sum_{i,j=1}^{l}\left(\sum_{k=1}^{q}\boldsymbol{\alpha}_i^T(\boldsymbol{h}_{ki}^T\boldsymbol{h}_{kj})\boldsymbol{\alpha}_j\right)(\boldsymbol{x}_i^T\boldsymbol{x}_j) + \frac{1}{2}\sum_{i,j=1}^{l}\left(\sum_{k=1}^{q}\boldsymbol{\alpha}_i^T(\boldsymbol{h}_{ki}^T\boldsymbol{h}_{kj})\boldsymbol{\alpha}_j\right) + \frac{1}{2}\sum_{i=1}^{l}\left(\frac{1}{C_i}\boldsymbol{\alpha}_i^T\boldsymbol{\alpha}_i\right)\\ &= \frac{1}{2}\sum_{i,j=1}^{l}\boldsymbol{\alpha}_i^T(\boldsymbol{H}_i^T\boldsymbol{H}_j)\boldsymbol{\alpha}_j(\boldsymbol{x}_i^T\boldsymbol{x}_j) + \frac{1}{2}\sum_{i,j=1}^{l}\boldsymbol{\alpha}_i^T(\boldsymbol{H}_i^T\boldsymbol{H}_j)\boldsymbol{\alpha}_j + \frac{1}{2}\sum_{i=1}^{l}\left(\frac{1}{C_i}\boldsymbol{\alpha}_i^T\boldsymbol{\alpha}_i\right)\\ &= \frac{1}{2}\boldsymbol{\alpha}^T\left((\boldsymbol{H}^T\boldsymbol{H})\odot\boldsymbol{K} + \boldsymbol{H}^T\boldsymbol{H} + \boldsymbol{D}\right)\boldsymbol{\alpha} = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\Theta}\boldsymbol{\alpha},\\ &\quad \text{s.t. } \boldsymbol{u}^T\boldsymbol{\alpha}=1,\ \boldsymbol{\alpha}\ge\boldsymbol{0},\end{aligned} \tag{37}$$

where

$$\boldsymbol{D} = diag(\boldsymbol{u}_1^T/C_1,\ldots,\boldsymbol{u}_l^T/C_l),$$
$$\boldsymbol{\Theta} = (\boldsymbol{H}^T\boldsymbol{H})\odot\boldsymbol{K} + (\boldsymbol{H}^T\boldsymbol{H}) + \boldsymbol{D}. \tag{38}$$

Similarly, our Rank-CVM (37) and its q discriminant functions (15) can also be kernelized by various kernels satisfying Mercer theorem [32]. Additionally, Rank-CVM also needs a threshold function, which is estimated using the same method as in Rank-SVM. Although our Rank-CVM (37) has the same number of variables to be solved as Rank-SVM (26), due to a unit simplex constraint in (37), we can construct a fast training algorithm based on FW for our Rank-CVM.

7. A fast training algorithm for Rank-CVM

In this section, a fast training algorithm for our Rank-CVM is constructed based on the Frank–Wolfe iterative linearization algorithm.

7.1. Frank–Wolfe algorithm

In this sub-section we briefly review the Frank–Wolfe (FW) method and its three special cases, from which both Rank-SVM and Rank-CVM will benefit a lot. FW is a simple and classical first order feasible direction optimization method [19], which was originally proposed to solve quadratic programming and then extended to solve convex problems with continuously differential objective function and linear (and box) constraints:

$$\min f(\boldsymbol{x}),\ \text{s.t.}\boldsymbol{x}\in S. \tag{39}$$

The set S is a nonempty and bounded polyhedron of the form,

$$S = \{\boldsymbol{x}|\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}, \boldsymbol{l}\le\boldsymbol{x}\le\boldsymbol{u}\}, \tag{40}$$

where $\boldsymbol{A}$ is a $m\times n$ matrix, $\boldsymbol{b}$ represents a m-dimensional column vector, and $\boldsymbol{l}$ and $\boldsymbol{u}$ denote the lower and upper bounds of $\boldsymbol{x}$. FW generates a sequence of feasible vectors $\{\boldsymbol{x}^{(p)}\}$ using a linear search $\boldsymbol{x}^{(p+1)}=\boldsymbol{x}^{(p)}+\lambda^{(p)}\boldsymbol{d}^{(p)}$, where $\lambda^{(p)}\in[0,1]$ is a step size and $\boldsymbol{d}^{(p)}=\bar{\boldsymbol{x}}^{(p)}-\boldsymbol{x}^{(p)}$ a feasible descent direction satisfying $\bar{\boldsymbol{x}}^{(p)}\in S$ and $z^{(p)}=(\boldsymbol{d}^{(p)})^T\nabla f(\boldsymbol{x}^{(p)})<0$. To find out a best feasible direction, i.e., the best $\bar{\boldsymbol{x}}^{(p)}$, FW utilizes the first order Taylor series expansion of $f(\boldsymbol{x})$ around the vector $\boldsymbol{x}^{(p)}$ and then solves a linear programming problem with linear constraints (40):

$$\bar{\boldsymbol{x}}^{(p)} = \underset{\boldsymbol{x}\in S}{\operatorname{argmin}}\left(f(\boldsymbol{x}^{(p)}) + (\boldsymbol{x}-\boldsymbol{x}^{(p)})^T\nabla f(\boldsymbol{x}^{(p)})\right) = \underset{\boldsymbol{x}\in S}{\operatorname{argmin}}\ \boldsymbol{x}^T\nabla f(\boldsymbol{x}^{(p)}). \tag{41}$$

Such a linear programming problem is usually optimized by the widely used simplex or interior point method. The basic FW algorithm for (39) can be stated as follows:

Step 1 (initializing): Select an initial feasible vector $\boldsymbol{x}^{(p)}\in S$ with $p=1$ and set a stop criterion ε.
Step 2 (solving a linear programming problem): Let $\bar{\boldsymbol{x}}^{(p)} = \underset{\boldsymbol{x}\in S}{\operatorname{argmin}}\ \boldsymbol{x}^T\nabla f(\boldsymbol{x}^{(p)})$. If $|z^{(p)}|\le\varepsilon$, then stop.
Step 3 (executing a linear search for step size): Let $\lambda^{(p)} = \underset{\lambda\in[0,1]}{\operatorname{argmin}}\ f(\boldsymbol{x}^{(p)}+\lambda(\bar{\boldsymbol{x}}^{(p)}-\boldsymbol{x}^{(p)}))$.
Step 4 (updating): Set $\boldsymbol{x}^{(p+1)}=\boldsymbol{x}^{(p)}+\lambda^{(p)}(\bar{\boldsymbol{x}}^{(p)}-\boldsymbol{x}^{(p)})$ and $p=p+1$, go to Step 2.

It was proved that the above FW algorithm has a sub-linear convergence [19,20]. We rewrite the objective functions in Rank-SVM and our Rank-CVM into a unified concise quadratic form:

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Theta}\boldsymbol{x} + \boldsymbol{\theta}^T\boldsymbol{x}, \tag{42}$$

where $\boldsymbol{\Theta}=(\boldsymbol{H}^T\boldsymbol{H})\odot\boldsymbol{K}$ and $\boldsymbol{\theta}=\boldsymbol{u}$ in Rank-SVM, and $\boldsymbol{\Theta}=(\boldsymbol{H}^T\boldsymbol{H})\odot\boldsymbol{K}+(\boldsymbol{H}^T\boldsymbol{H})+\boldsymbol{D}$ and $\boldsymbol{\theta}=\boldsymbol{0}$ in Rank-CVM. For some widely used Mercer kernels, e.g., linear, polynomial and RBF kernels, such an objective function (42) is strictly convex generally [32]. To speed up the training procedures of Rank-SVM and our Rank-CVM, three special cases can be exploited, i.e.,

Case 1. For the strictly convex objective function (42), the analytical step size of FW becomes

$$\lambda^{(p)} = \min\left\{-\frac{z^{(p)}}{(\boldsymbol{d}^{(p)})^T\boldsymbol{\Theta}\boldsymbol{d}^{(p)}}, 1\right\}. \tag{43}$$

This formula can be used to estimate the step size for Rank-SVM and our Rank-CVM efficiently.

Case 2. If the strictly convex function $f(\boldsymbol{x})$ in (42) has an optimal value f^*, and $\{\boldsymbol{x}^{(p)}\}$ and $\{z^{(p)}\}$ are derived by FW ($p=1,2\ldots$, $\boldsymbol{x}^{(1)}\in S$), then,

$$f(\boldsymbol{x}^{(p)})-f^*\le -z^{(p)} = -(\boldsymbol{\Theta}\boldsymbol{x}^{(p)}+\boldsymbol{\theta})^T\boldsymbol{d}^{(p)}. \tag{44}$$

Therefore, when $|z^{(p)}|\le\varepsilon$, we have $f(\boldsymbol{x}^{(p)})-f^*\le\varepsilon$. This implies that, when the stopping criterion ε is satisfied in Step 2 the difference between the current function value and the optimal one is less than ε.

