Progressive Random k-Labelsets for Cost-Sensitive Multi-Label Classification

Yu-Ping Wu R02922167@CSIE.NTU.EDU.TW

Hsuan-Tien Lin HTLIN@CSIE.NTU.EDU.TW

Department of Computer Science and Information Engineering, National Taiwan University, Taiwan

Abstract

In multi-label classification, an instance is associated with multiple relevant labels, and the goal is to predict these labels simultaneously. Many real-world applications of multi-label classification come with different performance evaluation criteria. It is thus important to design general multi-label classification methods that can flexibly take different criteria into account. Such methods tackle the problem of cost-sensitive multi-label classification (CSMLC). Most existing CSMLC methods either suffer from high computational complexity or focus on only certain specific criteria. In this work, we propose a novel CSMLC method, named progressive random k-labelsets (PRAkEL), to resolve the two issues above. The method is extended from a popular multi-label classification method, random k-labelsets, and hence inherits its efficiency. Furthermore, the proposed method can handle arbitrary example-based evaluation criteria by progressively transforming the CSMLC problem into a series of cost-sensitive multi-class classification problems. Experimental results demonstrate that PRAkEL is competitive with existing methods under the specific criteria they can optimize, and is superior under several popular criteria.

Keywords: machine learning, multi-label classification, loss function, cost-sensitive learning, labelset, ensemble method

1. Introduction

Multi-label classification (MLC) extends traditional multi-class classification by allowing each instance to be associated with a set of relevant labels. For example, in text classification, a document (instance) can belong to several topics (labels). Given a set of instances as well as their relevant labels, the goal of an MLC method is to predict the relevant labels of a new instance. Recently, MLC has attracted much research attention with a wide range of applications including music tag annotation (Trohidis et al., 2008; Lo et al., 2011), image classification (Boutell et al., 2004), and video classification (Qi et al., 2007).

In contrast to multi-class classification, one important characteristic of MLC is the possible correlations between different labels. Many approaches have been proposed to exploit the correlations. Chaining methods learn a label by treating other labels as features (Read et al., 2011; Dembczynski et al., 2010). Labelset-based methods learn several labels jointly (Tsoumakas et al., 2010; Tsoumakas and Vlahavas, 2007; Lo et al., 2014; Lo, 2013). Other methods transform the space of labels to capture the correlations (Hsu et al., 2009; Tai and Lin, 2012; Hardoon et al., 2004).

A key challenge of MLC is to automatically adapt a method to the evaluation criterion of interest. In real-world applications, different criteria are often required to evaluate the performance of an MLC method. For example, Hamming loss measures the proportion of the misclassified labels to the total number of labels; F1 score originates from information retrieval applications and is the harmonic mean of the precision and recall; subset 0/1 loss requires all labels to be correctly predicted. Because of the different natures of those criteria, a method that performs well under one criterion may not be well-suited for other criteria. It is therefore important to design general MLC methods that take the evaluation criterion into account, either in the training or prediction stage. Since the evaluation criterion, or metric, determines the *cost* for misclassifying an instance, this type of problem is generally called cost-sensitive multi-label classification (CSMLC) (Lo et al., 2014; Li and Lin, 2014), which is formally defined in Section 2.

We shall explain in Section 3 that most existing MLC methods either aim for optimizing a certain evaluation metric or require extra efforts to be adapted to each metric. For example, binary relevance (BR) (Tsoumakas et al., 2010) minimizes Hamming loss by learning each label independently. Label powerset (LP) (Tsoumakas et al., 2010) minimizes subset 0/1 loss by transforming the MLC problem to a multi-class classification problem with a huge number of hyper-classes. The well-known random k-labelsets (RAkEL) (Tsoumakas and Vlahavas, 2007) method focuses on many smaller multi-class classification problems to be computationally efficient, but it is only loosely connected to subset 0/1 loss (Ferng and Lin, 2013).

There are currently a few methods for dealing with general CSMLC problems (Dembczynski et al., 2010; Tsochantaridis et al., 2005; Li and Lin, 2014; Doppa et al., 2014). RAkEL has been extended to cost-sensitive random k-labelsets (CS-RAkEL) (Lo, 2013) and generalized k-labelsets ensemble (GLE) (Lo et al., 2014) to handle example-dependent weighted Hamming loss, but not general metrics. Probabilistic classifier chain (Dembczynski et al., 2010) requires designing an efficient inference rule with respect to the metric, and covers many, but not all, of the metrics of interest (Li and Lin, 2014). Condensed filter tree (Li and Lin, 2014) is a chaining method that takes any evaluation metric into account during the training stage, but its training time is quadratic to the number of labels. The structured support vector machine (Tsochantaridis et al., 2005) can also handle arbitrary metric, but it relies on solving a sophisticated optimization problem depending on the metric and is thus also inefficient. To the best of our knowledge, no existing CSMLC methods are both general and efficient.

In this work, we design a general and efficient CSMLC method in Section 4. This novel method, named progressive random k-labelsets (PRAkEL), is extended from RAkEL to inherit its efficiency. In particular, PRAkEL practically enjoys linear training time in terms of the number of labels. Moreover, PRAkEL is able to optimize any example-based metric by modifying the training stage of RAkEL. More specifically, RAkEL reduces the original problem to many regular multi-class problems and ignores the original cost information; PRAkEL reduces the CSMLC problem to many *cost-sensitive* multi-class ones by transferring the cost information to the sub-problems. The transferring task is non-trivial, however, because each sub-problem involves only a subset of labels of the original problem. We therefore introduce the notion of *reference labels* to determine the costs in the sub-problems.

We carefully propose two strategies for defining the reference labels, which lead to different advantages and disadvantages in both theoretical and empirical aspects.

We conducted experiments on seven benchmark datasets with various sizes and domains. The experimental results in Section 5 show that PRAkEL is competitive with state-of-the-art MLC methods under the specific metrics associated with the methods. Furthermore, in terms of general metrics, PRAkEL usually outperforms other methods. The results demonstrate that the proposed method is indeed more general, and more suitable for solving real-world problems.

2. Problem Setup

In CSMLC, we denote an instance by a vector $\mathbf{x} \in \mathcal{X} = \mathbb{R}^d$ and the relevant labels of $\mathbf{x}$ by a set $Y \subseteq \{1, 2, \cdots, K\}$, where K is the total number of labels. Equivalently, this set of labels can be represented by a bit vector $\mathbf{y} \in \mathcal{Y} = \{0, 1\}^K$, where the l-th component $\mathbf{y}[l]$ is 1 if and only if the l-th label is relevant, i.e., $l \in Y$. Here, $\mathcal{X}$ and $\mathcal{Y}$ are called the input space and label space, respectively; the pair $(\mathbf{x}, \mathbf{y})$ is called an example. In this work, we consider a particular CSMLC setup that allows each example to carry its own cost information. The example-based setup, which assumes example-dependent costs, is more general than the setup with label-dependent costs, in which all examples share the same cost functions. The more general setup makes it possible to express the importance of different instances easily through embedding the importance in the example-dependent cost, and has been considered in several studies of cost-sensitive learning (Fan et al., 1999; Zadrozny et al., 2003; Sun et al., 2007). Formally, given a training set $\{(\mathbf{x}_n, \mathbf{y}_n, \mathbf{c}_n)\}_{n=1}^N$ consisting of N examples, where $\mathbf{c}_n \colon \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is a non-negative cost function and each $(\mathbf{x}_n, \mathbf{y}_n, \mathbf{c}_n)$ is drawn independently from an unknown distribution $\mathcal{D}$, the goal of CSMLC is to learn a classifier $h \colon \mathcal{X} \to \mathcal{Y}$ such that the expected cost $\mathrm{E}_{(\mathbf{x},\mathbf{y},\mathbf{c})\sim\mathcal{D}}[\mathbf{c}(h(\mathbf{x}))]$ is small.

Note that the example-based setup cannot cover all popular evaluation criteria in multi-label classification. For instance, the micro-F1 and macro-F1 criteria, which are defined on a set of $\mathbf{y}$ rather than a single one, cannot be expressed as an example-dependent cost function. Nonetheless, as highlighted by earlier CSMLC works (Li and Lin, 2014), studying the example-based setup can be viewed as an intermediate step toward those more complicated criteria.

Two remarks about this setup are in order. First, for a classifier h, since $\mathbf{c}(h(\mathbf{x}))$ is being minimized, it is natural to assume $\mathbf{c}$ has a minimum of 0 at $\mathbf{y}$, the true label vector of $\mathbf{x}$. With this assumption, although $\mathbf{y}$ does not appear in the learning goal, its information is implicitly stored in the cost function. Second, we can similarly define the problem of cost-sensitive multi-class classification (CSMCC) by replacing the label space $\mathcal{Y}$ with $\{1, 2, \cdots, K\}$, which stands for K different classes. In fact, this setup is widely adopted in many existing works (Tu and Lin, 2010; Zhou and Liu, 2010; Abe et al., 2004).

Modern CSMCC works (Zhou and Liu, 2010) allow flexibly taking any cost functions into account based on application needs. While the proposed method shares the same flexibility in its derivation, we consider a more realistic scenario of CSMLC in the experiments. In particular, many CSMLC problems are actually associated with a global, label-dependent cost $L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, typically called a loss function, where $L(\mathbf{y}, \hat{\mathbf{y}})$ is the loss when predicting $\mathbf{y}$ as $\hat{\mathbf{y}}$. Those problems aim to learn a classifier $h \colon \mathcal{X} \to \mathcal{Y}$ such that $\mathrm{E}[L(\mathbf{y}, h(\mathbf{x}))]$ is

small (Dembczynski et al., 2010; Li and Lin, 2014). The aim can be easily expressed in our setup by assigning

$$\mathbf{c}_n(\hat{\mathbf{y}}) = L(\mathbf{y}_n, \hat{\mathbf{y}}). \tag{1}$$

We focus on CSMLC with such loss functions to demonstrate the applicability of the proposed method and to make a fair comparison with existing CSMLC methods (Li and Lin, 2014; Dembczynski et al., 2010). Popular loss functions include

- Hamming loss[1]
$$L_H(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{K}\sum_{l=1}^{K}[\![\hat{\mathbf{y}}[l] \neq \mathbf{y}[l]]\!];$$
- weighted Hamming loss with respect to the weight $\mathbf{w} \in {\mathbb{R}_{\geq 0}}^K$
$$L_{H,\mathbf{w}}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{l=1}^{K}\mathbf{w}[l]\cdot[\![\hat{\mathbf{y}}[l] \neq \mathbf{y}[l]]\!];$$
- ranking loss
$$L_r(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{R(\mathbf{y})}\sum_{(k,l):\, \mathbf{y}[k]<\mathbf{y}[l]}[\![\hat{\mathbf{y}}[k] > \hat{\mathbf{y}}[l]]\!] + \frac{1}{2}[\![\hat{\mathbf{y}}[k] = \hat{\mathbf{y}}[l]]\!],$$
where $R(\mathbf{y}) = |\{(k,l) \mid \mathbf{y}[k] < \mathbf{y}[l]\}|$ is a normalizer;
- F1 loss[2]
$$L_F(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \frac{2\mathbf{y}\cdot\hat{\mathbf{y}}}{\|\mathbf{y}\|_1 + \|\hat{\mathbf{y}}\|_1},$$
which is one minus the F1 score;
- subset 0/1 loss
$$L_s(\mathbf{y}, \hat{\mathbf{y}}) = [\![\hat{\mathbf{y}} \neq \mathbf{y}]\!].$$

For those loss functions defined above, we follow the convention that when the denominator is zero, the loss is defined as zero.

