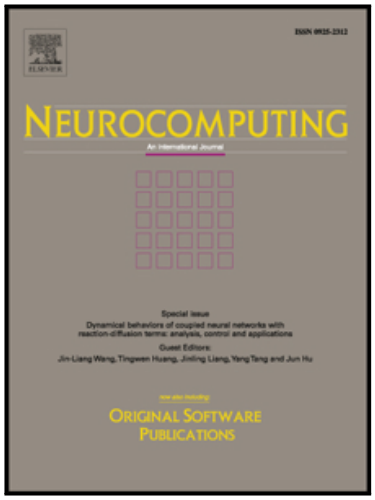
Communicated by Dr. Deng Cheng

Accepted Manuscript



Addressing Class-Imbalance in Multi-Label Learning via Two-Stage Multi-Label Hypernetwork

Kai Wei Sun, Chong Ho Lee

PII: S0925-2312(17)30899-8
DOI: 10.1016/j.neucom.2017.05.049
Reference: NEUCOM 18467

To appear in: *Neurocomputing*

Received date: 21 October 2016
Revised date: 23 February 2017
Accepted date: 22 May 2017

Please cite this article as: Kai Wei Sun, Chong Ho Lee, Addressing Class-Imbalance in Multi-Label Learning via Two-Stage Multi-Label Hypernetwork, *Neurocomputing* (2017), doi: 10.1016/j.neucom.2017.05.049

Addressing Class-Imbalance in Multi-Label Learning via Two-Stage Multi-Label Hypernetwork

Kai Wei Sun[a,1], Chong Ho Lee[a,*]

[a]Department of Information and Communication Engineering, Inha University, Incheon, 402-751, Korea

Abstract

Multi-label learning is concerned with learning from data examples that are represented by a single feature vector while associated with multiple labels simultaneously. Existing multi-label learning approaches mainly focus on exploiting label correlations to facilitate the learning process. However, an intrinsic characteristic of multi-label learning, i.e. class-imbalance has not been well studied. In this paper, we propose a two-stage multi-label hypernetwork (TSMLHN) to exploit label correlations and to make use of them to address the class-imbalance problem in multi-label learning. In TSMLHN, labels of a multi-label data set are divided into two groups, i.e. imbalanced labels and common labels based on their imbalance ratios. In the first stage of TSMLHN, we train a multi-label hypernetwork (MLHN) which generates basic predictions for all labels. In the second stage of TSMLHN, we train TSMLHN based on the predictions obtained from MLHN and utilize the correlations between common labels and imbalanced labels to improve the learning performance of imbalanced labels. Our proposed TSMLHN is conceptually simple, yet it is effective in addressing the class-imbalance problem in multi-label learning. Empirical studies on a broad range of multi-label data sets demonstrate that TSMLHN achieves competitive performance against state-of-the-art multi-label learning algorithms.

Keywords: Multi-label learning, imbalanced learning, hypernetwork

1. Introduction

Multi-label learning is prevalent in various real-world applications, such as text categorization [1, 2], automatic image annotation [3, 4], and bioinformatics [5, 6]. In multi-label learning, each data example is represented by a single feature vector while associated with multiple labels simultaneously. Let $\mathcal{X} \subset \mathbb{R}^d$ be a d-dimensional feature space, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ be a label space with m labels. A multi-label data example is represented as $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$ and $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{im}]$, with $y_{ij} = 1$ indicating that label j is in the label set of $\mathbf{x}_i$, $y_{ij} = 0$ otherwise. Given a training data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) \mid 1 \leq i \leq N\}$ with N multi-label examples, the task of multi-label learning is to learn a function $h : \mathcal{X} \rightarrow 2^{|\mathcal{Y}|}$ which can predict a proper label set $Y^* \subseteq \mathcal{Y}$ for a given multi-label example.