Case 3. For a unit simplex constraint in our Rank-CVM, the linear programming in Step 3 is

$$\min \sum_{i=1}^{n} (\boldsymbol{\Theta x}^{(p)})_i \bar{x}_i^{(p)}, \text{ s.t.} \sum_{i=1}^{n} \bar{x}_i^{(p)} = 1, \tag{45}$$

where $(\boldsymbol{\Theta x}^{(p)})_i$ denotes the ith component of gradient vector. Then, the above (45) has an analytical solution:

$$\bar{x}_j^{(p)} = \begin{cases} 1 & \text{if } j = \underset{i=1,\ldots,n}{\operatorname{argmin}} (\boldsymbol{\Theta x}^{(p)})_i, \\ 0 & \text{otherwise}. \end{cases} \tag{46}$$

That is, the component with a minimum derivate is 1, and the other components all are 0. Therefore, we can obtain an analytical solution directly for our Rank-CVM, which greatly reduces the computational time of the training procedure.

To decide how many iterations are theoretically needed to solve Rank-SVM and our Rank-CVM, we provide the rate of convergence of FW for (42) using the following theorem 1, in which we utilize some techniques in [19,20,23].

Theorem 1. Let $\boldsymbol{x}^*$ be optimal for the quadratic programming (42) with the feasible set S (40), and $\{\boldsymbol{x}^{(p)}\}$ a sequence generated by FW. Then there exists a constant $\xi \geq -1$ such that

$$f(\boldsymbol{x}^{(p)}) - f(\boldsymbol{x}^*) \leq \frac{2\|\boldsymbol{\Theta}\|_F \Delta^2}{p+\xi}, \tag{47}$$

where $\|\boldsymbol{\Theta}\|_F$indicates Frobenius norm of matrix, and Δ is the diameter of S.

Proof. According to $\boldsymbol{x}^{(p+1)} = \boldsymbol{x}^{(p)} + \lambda^{(p)}\boldsymbol{d}^{(p)}$ and the step size $\lambda^{(p)}$ in (43), we have

$$\begin{aligned} f(\boldsymbol{x}^{(p+1)}) &= \frac{1}{2}\left(\boldsymbol{x}^{(p)} + \lambda^{(p)}\boldsymbol{d}^{(p)}\right)^T \boldsymbol{\Theta}\left(\boldsymbol{x}^{(p)} + \lambda^{(p)}\boldsymbol{d}^{(p)}\right) + \boldsymbol{\theta}^T\left(\boldsymbol{x}^{(p)} + \lambda^{(p)}\boldsymbol{d}^{(p)}\right) \\ &= f(\boldsymbol{x}^{(p)}) - \frac{1}{2}\frac{\left((\boldsymbol{\Theta x}^{(p)} + \boldsymbol{\theta})^T \boldsymbol{d}^{(p)}\right)^2}{(\boldsymbol{d}^{(p)})^T \boldsymbol{\Theta d}^{(p)}}. \end{aligned} \tag{48}$$

Using (44), the above (48) can be rewritten as

$$f(\boldsymbol{x}^{(p+1)}) - f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}^{(p)}) - f(\boldsymbol{x}^*) - \frac{1}{2}\frac{\left(f(\boldsymbol{x}^{(p)}) - f(\boldsymbol{x}^*)\right)^2}{(\boldsymbol{d}^{(p)})^T \boldsymbol{\Theta d}^{(p)}}. \tag{49}$$

Set $\delta^{(p)} = 2(\boldsymbol{d}^{(p)})^T \boldsymbol{\Theta d}^{(p)} \leq 2\|\boldsymbol{\Theta}\|_F \|\boldsymbol{d}^{(p)}\|_2^2 \leq 2\|\boldsymbol{\Theta}\|_F \Delta^2 = \delta$, where $\Delta = \max_p \|\boldsymbol{d}^{(p)}\|_2$, namely, the diameter of S. Since $(1-a)(1+a) = 1 - a^2 \leq 1$ if $a \geq 0$, the inequality holds: $1 - a \leq 1/(1+a)$.

Denoting $\beta^{(p)} = f(\boldsymbol{x}^{(p)}) - f(\boldsymbol{x}^*) \geq 0$, it follows that

$$\beta^{(p+1)} \leq \beta^{(p)} - \frac{(\beta^{(p)})^2}{\delta^{(p)}} \leq \frac{\beta^{(p)}}{1 + \beta^{(p)}/\delta^{(p)}} = \frac{1}{1/\beta^{(p)} + 1/\delta^{(p)}} \leq \frac{1}{1/\beta^{(p)} + 1/\delta} \tag{50}$$

For $k=1$, its step size $\lambda^{(1)} = -(\boldsymbol{\Theta x}^{(1)} + \boldsymbol{\theta})^T \boldsymbol{d}^{(1)}/\delta^{(1)} \leq 1$ and according to (44), we have $\beta^{(1)} \leq \lambda^{(1)}\delta^{(1)} \leq \lambda^{(1)}\delta$.

By inducing, we derive the following inequalities:

$$\beta^{(2)} \leq \frac{1}{1/\delta + 1/\lambda^{(1)}\delta} = \frac{\delta}{1 + 1/\lambda^{(1)}},\ \beta^{(3)} \leq \frac{1}{1/\delta + 1/\beta^{(2)}} = \frac{\delta}{2 + 1/\lambda^{(1)}} \ldots,$$

$$\beta^{(p)} \leq = \frac{\delta}{p - 1 + 1/\lambda^{(1)}}. \tag{51}$$

Since $0 \leq \lambda^{(1)} \leq 1$, let $\xi = 1/\lambda_1 - 1 \geq -1$, and then we have

$$f(\boldsymbol{x}^{(p)}) - f(\boldsymbol{x}^*) \leq \frac{\delta}{p+\xi} = \frac{2\|\boldsymbol{\Theta}\|_F \Delta^2}{p+\xi}, \tag{52}$$

which completes our proof. This theorem implies that, to achieve a ε-accuracy solution, i.e., $\varepsilon = f(\boldsymbol{x}^{(p)}) - f(\boldsymbol{x}^*)$, FW needs $O(2\|\boldsymbol{\Theta}\|_F \Delta^2/\varepsilon)$ iterations for a convex quadratic programming (42). □

7.2. Initialization for training algorithm of Rank-CVM

In order to apply the above FW method to our Rank-CVM, we have to initialize a solution vector and some other key quantities. According to (38), the elements in the matrix $\boldsymbol{\Theta}$ are

$$\Theta_{ii'}^{jj'} = \sum_{k=1}^{q} h_{ki}^{j} h_{ki'}^{j'} (\boldsymbol{x}_i^T \boldsymbol{x}_{i'} + 1) + \frac{1}{C_i}\delta_{ii'}^{jj'};\ i,i' = 1,\ldots,l,\ j = 1,\ldots,l_i,\ j' = 1,\ldots,l_{i'}, \tag{53}$$

where $\delta_{ii'}^{jj'} = 1$, if $i = i'$, $j = j'$ and $\delta_{ii'}^{jj'} = 0$ otherwise. This element in (53) corresponds to the jth row of the ith instance and the j'th column of the i'th instance. But, the diagonal elements only have l different values,

$$\Theta_{ii}^{jj} = 2\left((\boldsymbol{x}_i^T \boldsymbol{x}_i) + 1\right) + \frac{1}{C_i};\ \ i = 1,\ldots,l \text{ and any } j. \tag{54}$$

We search for the i'th instance which minimize the above (54), i.e.,

$$i' = \underset{i=1,\ldots,l}{\operatorname{argmin}}\ \Theta_{ii}^{jj}. \tag{55}$$