To simplify the explanations of the proposed method, we further introduce some terminology. We denote the set of K labels by $\mathcal{L}_K = \{1, \cdots, K\}$. A subset S of $\mathcal{L}_K$ with $|S| = k$ is called a k-labelset. If $S = \{s_1, \cdots, s_k\}$ is a k-labelset with $s_1 < \cdots < s_k$, then we denote $(\mathbf{y}[s_1], \cdots, \mathbf{y}[s_k]) \in \{0,1\}^k$ by $\mathbf{y}[S]$. When the number of labels, K, is clear in the context, we also use the notation S^c to represent the $(K-k)$-labelset $\mathcal{L}_K \setminus S = \{1 \leq l \leq K \mid l \notin S\}$. We summarize the main notation used throughout the paper in Table 1.

1. $[\![\cdot]\!]$ is the indicator function.
2. $\|\cdot\|_1$ is the ℓ_1 norm.

Notation	Description
N	Number of training examples
d	Dimension of input space (number of features)
K	Number of labels
$\mathcal{X} = \mathbb{R}^d$	Input space (feature space)
$\mathcal{Y} = \{0, 1\}^K$	Output space (label space)
$\mathbf{x} \in \mathcal{X}$	Instance (feature vector)
$\mathbf{y} \in \mathcal{Y}$	True label vector
$\hat{\mathbf{y}} \in \mathcal{Y}$	Predicted label vector
$\check{\mathbf{y}} \in \mathcal{Y}$	Reference label vector (see Section 4.2)
$h \colon \mathcal{X} \to \mathcal{Y}$	Multi-label classifier
$\mathbf{c} \colon \mathcal{Y} \to \mathbb{R}_{\geq 0}$	Example-dependent cost function
$L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$	Label-dependent loss function
$\mathcal{L}_K = \{1, \cdots, K\}$	The set of K labels
$S \subseteq \mathcal{L}_K$ with $\lvert S \rvert = k$	k-labelset
$\mathbf{y}[S]$	The ordered set of labels of $\mathbf{y}$ within S
M	Number of iterations (labelsets) for the proposed method

Table 1: Main notation used in the paper.

3. Related Work

Multi-label classification methods can be divided into two main categories, namely, algorithm adaptation and problem transformation (Tsoumakas and Katakis, 2007). Algorithm adaptation methods directly extend a specific learning algorithm to tackle MLC problems. Multi-label k-nearest neighbor (ML-kNN) (Zhang and Zhou, 2007) is adapted from the famous k-nearest neighbors algorithm. AdaBoost.MH and AdaBoost.MR (Schapire and Singer, 2000) are an two multi-label extensions of the AdaBoost algorithm (Freund and Schapire, 1999). ML-C4.5 (Clare and King, 2001) is an adaptation of the popular C4.5 algorithm. BP-MLL (Zhang and Zhou, 2006) is derived from the back-propagation algorithm of neural networks.

Problem transformation methods transform MLC problems into other types of learning problems and solve them by existing algorithms. Such methods are general and can be coupled with any mature algorithms. Our proposed method in Section 4 belongs to this category.

Binary relevance (BR) (Tsoumakas et al., 2010) is arguably the simplest problem transformation method, which transforms the MLC problem into several binary classification problems by learning and predicting each label independently. Classifier chain (CC) (Read et al., 2011) iteratively learns a binary classifier to predict the l-th label using $\{(\mathbf{x}_n, \hat{\mathbf{y}}_n[1], \cdots, \hat{\mathbf{y}}_n[l-1])\}$ as the training set, where $\hat{\mathbf{y}}_n$ contains the previously predicted labels. Although it considers the label dependencies, the order of labels becomes crucial to the performance of CC. Many approaches have been proposed to address this issue (Read et al., 2011, 2014; Goncalves et al., 2013). In particular, the ensemble of classifier chains (ECC) (Read et al., 2011) learns several CC classifiers, each with a random ordering of labels, and it averages the predictions from all the classifiers to classify a new instance.

Instead of learning one binary classifier for each label, probabilistic classifier chain (PCC) (Dembczynski et al., 2010) learns probabilistic classifiers to estimate $P(\mathbf{y} \mid \mathbf{x})$ by the chain rule

$$P(\mathbf{y} \mid \mathbf{x}) = P(\mathbf{y}[1] \mid \mathbf{x}) \cdot \prod_{l=2}^{K} P(\mathbf{y}[l] \mid \mathbf{x}, \mathbf{y}[1], \cdots, \mathbf{y}[l-1])$$

and then applies Bayes optimal inference rule designed for the evaluation metric to produce the final prediction. In principle, PCC is able to be adapted to any different metric to tackle CSMLC problems by designing proper inference rules for the metric. However, deriving efficient inference rules for different metrics is practically challenging. Inference rules for Hamming, ranking, F1 and subset 0/1 loss have been designed (Dembczynski et al., 2010, 2011), but the rules for other metrics remain an open question. Similar to ECC, the ensembled probabilistic classifier chain (EPCC) (Dembczynski et al., 2010) resolves the issue of label ordering by random orderings.

The Monte Carlo optimization for classifier chains (MCC) (Read et al., 2014) employs the Monte Carlo scheme to find a good label ordering in the training stage of PCC. A recently proposed method, the classifier trellis (CT) (Read et al., 2015), is extended from MCC to consider a trellis structure of labels rather than a chain to improve efficiency. During the prediction stage of both methods (Read et al., 2014, 2015), the Monte Carlo scheme is applied to generate samples from $P(\mathbf{y} \mid \mathbf{x})$. A large number of samples may be required for Monte Carlo simulation, which results in possible computational challenges during prediction. While those samples can in principle be used to make cost-sensitive predictions, the possibility has not been fully studied in both works. In fact, the original works consider only approximate inference for Hamming loss and subset 0/1 loss.

A group of methods take label dependencies into account by learning multiple labels jointly. Label powerset (LP) (Tsoumakas et al., 2010) transforms each label combination into a unique hyper-class and learns a multi-class classifier. If there are K labels in total, then the number of classes may be as large as 2^K. Hence, when the number of labels is large, LP suffers from computational issues and an insufficient number of training examples within each class.

To overcome the drawback, a method called random k-labelsets (RAkEL) (Tsoumakas and Vlahavas, 2007) focuses on one labelset at a time. Recall that a k-labelset is a size-k subset of $\{1, 2, \cdots, K\}$. RAkEL iteratively selects a random k-labelset S_m and learns an LP classifier h_m for the training set restricted to the labels within S_m, i.e., $\{(\mathbf{x}_n, \mathbf{y}_n[S_m])\}$. Each classifier h_m predicts the k labels within S_m, and the final prediction of an instance is produced by a majority vote of all the classifiers. Because the number of classes in each LP classifier is decreased, RAkEL is more efficient than LP. In addition, it achieves better performance than LP in terms of Hamming and F1 loss.

Nonetheless, there is a noticeable issue of RAkEL. In each multi-class sub-problem, a one-bit prediction error and a two-bit error are equally penalized. That is, the LP classifiers cannot distinguish between small and big errors. Because these classifiers are learned without considering the evaluation metric, RAkEL is not a cost-sensitive method.

Two extensions of RAkEL were proposed to address the above issue, but they both consider only the example-dependent weighted Hamming loss rather than general metrics. The cost-sensitive random k-labelsets (CS-RAkEL) (Lo, 2013) method reduces the CSMLC

problem to several multi-class ones with instance weights. The weight of each instance is defined as the sum of the misclassified costs of the relevant labels. Despite the restriction, one advantage of CS-RAkEL is that it only requires re-weighting of the instances and can hence be coupled with many traditional multi-class classification algorithms.

Generalized k-labelsets ensemble (GLE) (Lo et al., 2014) learns a set of LP classifiers and determines a linear combination of them by minimizing the averaged loss of training examples. The minimization is formulated as a quadratic optimization problem without any constraints and hence can be solved efficiently. While both CS-RAkEL and GLE are pioneering works on extending RAkEL for CSMLC, they focus on specific applications of tagging. As a consequence, the two methods do not come with much theoretical guarantee, and it is non-trivial to extend them to handle example-dependent costs.

For the methods introduced above, BR and CC optimize Hamming loss; CS-RAkEL and GLE deal with weighted Hamming loss; MCC and CT minimize Hamming and subset 0/1 loss currently, with the potential of handling general metrics yet to be studied; PCC is designed to deal with general metrics, but is computationally demanding for arbitrary metrics that come without efficient inference rules. Another method that deals with general metrics is the structured support vector machine (SSVM) (Tsochantaridis et al., 2005). The SSVM optimizes a metric by re-scaling certain variables in the traditional SVM optimization problem based on the metric. However, the complexity of solving the problem depends on the metric and is usually too high for practical applications.

Condensed filter tree (CFT) (Li and Lin, 2014) is a state-of-the-art CSMLC method, extended from the well-known filter tree algorithm (Beygelzimer et al., 2009) to handle multi-label data. Similarly, the divide-and-conquer tree algorithm (Beygelzimer et al., 2009) for multi-class problems can be directly adapted to CSMLC problems to design the top-down tree (TT) method (Li and Lin, 2014). Both CFT and TT can be viewed as cost-sensitive extensions of CC. CFT suffers from its training time, which is quadratic to the number of labels; TT suffers from its weaker performance as compared with CFT (Li and Lin, 2014).

Multi-label search (MLS) (Doppa et al., 2014) optimizes a metric by adapting the $\mathcal{HC}$-search framework to multi-label problems. It learns a heuristic function and estimates the evaluation metric in the training stage. Then, during the prediction stage, MLS conducts a heuristic search towards minimizing the estimated cost. Despite its generality, MLS suffers from high computational complexity. To learn the heuristic function during training, it needs to solve a ranking problem consisting of $O(NK)$ examples, where N is the number of training examples and K is the number of labels.

In summary, many existing MLC methods are not applicable to arbitrary example-based metrics of CSMLC (BR, CC, LP, RAkEL). There are some extensions dealing with restricted metrics of CSMLC (CS-RAkEL, GLE). For general metrics, current methods suffer from computational issues (CFT, MLS, SSVM), performance issues (TT), or require elegant design of inference rules or more studies to handle different metrics (PCC, MCC, CT). In the next section, we present a general yet efficient cost-sensitive multi-label method, which is competitive with state-of-the-art CSMLC methods.

4. Proposed Method

Recall that the LP method solves an MLC problem by transforming it into a single multi-class problem. Similarly, a CSMLC problem can be transformed into a cost-sensitive multi-class classification (CSMCC) problem, as illustrated in the CFT work (Li and Lin, 2014). The resulting method, however, suffers from the same computational issue as LP, and hence is not feasible for large problems. CFT solves the computational issue by considering an efficient multi-class classification model—the filter tree.