Multi-label learning can be regarded as a general version of traditional supervised learning. If data examples are confined to have only one class label, multi-label learning degenerates into traditional supervised learning. The main difference between multi-label learning and traditional supervised learning lies in that labels in multi-label learning are not mutually exclusive but may be correlated. In the past decades, most of the multi-label learning researches have been focused on how to effectively and efficiently exploit correlations among labels to facilitate the learning process. In [7, 8], instance-based learning is applied to multi-label learning. These approaches utilize neighborhood information and interdependencies between labels to improve predictive accuracy. As the merits of lazy learning are inherited, these instance-based multi-label learning approaches are usually conceptually simple and efficient. The calibrate label ranking (CLR) [9] method considers correlations between any two labels by building a classifier for each label pair. Other multi-label learning algorithms such as classifier chain (CC) [10] and random k-labelset (RAkEL) [11] are designed to exploit correlations among more than two labels. Label correlations are crucial for multi-label learning, however, the exploitation of label correlations is challenged by the overwhelming size of label combinations, i.e. the number of possible label combinations grows exponentially as the number of labels increases.

Class-imbalance which is common in machine learning community has long been regarded as a threat to the performance of standard machine learning algorithms. Similar to the traditional supervised learning, multi-label learning also confronts the class-imbalance problem because in multi-label learning some labels may have much less positive examples than negative examples. In traditional supervised learning, many approaches such as under-sampling, over-sampling, synthetic minority over-sampling (SMOTE), and cost-sensitive learning have been proposed to address the class-imbalance problem [12–15]. In multi-label learning, re-sampling techniques based methods [16, 17] and other techniques that make use of label correlations [18, 19] have been proposed to deal with class-

*Corresponding author.
Email addresses: skw@inhaian.net (Kai Wei Sun), chlee@inha.ac.kr (Chong Ho Lee)

imbalance problem. However, learning from imbalanced multi-label data still remains a challenging problem that has not been well studied. The main challenge for imbalanced multi-label learning lies in the effectiveness and efficiency of learning algorithms as multi-label learning usually demands more complex models than traditional supervised learning and the imbalance level in multi-label learning usually trends to be higher.

Unlike traditional supervised learning, correlations exist among labels in multi-label scenario which can be treated as extra information to help learn from imbalanced multi-label data. In this paper, we propose a two-stage multi-label hypernetwork (TSMLHN) to address the class-imbalance problem in multi-label learning by making use of the correlations among labels. TSMLHN is an extension of our previous works on multi-label learning [20, 21] which are effective and efficient in exploiting correlations among labels. In TSMLHN, labels of a multi-label data set are divided into two groups, i.e. imbalanced labels and common labels based on their imbalance ratios, and correlations among them are explicitly used to facilitate the learning of imbalanced labels. The learning process of TSMLHN consists of two stages. In the first stage, we build a multi-label hypernetwork (MLHN) [21] model for all labels. From MLHN, we can obtain basic predictions for labels. However, the predictions of imbalanced labels trend to be negative due to the lack of enough positive examples for these labels. To address this problem, we initialize and train a TSMLHN model in the second stage where the co-occurrences among common labels and imbalanced labels are utilized to facilitate the learning of imbalanced labels. The learning process of TSMLHN in the second stage can be viewed as an imbalance-specific fine-tuning operation on the learned MLHN model in the first stage with extra information provided by correlations among common labels and imbalanced labels. The main difference between TSMLHN and MLHN lies in that a two-stage learning algorithm is introduced into the learning process of MLHN to improve the learning performance of imbalanced labels by explicitly utilizing the correlations among common labels and imbalanced labels.

The rest of this paper is organized as follows: Section 2 briefly describes some related works on multi-label learning and the learning from imbalanced multi-label data. Section 3 introduces the proposed two-stage multi-label hypernetwork (TSMLHN). Experimental studies on a broad range of multi-label data are presented in Section 4. Finally, the conclusions are given in Section 5.

2. Related work

Modeling label correlations as well as dealing with class-imbalance problem are two challenging tasks in multi-label learning. In this section, we briefly introduce some related works on multi-label learning and the learning from imbalanced multi-label data, respectively.