Therefore, according to the unit simplex constraint, we choose an initial solution as

$$\alpha_i^{j(1)} = \begin{cases} 1, & \text{if } i = i', j = 1, \\ & ; i = 1,\ldots,l,\ j = 1,\ldots,l_i, \\ 0, & \text{otherwise}, \end{cases} \tag{56}$$

where the superscript (1) indicates the initial solution. In (56), the first component of the i'th instance is set to be 1 only. In this case, the initial objective function value is

$$W^{(1)}(\boldsymbol{\alpha}) = \left((\boldsymbol{x}_i^T \boldsymbol{x}_i) + 1\right) + \frac{1}{2C_{i'}} \tag{57}$$

and the initial $\beta_{ki}^{(1)}$ becomes

$$\beta_{ki}^{(1)} = \begin{cases} h_{ki}^1 & \text{if } i = i', \\ 0 & \text{otherwise}, \end{cases} ; i = 1,\ldots,l,\ k = 1,\ldots,q, \tag{58}$$

where h_{ki}^1 denotes the first element of the row vector $\boldsymbol{h}_{ki}$. This indicates that the initial $\beta_{ki'}^{(1)} (k = 1,\ldots,q)$ are set to be the first column vector of $\boldsymbol{H}_{i'}$. The gradient vector of the objective function in (37) is

$$\boldsymbol{g}_i = \sum_{j=1}^{l}\sum_{k=1}^{q} (\boldsymbol{h}_{ki}^T \boldsymbol{h}_{kj})\boldsymbol{\alpha}_j\left((\boldsymbol{x}_i^T \boldsymbol{x}_j) + 1\right) + \frac{1}{C_i}\boldsymbol{\alpha}_i, \text{or } \boldsymbol{g} = \boldsymbol{\Theta\alpha}. \tag{59}$$

where $\boldsymbol{g}_i$ corresponds to $\boldsymbol{\alpha}_i$ of the ith instance, and $\boldsymbol{g} = \begin{bmatrix} \boldsymbol{g}_1^T & \cdots & \boldsymbol{g}_l^T \end{bmatrix}^T$. Therefore the initial gradients are

$$g_i^{j(1)} = \begin{cases} \displaystyle\sum_{k=1}^{q} h_{ki}^j h_{ki'}^1 \left((\boldsymbol{x}_i^T \boldsymbol{x}_{i'}) + 1\right) + \frac{1}{C_i} & \text{if } i = i', j = 1, \\ & ; i = 1,\ldots,l, j = 1,\ldots,l_i. \\ \displaystyle\sum_{k=1}^{q} h_{ki}^j h_{ki'}^1 \left((\boldsymbol{x}_i^T \boldsymbol{x}_{i'}) + 1\right) & \text{otherwise}, \end{cases} \tag{60}$$

7.3. Some useful recursive formulae for training algorithm of Rank-CVM

After the above quantities have been initialized, we derive some useful recursive formulae in this sub-section. At the pth ($p = 1, 2, \ldots$) iteration, assume the minimum gradient component to be $g_{i'}^{j'(p)}$. According to the special case 3 in Section 7.1, the corresponding solution of linear programming becomes

$$\bar{\alpha}_i^{j(p)} = \begin{cases} 1, & \text{if } i = i', j = j', \\ 0, & \text{otherwise,} \end{cases} ; i = 1\ldots, l, j = 1, \ldots, l_i, \tag{61}$$

where $\bar{\alpha}_i^j$ denotes the jth component of the ith instance. The quantity $z^{(p)}$ is used to decide whether the iterative procedure should be terminated, and to calculate the step size (43),

$$z^{(p)} = (\mathbf{g}^{(p)})^T(\bar{\boldsymbol{\alpha}}^{(p)} - \boldsymbol{\alpha}^{(p)}) = (\boldsymbol{\Theta}\boldsymbol{\alpha}^{(p)})^T(\bar{\boldsymbol{\alpha}}^{(p)}) - (\boldsymbol{\Theta}\boldsymbol{\alpha}^{(p)})^T(\boldsymbol{\alpha}^{(p)}) \\ = g_{i'}^{j'(p)} - 2W(\boldsymbol{\alpha}^{(p)}). \tag{62}$$

The denominator for estimating the step size (43) becomes

$$(\mathbf{d}^{(p)})^T\boldsymbol{\Theta}\mathbf{d}^{(p)} = (\bar{\boldsymbol{\alpha}}^{(p)} - \boldsymbol{\alpha}^{(p)})^T\boldsymbol{\Theta}(\bar{\boldsymbol{\alpha}}^{(p)} - \boldsymbol{\alpha}^{(p)}) = \Theta_{i'i'}^{j'j'} - 2g_{i'}^{j'(p)} + 2W(\boldsymbol{\alpha}^{(p)}). \tag{63}$$

Therefore, the step size (43) is

$$\lambda^{(p)} = \min\left\{ -\frac{g_{i'}^{j'(p)} - 2W(\boldsymbol{\alpha}^{(p)})}{\Theta_{i'i'}^{j'j'} - 2g_{i'}^{j'(p)} + 2W(\boldsymbol{\alpha}^{(p)})}, 1 \right\}. \tag{64}$$

Next we update some key quantities. For the solution vector $\boldsymbol{\alpha}$, we have its vector form:

$$\boldsymbol{\alpha}^{(p+1)} = (1-\lambda^{(p)})\boldsymbol{\alpha}^{(p)} + \lambda^{(p)}\bar{\boldsymbol{\alpha}}^{(p)} \tag{65}$$

and its component form,

$$\alpha_i^{j(p+1)} = \begin{cases} (1-\lambda^{(p)})\alpha_i^{j(p)} + \lambda^{(p)} & \text{if } i = i', j = j', \\ (1-\lambda^{(p)})\alpha_i^{j(p)} & \text{otherwise,} \end{cases} ; i = 1, \ldots, l, j = 1, \ldots, l_i. \tag{66}$$

The recursive formula for β_{ki} is

$$\beta_{ki}^{(p+1)} = \mathbf{h}_{ki}\boldsymbol{\alpha}_i^{(p+1)} = (1-\lambda^{(p)})\beta_{ki}^{(p)} + \lambda^{(p)}\mathbf{h}_{ki}\bar{\boldsymbol{\alpha}}_i^{(p)} \\ = \begin{cases} (1-\lambda^{(p)})\beta_{ki}^{(p)} + \lambda^{(p)}h_{ki}^{j'} & \text{if } i = i', \\ (1-\lambda^{(p)})\beta_{ki}^{(p)}, & \text{otherwise,} \end{cases} ; i = 1, \ldots, l, \ k = 1, \ldots, q. \tag{67}$$

For the objective function value $W(\boldsymbol{\alpha})$, we have

$$W(\boldsymbol{\alpha}^{(p+1)}) = \frac{1}{2}(\boldsymbol{\alpha}^{(p+1)})^T\boldsymbol{\Theta}(\boldsymbol{\alpha}^{(p+1)}) \\ = (1-\lambda^{(p)})^2W(\boldsymbol{\alpha}^{(p)}) + \lambda^{(p)}(1-\lambda^{(p)})g_{i'}^{j'(p)} + \frac{1}{2}(\lambda^{(p)})^2\Theta_{i'i'}^{j'j'}. \tag{68}$$

The recursive formula for gradient vector is

$$\mathbf{g}^{(p+1)} = \boldsymbol{\Theta}\boldsymbol{\alpha}^{(p+1)} = (1-\lambda^{(p)})\boldsymbol{\Theta}\boldsymbol{\alpha}^{(p)} + \lambda^{(p)}\boldsymbol{\Theta}\bar{\boldsymbol{\alpha}}^{(p)} = (1-\lambda^{(p)})\mathbf{g}^{(p)} + \lambda^{(p)}\boldsymbol{\Theta}\bar{\boldsymbol{\alpha}}^{(p)}, \tag{69}$$

and the corresponding component form is

$$g_i^{j(p+1)} = (1-\lambda^{(p)})g_i^{j(p)} + \lambda^{(p)}\Theta_{ii'}^{ij'}; \ i = 1, \ldots, l, j = 1, \ldots, l_i. \tag{70}$$

Our Rank-CVM utilizes the same discriminant functions (15) or (27) as Rank-SVM, so we only need to update β_{ki}, which can be referred to as a proxy solution to replace $\boldsymbol{\alpha}$ in Rank-CVM. Now, our new method has an analytical step size (64), and some efficient recursive formulae for the objective function value (68), gradients (70) and proxy solution (67), which can speed up the training procedure of Rank-CVM dramatically.

7.4. A fast training algorithm for Rank-CVM

Now, we list our fast training algorithm for Rank-CVM as follows, in which two terminated indexes are used at the same time, i.e., ε and M for the objective function value and maximal epochs, and the time complexity of the most time consuming computations are given too.