In this work, we deal with the computational issue differently. We extend the idea of RAkEL and propose a novel labelset-based method, which iteratively transforms the CSMLC problem into a series of CSMCC problems. Different from RAkEL, the critical part of the proposed method is the transfer of the cost information to the sub-problems in the training stage. This is not a trivial task, since each sub-problem involves only a subset of labels and hence the costs in each sub-problem cannot be easily connected to those in the original problem. Therefore, we introduce the notion of reference label vectors to determine the costs in the sub-problems. While the overall idea sounds simple, it advances the study of CSMLC in several aspects:

- Compared with traditional MLC methods such as RAkEL, the proposed method is sensitive to the evaluation metric and hence is able to optimize arbitrary example-based metrics.
- Compared with CS-RAkEL and GLE, the proposed method handles more general metrics and comes with solid theoretical analysis.
- Compared with PCC, MCC and SSVMs, our method alternatively considers label dependencies through labelsets and requires no manual adaptation to each evaluation metric.
- Compared with existing CSMLC methods such as CFT, our method is more efficient in terms of training time complexity while reaching similar level of performance.

We first provide the framework of the proposed method. Then, we describe it in great detail and present its analysis.

4.1. Framework

Let $\mathcal{T} = \{(\mathbf{x}_n, \mathbf{y}_n, \mathbf{c}_n)\}_{n=1}^N$ be the training set and M be the number of iterations. Inspired by RAkEL, in the m-th iteration, our method selects a random k-labelset S_m and constructs a CSMCC training set $\mathcal{T}'_m = \{(\mathbf{x}_n, \mathbf{y}_n[S_m], \mathbf{c}'_n)\}_{n=1}^N$ of $K' = 2^k$ classes, where $\mathbf{c}'_n \colon \{0,1\}^k \to \mathbb{R}$. The main difference between our method and RAkEL is that, the multi-class sub-problems defined here contain the costs $\mathbf{c}'_n$, and hence our method is able to carry the information of the evaluation metric. The two issues of RAkEL discussed in Section 3 can also be resolved by properly defining these $\mathbf{c}'_n$. Although in our problem setup described in Section 2, the label space of a CSMCC problem should be $\mathcal{L}_{K'}$, by considering a bijection between $\mathcal{L}_{K'}$ and $\{0,1\}^k$, we may treat $\mathbf{y}_n[S_m]$ as an element of $\mathcal{L}_{K'}$ and assume $\mathbf{c}'_n \colon \mathcal{L}_{K'} \to \mathbb{R}$. Then, any CSMCC algorithm can be employed to learn a multi-class classifier $h'_m \colon \mathcal{X} \to \{0,1\}^k$ for $\mathcal{T}'_m$.

Similar to RAkEL, the final prediction of a new instance $\mathbf{x}$ is produced by a majority vote of all the classifiers h'_m. More precisely, if we define $h_m \colon \mathcal{X} \to \{-1, 0, 1\}^K$ by

$$\begin{cases} h_m(\mathbf{x})[S_m] = 2 \cdot h'_m(\mathbf{x}) - 1 \in \{-1, 1\}^k \\ h_m(\mathbf{x})[S_m^c] = \mathbf{0} \in \{0\}^{K-k}, \end{cases} \tag{2}$$

then the final prediction $\hat{\mathbf{y}} \in \mathcal{Y}$ can be obtained by setting $\hat{\mathbf{y}}[l] = 1$ if and only if $\sum_{m=1}^{M} h_m[l] > 0$.

4.2. Cost Transformation

Having described the framework, we now turn our attention to the multi-class cost functions $\mathbf{c}'_n$ in the sub-problems, which must be defined in each iteration. At this point, notice that if we define $\mathbf{c}'_n(\hat{\mathbf{y}}') = [\![\hat{\mathbf{y}}' \neq \mathbf{y}_n[S_m]]\!]$, then the proposed method degenerates into RAkEL. Since this $\mathbf{c}'_n$ is independent of the original cost function $\mathbf{c}_n$, it can also be seen from this assignment that RAkEL is not a cost-sensitive method.

To establish the connections between these two cost functions, $\mathbf{c}'_n$ must carry a certain amount of information of $\mathbf{c}_n$. Note that the domain of $\mathbf{c}'_n$ is $\{0, 1\}^k$ and $\mathbf{c}_n$ is defined on $\mathcal{Y} = \{0, 1\}^K$. To extend $\mathbf{c}'_n$ to the domain of $\mathbf{c}_n$, we propose considering a *reference* label vector $\tilde{\mathbf{y}}_n \in \mathcal{Y}$ and setting the value of $\mathbf{c}'_n$ to be the cost $\mathbf{c}_n$ assuming the labels outside S_m were predicted the same as $\tilde{\mathbf{y}}_n$. Mathematically,

$$\mathbf{c}'_n(\hat{\mathbf{y}}') = \mathbf{c}_n(\hat{\mathbf{y}}' \cup \tilde{\mathbf{y}}_n[S_m^c]). \tag{3}$$

Here, we treat $\hat{\mathbf{y}}'$ and $\tilde{\mathbf{y}}_n[S_m^c]$ as subsets of S_m and S_m^c, respectively, and therefore, their union is considered as a subset of $\mathcal{L}_K$, or equivalently a bit vector in $\{0, 1\}^K$.

It then remains to define these $\tilde{\mathbf{y}}_n$ in *each* iteration to complete the transformation. We shall see in the next section that these reference vectors may depend on the classifiers learned in the previous iterations, and hence, the multi-class cost functions would be obtained progressively. As a consequence, the proposed method is called progressive random k-labelsets (PRAkEL). The training and prediction algorithms of PRAkEL are presented in Algorithms 1 and 2, where the weighting strategy mentioned in line 8 of Algorithm 1 is described in Section 4.4. For now we simply assume $\alpha_m = 1$ for $1 \leq m \leq M$. Another thing to note is that, we do not explicitly require selecting a labelset that has not been chosen before. However, in practice we give higher priority to those labels that were selected fewer times in the previous iterations. In particular, we guarantee that all labels are selected at least once if $kM \geq K$.

4.3. Defining Reference Label Vectors

We propose two strategies for defining the reference label vectors. The first, and also the most intuitive, is to let $\tilde{\mathbf{y}}_n = \mathbf{y}_n$ in every iteration. The proposed method with this assignment is denoted by PRAkEL$_\mathrm{t}$ to indicate the usage of the true label vectors. In this strategy, we implicitly assume that the labels outside the labelset can be perfectly predicted by the other classifiers.

In real-world situations, however, this is usually not the case. Therefore, in the second strategy, we define $\tilde{\mathbf{y}}_n$ to be the predicted label vector of $\mathbf{x}_n$ obtained thus far. Thus, the

Algorithm 1: Training algorithm of PRAkEL

Input: Training set $\mathcal{T} = \{(\mathbf{x}_n, \mathbf{y}_n, \mathbf{c}_n)\}_{n=1}^N$, cost-sensitive multi-class training algorithm $\mathcal{A}$
Parameter: Size of labelset k, number of iterations M
Output: Labelsets S_m, weights α_m and multi-class classifiers $h'_m \colon \mathcal{X} \to \{0,1\}^k$ for $1 \leq m \leq M$

for $m \leftarrow 1$ **to** M **do**
  Select a random k-labelset S_m;
  Define the reference label vector $\tilde{\mathbf{y}}_n$ for $1 \leq n \leq N$;
  Transform each example $(\mathbf{x}_n, \mathbf{y}_n, \mathbf{c}_n)$ to $(\mathbf{x}_n, \mathbf{y}_n[S_m], \mathbf{c}'_n)$ by Equation 3;
  $\mathcal{T}' \leftarrow \{(\mathbf{x}_n, \mathbf{y}_n[S_m], \mathbf{c}'_n)\}_{n=1}^N$ based on (3);
  $h'_m \leftarrow \mathcal{A}(\mathcal{T}')$;
  Define h_m by Equation 2;
  Assign weight α_m according to the weighting strategy;
  $F_n \leftarrow F_n + \alpha_m h_m(\mathbf{x}_n)$ for $1 \leq n \leq N$;
end

Algorithm 2: Prediction algorithm of PRAkEL

Input: New instance $\mathbf{x}$, labelsets S_m, weights α_m and multi-class classifiers $h'_m \colon \mathcal{X} \to \{0,1\}^k$ for $1 \leq m \leq M$
Output: Predicted label vector $\hat{\mathbf{y}} \in \mathcal{Y}$

Define h_m for $1 \leq m \leq M$ by (2);
$F \leftarrow \sum_{m=1}^M \alpha_m h_m(\mathbf{x})$;
$\hat{\mathbf{y}} = \mathbf{0} \in \{0\}^K$;
for $l \leftarrow 1$ **to** K **do**
  if $F[l] > 0$ **then**
    $\hat{\mathbf{y}}[l] \leftarrow 1$;
  end
end

optimization in each sub-problem no longer depends on the perfect predictions from the previous classifiers. Formally, let $F_{m,n} = \sum_{p=1}^m h_p(\mathbf{x}_n)$ for $1 \leq n \leq N$ and define $H_{m,n} \in \mathcal{Y}$ by $H_{m,n}[l] = [\![F_{m,n}[l] > 0]\!]$. That is, $H_{m,n}$ is the prediction of $\mathbf{x}_n$ by a majority vote of the first m classifiers. We then define $\tilde{\mathbf{y}}_n$ in the m-th iteration to be $H_{m-1,n}$ for $m \geq 2$, and let $\tilde{\mathbf{y}}_n = \mathbf{y}_n$ in the first iteration. Since the reference label vectors as well as the multi-class sub-problems are obtained progressively, the proposed method coupled with this strategy is denoted simply by PRAkEL.

Recall that in our problem setup we assume the minimum of each $\mathbf{c}_n$ is 0. Therefore, for PRAkEL$_\mathrm{t}$ we have $\min_{\hat{\mathbf{y}}' \in \{0,1\}^k} \mathbf{c}'_n(\hat{\mathbf{y}}') = \min_{\hat{\mathbf{y}} \in \mathcal{Y}} \mathbf{c}_n(\hat{\mathbf{y}}[S_m] \cup \mathbf{y}_n[S_m^c]) = \mathbf{c}_n(\mathbf{y}_n) = 0$. In other words, the minimum cost for every example in each sub-problem is 0, which is a consequence of $\tilde{\mathbf{y}}_n = \mathbf{y}_n$. For PRA$k$EL, however, this identity may not hold. Since the predicted labels outside S_m cannot be altered in the m-th iteration, it is natural to add a constant to each of the functions $\mathbf{c}'_n$ such that $\min_{\hat{\mathbf{y}}' \in \{0,1\}^k} \mathbf{c}'_n(\hat{\mathbf{y}}') = 0$. Therefore, the transformed cost functions

for PRAkEL are all shifted to satisfy this equality by the following formula.

$$\mathbf{c}'_n(\hat{\mathbf{y}}') = \mathbf{c}_n(\hat{\mathbf{y}}' \cup \tilde{\mathbf{y}}_n[S_m^c]) - \mathbf{c}_n(\hat{\mathbf{y}}' \cup \mathbf{y}_n[S_m^c]) \tag{4}$$

Interestingly, after shifting the costs, PRAkEL$_\mathrm{t}$ and PRAkEL become equivalent under Hamming loss and ranking loss. To show this, we first present two lemmas.