2.1. Multi-label learning

In the past decades, modeling label correlations has been one of the core goals in multi-label learning. In [22, 23], Zhang and Zhou categorize the existing multi-label learning algorithms into three families, i.e. *first-order strategy*, *second-order strategy*, and *high-order strategy* based on the order of label correlations that the learning algorithms have considered. The *first-order strategy* treats each label independently and ignores label correlations. Binary relevance (BR) [3] which is usually used as building blocks in many advanced multi-label learning algorithms tackles multi-label learning by building a binary classifier for each label independently. BR is conceptually simple but it cannot make use of label correlations to improve the learning performance. ML-*k*NN [7], IBLR [8] and multi-label decision tree (ML-DT) [24] are also representative multi-label learning algorithms of the *first-order strategy*. ML-*k*NN is an adaptation of the *k*-nearest neighbors techniques. IBLR combines instance-based learning and logistic regression for multi-label learning. In IBLR, the information that derives from examples similar to a query example is considered as a feature of the query example. ML-DT adapts decision tree techniques to deal with multi-label learning. In ML-DT, an information gain based on multi-label entropy is utilized to build the tree. Algorithms of the *second-order strategy* such as CLR [9], Rank-SVM [25], and CML [26] exploit correlations between two labels. For a multi-label data set with m labels, CLR builds $\frac{m(m-1)}{2}$ binary classifiers, one for each label pair. Rank-SVM tackles multi-label learning by adapting maximum margin strategy and building a set of linear classifiers to minimize the empirical ranking loss. CML adapts the maximum entropy principle to deal with multi-label data and encodes the label correlations in the constrains that the resulting distribution must satisfy. The *high-order strategy* is capable of modeling more complex correlations among labels. RA*k*EL [11] transforms multi-label learning problem into an ensemble of multi-class learning problems, where each multi-class classifier considers a random k labels only. CC [10] builds a chain of binary classifiers where subsequent binary classifiers are built upon the prediction of preceding ones. ECC [27] is regarded as an ensemble of CC, it is designed to account for the effect of ordering confronted by CC. In recent years, many algorithms that are concerned with other characteristics of multi-label learning have also been proposed. In [28], five multi-label dimensionality reduction algorithms, i.e. canonical correlation analysis (CCA), orthogonal partial least squares (OPLS), hypergraph spectral learning (HSL), linear discriminate analysis (LDA), and shared-subspace learning framework (SSL) are introduced to tackle the *curse-of-dimensionality* in multi-label learning. In [29], a multi-label learning algorithm that utilizes label-specific features (LIFT) is proposed.

2.2. Class-imbalance in multi-label learning

In many multi-label learning tasks, data sets are usually associated with a large number of labels, however, some of the labels have much less positive examples than negative examples, which leads to class-imbalance in multi-label learning. In [16], two measures, i.e. imbalance ratio per label (IRLbl) and mean imbalance ratio (MeanIR) have been proposed to assess the imbalance level of multi-label data, and four algorithms based on re-sampling techniques and label power-

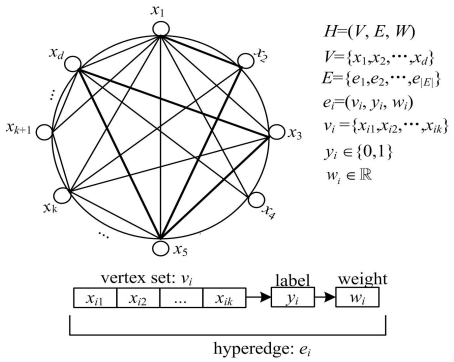


Figure 1: An example of traditional hypernetwork model.

set (LP) are also proposed and compared. These four algorithms are LP based random under-sampling (LP-RUS), LP based random over-sampling (LP-ROS), individual label random over-sampling (ML-ROS), and individual label random under-sampling (ML-RUS). In [17], a multi-label synthetic minority over-sampling (MLSMOTE) technique has been proposed to produce synthetic examples for imbalanced multi-label data sets. In MLSMOTE, for a data example, the feature values of its synthetic example are interpolated along the line connects this data example to one example randomly selected from its neighbors, and the labels that appear in more than half of the neighbors are considered as the labels of the synthetic example. In [19], a multi-label learning method named DEML is proposed to deal with the class-imbalance in multi-label learning by combining labels. In [30], a multiple window approach is proposed to deal with the concept drift and class-imbalance problem in multi-label stream classification (MLSC). In [18], a new multi-label learning approach named cross-coupling aggregation (COCOA) is proposed which is aimed at leveraging the exploitation of label correlations as well as the exploration of class-imbalance.