Step 1: Set two terminated indexes: ε and M, and then initialize $\beta_{ki}^{(p)}(k = 1, \ldots, q, i = 1, \ldots, l)$ using (58), $g_i^{j(p)}(i = 1, \ldots, l, j = 1, \ldots, l_i)$ using (60) and $W^{(p)}(\boldsymbol{\alpha})$ using (57), where $p = 1$.
Step 2: Search for the minimal component of gradient vector, i.e., $g_{i'}^{j'(p)}(O(l_t))$.
Step 3: Calculate $z^{(p)}$ according to (62). If $|z^{(p)}| < \varepsilon$, then stop.
Step 4: Estimate the step size $\lambda^{(p)}$ according to (64).
Step 5: Update $\beta_{ki}^{(p+1)}(O(4ql))$ using (67), $g_i^{j(p+1)}(O((3q+6)l_t))$ using (70) and $W^{(p+1)}(\boldsymbol{\alpha})$ using (68).
Step 6: Let $p = p+1$. If p reaches to $(M \times l_t)$, then stop; else go to Step 2.

When FW is applied to Rank-SVM, at its each iteration or epoch, the entire solution vector of size l_t is updated at the same time. For the sake of comparison, in the above training procedure for our Rank-CVM, each epoch indicates to execute l_t iterations.

7.5. Time complexity analysis

For Rank-SVM, it is pointed out that the time complexity of any epoch is denoted by $O(ql_t^2)$ at most [16]. In our Rank-CVM, the most computational time comes from updating the gradient vector of length l_t in Step 5 where the elements $\Theta_{ii'}^{ij'}$ are recalculated, updating the proxy solution in Step 5, and searching for the minimal gradient component in Step 2, whose total time complexity is $O((3q+7)l_t + 4ql)$. Since one epoch indicates to execute l_t iterations in our Rank-CVM, the time complexity is $O((3q+7)l_t^2 + 4qll_t)$. This means that for any epoch, the time complexity of our Rank-CVM is at least three times as high as that of Rank-SVM.

To achieve a ε-accuracy solution, the number of iterations depends on $O(2\|\boldsymbol{\Theta}\|_F\Delta^2/\varepsilon)$ according to Theorem 1. Now we estimate such a quantity at the worst case for Rank-SVM and our Rank-CVM respectively. Since each column of $\boldsymbol{H}$ consists of "+1", "-1" and q-2 "0 s", we have

$$||\boldsymbol{H}^T\boldsymbol{H}||_F \le ||\boldsymbol{H}||_F^2 = 2l_t \tag{71}$$

and thus

$$||(\boldsymbol{H}^T\boldsymbol{H}) \odot \boldsymbol{K}||_F \le K^{\max}||(\boldsymbol{H}^T\boldsymbol{H})||_F \le 2l_tK^{\max}, \tag{72}$$

where $K^{\max}$indicates the absolute maximum of kernel matrix elements. Additionally, for any instance, we have the inequalities, $(q-1) \le |L_i||\bar{L}_i| \le \lfloor q/2 \rfloor(1 - \lfloor q/2 \rfloor)$,where $\lfloor \cdot \rfloor$ represents floor operation, e.g., $\lfloor 2.9 \rfloor = 2$. For Rank-SVM, we estimate,

$$\|\boldsymbol{\Theta}\|_F^{Rank-SVM} = \|(\boldsymbol{H}^T\boldsymbol{H}) \odot \boldsymbol{K}\|_F \le 2l_tK^{\max}, \\ (\Delta^{Rank-SVM})^2 \le \|\mathbf{C}\|_2^2 \le l_tC^2/(q-1)^2. \tag{73}$$

For Rank-CVM, we have $\|\boldsymbol{D}\|_F \le \sqrt{l_t}\lfloor q/2 \rfloor(q - \lfloor q/2 \rfloor)/C$ and $\boldsymbol{u}^T\boldsymbol{\alpha} = 1$, and thus

$$\|\boldsymbol{\Theta}\|_F^{\text{Rank-CVM}} = ||(\boldsymbol{H}^T\boldsymbol{H}) \odot \boldsymbol{K} + (\boldsymbol{H}^T\boldsymbol{H}) + \boldsymbol{D}||_F \le ||(\boldsymbol{H}^T\boldsymbol{H}) \\ \odot \boldsymbol{K}||_F + ||\boldsymbol{H}^T\boldsymbol{H}||_F + ||\boldsymbol{D}||_F \\ \le 2l_tK^{\max} + 2l_t + \sqrt{l_t}\lfloor q/2 \rfloor(q - \lfloor q/2 \rfloor)/C, \\ (\Delta^{\text{Rank-CVM}})^2 = \max_p ||\boldsymbol{d}^{(p)}||_2^2 = \max_p ||\bar{\boldsymbol{\alpha}}^{(p)} - \boldsymbol{\alpha}^{(p)}||_2^2 \\ \le \max_p \left(||\bar{\boldsymbol{\alpha}}^{(p)}||_2^2 + ||\boldsymbol{\alpha}^{(p)}||_2^2\right) \\ \le \max_p \left(||\bar{\boldsymbol{\alpha}}^{(p)}||_1 + ||\boldsymbol{\alpha}^{(p)}||_1\right) = \max_p \left(\boldsymbol{u}^T\bar{\boldsymbol{\alpha}}^{(p)} + \boldsymbol{u}^T\boldsymbol{\alpha}^{(p)}\right) = 2. \tag{74}$$

Since one epoch represents one iteration in Rank-SVM and l_t iterations in our Rank-CVM, with RBF kernel ($K^{\max} = 1$), at the worst case the number of epochs for Rank-SVM and Rank-CVM are

$$O^{\text{Rank-SVM}}(4l_t^2C^2/(q-1)^2/\varepsilon), \quad (75)$$

$$O^{\text{Rank-CVM}}\left(\left[16+4\lfloor q/2\rfloor(q-\lfloor q/2\rfloor)/C\sqrt{l_t}\right]/\varepsilon\right). \quad (76)$$

It can be observed that Rank-CVM needs much less epochs than Rank-SVM to achieve a ε-accuracy solution.

8. Experiments

In this section, we compare our Rank-CVM with four existing multi-label classification approaches including Rank-SVM experimentally. Before presenting our experimental results, we briefly introduce four existing methods and nine benchmark data sets.

8.1. Four existing multi-label methods

In this paper, we selected four existing multi-label classification methods: Rank-SVM [15], ADTree [24], BP-MLL [6] and ML-kNN [9], which will be compared with our Rank-CVM experimentally. It is worth noting that the first three methods and our Rank-CVM belong to algorithm extension methods considering all training instances and all labels at the same time, while ML-kNN is a typical hybrid method based on the OVR data decomposition trick. We downloaded the original C/C++ software package of ADTree from [42] and the Matlab ones of BP-MLL and ML-kNN from [43]. For ADTree, we utilize its original name rather than ADTBoost.MR in [6,9], and slightly modified its software to calculate the ranking loss. On the other hand, we accept their recommended parameter settings. For ADTree, the number of boosting rounds is set to be 50. For BP-MLL, the number of hidden neurons is 20% 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 ML-kNN, the Laplacian estimator ($s=1$) is used and the number of nearest instances $k=10$.

In order to improve the training efficiency of Rank-SVM, we utilized an analytical step size (64) and coded its implementation using C/C++ language, in which the free C language package LPSOL5.5 based on improved simplex method from [44] is adopted as our linear programming solver.

8.2. Nine data sets

To compare our Rank-CVM with the aforementioned four classification methods, we collected seven existing benchmark data sets: Emotions, Image, Scene, Genbase, Medical, Slashdot, and Yeast from [43,45,46], and constructed two new biological data sets: Plant and Human, as summarized in Table 1, according to the number of variables to be solved in the training algorithms of Rank-CVM and our Rank-SVM in the last column. Table 1 also shows some useful statistics of these data sets, such as, the number of instances in the training and test sets, the number of features, the number of labels, and the average labels. These data sets cover four distinct domains: text, scene, music and biology.

For seven existing data sets, the Image and Slashdot are partitioned into training and test sets randomly according to 60% versus 40% by us, while the other five training and test sets are downloaded directly from their web sites [45]. For more detailed information and description for these seven data sets, please refer to their corresponding web sites and references therein [43,45,46].

In bioinformatics, an important task is to develop computational methods to predict the subcellular locations of proteins according to their sequences. It has been observed that some multiplex proteins can simultaneously exist at two, or move between, two or more different location sites. Such a problem was identified as a multi-label classification, handled by the OVR decomposition trick simply, and evaluated by accuracy of each class [13]. But, this problem is dealt with under standard multi-label classification setting in this paper. We downloaded 978 and 3106 sequences for Plant and Human species from [47], and extracted 440 features (20 amino acid, 20 pseudo-amino acid and 400 dipetide compositions) from each protein sequence under the default setting using the free off-line software PseAAC-Builder from [48]. The Plant and Human include 12 location sites (cell membrace, cell wall, chloroplast, cytoplasm, endoplasmic reticulum, extracellular, golgi apparatus, mitochondrion, nucleus, peroxisome, plastid, and vacuole), and 14 ones (centriole, cytoplasm, cytoskeleton, endoplasm reticulum, endosome, extracell, golgi, apparatus, lysosome, microsome, mitochondrion, nucleus, peroxisome, plasma membrace, and synapse) respectively. Their average labels are 1.08 and 1.19. Finally we parse these two sets into training and test data sets randomly in terms of 60% versus 40%.