Lemma 1 *Let L_r be the function of ranking loss and $\mathbf{y} \in \mathcal{Y} = \{0,1\}^K$. Then, there exists a unique $\mathbf{w} \in {\mathbb{R}_{\geq 0}}^K$ such that $L_r(\mathbf{y}, \cdot) = L_{H,\mathbf{w}}(\mathbf{y}, \cdot)$, where $L_{H,\mathbf{w}}$ is the function of weighted Hamming loss with respect to $\mathbf{w}$.*

Proof See Appendix A. ■

Lemma 2 *Let $L_{H,\mathbf{w}}$ be the function of weighted Hamming loss and S be a k-labelset. For any subsets $\mathbf{y}'_0$ and $\mathbf{y}'_1$ of S, $L_{H,\mathbf{w}}(\mathbf{y}, \mathbf{y}'_0 \cup \tilde{\mathbf{y}}[S^c]) - L_{H,\mathbf{w}}(\mathbf{y}, \mathbf{y}'_1 \cup \tilde{\mathbf{y}}[S^c])$ is independent of $\tilde{\mathbf{y}} \in \{0,1\}^K$.*

Proof See Appendix A. ■

Theorem 3 *Under Hamming loss and ranking loss, PRAkEL$_t$ and PRAkEL are equivalent.*

Proof Let L be the loss function of interest and consider the m-th iteration. For any instance $\mathbf{x}$, let $\mathbf{b}'$ and $\mathbf{c}'$ be the cost functions of $\mathbf{x}$ in the m-th multi-class sub-problem, in the training of PRAkEL$_\mathrm{t}$ and PRAkEL, respectively. We show that $\mathbf{b}'(\mathbf{y}') = \mathbf{c}'(\mathbf{y}') - \min \mathbf{c}'$. Let $\tilde{\mathbf{y}}$ be the reference label vector of $\mathbf{x}$ for PRAkEL. Since we are considering a single instance, by Lemma 1, we may assume L is the function of weighted Hamming loss. Let S be the k-labelset in the current iteration and $\mathbf{y}$ be the true label vector of $\mathbf{x}$.

If $\mathbf{y}' \subseteq S$, then by definition,

$$\begin{aligned}
\mathbf{c}'(\mathbf{y}') - \min \mathbf{c}' &= \mathbf{c}(\mathbf{y}' \cup \tilde{\mathbf{y}}[S^c]) - \min_{\hat{\mathbf{y}}\colon \hat{\mathbf{y}}[S^c] = \tilde{\mathbf{y}}[S^c]} \mathbf{c}(\hat{\mathbf{y}}) \\
&= L(\mathbf{y}, \mathbf{y}' \cup \tilde{\mathbf{y}}[S^c]) - \min_{\hat{\mathbf{y}}\colon \hat{\mathbf{y}}[S^c] = \tilde{\mathbf{y}}[S^c]} L(\mathbf{y}, \hat{\mathbf{y}}) \\
&= L(\mathbf{y}, \mathbf{y}' \cup \tilde{\mathbf{y}}[S^c]) - \min_{\hat{\mathbf{y}}' \subseteq S} L(\mathbf{y}, \hat{\mathbf{y}}' \cup \tilde{\mathbf{y}}[S^c]) \\
&= \max_{\hat{\mathbf{y}}' \subseteq S} (L(\mathbf{y}, \mathbf{y}' \cup \tilde{\mathbf{y}}[S^c]) - L(\mathbf{y}, \hat{\mathbf{y}}' \cup \tilde{\mathbf{y}}[S^c])).
\end{aligned}$$

In addition, by Lemma 2, $L(\mathbf{y}, \mathbf{y}' \cup \tilde{\mathbf{y}}[S^c]) - L(\mathbf{y}, \hat{\mathbf{y}}' \cup \tilde{\mathbf{y}}[S^c])$ is independent of $\tilde{\mathbf{y}}[S^c]$ for all $\hat{\mathbf{y}}' \subseteq S$. Therefore, we have

$$\begin{aligned}
\mathbf{c}'(\mathbf{y}') - \min \mathbf{c}' &= \max_{\hat{\mathbf{y}}' \subseteq S} (L(\mathbf{y}, \mathbf{y}' \cup \mathbf{y}[S^c]) - L(\mathbf{y}, \hat{\mathbf{y}}' \cup \mathbf{y}[S^c])) \\
&= L(\mathbf{y}, \mathbf{y}' \cup \mathbf{y}[S^c]) - L(\mathbf{y}, \mathbf{y}[S] \cup \mathbf{y}[S^c]) \\
&= L(\mathbf{y}, \mathbf{y}' \cup \mathbf{y}[S^c]) \\
&= \mathbf{b}(\mathbf{y}' \cup \mathbf{y}[S^c]) \\
&= \mathbf{b}'(\mathbf{y}').
\end{aligned}$$

■

Moreover, for these two loss functions, it is easy to derive an upper bound of the training cost. Consider a training example $(\mathbf{x}, \mathbf{y}, \mathbf{c})$. Let e_m be the training cost of $\mathbf{x}$ in the m-th CSMCC sub-problem. We hope to bound the overall multi-label training cost of $\mathbf{x}$ in terms of these e_m.

By Lemma 1, again, it suffices to consider weighted Hamming loss. Recall that K is the number of labels, k is the size of the labelsets, and M is the number of iterations. For simplicity, assume kM is a multiple of K. In addition, we assume that each label appears in exactly $r = kM/K$ labelsets. That is, the labelsets are selected uniformly. Let $h_m \in \{-1, 0, 1\}^K$ be the prediction of $\mathbf{x}$ in the m-th iteration as defined in Section 4.1 and $\hat{\mathbf{y}} \in \mathcal{Y}$ be the final prediction, which is obtained by averaging these h_m. Now, focus on the l-th label. If $\hat{\mathbf{y}}[l] \neq \mathbf{y}[l]$, then there must be at least half of those m with $l \in S_m$ such that $h_m[l]$ is predicted incorrectly. Hence, the part of the overall training cost contributed by the l-th label cannot exceed $e_m/(r/2) = 2e_m/r$. As a result, by the property of weighted Hamming loss, the training cost is no more than $\sum_{m=1}^{M} 2e_m/r = (2K/k)\bar{e}$, where $\bar{e} = \sum_{m=1}^{M} e_m/M$. By the above arguments, we have the following theorem.

Theorem 4 *Let E_m be the multi-class training cost of the training set in the m-th iteration. Then, under Hamming loss and ranking loss, the overall training cost of the CSMLC problem for both PRAkEL$_t$ and PRAkEL is no more than $(2K/k)\bar{E}$, where $\bar{E}$ is the mean of E_m.*

Proof Since the statement is true for each example, the proof is straightforward. ■

Despite the equivalence between PRAkEL$_\mathrm{t}$ and PRAkEL for Hamming and ranking loss, they are not the same for arbitrary cost functions. In the experiment section, we demonstrate that PRAkEL is more effective under F1 loss. For now, we present an explanation, by restricting ourselves to the case where the labelsets are disjoint. In this case, $K/k = M$, and the upper bound in Theorem 4 can be improved to $(K/k)\bar{E} = M\bar{E}$ because the final prediction of each label is determined by a single LP classifier. Under this restriction, we have a similar result for PRAkEL. Before stating the next theorem, we have to make some *normality* assumption about the cost functions. For a label vector $\mathbf{y}$ and its corresponding cost function $\mathbf{c}$, we assume that if $\hat{\mathbf{y}}' \in \mathcal{Y}$ is one bit closer to $\mathbf{y}$ than $\hat{\mathbf{y}}'' \in \mathcal{Y}$, then $\mathbf{c}(\hat{\mathbf{y}}') \leq \mathbf{c}(\hat{\mathbf{y}}'')$. That is, a more correct prediction does not result in a larger cost. In fact, this simple assumption has been implicitly made by many MLC methods such as BR, CC and RAkEL.

Theorem 5 *Assume the labelsets are disjoint. Then, for any cost function satisfying the above assumption, the overall training cost for PRAkEL is no more than $M\bar{E}$.*

Proof We may assume there is only one training example $(\mathbf{x}, \mathbf{y}, \mathbf{c})$, where the subscript n is dropped here for simplicity. Recall that the reference label vector of $\mathbf{x}$ in the m-th iteration,

denoted by $\tilde{\mathbf{y}}^{(m)}$, is defined to be H_{m-1} for $m \geq 2$. Then, for $m \geq 2$,

$$\begin{aligned}\mathbf{c}(H_m) &= \mathbf{c}(H_m[S_m] \cup H_m[S_m^c]) \\ &= \mathbf{c}(h'_m(\mathbf{x}) \cup H_{m-1}[S_m^c]) \\ &= E_m + \mathbf{c}(\mathbf{y}[S_m] \cup H_{m-1}[S_m^c]) \\ &\leq E_m + \mathbf{c}(H_{m-1}),\end{aligned}$$

where the third equality is by definition of E_m, and the inequality follows from the assumption we just made. Hence, by induction, the overall training cost is $\mathbf{c}(H_M) \leq \mathbf{c}(\tilde{\mathbf{y}}^{(1)}) + \sum_{m=1}^{M} E_m = \mathbf{c}(\mathbf{y}) + M\bar{E} = M\bar{E}$. ■

Note that this bound cannot be improved since all inequalities in the proof become equalities under Hamming loss. Nonetheless, there is no analogous result for PRAkEL$_{\mathrm{t}}$, as shown in the following theorem.

Theorem 6 *Assume $k < K$. For PRAkEL$_t$, there is no constant $B > 0$ such that the bound $B\bar{E}$ of the overall training cost holds for any cost functions.*

Proof Again, assume the labelsets are disjoint and there is only one instance $\mathbf{x}$. Consider the special case where the true label vector of $\mathbf{x}$ is $\mathbf{y} = (1, \cdots, 1) \in \mathcal{Y}$, and assume $h_m[l] = -1$ for all $l \in S_m$ and all m. In this case, $\hat{\mathbf{y}} = (0, \cdots, 0) \in \mathcal{Y}$, and therefore, its F1 loss is $L_F(\mathbf{y}, \hat{\mathbf{y}}) = 1$. In addition, if we define $\hat{\mathbf{y}}_m = \hat{\mathbf{y}}[S_m] \cup \mathbf{y}[S_m^c]$, then

$$E_m = L_F(\mathbf{y}, \hat{\mathbf{y}}_m) \tag{5}$$

$$= \frac{\sum_l [\![\mathbf{y}[l] \neq \hat{\mathbf{y}}_m[l]]\!]}{\sum_l [\![\mathbf{y}[l] \neq \hat{\mathbf{y}}_m[l]]\!] + 2\sum_l [\![\mathbf{y}[l] = \hat{\mathbf{y}}_m[l] = 1]\!]} \tag{6}$$

$$= \frac{k}{k + 2(K-k)}. \tag{7}$$

Hence, we have $L_F(\mathbf{y}, \hat{\mathbf{y}}) = 1 = ((2K-k)/k)\bar{E}$. Note that if the factor 2 in the equation (7) is replaced by a larger constant, then the bound needs to be larger. Moreover, we can freely define a loss function L similar to L_F by replacing the constant 2 in (6) with an arbitrary positive one. Letting the constant tend to infinity, the proof is complete. ■

Theorems 5 and 6 suggest we define the reference label vectors to be the predicted instead of the true ones. Empirical results in the experiment section also support this finding. In fact, a previous study on multi-target regression has already revealed the problem of treating true targets as additional input variables (Spyromitros-Xioufis et al., 2016). Besides, the authors showed that in-sample estimates of target variables are still problematic, and proposed an approach of out-of-sample estimates to tackle the issue. Although we do not consider this kind of estimates in this paper, we believe that a similar approach for PRAkEL could be considered in future work.