3. Two-stage multi-label hypernetwork

In this section, we present details of TSMLHN method. Traditional hypernetwork model is introduced in Section 3.1. Section 3.2 describes the TSMLHN model used in the first-stage and the second-stage, respectively. Section 3.3 presents a two-stage learning algorithm which is used to learn TSMLHN from multi-label data. The computational complexity of TSMLHN is presented in Section 3.4.

3.1. Traditional hypernetwork

Hypernetwork is a weighted random hypergraph where high-order interactions among vertices are represented in hyperedges [31]. Because of its capability of modeling high-order interactions, hypernetwork has been successfully applied to various machine learning tasks [32–35]. Formally, a hypernetwork is defined by a triple $H = (V, E, W)$, where V, E, and W denote vertices, hyperedges, and hyperedge weights, respectively. An example of hypernetwork model is illustrated in Fig. 1, where the vertices V correspond to the features (or attributes) of data, hyperedges E are generated by randomly sampling features from the training data, and W represents model parameters need to learn from training data. Hypernetwork consists of a large number of hyperedges. Each hyperedge e_i is composed of three parts, i.e. a vertex set v_i, a class label y_i, and a real-valued weight w_i. The vertex set is viewed as a fraction of a training example, and the weight of a hyperedge indicates the importance of hyperedge to the corresponding class label. Hypernetwork classifies a data example by computing the conditional probabilities with respect to all possible labels and selecting the class label that has the highest conditional probability as the predicted label $\hat{y}$. The classification of a data example $\mathbf{x}$ using hypernetwork is shown in Eq. (1), where $P(\mathbf{x}, y)$ denotes the joint probability of data $\mathbf{x}$ and label y, $P(\mathbf{x})$ is the probability of the data example being observed by hypernetwork model. In hypernetwork model, $P(\mathbf{x}, y)$ is represented by a set of point estimators on hyperedges [33]. In binary classification scenario, $P(\mathbf{x})$ is calculated as $P(\mathbf{x}) = P(\mathbf{x}, y = 1) + P(\mathbf{x}, y = 0)$. The definition of $P(\mathbf{x}, y)$ is shown in Eq. (2), where $\varepsilon(\mathbf{x}, y; W)$ is the energy function which can be expressed in many ways, such as log function and sigmoid function, $Z(W)$ is a normalizing term, and W denotes the weights of hyperedges.

$$\hat{y} = \arg\max_{y\in\{0,1\}} P(y|\mathbf{x}) = \arg\max_{y\in\{0,1\}} \left(\frac{P(\mathbf{x}, y)}{P(\mathbf{x})}\right) \tag{1}$$

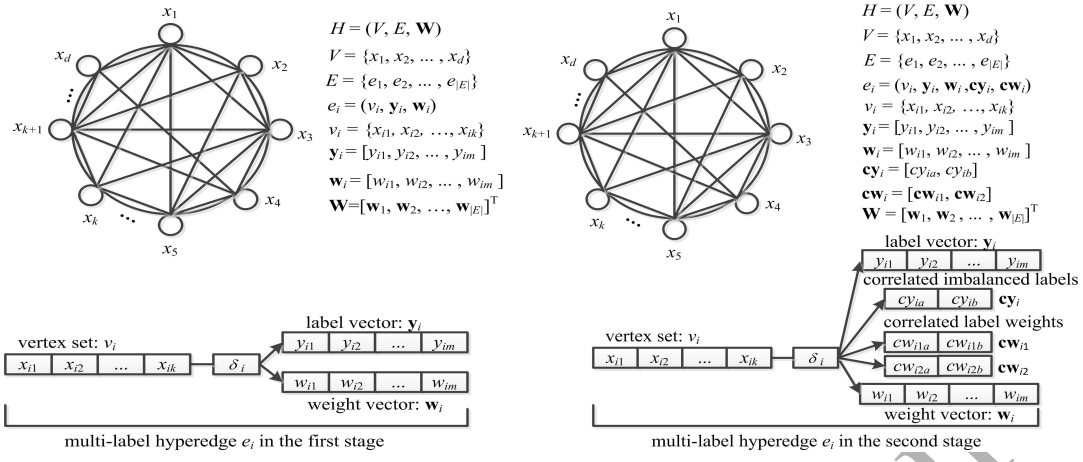
$$P(\mathbf{x}, y) = \frac{1}{Z(W)} \exp(-\varepsilon(\mathbf{x}, y; W)) \tag{2}$$