Table 1
Statistics for nine benchmark data sets used in our experiments.

Data set	Domain	Instances		Features	Classes	Average labels	Variables
		Train	Test				
Emotions[45]	Music	391	202	72	6	1.87	2793
Image[43]	Scene	1200	800	294	5	1.24	5342
Scene[45]	Scene	1211	1196	294	6	1.07	6278
Plant	Biology	588	390	440	12	1.08	6959
Genbase[45]	Biology	463	199	1185	27	1.35	14808
Human	Biology	1862	1244	440	14	1.19	28036
Medical[45]	Text	645	333	1449	45	1.25	34878
Slashdot[46]	Text	2269	1513	1079	22	1.18	55306
Yeast[45]	Biology	1500	917	103	14	4.24	58248

8.3. Tuning two key parameters for Rank-CVM and Rank-SVM on training sets

In this work, the RBF kernel $K(\boldsymbol{x},\boldsymbol{y})=\exp(-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|_2^2)$ is tested for Rank-SVM and our Rank-CVM, where γ denotes the kernel scale factor. Additionally, such two methods involve a regularization constant C. To reach an acceptable classification performance, $M=50$ are recommended without achieving a $\varepsilon=10^{-3}$ accuracy solution for Rank-SVM in [6,9], whereas $\varepsilon=10^{-3}$ is quickly satisfied within 1–2 epochs for our Rank-CVM. In this case, we set $\varepsilon=10^{-3}$ and $M=50$ for such two SVM-type methods. Therefore, now there are two key tunable parameters: γ and C. To reduce the search space of possible parameter combinations, we construct a lazy procedure to tune such two key parameters on training sets using three-fold cross validation, in which through fixing one parameter by turns we investigate a proper criterion function (some measure mentioned in Section 2, or some combination of these measures) as a function of the other parameter, and then detect a corresponding optimal value.

First, we validate whether all five measures defined in Section 2 have an identical optimal γ or C value. For the Emotions data set, we investigate five measures as different functions of 13 different γ values: $2^2, 2^1, \ldots, 2^{-10}$ given $C=1$, and 10 different C values: $2^8, 2^7, \ldots, 2^{-1}$ given $\gamma=2^{-1}$, as shown in Fig. 3. It is found out that the different measure corresponds to a slightly different optimal γ or C value, e.g., $\gamma=2^{-1}$ or 2^{-2} in Fig. 3(a), and $C=2^0, 2^1$ or 2^2 in Fig. 3 (b). This means that it is improper to use one of the five measures as a criterion function to tune two key parameters.

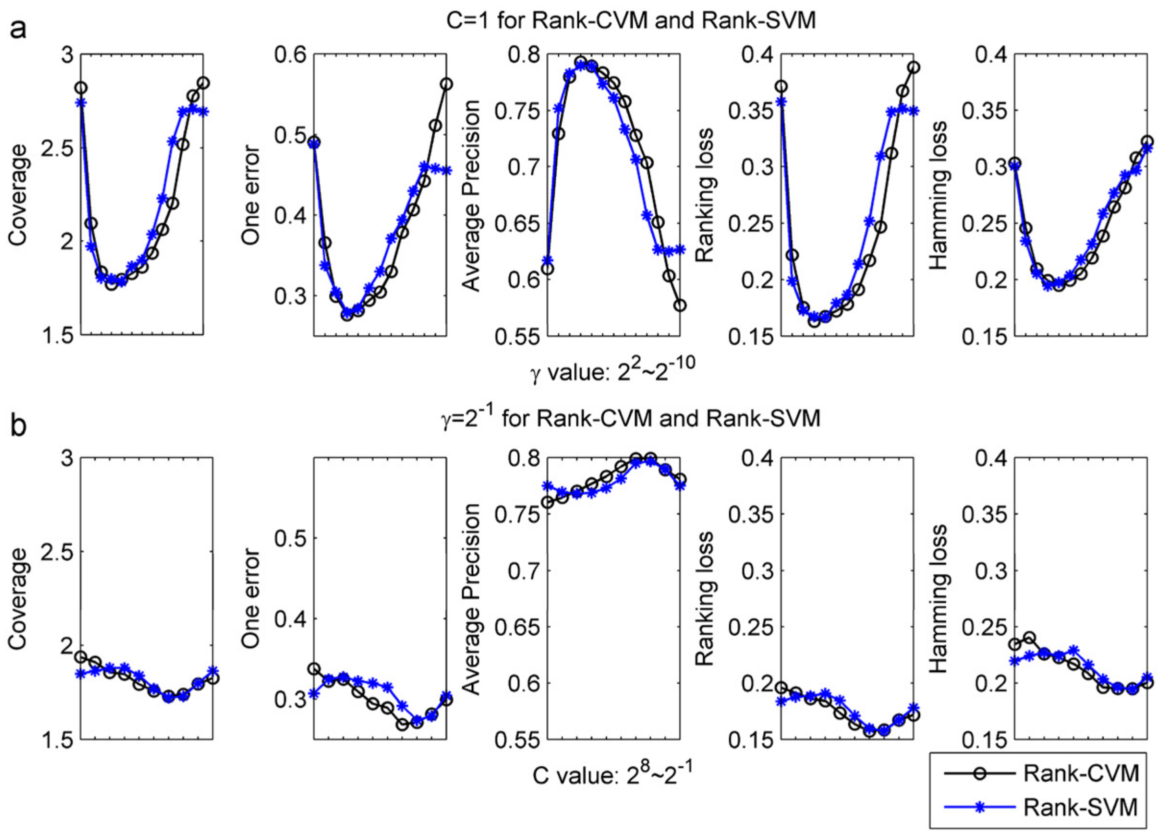

Fig. 3. Five measures as functions of different parameter (γ or C) on Emotions.

Therefore we define a tradeoff criterion function as follows:

$$\text{Criterion function} = (\text{Ranking loss} + \text{Hamming loss})/2. \quad (77)$$

According to (77), we conducted our lazy tuning procedure for our Rank-CVM and Rank-SVM on nine training data set listed in Table 1, as shown in Fig. 4. Fig. 4(a) shows the criterion function (77) as a function of 13 γ values from 2^2 to 2^{-10} given $C=1$, whose curves behave smoothly. Fig. 4(b) displays the criterion (77) as a function of 10 different C values from 2^8 to 2^{-1} given the optimal γ values form Fig. 4(a), whose curves oscillate slightly. According to Fig. 4, the optimal γ and C values, and their corresponding criterion function values are detected and listed in Table 2. It is inspiring that tuning C value indeed reduces the criterion function values, since all C values but one for Rank-CVM on Image is greater than 1. Among nine sets, our Rank-CVM performs better than Rank-SVM on five sets. But note that such two methods have no statistically significant difference based on the paired Wilcoxon sign ranked test with 5% significance level [49].

8.4. *Performance comparison with four existing methods on test sets*

In this sub-section, we compare our Rank-CVM with four existing methods: Rank-SVM, ADTree, BP-MLL, and ML-*k*NN using a train-test mode. According to those optimal parameter combinations in Table 2 for Rank-SVM and Rank-CVM from training sets, and those recommended parameter settings for the other three methods in Section 8.1, we retrained all five multi-label classification methods on nine training sets, and then verified their performance using nine independent test sets in Table 1 respectively. The detailed experimental results are shown in Tables 3–7 according to five different measures, where the best value of each data set among five methods is highlighted in boldface. To compare these methods statistically, we choose our Rank-CVM as a baseline, and compare whether our method is statistically equal to, or better or worse than, one of the remained four methods, denoted by "==", "»" and "«" respectively, using the paired Wilcoxon sign ranked test with 5% significance level [49]. The statistical comparison results are listed in the last row (denoted by W-test) of Tables 3–7.

In Table 3, Rank-CVM and Rank-SVM achieve two and seven best coverage values, which looks like that Rank-SVM is superior to our Rank-CVM. But based on the statistical test, there is no statistically significant difference between such two methods. Furthermore, our Rank-CVM performs statistically better than the remained three methods: ADTree, BP-MLL and ML-*k*NN.

In Table 4 of one error measure, our Rank-CVM and Rank-SVM work the best on five and four data sets respectively. On the Genbase data set, Rank-CVM, ADTree and BP-MLL achieve the same one error value. According to the Wilcoxon test, our Rank-CVM performs as well as Rank-SVM, and better than ADTree, BP-MLL and ML-*k*NN.