One disadvantage of employing the predicted labels is that the sub-problems need to be learned iteratively, while the training process of the LP classifiers of RAkEL can be parallelized. In addition, the two cost-sensitive extensions of RAkEL, CS-RAkEL and GLE, as well as PRAkEL$_{\mathrm{t}}$, apparently do not have this drawback. There is thus a tradeoff between performance and efficiency.

4.4. Weighting of Base Classifiers

In general, some sub-problems of PRAkEL are easier to solve, while others are more difficult. Thus, the performance of each LP classifier within PRAkEL can be different, and the majority vote of these classifiers may be sub-optimal. Inspired by GLE (Lo et al., 2014), we can further assign different weights to the LP classifiers to represent the importance of them. To achieve this, a linear combination of the classifiers is learned by minimizing the training cost.

Formally, given a new instance $\mathbf{x}$, its prediction $\hat{\mathbf{y}} \in \mathcal{Y}$ is produced by setting $\hat{\mathbf{y}}[l] = 1$ if and only if $\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})[l] > 0$, where these $\alpha_m > 0$ are called the *weights* of the base classifiers. Accordingly, the assignment $F_{m,n} = \sum_{p=1}^{m} h_p(\mathbf{x}_n)$ in the previous section should be changed to $F_{m,n} = \sum_{p=1}^{m} \alpha_p h_p(\mathbf{x}_n)$.

One approach for determining these weights is to solve an optimization problem after all the h_m are learned, just as GLE does. However, this overall optimization ignores the iterative nature of PRAkEL, where the value of $F_{m,n}$ depends on α_p for $1 \leq p < m$ in the m-th iteration. We therefore iteratively determine α_m by greedily minimizing the training cost. More precisely, let $\alpha_1 = 1$ for simplicity, and for $m \geq 2$, by regarding $H_{m,n}$ as a function of α_m, we solve the following single-variable optimization problem and define α_m to be an optimal solution.

$$\min_{\alpha \in \mathbb{R}} \frac{1}{N} \sum_{n=1}^{N} \mathbf{c}_n(H_{m,n}(\alpha)) \tag{8}$$

It is not easy to solve this type of problems in general. Nevertheless, since the objective function is piece-wise constant, the optimization problem (8) can be solved by considering only finitely many α, and the remaining task is to obtain these candidate α. It then suffices to find the discontinuities of the objective function, and therefore the zeros of each component of the function $F_{m,n}(\alpha)$ for all n, denoted by a set $E_{m,n} \subseteq \mathbb{R}$. Since $F_{m,n}(\alpha) = F_{m-1,n} + \alpha h_m(\mathbf{x}_n)$, we have $E_{m,n} \subseteq \{\alpha \mid F_{m,n}(\alpha)[l] = 0 \text{ for some } l \in S_m\} = \{-F_{m-1,n}[l]/h_m(\mathbf{x})[l] \mid l \in S_m\}$, implying $|E_{m,n}| \leq |S_m| = k$. If $(\cup_n E_{m,n}) \cap \mathbb{R}_{>0} = \{a_1, \cdots, a_P\}$ with $0 < a_1 < \cdots < a_P$, then clearly $P \leq Nk$, and the set of candidate α can be chosen to be $\{(a_i + a_{i+1})/2 \mid 1 \leq i < P\} \cup \{a_1/2, a_P + 1\}$. This weighting strategy is called greedy weighting (GW).

Certainly, one can simplify the process of solving (8) by minimizing it over a fixed finite set, E, the *candidate set* of α, to ease the burden of computation and decrease the possibility of overfitting. For example, let $E = \{i/P \mid 1 \leq i \leq P\} \cup \{\epsilon\}$ for some $P \in \mathbb{N}$, where $0 < \epsilon < 1/PM$ is a small number for tie breaking. This weighting strategy is called simple weighting (SW).

4.5. Analysis of Time Complexity

First, we analyze the training time complexity of PRAkEL without considering the weighting of the base classifiers. The trivial steps of Algorithm 1 to form the sub-problems are of time complexity at most $O(N)$ multiplied by the time needed to calculate the reference label $\tilde{\mathbf{y}}_n$ and the cost $\mathbf{c}_n$. The more time-consuming step of PRAkEL, similar to RAkEL, depends on the time spent on the CSMCC base classifier, which is denoted as $T_0(N, d, K')$ for N examples, d features, and K' classes. The empirical results of PRAkEL in the next section demonstrate that it suffices to let each label appear in a fixed number of labelsets on

average. That is, only $M = O(K/k)$ iterations are needed, and hence, the practical training time of PRAkEL is $T_0(N, d, 2^k) \cdot O(K/k)$, which is linear in K. In contrast, as discussed in Section 3, the training time of CFT (Li and Lin, 2014) is $O(NK^2)$ multiplied by the time needed to calculate the cost $\mathbf{c}_n$, and summed with $O(K)$ calls to the base classifier. The complexity analysis reveals the asymptotic efficiency of PRAkEL over CFT.

When considering the weighting, in each iteration, GW (which is generally more time consuming than SW) needs $O(k)$ to determine the zeros of each $F_{m,n}$, and evaluating the goodness of all candidate α can be done within $O(Nk)$, multiplied by the time needed to calculate $\mathbf{c}_n$. That is, the running time of PRAkEL-GW with $M = O(K/k)$ iterations needs an additional $O(NK)$ multiplied by the time needed to calculate the cost $\mathbf{c}_n$. The additional time of PRAkEL-GW is still asymptotically more efficient than the training time of CFT.

5. Experiment

5.1. Experimental Setup

The experiments were conducted on seven benchmark datasets (Tsoumakas et al., 2011).[3] These datasets were taken because of their diversity of domains and popularity in multi-label research community. Their basic statistics are provided in Table 2, where N is the number of examples, d is the dimension of the input space, and K is the number of labels.

Dataset	Domain	N	d	K	Cardinality	Density
`CAL500`	music	502	68	174	26.044	0.150
`emotions`	music	593	72	6	1.868	0.311
`enron`	text	1702	1001	53	3.378	0.064
`medical`	text	978	1449	45	1.245	0.028
`scene`	image	2407	294	6	1.074	0.179
`tmc2007`	text	28596	500	22	2.220	0.101
`yeast`	biology	2417	103	14	4.237	0.303

Table 2: Statistics of the datasets.

For statistical significance, all results reported in Section 5.2 were averaged over 30 independent runs. For each run, we randomly sampled 75% of the dataset for training and used the remaining data for testing. One third of the training set was reserved for validation.

We compared four variants of the proposed method, namely, PRAkEL$_\mathrm{t}$, PRAkEL, PRAkEL-GW and PRAkEL-SW, with three types of methods: (a) labelset-related methods, including RAkEL (Tsoumakas and Vlahavas, 2007) and CS-RAkEL (Lo, 2013); (b) state-of-the-art CSMLC methods, including EPCC (Dembczynski et al., 2010, 2011, 2012) and CFT (Li and Lin, 2014); (c) a state-of-the-art cost-insensitive MLC method, ML-kNN (Zhang and Zhou, 2007). All hyper-parameters of all the compared methods and the base classifiers were selected by grid search on the validation set. For our method and the labelset-related methods, the parameter k was selected from $\{2, \cdots, 9\}$, and for each k, the maximum M was fixed to $10K/k$. The ensemble size of EPCC was selected from $\{1, \cdots, 7\}$ for efficiency, and on datasets with more than 20 labels, the Monte Carlo sampling technique was employed

3. They were obtained from http://mulan.sourceforge.net/datasets-mlc.html.

with a sample size of 200 (Dembczynski et al., 2012). For CFT, the number of internal iterations was selected from $\{2, \cdots, 8\}$, as suggested by the original authors.[4]

For the base classifier of EPCC, we employed logistic regression implemented in LIBLINEAR (Fan et al., 2008). For the methods requiring a regular binary or multi-class classifier, we used linear one-versus-all support vector machines (SVMs) implemented in LIBLINEAR. Our method was coupled with linear RED-OSSVR (Tu and Lin, 2010).[5] The regularization parameter in linear SVMs and RED-OSSVR was also selected by grid search on the validation set. The cost functions we considered in the experiments are all derived from loss functions, as explained in Section 2.

5.2. Results and Discussion

Dataset	PRAkEL$_t$	PRAkEL	PRAkEL-GW	PRAkEL-SW
CAL500	$\mathbf{0.1370 \pm 0.0004}$	$\mathbf{0.1370 \pm 0.0004}$	0.1379 ± 0.0004	0.1372 ± 0.0004
emotions	$\mathbf{0.1951 \pm 0.0026}$	$\mathbf{0.1951 \pm 0.0026}$	0.1974 ± 0.0024	0.1961 ± 0.0025
enron	0.0465 ± 0.0002	0.0465 ± 0.0002	0.0466 ± 0.0003	0.0465 ± 0.0003
medical	0.0103 ± 0.0002	0.0103 ± 0.0002	0.0103 ± 0.0002	0.0102 ± 0.0002
scene	0.0919 ± 0.0008	0.0919 ± 0.0008	0.0915 ± 0.0008	0.0915 ± 0.0008
tmc2007	$\mathbf{0.0532 \pm 0.0001}$	$\mathbf{0.0532 \pm 0.0001}$	0.0538 ± 0.0002	0.0533 ± 0.0001
yeast	$\mathbf{0.1950 \pm 0.0008}$	$\mathbf{0.1950 \pm 0.0008}$	0.1957 ± 0.0008	0.1955 ± 0.0009

Dataset	EPCC	CFT	RAkEL	ML-kNN
CAL500	$\mathbf{0.1370 \pm 0.0004}$	0.1371 ± 0.0004	0.1372 ± 0.0004	0.1466 ± 0.0004
emotions	0.1987 ± 0.0020	0.2012 ± 0.0025	0.2048 ± 0.0027	0.2032 ± 0.0026
enron	$\mathbf{0.0461 \pm 0.0002}$	0.0466 ± 0.0002	0.0466 ± 0.0002	0.0548 ± 0.0003
medical	0.0104 ± 0.0001	0.0105 ± 0.0002	$\mathbf{0.0100 \pm 0.0002}$	0.0157 ± 0.0002
scene	0.0923 ± 0.0009	0.0989 ± 0.0008	0.0919 ± 0.0008	$\mathbf{0.0885 \pm 0.0009}$
tmc2007	0.0568 ± 0.0001	0.0559 ± 0.0001	0.0546 ± 0.0001	0.0671 ± 0.0001
yeast	0.1990 ± 0.0008	0.1993 ± 0.0009	0.2160 ± 0.0012	0.1988 ± 0.0010

Table 3: Performance of each method in terms of Hamming loss (mean ± standard error).