From above description, we can see that hypernetwork represents a probabilistic model of data and class labels via a large population of hyperedges and their weights. The goal of hypernetwork learning is to learn proper weight values for hyperedges to fit the training data.

3.2. Two-stage multi-label hypernetwork model

In multi-label learning, each data example $\mathbf{x}$ is associated with a label vector $\mathbf{y} = [y_1, y_2, \ldots, y_m]$, with $y_j = 1$ indicating that $\mathbf{x}$ has label j, and $y_j = 0$ otherwise. In order to deal with class-imbalance problem in multi-label learning, we extend hypernetwork into a two-stage multi-label hypernetwork (TSMLHN). In the first stage learning of TSMLHN, we train a multi-label hypernetwork (MLHN) model which is used to generate basic predictions for all labels. In the second stage learning, TSMLHN is initialized and trained based on the predictions obtained from MLHN in the first stage, and correlations among labels are utilized to improve the learning performance of imbalanced labels. An example of TSMLHN model is illustrated in Fig. 2. In the following passages, we describe the model related to each stage of TSMLHN, respectively.

The goal of the first stage learning of TSMLHN is to build a multi-label model which can generate basic predictions for labels. To this end, we extend the traditional hypernetwork into a multi-label hypernetwork (MLHN). Fig. 2(a)) shows the model of MLHN used in the first stage. Similar to hypernetwork, MLHN is also defined by a triple $H = (V, E, \mathbf{W})$, where V are



(a) TSMLHN model in the first stage

(b) TSMLHN model in the second stage

Figure 2: An example of two-stage multi-label hypernetwork (TSMLHN) model.

vertices which correspond to data features, E is a set of multi-label hyperedges, and $\mathbf{W}$ is a real-valued matrix where each row represents the weight vector of a multi-label hyperedge. Each multi-label hyperedge is composed of four parts: a vertex set v_i, a label vector $\mathbf{y}_i$, a real-valued threshold δ_i for multi-label hyperedge matching, and a weight vector $\mathbf{w}_i$. The vertex set and label vector of a multi-label hyperedge are generated from a training example, the weight vector contains parameters need to learn from training data. Like hypernetwork, MLHN also represents the joint probabilities $P(\mathbf{x}, y_i)$ $(1 \leq i \leq m)$ between data and class labels via multi-label hyperedges and their weight vectors. The joint probability is defined in Eq. (3), where $\varepsilon(\mathbf{x}, y_i; \mathbf{W})$ is a log function shown in Eq. (4), $Z(\mathbf{W})$ is a normalizing term, $\mathbf{W}$ is the weight matrix of MLHN, and $y_i \in \{0, 1\}$.

$$P(\mathbf{x}, y_i) = \frac{1}{Z(\mathbf{W})} \exp\left(-\varepsilon(\mathbf{x}, y_i; \mathbf{W})\right) \tag{3}$$

$$\varepsilon(\mathbf{x}, y_i; \mathbf{W}) = -\ln\left(\sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i; e_j)\right) \tag{4}$$

In Eq. (4), $I(\mathbf{x}, y_i; e_j)$ is a matching function which verifies whether a data example $\mathbf{x}$ matches a multi-label hyperedge e_j. The definition of function $I(\mathbf{x}, y_i; e_j)$ is shown in Eq. (5).