As to the average precision in Table 5, our Rank-CVM and Rank-SVM behave the best on four and five data sets. Based on the statistical test, again, our Rank-CVM is as good as Rank-SVM, and superior to ADTree, BP-MLL, and ML-*k*NN.

In Table 6 of ranking loss, our Rank-CVM and Rank-SVM work the best on two and seven data sets. Since the difference between such two algorithms is small, no significant difference is detected by the statistical test. According to the Wilcoxon test, our Rank-CVM is better than ADTree, BP-MLL and ML-*k*NN.

As per Hamming loss in Table 7, Rank-CVM, Rank-SVM, ADTree and ML-*k*NN work the best on one, five, one and two data sets. It is surprising that no statistical significant difference is detected between our Rank-CVM and the other four multi-label classifiers.

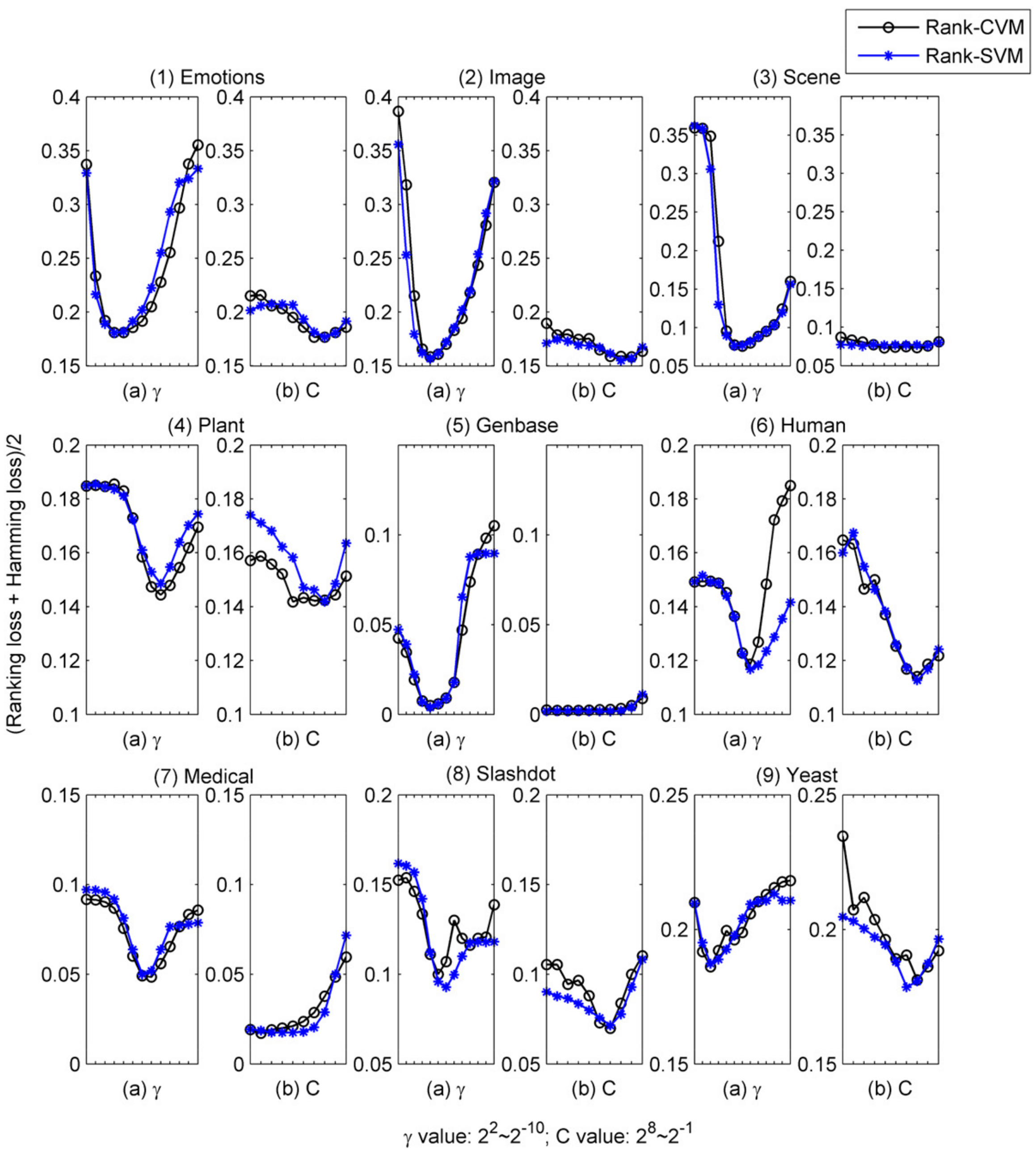

Fig. 4. Two key parameter tuning procedure on training sets for Rank-CVM and Rank-SVM.

Table 2
The optimal γ and C values, and corresponding criterion function values for nine training sets.

Data set	Rank-CVM			Rank-SVM		
	γ	C	Criterion	γ	C	Criterion
Emotions	2^{-2}	2^{1}	**0.17682**	2^{-1}	2^{1}	0.17687
Image	2^{-2}	2^{0}	0.15834	2^{-2}	2^{1}	**0.15505**
Scene	2^{-4}	2^{4}	**0.07317**	2^{-3}	2^{6}	0.07557
Plant	2^{-6}	2^{4}	**0.14179**	2^{-6}	2^{1}	0.14190
Genbase	2^{-2}	2^{5}	0.00224	2^{-2}	2^{3}	**0.00165**
Human	2^{-5}	2^{1}	0.11404	2^{-5}	2^{1}	**0.11267**
Medical	2^{-5}	2^{7}	**0.01721**	2^{-4}	2^{5}	0.01752
Slashdot	2^{-3}	2^{2}	**0.06981**	2^{-4}	2^{2}	0.07147
Yeast	2^{0}	2^{1}	0.18122	2^{0}	2^{2}	**0.17853**

From Tables 3–7, it is illustrated that our Rank-CVM performs as well as Rank-SVM, and better than the other three existing state-of-the-art multi-label classification techniques: ADTree, BP-MLL and ML-*k*NN.

8.5. Computational time comparison between Rank-CVM and Rank-SVM

In this sub-section, we compare the training and test time between Rank-CVM and Rank-SVM on nine data sets with the same parameter settings as in the above sub-section, where the test time is measured by the number of support vectors. Our computational platform is one desk computer with Intel Core i5 micro-processor including four 2.8 G CPUs and 8 G RAM, and MFC6.0.

In the training phases of nine data sets, our Rank-CVM experimentally converges within one epoch for satisfying $\varepsilon = 10^{-3}$, whereas Rank-SVM is terminated by 50 epochs, which verifies our theoretical analysis according to Theorem 1 in Sub-Section 7.5, that is, our Rank-CVM needs much less epochs than Rank-SVM. In Fig. 5(a), we investigate the training time as a function of the number of variables (listed in the last column of Table 1) in

Table 3
Coverage measure from five methods on nine data sets.

Data set	Rank-CVM	Rank-SVM	ADTree	BP-MLL	ML-*k*NN
Emotions	1.85149	**1.80693**	2.20792	1.95050	1.87620
Image	0.81500	**0.80375**	1.10000	1.62750	0.96625
Scene	**0.43896**	0.49415	0.68645	2.06270	0.56856
Plant	2.00513	**1.84359**	2.59744	5.98460	2.42310
Genbase	0.47236	**0.29146**	0.45226	0.59799	0.55276
Human	1.92122	**1.90997**	2.49598	5.79020	2.40510
Medical	**1.24324**	1.42943	1.95195	2.02100	2.72370
Slashdot	2.36550	**2.28949**	3.15400	2.33110	4.26240
Yeast	6.79171	**6.29335**	6.51908	6.42310	6.41440
W-test		= =	»	»	»

Table 4
One error measure from five methods on nine data sets.

Data set	Rank-CVM	Rank-SVM	ADTree	BP-MLL	ML-*k*NN
Emotions	0.29208	**0.28218**	0.37129	0.30693	0.30198
Image	0.25750	**0.25000**	0.38750	0.52625	0.32375
Scene	**0.19398**	0.20067	0.31940	0.82692	0.24247
Plant	**0.60513**	0.61026	0.73077	0.94872	0.66410
Genbase	**0.00000**	0.00503	**0.00000**	**0.00000**	0.00503
Human	0.51608	**0.51447**	0.61415	0.88746	0.60370
Medical	**0.13514**	0.13814	0.18919	0.36036	0.26426
Slashdot	**0.38929**	0.39458	0.50496	0.40582	0.66424
Yeast	0.25736	**0.22574**	0.25518	0.23664	0.23446
W-test		= =	»	»	»

Table 5
Average precision measure from five methods on nine data sets.