Dataset	PRAkEL$_t$	PRAkEL	PRAkEL-GW	PRAkEL-SW
CAL500	0.2619 ± 0.0009	0.2619 ± 0.0009	0.2555 ± 0.0009	0.2579 ± 0.0009
emotions	0.2179 ± 0.0029	0.2179 ± 0.0029	0.2186 ± 0.0030	0.2182 ± 0.0031
enron	0.1424 ± 0.0010	0.1424 ± 0.0010	0.1424 ± 0.0010	0.1420 ± 0.0010
medical	0.0464 ± 0.0011	0.0464 ± 0.0011	0.0497 ± 0.0012	0.0465 ± 0.0012
scene	0.1285 ± 0.0016	0.1285 ± 0.0016	$\mathbf{0.1258 \pm 0.0015}$	0.1274 ± 0.0017
tmc2007	0.0856 ± 0.0002	0.0856 ± 0.0002	$\mathbf{0.0844 \pm 0.0002}$	0.0848 ± 0.0003
yeast	0.2290 ± 0.0014	0.2290 ± 0.0014	0.2291 ± 0.0014	0.2288 ± 0.0015

Dataset	EPCC	CFT	RAkEL	ML-kNN
CAL500	$\mathbf{0.2501 \pm 0.0007}$	0.2534 ± 0.0006	0.3902 ± 0.0018	0.3885 ± 0.0010
emotions	$\mathbf{0.2121 \pm 0.0030}$	0.2227 ± 0.0031	0.2460 ± 0.0036	0.2505 ± 0.0032
enron	$\mathbf{0.1409 \pm 0.0007}$	0.1415 ± 0.0008	0.2533 ± 0.0014	0.3054 ± 0.0016
medical	$\mathbf{0.0395 \pm 0.0011}$	0.0483 ± 0.0010	0.1067 ± 0.0019	0.2031 ± 0.0027
scene	0.1263 ± 0.0011	0.1411 ± 0.0014	0.1658 ± 0.0016	0.1651 ± 0.0019
tmc2007	0.0866 ± 0.0001	$\mathbf{0.0844 \pm 0.0002}$	0.1554 ± 0.0006	0.2054 ± 0.0006
yeast	$\mathbf{0.2283 \pm 0.0013}$	0.2322 ± 0.0013	0.2628 ± 0.0014	0.2498 ± 0.0017

Table 4: Performance of each method in terms of ranking loss (mean ± standard error).

4. Because of its efficiency issues, we restricted the maximum number of iterations to 4 on datasets with $K > 20$.
5. RED-OSSVR can be shown to be equivalent to one-versus-all SVMs for cost functions $\mathbf{c}_n(\hat{\mathbf{y}}) = [\![\hat{\mathbf{y}} \neq \mathbf{y}_n]\!]$.

Dataset	PRAkEL$_{\mathrm{t}}$	PRAkEL	PRAkEL-GW	PRAkEL-SW
CAL500	0.6498 ± 0.0015	0.5246 ± 0.0014	0.5217 ± 0.0016	0.5216 ± 0.0014
emotions	0.3347 ± 0.0046	0.3308 ± 0.0040	0.3326 ± 0.0044	0.3322 ± 0.0041
enron	0.4545 ± 0.0026	0.4143 ± 0.0028	0.4169 ± 0.0029	0.4138 ± 0.0030
medical	0.1969 ± 0.0029	$\mathbf{0.1899 \pm 0.0032}$	0.1921 ± 0.0037	0.1902 ± 0.0035
scene	0.2469 ± 0.0023	0.2467 ± 0.0023	0.2478 ± 0.0025	$\mathbf{0.2466 \pm 0.0025}$
tmc2007	0.2753 ± 0.0005	0.2671 ± 0.0005	0.2670 ± 0.0006	$\mathbf{0.2661 \pm 0.0005}$
yeast	0.3644 ± 0.0020	0.3455 ± 0.0022	0.3453 ± 0.0021	$\mathbf{0.3449 \pm 0.0021}$

Dataset	EPCC	CFT	RAkEL	ML-kNN
CAL500	$\mathbf{0.5160 \pm 0.0014}$	0.5248 ± 0.0013	0.6579 ± 0.0028	0.6545 ± 0.0020
emotions	$\mathbf{0.3282 \pm 0.0039}$	0.3330 ± 0.0044	0.3859 ± 0.0048	0.3938 ± 0.0058
enron	0.4064 ± 0.0017	$\mathbf{0.3951 \pm 0.0027}$	0.4648 ± 0.0028	0.5675 ± 0.0032
medical	0.2145 ± 0.0030	0.2082 ± 0.0033	0.2163 ± 0.0033	0.4028 ± 0.0052
scene	0.2481 ± 0.0024	0.2790 ± 0.0022	0.2789 ± 0.0027	0.2976 ± 0.0036
tmc2007	0.2788 ± 0.0004	0.2805 ± 0.0005	0.3020 ± 0.0009	0.3871 ± 0.0010
yeast	0.3468 ± 0.0020	0.3551 ± 0.0026	0.3936 ± 0.0025	0.3818 ± 0.0028

Table 5: Performance of each method in terms of F1 loss (mean $\pm$ standard error).

Tables 3, 4 and 5 present the results of the four variants of our method, EPCC, CFT, RAkEL and ML-kNN in terms of Hamming, ranking and F1 loss. The best results for each dataset are marked in bold.

Comparison of Variants of PRAkEL In this subsection, we draw a comparison between the four variants of the proposed method, namely, PRAkEL$_{\mathrm{t}}$, PRAkEL, PRAkEL-GW and PRAkEL-SW. We first compare PRAkEL$_{\mathrm{t}}$ and PRAkEL to understand the difference between using the true and the predicted ones as the reference label vectors. Recall that PRAkEL$_{\mathrm{t}}$ and PRAkEL are theoretically equivalent under Hamming and ranking loss, and therefore, it is not a coincidence that the results of the first two variants in Tables 3 and 4 are exactly the same. Table 5 shows that on all the datasets PRAkEL has lower costs than PRAkEL$_{\mathrm{t}}$ in terms of F1 loss. We also present in Table 6 the results of the Student's t-test at a significance level of 0.05, on two pairs of variants. The comparison of PRAkEL and PRAkEL$_{\mathrm{t}}$ under F1 loss reveals that PRAkEL is significantly superior on five datasets. This demonstrates the benefits of exploiting previous predictions, and is also consistent with the theoretical results in Theorems 5 and 6. Thus, for the remaining experiments, the results of PRAkEL$_{\mathrm{t}}$ are not presented.

Loss function	PRAkEL v.s. PRAkEL$_{\mathrm{t}}$	PRAkEL-GW v.s. PRAkEL	PRAkEL-SW v.s. PRAkEL
Hamming loss	0/7/0	0/5/2	1/6/0
Ranking loss	0/7/0	3/3/1	3/4/0
F1 loss	5/2/0	1/5/1	2/5/0
Total	5/16/0	4/13/4	6/15/0

Table 6: Variants of PRAkEL versus other variants by the Student's t-test at a significance level of 0.05 (superior/comparable/inferior).

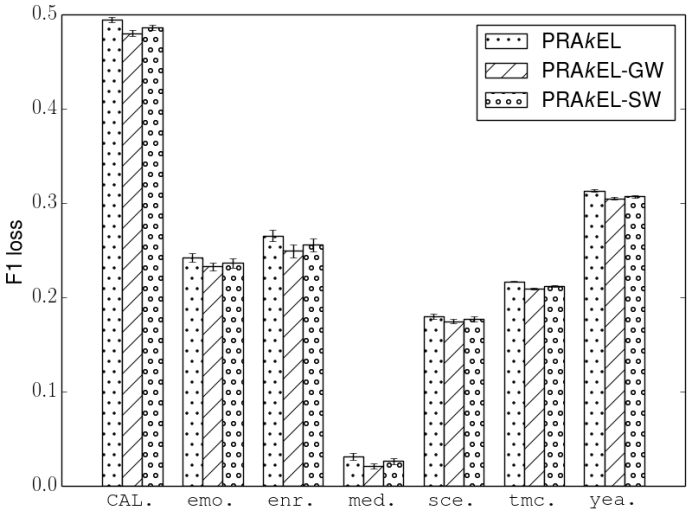


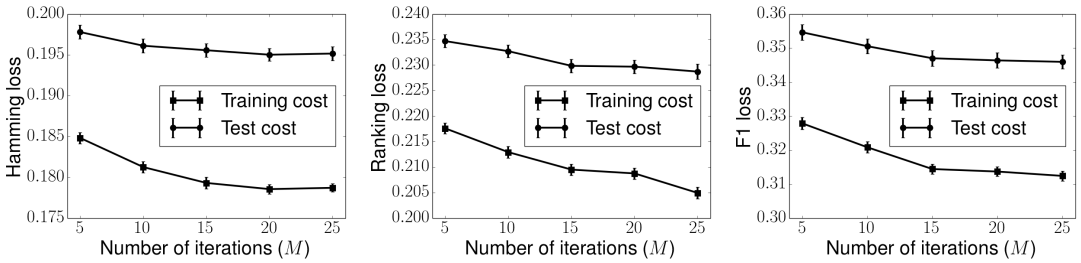
Figure 1: Training costs of PRAkEL, PRAkEL-GW and PRAkEL-SW in terms of F1 loss with the standard errors.

Dataset	PRAkEL	PRAkEL-GW	PRAkEL-SW
CAL500	0.4947 ± 0.0027	$\mathbf{0.4804 \pm 0.0030}$	0.4866 ± 0.0029
emotions	0.2425 ± 0.0048	$\mathbf{0.2327 \pm 0.0045}$	0.2366 ± 0.0049
enron	0.2658 ± 0.0064	$\mathbf{0.2493 \pm 0.0072}$	0.2559 ± 0.0070
medical	0.0313 ± 0.0036	$\mathbf{0.0210 \pm 0.0026}$	0.0264 ± 0.0032
scene	0.1797 ± 0.0026	$\mathbf{0.1747 \pm 0.0024}$	0.1773 ± 0.0025
tmc2007	0.2170 ± 0.0008	$\mathbf{0.2092 \pm 0.0011}$	0.2122 ± 0.0009
yeast	0.3133 ± 0.0013	$\mathbf{0.3050 \pm 0.0015}$	0.3074 ± 0.0014

Table 7: Training costs of PRAkEL, PRAkEL-GW and PRAkEL-SW in terms of F1 loss (mean $\pm$ standard error).