$$I(\mathbf{x}, y_i; e_j) = \begin{cases} 1 & dist(\mathbf{x}; e_j) \leq \delta_j \quad and \quad y_{ji} = y_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In Eq. (5), $dist(\mathbf{x}; e_j)$ is the Euclidean distance between $\mathbf{x}$ and multi-label hyperedge e_j, δ_j is the matching threshold associated with multi-label hyperedge e_j, y_{ji} is the value of label i in multi-label hyperedge e_j. In MLHN model, the matching threshold δ_j for a given multi-label hyperedge e_j is calculated as the average Euclidean distance between the training example from which e_j is generated and its k nearest neighbor examples. Formally, the calculation of δ_j is defined in Eq. (6), where $G_{\mathbf{x}_j}$ represents the set of nearest neighbor examples of data example $\mathbf{x}_j$, and $\mathbf{x}_j$ is the training example from which e_j is generated. It is worth mentioning that the distances between data examples and their neighbors and label information of the neighbors are utilized in instance-based multi-label learning algorithms, such as ML-kNN [7] and IBLR [8]. However, in TSMLHN only the distance information is used to form a threshold for multi-label hyperedge matching. This makes TSMLHN very different from other multi-label learning algorithms that utilize label interdependencies among neighbors.

$$\delta_j = \frac{1}{|G_{\mathbf{x}_j}|} \sum_{\hat{\mathbf{x}} \in G_{\mathbf{x}_j}} \|\mathbf{x}_j - \hat{\mathbf{x}}\| \tag{6}$$

$$P(y_i = 1 \mid \mathbf{x}) = \frac{P(\mathbf{x}, y_i = 1)}{P(\mathbf{x})} \tag{7}$$

$$\begin{aligned} P(y_i = 1 \mid \mathbf{x}) &= \frac{P(\mathbf{x}, y_i = 1)}{P(\mathbf{x}, y_i = 1) + P(\mathbf{x}, y_i = 0)} \\ &= \frac{\sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 1; e_j)}{\sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 1; e_j) + \sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 0; e_j)} \end{aligned} \tag{8}$$

For the classification of a multi-label data example $\mathbf{x}$, MLHN returns a real-valued vector $\mathbf{P} = [P(y_1 = 1 \mid \mathbf{x}), P(y_2 = 1 \mid \mathbf{x}), \ldots, P(y_m = 1 \mid \mathbf{x})]$ to indicate the probabilities of having corresponding labels. The calculation of the probability is shown in Eq. (7), where $P(\mathbf{x})$ is the probability of example $\mathbf{x}$ being observed by MLHN. Because each label y_i in multi-label learning has only two values, i.e. $y_i \in \{0, 1\}$, $P(\mathbf{x})$ can be calculated by $P(\mathbf{x}) = P(\mathbf{x}, y_i = 0 \mid \mathbf{W}) + P(\mathbf{x}, y_i = 1 \mid \mathbf{W})$. Thus, Eq. (7) can be also represented by Eq. (8). From Eq. (8), we note that if $\sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 1; e_j) \geq \sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 0; e_j)$, $P(y_i = 1 \mid \mathbf{x}) \geq 0.5$, and $P(y_i = 1 \mid \mathbf{x}) < 0.5$ otherwise.

In the learning process of MLHN, we need to calculate the derivative of $P(y_i = 1 \mid \mathbf{x})$ with respect to w_{ji}, i.e. $\frac{\partial P(y_i = 1 \mid \mathbf{x})}{\partial w_{ji}}$. However, the calculation of derivative is very complicated if Eq. (8) is used to calculate $P(y_i = 1 \mid \mathbf{x})$. In this paper, we utilize the logistic function to calculate $P(y_i = 1 \mid \mathbf{x})$, which is shown

in Eq. (9). We can see that Eq. (9) has the same property with Eq. (8), i.e. if $\sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 1; e_j) \geq \sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 0; e_j)$, $P(y_i = 1 \mid \mathbf{x}) \geq 0.5$, and $P(y_i = 1 \mid \mathbf{x}) < 0.5$ otherwise, but the calculation of derivative of Eq. (9) is more simple than that of Eq. (8).

$$\begin{aligned} P(y_i = 1 \mid \mathbf{x}) &= \frac{1}{1 + \exp(-(w_i^+ - w_i^-)} \\ w_i^+ &= \sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 1; e_j) \\ w_i^- &= \sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 0; e_j) \end{aligned} \tag{9}$$

As mentioned above, MLHN returns a real-valued vector $\mathbf{P} = [P(y_1 = 1 \mid \mathbf{x}), P(y_2 = 1 \mid \mathbf{x}), \ldots P(y_m = 1 \mid \mathbf{x})]$ to indicate the probabilities of having corresponding labels. In order to decide a proper label set for an unseen example $\mathbf{x}$, the probabilities are calibrated against a threshold which is set to be 0.5 in this paper.