Data set	Rank-CVM	Rank-SVM	ADTree	BP-MLL	ML-*k*NN
Emotions	0.80105	**0.80575**	0.73629	0.77910	0.79073
Image	0.83278	**0.83740**	0.75262	0.63793	0.79163
Scene	**0.88314**	0.87442	0.80664	0.46483	0.85118
Plant	0.58294	**0.58912**	0.48275	0.23581	0.53646
Genbase	**0.99619**	0.99456	0.99422	0.99145	0.99144
Human	**0.65166**	0.65134	0.57115	0.33170	0.58114
Medical	**0.89898**	0.89445	0.84046	0.75889	0.79572
Slashdot	0.70452	**0.70662**	0.60862	0.69645	0.47754
Yeast	0.73944	**0.76902**	0.73723	0.75049	0.75846
W-test		= =	»	»	»

Table 6
Ranking loss measure from five methods on nine data sets.

Data set	Rank-CVM	Rank-SVM	ADTree	BP-MLL	ML-*k*NN
Emotions	0.15751	**0.15318**	0.22847	0.17992	0.16148
Image	0.13385	**0.13146**	0.21292	0.33948	0.17552
Scene	**0.06695**	0.07736	0.11796	0.39452	0.09308
Plant	0.17199	**0.15842**	0.22815	0.52593	0.21099
Genbase	0.00412	**0.00115**	0.00390	0.00762	0.00647
Human	0.12450	**0.12424**	0.16927	0.41341	0.16112
Medical	**0.01708**	0.01953	0.02966	0.03043	0.04245
Slashdot	0.09089	**0.08764**	0.12767	0.09016	0.17847
Yeast	0.18038	**0.16200**	0.18726	0.17504	0.17150
W-test		= =	»	»	»

Table 7
Hamming loss measure from five methods on nine data sets.

Data set	Rank-CVM	Rank-SVM	ADTree	BP-MLL	ML-*k*NN
Emotions	0.20627	**0.20380**	0.25660	0.21782	0.20875
Image	0.15300	**0.15200**	0.19175	0.22950	0.17225
Scene	0.08877	**0.08417**	0.11859	0.29097	0.09894
Plant	0.10620	0.10769	0.09551	0.09850	**0.08697**
Genbase	0.00149	0.00149	**0.00093**	0.00335	0.00462
Human	0.08860	0.08883	0.08331	0.09239	**0.08309**
Medical	0.01081	**0.01074**	0.01522	0.02002	0.01708
Slashdot	**0.04368**	0.04443	0.04957	0.04521	0.052785
Yeast	0.22901	**0.19263**	0.21234	0.20922	0.19801
W-test		= =	= =	= =	= =

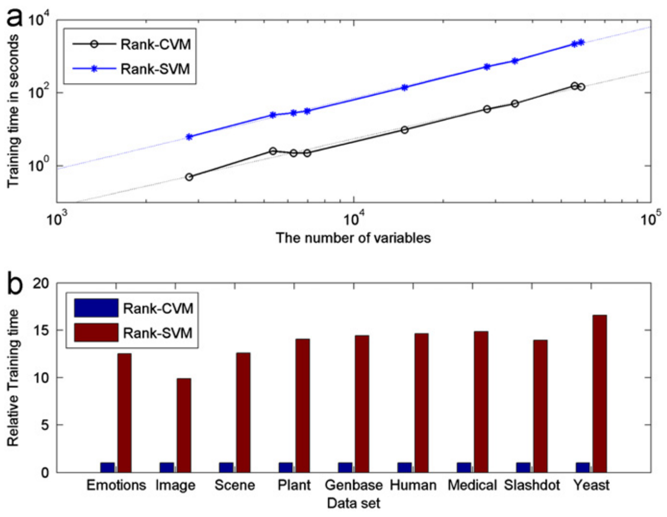

Fig. 5. The comparison of training time on nine data sets for Rank-CVM and Rank-SVM. (a) The training time as a function of the number of variables in log-scale. (b) The ratio of training time of Rank-SVM to our Rank-CVM.

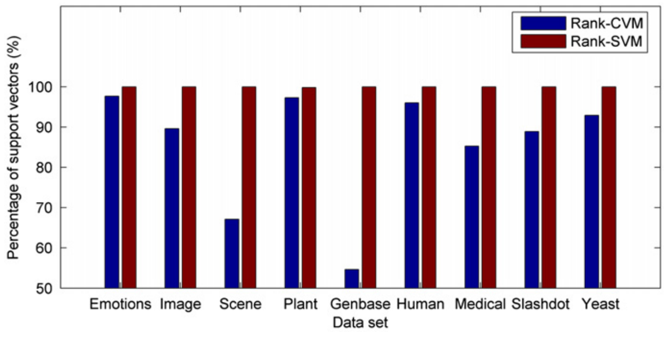

Fig. 6. The number of support vectors from Rank-CVM and Rank-SVM.

Rank-SVM and our Rank-CVM in log-scale, where $1.17 \times 10^{-6} l_t^{1.95}$ for Rank-SVM and $2.13 \times 10^{-7} l_t^{1.85}$ for Rank-CVM are fitted and shown in dotted lines. Further, we calculate the ratio of training time of Rank-SVM to our Rank-CVM, as shown in Fig. 5(b), where the ratio is at least 9.88 from Image, at most 16.56 from Yeast, 13.72 on an average. Therefore, Rank-CVM averagely runs 3.64 times as slow as Rank-SVM for each epoch, which is less than the theoretical value 4.04 estimated from Sub-Section 7.5.

For SVM-type classification algorithms, the test time is mainly determined by the number of support vectors, since evaluating kernel functions would spend the most fraction of test time. In this study, a training vector is regarded as a support vector if only its one non-zero coefficient occurs in its q discriminant functions (27). Fig. 6 shows the ratio of the number of support vectors to training vectors on nine data sets for Rank-CVM and Rank-SVM. The average ratios over nine training data sets are 85.50% for our Rank-CVM and 98.98% for Rank-SVM respectively. This means that the solution sparseness of such two methods is not good enough. But it is verified that the sparseness of Rank-CVM are statistically significantly better than that of Rank-SVM according the Wilcoxon signed rank test with 5% significance level.

According to the training time and the number of support vectors, it can be demonstrated that our Rank-CVM runs much faster than its rival Rank-SVM in the training and test phases.

9. Conclusions

The existing multi-label support vector machine (Rank-SVM) is a typical algorithm extension method, which considers all instances and all labels simultaneously to characterize label correlation sufficiently, resulting into an extremely complicated quadratic programming problem. Therefore it is highly desirable to design and implement a novel SVM-type multi-label algorithm. In this paper, a novel SVM-type algorithm for multi-label classification, i.e., Rank-CVM, is presented, which is derived through integrating Rank-SVM and binary core vector machine (CVM). Our quadratic programming problem has a special unit simplex constraint. When the Frank–Wolfe method is applied, our Rank-CVM has analytical solution and step size, and several recursive formulae for objective function, and gradient vector and proxy solution, at any iteration. All these tricks speed up the training procedure of our Rank-CVM greatly. Theoretical analysis shows that our Rank-has a lower time complexity than Rank-SVM. For nine data sets, our novel method runs on average 13 times faster and has less support vectors than Rank-SVM in the training phase under C++ environment. Experimental study also demonstrates that our Rank-CVM achieves rather competitive performance, compared with five typical multi-label classification methods, including Rank-SVM, ADTree, BP-MLL and ML-*k*NN, according to five indicative performance measures.

In future we will conduct a detailed model selection work to search optimal key parameters more efficiently for our Rank-CVM and construct a feature selection technique using our Rank-CVM.

Acknowledgments

This work is supported by Natural Science Foundation of China under grant no. 61273246 and 60875001.