Next, we compare the three weighting strategies, i.e., uniform, greedy and simple weighting. From table 6, overall PRAkEL is competitive with PRAkEL-GW, although under ranking loss the performance of PRAkEL-GW is slightly better. In addition, from the last comparison wee see that PRAkEL-SW is never outperformed by PRAkEL under these three loss functions. For Hamming loss, there is no significant difference between the performance of PRAkEL and PRAkEL-SW. For ranking loss and F1 loss, however, PRAkEL-SW performs slightly better than PRAkEL.

Since the last two variants greedily minimize the training costs in every iteration, it is expected that their training costs are much lower than PRAkEL's. Table 7 and Fig. 1, which show the training costs in terms of F1 loss, verify this deduction. Under other loss functions we also observe similar behavior. The reason is that, for PRAkEL-GW, the weights of the classifiers are determined from an optimization problem with no constraints, while for PRAkEL-SW, the weights are restricted to the candidate set. From a holistic point of view, the candidate set acts as a regularizer, which prevents PRAkEL-SW from



(a) Hamming loss. (b) Ranking loss. (c) F1 loss.

Figure 2: Training and test costs of PRAkEL versus the number of iterations (M) on the `yeast` dataset.

Loss function	Significance by Nemenyi test
Hamming loss	None
Ranking loss	{PRAkEL-SW, EPCC} $\succ$ {RAkEL, ML-kNN}
F1 loss	{PRAkEL, PRAkEL-SW} $\succ$ {RAkEL, ML-kNN}, {PRAkEL-GW, EPCC} $\succ$ {ML-kNN}

Table 8: Significance indicated by the Nemenyi test at a significance level of 0.05 ($\succ$ means significantly better than).

excessively overfitting the training set. In conclusion, among the four variants of our method, PRAkEL-SW is the most stable.

Finally, we demonstrate the effectiveness of ensemble. Fig. 2 shows the training and test costs versus the number of iterations M on `yeast` dataset. We can see that all the costs are decreasing as a function of M. The behavior of the costs on the other datasets is similar.

Comparison with State-of-the-art Methods We compare our method with EPCC, CFT, RAkEL and ML-kNN in terms of Hamming, ranking and F1 loss. Table 3 shows the performance of each method under Hamming loss. RAkEL and ML-kNN individually achieve the best performance on one dataset. On the other datasets, the method with the lowest cost is either PRAkEL or EPCC. Overall, all the methods perform fairly well under Hamming loss.

The results for the other two loss functions are shown in Tables 4 and 5. In terms of ranking loss, EPCC is the most stable method, which outperforms the others on five datasets, and the proposed method reaches the lowest cost on the remaining two datasets. Under F1 loss, our method is superior to the others on half of the datasets, and EPCC has the best performance on two datasets. In addition, it can be seen that under these two loss functions, the two cost-insensitive methods, RAkEL and ML-kNN, are completely not comparable to either of the other cost-sensitive methods. This observation also demonstrates the effectiveness of cost sensitivity.

To compare all the classifiers over multiple datasets, we conducted the Friedman test with the corresponding Nemenyi post-hoc test (Demšar, 2006). For all the three loss functions, the p-values of the Friedman test were 6.6×10^{-3}, 3.6×10^{-5} and 8.7×10^{-6}, respectively. Therefore, the null hypothesis was rejected at $\alpha = 0.05$, and the post-hoc test was performed

Loss function	EPCC	CFT	RAkEL	ML-kNN
Hamming loss	2/4/1	4/3/0	3/3/1	6/0/1
Ranking loss	1/3/3	4/1/2	7/0/0	7/0/0
F1 loss	2/3/2	4/2/1	7/0/0	7/0/0
Total	5/10/6	12/6/3	17/3/1	20/0/1

Table 9: PRAkEL versus each method by the Student's t-test at a significance level of 0.05 (superior/comparable/inferior).

Dataset	PRAkEL	EPCC-Ham	EPCC-F1	CFT
CAL500	$\mathbf{0.2290 \pm 0.0005}$	0.2433 ± 0.0006	0.2497 ± 0.0006	0.2324 ± 0.0005
emotions	$\mathbf{0.2268 \pm 0.0031}$	0.2391 ± 0.0022	0.2503 ± 0.0028	0.2317 ± 0.0032
enron	0.1228 ± 0.0007	0.1321 ± 0.0005	0.1238 ± 0.0005	$\mathbf{0.1183 \pm 0.0007}$
medical	$\mathbf{0.0469 \pm 0.0007}$	0.0509 ± 0.0008	0.0517 ± 0.0007	0.0507 ± 0.0009
scene	$\mathbf{0.1281 \pm 0.0012}$	0.1412 ± 0.0013	0.1387 ± 0.0012	0.1382 ± 0.0014
tmc2007	$\mathbf{0.0974 \pm 0.0002}$	0.1088 ± 0.0002	0.1034 ± 0.0001	0.1028 ± 0.0002
yeast	$\mathbf{0.2300 \pm 0.0012}$	0.2371 ± 0.0009	0.2520 ± 0.0013	0.2350 ± 0.0012

Table 10: Performance of each method in terms of composite loss (mean $\pm$ standard error).

afterwards. The results of Nemenyi test are shown in Table 8, which agree with the discussion in the last paragraph. Basically the proposed method and EPCC outperform the two cost-insensitive methods, RAkEL and ML-kNN. However, according to the Nemenyi test, the performance of the other cost-sensitive methods does not have significant differences. To make further comparisons of our method, EPCC and CFT, we conducted the pairwise Student's t-test at a significance level of 0.05 for each dataset. Table 9 shows the number of datasets on which PRAkEL is statistically superior, comparable, or inferior to each of the other methods. We conclude that under these three metrics, PRAkEL performs significantly better than both RAkEL and ML-kNN and generally better than CFT. As compared with EPCC, PRAkEL is competitive under Hamming and F1 loss, but performs slightly worse than EPCC under ranking loss.

Loss function	EPCC	CFT
Composite loss	7/0/0	6/0/1

Table 11: PRAkEL versus EPCC and CFT under composite loss by the Student's t-test at a significance level of 0.05 (superior/comparable/inferior).

Comparison with EPCC and CFT under Composite Loss To demonstrate our method's capability to optimize general metrics, we defined the function of composite loss as $L_c = 0.8L_H + 0.2L_F$, where L_H and L_F are the functions of Hamming and F1 loss, respectively. This loss function was similarly defined in one experiment on CFT (Li and Lin, 2014).

Because there is no inference rule for EPCC, we used the rules for both Hamming and F1 loss. The results are shown in Table 10, where EPCC-Ham is EPCC with the inference rule for Hamming loss, and EPCC-F1 is the one with the rule for F1 loss. Since CFT is a general cost-sensitive method, we also included it in this experiment. The null hypothesis of the Friedman test was rejected at $\alpha = 0.05$ with the p-value being 4.6×10^{-4}, and the average rank of PRAkEL is 1.14. According to the Nemenyi test, the performance of PRAkEL is significantly better than that of both EPCCl-Ham and EPCC-F1. In addition, according to the results of the t-test in Table 11, the proposed method is superior to CFT except on the `enron` dataset.

Dataset	PRAkEL	CS-RAkEL	GLE
`CAL500`	$\mathbf{0.1370 \pm 0.0004}$	0.1476 ± 0.0005	0.1371 ± 0.0004
`emotions`	$\mathbf{0.1951 \pm 0.0026}$	0.2154 ± 0.0028	0.2039 ± 0.0031
`enron`	$\mathbf{0.0465 \pm 0.0002}$	0.0527 ± 0.0003	$\mathbf{0.0465 \pm 0.0002}$
`medical`	0.0103 ± 0.0002	0.0105 ± 0.0002	$\mathbf{0.0100 \pm 0.0002}$
`scene`	$\mathbf{0.0919 \pm 0.0008}$	0.1078 ± 0.0007	0.0926 ± 0.0009
`tmc2007`	$\mathbf{0.0532 \pm 0.0001}$	0.0576 ± 0.0002	0.0546 ± 0.0001
`yeast`	$\mathbf{0.1950 \pm 0.0008}$	0.2187 ± 0.0017	0.2152 ± 0.0012

Table 12: Performance of PRAkEL and CS-RAkEL in terms of Hamming loss (mean $\pm$ standard error).

Dataset	PRAkEL	CS-RAkEL	GLE
`CAL500`	$\mathbf{0.1421 \pm 0.0004}$	0.1559 ± 0.0005	0.1424 ± 0.0004
`emotions`	$\mathbf{0.2064 \pm 0.0031}$	0.2432 ± 0.0032	0.2127 ± 0.0039
`enron`	$\mathbf{0.0527 \pm 0.0003}$	0.0591 ± 0.0003	0.0530 ± 0.0003
`medical`	0.0110 ± 0.0002	0.0112 ± 0.0002	$\mathbf{0.0108 \pm 0.0002}$
`scene`	$\mathbf{0.0726 \pm 0.0007}$	0.0876 ± 0.0007	0.0753 ± 0.0009
`tmc2007`	$\mathbf{0.0532 \pm 0.0001}$	0.0581 ± 0.0003	0.0550 ± 0.0001
`yeast`	$\mathbf{0.1785 \pm 0.0009}$	0.2052 ± 0.0022	0.1988 ± 0.0012

Table 13: Performance of PRAkEL and CS-RAkEL in terms of weighted Hamming loss (mean $\pm$ standard error).

Comparison with CS-RAkEL and GLE Recall that under the problem setup of CS-RAkEL and GLE, they can handle only weighted Hamming loss, as defined in Section 2. Therefore, in this subsection we compare PRAkEL with these two methods in terms of Hamming loss and weighted Hamming loss. For each dataset, each component of the weight, $\mathbf{w}[l]$, was drawn independently from the uniform distribution over $[0, 1]$, and then the weight $\mathbf{w}$ was normalized such that $\sum_{l=1}^{K} \mathbf{w}[l] = 1$. The results are shown in Tables 12 and 13. We see that PRAkEL reaches lower costs than both CS-RAkEL and GLE on nearly all the datasets. It is clear from these two tables that PRAkEL performs significantly better than the other methods in terms of both loss functions. For completeness, we also provide the results of the Student's t-test in Table 14. The reason for such a significant improvement is that our method considers not only the differences between misclassified costs for each example, but also the varying costs among all the examples. In contrast, CS-RAkEL and GLE take only the latter into account.

Loss function	CS-RAkEL	GLE
Hamming loss	6/1/0	3/3/1
Weighted Hamming loss	6/1/0	4/3/0
Total	12/2/0	7/6/1

Table 14: PRAkEL versus CS-RAkEL by the Student's t-test at a significance level of 0.05 (superior/comparable/inferior).

6. Conclusion

We proposed an efficient cost-sensitive extension of RAkEL, named PRAkEL, which meets the needs of different MLC applications by taking into account the evaluation metric. Experimental results demonstrate that PRAkEL is competitive with other methods designed for certain specific metrics, and frequently outperforms others under general cost functions. The generality of PRAkEL allows it to optimize arbitrary evaluation metrics without additional knowledge, inference rule, or approximation, and thus, it is more suitable for solving real-world problems.

References

Naoki Abe, Bianca Zadrozny, and John Langford. An iterative method for multi-class cost-sensitive learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3–11, 2004.