From Eq. (9), we note that the predictions for imbalanced labels trend to be negative because the value of w_i^+ is usually smaller than that of w_i^- due to the lack of positive examples. In this paper, we attempt to address this problem by making use of the correlations among common labels and imbalanced labels. In the second stage of TSMLHN, we initialize and train TSMLHN based on the basic predictions obtained from MLHN in the first stage. The goal of the second stage learning of TSMLHN is to leverage the learning performance of imbalanced labels by utilizing the correlations among common labels and imbalanced labels. In the second stage, labels of a multi-label data set are divided into imbalanced labels and common labels based on the imbalanced ratio ImR_i of labels. The definition of ImR_i for each label i is shown in Eq. (10). For a class label $i \in \{1, 2, \ldots, m\}$, if $ImR_i \geq \theta$ then label i is regarded as an imbalanced label, label i is regarded as a common label otherwise.

$$\begin{aligned} ImR_i &= \frac{\max(|\mathcal{D}_i^+|, |\mathcal{D}_i^-|)}{\min(|\mathcal{D}_i^+|, |\mathcal{D}_i^-|)} \\ \mathcal{D}_i^+ &= \{\mathbf{x}_j \mid (\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}, y_{ji} = 1\} \\ \mathcal{D}_i^- &= \{\mathbf{x}_j \mid (\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}, y_{ji} = 0\} \end{aligned} \tag{10}$$

The TSMLHN model used in the second stage is shown in Fig. 2(b). Here, we take an example to explain the TSMLHN model in the second stage. Suppose we have totally m class labels which are divided into two groups, i.e. common labels $\{1, 2\}$ and imbalanced labels $\{3, \ldots, m\}$. Consider the multi-label hyperedge e_i shown in Fig. 2(b), the correlated imbalance labels of e_i are cy_{ia} and cy_{ib}, which means that imbalanced labels $a, b \in \{3, \ldots, m\}$ co-occur with common labels y_{i1}, y_{i2}. The correlated label weights $cw_{i1a}, cw_{i1b}, cw_{i2a}, cw_{i2b}$ indicate how much information the common labels y_{i1}, y_{i2} in multi-label hyperedge e_i can provide for the imbalanced labels a and b, respectively. By combining the information provided by common labels, the calculation of the probabilities of labels (see Eq. (9)) can be updated as shown in Eq. (11), where q is the number of common labels, cw_{jki} is the weight of correlated label. From Eq. (11), we can see that the probabilities of common labels remain the same as in the first stage while the probabilities of imbalanced labels are updated based on the correlations between common labels and imbalanced labels. Given an unseen example $\mathbf{x}$, the method of TSMLHN classification is shown in **Algorithm 1**.

$$\begin{aligned} P(y_i = 1 \mid \mathbf{x}) &= \frac{1}{1 + \exp(-(w_i^+ - w_i^-))} \\ w_i^+ &= \sum_{j=1}^{|E|} \left(w_{ji} I(\mathbf{x}, y_i = 1; e_j) + \sum_{k=1}^{q} cw_{jki} \right) \\ w_i^- &= \sum_{j=1}^{|E|} w_{ji} I(\mathbf{x}, y_i = 0; e_j) \end{aligned} \tag{11}$$