References

[1] G. Tsoumakas, I. Katakis, Multi-label classification: an overview, International Journal of Data Warehousing and Mining 3 (3) (2007) 1–13.
[2] A.C.P.L.F. de Carvalho, A.A. Freitas, A tutorial on multi-label classification techniques, in: A. Abraham, A.E. Hassanien, V. Snasel (Eds.), Function Approximation and Classification, Foundations of Computational Intelligence, vol. 5, Springer, Berlin/Heidelberg, 2009, pp. 177–195.
[3] G. Tsoumakas, I. Katakis, I. Vlahavas, Mining multi-label data, in: O. Maimon, L. Rokach (Eds.), Data Mining and Knowledge Discovery Handbook, second ed., Springer, New York, 2010, pp. 667–685.
[4] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of the 10th European Conference on Machine Learning (ECML1998), Lecture Notes in Computer Science, vol. 1398, Chemnitz, Germany, 1998, pp. 137–142.
[5] R.E. Schapire, Y. Singer, Boostexter: a boosting-based system for text categorization, Machine Learning 39 (2/3) (2000) 135–168.
[6] M.L. Zhang, Z.H. Zhou, Multilabel neural networks with application to function genomics and text categorization, IEEE Transactions on Knowledge and Data Engineering 18 (10) (2006) 1338–1351.
[7] F. Brucker, F. Benites, E. Sapozhnikova, Multi-label classification and extracting predicted class hierarchies, Pattern Recognition 44 (3) (2011) 724–738.
[8] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognition 37 (9) (2004) 1757–1771.
[9] M.L. Zhang, Z.H. Zhou, ML-*k*NN: a lazy learning approach to multi-label learning, Pattern Recognition 40 (5) (2007) 2038–2048.
[10] A. Jiang, C. Wang, Y. Zhu, Calibrated Rank-SVM for multi-label image categorization, in: Proceedings of 2008 IEEE International Joint Conference on Neural Networks (IJCNN2008), Hongkong, China, 2008, pp.1450–1455.
[11] J.D. Wang, Y.H. Zhao, X.Q. Wu, X.S. Hua, A transductive multi-label learning approach for video concept detection, Pattern Recognition 44 (10/11) (2011) 2274–2286.
[12] P. Pavlidis, J. Weston, J. Cai, W.N. Grundy, Combining microarray expression data and phylogenetic profiles to learn functional categories using support vector machines, in: Proceedings of the 5th Annual international Conference on Computational Molecular Biology (RECOMB2001), Montreal, Canada, 2001, pp. 242–248.
[13] K.C. Chou, H.B. Shen, Cell-PLoc 2.0: an improved package of web-servers for predicting subcellular location of proteins in various organisms, Natural Science 2 (10) (2010) 1090–1103.
[14] K. Trohidis, G. Tsoumakas, G. Kalliris, I. Vlahavas, Multi-label classification of music into emotions, in: Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR2008), Philadelphia, PA, USA, 2008, pp. 325–330.
[15] A. Elisseeff, J. Weston, 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.
[16] A. Elisseeff, J. Weston, Kernel Methods for Multi-labelled Classification and Categorical Regression Problems, Technical Report, 2001, BIOwulf Technologies 〈http://www.kyb.tuebingen.mpg.de/bs/people/weston/publications〉.
[17] W. Cheng, E. Hullermeier, Combining instance-based learning and logistic regression for multi-label classification, Machine Learning 76 (2/3) (2009) 211–225.
[18] K. Dembczynski, W. Waegeman, W. Cheng, E. Hullermeier, On label dependencies in multi-label classification, in: Workshop Proceedings of Learning from Multi-label Data, Haifa, Israel, 2010, pp. 5–12.
[19] M. Frank, P. Wolfe, An algorithm for quadratic programming, Naval Research Logistic Quarterly 3 (1/2) (1956) 95–110.
[20] J. Guelat, P. Marcotte, Some comments on Wolfe's away step, Mathematical Programming 35 (1) (1986) 110–119.
[21] I.W. Tsang, J.T. Kwok, P.M. Cheung, Core vector machines: fast SVM training on very large data sets, Journal of Machine Learning Research 6 (2005) 363–392.
[22] I.W. Tsang, J.T. Kwok, J.M. Zurada, Generalized core vector machines, IEEE Transactions on Neural Networks 17 (5) (2006) 1126–1140.
[23] H. Ouyang, A. Gray, Fast stochastic Frank–Wolfe algorithms for nonlinear SVMS, in: Proceedings of the 10th SIAM International Conference on Data Mining (SDM2010), Columbus, Ohio, USA, 2010, pp. 245–256.
[24] F.D. Comite, R. Gilleron, M. Tommasi, Learning multi-label alternative decision tree from texts and data, in: Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition (MLMD2003), Lecture Notes in Computer Science, vol. 2734, Leipzig, Germany, 2003, pp. 35–49.
[25] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., John Wiley and Sons, New York, 2001.
[26] J. Ramon, O. Luaces, A. Bahamonde, Multi-label classification with a probabilistic thresholding strategy, Pattern Recognition 45 (2) (2012) 876–883.
[27] M. Petrovskiy, Paired comparisons method for solving multi-label learning problem, in: Proceedings of the 6th International Conference on Hybrid Intelligent Systems (HIS2006), Auckland, New Zealand, 2006, p. 42.
[28] J. Furnkranz, E. Hullermeier, E.L. Mencia, K. Brinker, Multi-label classification via calibrated label ranking, Machine Learning 73 (2) (2008) 133–153.
[29] G. Madjarov, D. Gjorgjevikj, S. Dzeroski, Two Stage architecture for multi-label learning, Pattern Recognition 45 (3) (2012) 1019–1034.
[30] G. Tsoumakas, I. Vlahavas, I. Katakis, Random *k*-labelsets for multi-label classification, IEEE Transactions on Knowledge and Data Mining 23 (7) (2011) 1079–1089.
[31] A. Clare, R.D. King, Knowledge discovery in multi-label phenotype data, in: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2001), Lecture Notes in Computer Science, vol. 2168, Freiburg, Baden-Württemberg, Germany, 2001, pp. 42–53.
[32] V.N. Vapnik, Statistical Learning Theory, John Wiley and Sons, New York, 1998.
[33] M.L. Zhang, J.M. Pena, V. Robles, Feature selection for multi-label naïve Bayes classification, Information Science 179 (19) (2009) 3218–3229.
[34] J. Reed, B. Pfahringer, G. Holmes, Classifier chain for multi-label classification, Machine Learning 85 (3) (2011) 333–359.
[35] J.H. Xu, An extended one-versus-rest support vector machine for multi-label classification, Neurocomputing 74 (17) (2011) 3114–3124.
[36] M.L. Zhang, ML-RBF: RBF neural networks for multi-label learning, Neural Processing Letters 29 (2) (2009) 61–74.

[37] L. Wang, M. Chang, J. Feng, Parallel and sequential support vector machines for multi-label classification, International Journal of Information Technology 11 (9) (2005) 11–18.
[38] S.P. Wan, J.H. Xu, A multi-label classification algorithm based on triple class support vector machine, in: Proceedings of 2007 IEEE International Conference on Wavelet Analysis and Pattern Recognition (ICPRWA2007), Beijing, China, 2007, pp. 1447–1452.
[39] J.Y. Li, J.H. Xu, A fast multi-label classification algorithm based on double label support vector machine, in: Proceedings of 2009 International Conference on Computational Intelligence and Security (CIS2009), vol. 2, Beijing, China, 2009, pp. 30–35.
[40] C.C. Chang, C.J. Lin, Training nu-support vector classifiers: theory and algorithm, Neural Computation 13 (9) (2001) 2119–2147.
[41] R.E. Fan, P.H. Chen, C.J. Lin, Working set selection using second order information for training support vector machines, Journal of Machine Learning Research 6 (2005) 1889–1918.
[42] F.D. Comite, R. Gilleron, M. Tommasi, ADTree, 2003 ⟨http://www.grappa.univlille3.fr/ftp/Softs/ADTree.tgz⟩.
[43] M.L. Zhang, Matlab software of BP-MLL and ML-*k*NN, and Image data set, 2009 ⟨http://cse.seu.edu.cn/people/zhangml⟩.
[44] S. Stiena, LPSOL5.5, 2006 ⟨http://www.cs.sunysb.edu/algorithm/implement/lpsolve/implement.shtml⟩.
[45] G. Tsoumakas, Multi-label data sets, 2009 ⟨http://mulan.sourceforge.net/datasets.html⟩.
[46] J. Read, Slashdot data set, 2011 ⟨http://meka.sourceforge.net/#datasets⟩.
[47] H.B. Shen, Cell_PLoc 2.0 data sets, 2010 ⟨http://www.csbio.sjtu.edu.cn/bioinf/Cell-PLoc-2/Data⟩.
[48] P.F. Du , PseAAC-Builder, 2011 ⟨http://www.sourceforge.net/projects/pseb/files⟩.
[49] J. Demsar, Statistical comparison of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

Jianhua Xu received his Ph.D. in Pattern Recognition and Intelligent Systems in 2002 (Department of Automation, Tsinghua University, Beijing, China). Since 2005, he is a professor in Computer Science, Nanjing Normal University, Nanjing, China. From September 2008 to September 2009, he was a visiting scholar at Department of Statistics, Harvard University, Cambridge MA, USA. His research interests are focused on pattern recognition, machine learning, and their applications to bioinformatics.