Alina Beygelzimer, John Langford, and Pradeep Ravikumar. Error-correcting tournaments. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory*, pages 247–262, 2009.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

Amanda Clare and Ross D King. Knowledge discovery in multi-label phenotype data. In *Principles of data mining and knowledge discovery*, pages 42–53. 2001.

Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning*, pages 279–286, 2010.

Krzysztof Dembczynski, Willem Waegeman, and Eyke Hüllermeier. An analysis of chaining in multi-label classification. In *Proceedings of the 21st European Conference on Artificial Intelligence*, pages 294–299, 2012.

Krzysztof J Dembczynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. An exact algorithm for F-measure maximization. In *Advances in Neural Information Processing Systems*, pages 1404–1412, 2011.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Janardhan Rao Doppa, Jun Yu, Chao Ma, Alan Fern, and Prasad Tadepalli. HC-search for multi-label prediction: An empirical study. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1795–1801, 2014.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874, 2008.

Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. Adacost: Misclassification cost-sensitive boosting. In *Proceedings of the 16th International Conference on Machine Learning*, pages 97–105, 1999.

Chun-Sung Ferng and Hsuan-Tien Lin. Multilabel classification using error-correcting codes of hard or soft bits. *IEEE Transactions on Neural Networks and Learning Systems*, 24 (11):1888–1900, 2013.

Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

Eduardo C. Goncalves, Alexandre Plastino, and Alex A. Freitas. A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In *Proceedings of the 25th International Conference on Tools with Artificial Intelligence*, pages 469–476, 2013.

David R. Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

Daniel Hsu, Sham Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. In *Advances in Neural Information Processing Systems*, pages 772–780, 2009.

Chun-Liang Li and Hsuan-Tien Lin. Condensed filter tree for cost-sensitive multi-label classification. In *Proceedings of the 31st International Conference on Machine Learning*, pages 423–431, 2014.

Hung-Yi Lo. *Cost-Sensitive Multi-Label Classification with Applications*. PhD thesis, National Taiwan University, January 2013.

Hung-Yi Lo, Ju-Chiang Wang, Hsin-Min Wang, and Shou-De Lin. Cost-sensitive multi-label learning for audio tag annotation and retrieval. *IEEE Transactions on Multimedia*, 13(3): 518–529, 2011.

Hung-Yi Lo, Shou-De Lin, and Hsin-Min Wang. Generalized k-labelsets ensemble for multi-label and cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1679–1691, 2014.

Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th International Conference on Multimedia*, pages 17–26, 2007.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

Jesse Read, Luca Martino, and David Luengo. Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, 47(3):1535–1546, 2014.

Jesse Read, Luca Martino, Pablo M. Olmos, and David Luengo. Scalable multi-output label prediction: From classifier chains to classifier trellises. *Pattern Recognition*, 48(6): 2096–2109, 2015.

Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2):135–168, 2000.

Eleftherios Spyromitros-Xioufis, Grigorios Tsoumakas, William Groves, and Ioannis Vlahavas. Multi-target regression via input space expansion: Treating targets as inputs. *Machine Learning*, 104(1):55–98, 2016.

Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

Farbound Tai and Hsuan-Tien Lin. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542, 2012.

Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P. Vlahavas. Multi-label classification of music into emotions. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 325–330, 2008.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Machine learning: ECML 2007*, pages 406–417. 2007.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685. 2010.

Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. MULAN: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

Han-Hsing Tu and Hsuan-Tien Lin. One-sided support vector regression for multiclass cost-sensitive classification. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1095–1102, 2010.

Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 435–442, 2003.

Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

Zhi-Hua Zhou and Xu-Ying Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.

Appendix A. Proof

Lemma 1 *Let L_r be the function of ranking loss and $\mathbf{y} \in \mathcal{Y} = \{0,1\}^K$. Then, there exists a unique $\mathbf{w} \in {\mathbb{R}_{\geq 0}}^K$ such that $L_r(\mathbf{y}, \cdot) = L_{H,\mathbf{w}}(\mathbf{y}, \cdot)$, where $L_{H,\mathbf{w}}$ is the function of weighted Hamming loss with respect to $\mathbf{w}$.*

Proof Let $p = |\{1 \leq k \leq K \mid \mathbf{y}[k] = 0\}|$. If $\mathbf{y} = \mathbf{0}$ or $\mathbf{1}$, then $L_r(\mathbf{y}, \hat{\mathbf{y}}) = 0$ for all $\hat{\mathbf{y}}$, and hence the proof is trivial. Therefore, for now we assume $0 < p < K$. Since the normalizer[6] $R(\mathbf{y}) = |\{(k,l) \mid \mathbf{y}[k] < \mathbf{y}[l]\}| = p(K-p)$, we write

$$\begin{aligned} p(K-p) \cdot L_r(\mathbf{y}, \hat{\mathbf{y}}) = &\sum_{k,l} [\![\mathbf{y}[k] < \mathbf{y}[l]]\!][\![\hat{\mathbf{y}}[k] > \hat{\mathbf{y}}[l]]\!] + \\ &\frac{1}{2}\sum_{k,l} [\![\mathbf{y}[k] < \mathbf{y}[l]]\!][\![\hat{\mathbf{y}}[k] = \hat{\mathbf{y}}[l] = 0]\!] + \\ &\frac{1}{2}\sum_{k,l} [\![\mathbf{y}[k] < \mathbf{y}[l]]\!][\![\hat{\mathbf{y}}[k] = \hat{\mathbf{y}}[l] = 1]\!]. \end{aligned}$$

Because $\mathbf{y}[k]$ and $\hat{\mathbf{y}}[k]$ are either 0 or 1, the second term in the right hand side of the last equation is

$$\frac{1}{2}\sum_{k,l} [\![\hat{\mathbf{y}}[k] = \mathbf{y}[k]]\!][\![\mathbf{y}[l] > \mathbf{y}[k]]\!][\![\hat{\mathbf{y}}[l] = 0]\!], \tag{9}$$

and similarly, the third term is

$$\frac{1}{2}\sum_{k,l} [\![\mathbf{y}[l] < \mathbf{y}[k]]\!][\![\hat{\mathbf{y}}[l] = \hat{\mathbf{y}}[k] = 1]\!] = \frac{1}{2}\sum_{k,l} [\![\hat{\mathbf{y}}[k] = \mathbf{y}[k]]\!][\![\mathbf{y}[l] < \mathbf{y}[k]]\!][\![\hat{\mathbf{y}}[l] = 1]\!]. \tag{10}$$

6. The normalizer of ranking loss was defined in Section 2.

In addition, by interchanging k and l, the first term can be written as

$$\begin{aligned}
&\frac{1}{2}\sum_{k,l}[\![\mathbf{y}[k]<\mathbf{y}[l]]\!][\![\hat{\mathbf{y}}[k]>\hat{\mathbf{y}}[l]]\!]+[\![\mathbf{y}[l]<\mathbf{y}[k]]\!][\![\hat{\mathbf{y}}[l]>\hat{\mathbf{y}}[k]]\!]\\
=&\frac{1}{2}\sum_{k,l}[\![\hat{\mathbf{y}}[k]>\mathbf{y}[k]]\!][\![\mathbf{y}[l]\neq\mathbf{y}[k]]\!][\![\hat{\mathbf{y}}[k]=0]\!]+\\
&\frac{1}{2}\sum_{k,l}[\![\hat{\mathbf{y}}[k]<\mathbf{y}[k]]\!][\![\mathbf{y}[l]\neq\mathbf{y}[k]]\!][\![\hat{\mathbf{y}}[k]=1]\!].
\end{aligned}\tag{11}$$

Hence, combining the first term of (11) with (10), and the second term with (9), we have

$$\begin{aligned}
p(K-p)\cdot L_r(\mathbf{y},\hat{\mathbf{y}})&=\frac{1}{2}\sum_{k,l}[\![\hat{\mathbf{y}}[k]<\mathbf{y}[k]]\!][\![\mathbf{y}[l]\neq\mathbf{y}[k]]\!]+[\![\hat{\mathbf{y}}[k]>\mathbf{y}[k]]\!][\![\mathbf{y}[l]\neq\mathbf{y}[k]]\!]\\
&=\frac{1}{2}\sum_{k,l}[\![\hat{\mathbf{y}}[k]\neq\mathbf{y}[k]]\!][\![\mathbf{y}[l]\neq\mathbf{y}[k]]\!]\\
&=\sum_k\left(\frac{1}{2}\sum_l[\![\mathbf{y}[l]\neq\mathbf{y}[k]]\!]\right)[\![\hat{\mathbf{y}}[k]\neq\mathbf{y}[k]]\!]\\
&=\sum_k p(K-p)\mathbf{w}[k][\![\hat{\mathbf{y}}[k]\neq\mathbf{y}[k]]\!]\\
&=p(K-p)\cdot L_{H,\mathbf{w}}(\mathbf{y},\hat{\mathbf{y}}),
\end{aligned}$$

where $\mathbf{w}[k]=\frac{1}{2}\sum_l[\![\mathbf{y}[l]=\mathbf{y}[k]]\!]/p(K-p)$. The uniqueness follows immediately from the above argument. ■

Lemma 2 *Let $L_{H,\mathbf{w}}$ be the function of weighted Hamming loss and S be a k-labelset. For any subsets $\mathbf{y}'_0$ and $\mathbf{y}'_1$ of S, $L_{H,\mathbf{w}}(\mathbf{y},\mathbf{y}'_0\cup\tilde{\mathbf{y}}[S^c])-L_{H,\mathbf{w}}(\mathbf{y},\mathbf{y}'_1\cup\tilde{\mathbf{y}}[S^c])$ is independent of $\tilde{\mathbf{y}}\in\{0,1\}^K$.*

Proof By induction, we may assume that $\mathbf{y}'_0$ and $\mathbf{y}'_1$ differ by only the j-th bit. That is, $\mathbf{y}'_0[j]=0$, $\mathbf{y}'_1[j]=1$, and $\mathbf{y}'_0[l]=\mathbf{y}'_1[l]$ for all $l\neq j$. It then suffices to prove the case where $k=1$.

Since $\mathbf{y}'_0[j]=0$, $L_{H,\mathbf{w}}(\mathbf{y},\mathbf{y}'_0\cup\tilde{\mathbf{y}}[S^c])=\sum_{l\neq j}\mathbf{w}[l]\cdot[\![\tilde{\mathbf{y}}[l]\neq\mathbf{y}[l]]\!]+\mathbf{w}[j]\cdot[\![\mathbf{y}[j]\neq 0]\!]$, and similarly $L_{H,\mathbf{w}}(\mathbf{y},\mathbf{y}'_1\cup\tilde{\mathbf{y}}[S^c])=\sum_{l\neq j}\mathbf{w}[l]\cdot[\![\tilde{\mathbf{y}}[l]\neq\mathbf{y}[l]]\!]+\mathbf{w}[j]\cdot[\![\mathbf{y}[j]\neq 1]\!]$. Therefore, the difference is $\mathbf{w}[j]\cdot|[\![\mathbf{y}[j]\neq 0]\!]-[\![\mathbf{y}[j]\neq 1]\!]|=\mathbf{w}[j]$, which is clearly independent of $\tilde{\mathbf{y}}$.

■