Algorithm 1. Method of TSMLHN classification.
Input: unseen example: $\mathbf{x}$.
Output: probability vector $\mathbf{P}$, predicted label vector $\hat{\mathbf{y}}$
Process:
Classification in first stage:
1: Identify k nearest neighbor examples for $\mathbf{x}$ from training data
2: Extract multi-label hyperedges that are generated from the neighbor examples and match with $\mathbf{x}$ into a set U
3: for i=1 to m do
4: $\mathbf{W}^+[i] \leftarrow 0$
5: $\mathbf{W}^-[i] \leftarrow 0$
6: for each $e_j \in U$ do
7: if $y_{ji} = 1$
8: $\mathbf{W}^+[i] = \mathbf{W}^+[i] + w_{ji}$
9: else
10: $\mathbf{W}^-[i] = \mathbf{W}^-[i] + w_{ji}$
11: end if
12: end for
13: end for
14: for i=1 to m do
15: $\mathbf{P}[i] = \frac{1}{1+\exp(-(\mathbf{W}^+[i] - \mathbf{W}^-[i]))}$
16: end for
Classification in second stage:
17: for $i = 1$ to m do
18: for each $e_j \in U$ do
19: for $k = 1$ to q do
20: if $\mathbf{P}[k] \geq 0.5$ do
21: $\mathbf{W}^+[i] = \mathbf{W}^+[i] + cw_{jki}$
22: end if
23: end for
24: end for
25: end for
26: for i=1 to m do
27: $\mathbf{P}[i] = \frac{1}{1+\exp(-(\mathbf{W}^+[i] - \mathbf{W}^-[i]))}$
28: if $\mathbf{P}[i] \geq 0.5$
29: $\hat{\mathbf{y}}[i] = 1$
30: else
31: $\hat{\mathbf{y}}[i] = 0$
32: end if
33: end for

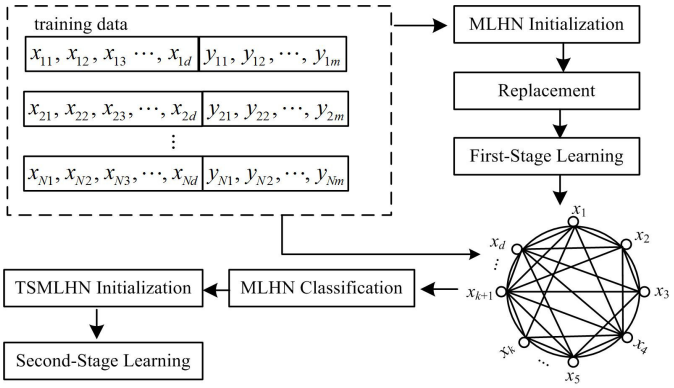


Figure 3: Two-stage learning framework of TSMLHN

34: return $\mathbf{P}$ and $\hat{\mathbf{y}}$

3.3. Two-stage learning of TSMLHN

The model of TSMLHN has been introduced above. In this section, we propose a two-stage learning algorithm to learn TSMLHN from multi-label data. The learning algorithm in the first stage is aimed at learning the weight matrix $\mathbf{W}$ (as shown in Fig. 2(a)) from the training data while the learning algorithm in the second stage is focused on learning the correlated label weights $\mathbf{cw}$ (as shown in Fig. 2(b)) of multi-label hyperedges and leveraging the learning performance of imbalanced labels by using the correlations between common labels and imbalanced labels. The learning in the second stage can also be viewed as a fine-tuning process of the learning of imbalanced labels in the first stage. The two-stage learning framework of TSMLHN is shown in Fig. 3. In the following passages, we will introduce each learning step in detail.

The first-stage learning begins with an initial multi-label hypernetwork (MLHN) where multi-label hyperedges are generated from training examples. Given a multi-label example $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = [x_1, x_2, \ldots, x_d]$, and $\mathbf{y} = [y_1, y_2, \ldots, y_m]$, a multi-label hyperedge is generated by randomly sampling a subset of features from $\mathbf{x}$, taking $\mathbf{y}$ as the label vector of multi-label hyperedge, and associating an initial weight vector to this multi-label hyperedge.

Due to the randomness of hyperedge generation, it is necessary to replace multi-label hyperedges of low fitness so that multi-label hyperedges with better representation of the training data are included in MLHN model. The fitness of a multi-label hyperedge is calculated in Eq. (12), where G is a set of training examples that match with multi-label hyperedge e_i, m is the number of labels, y_{jk} is the value of label k of example $(\mathbf{x}_j, \mathbf{y}_j)$, and y_{ik} is the value of label k in multi-label hyperedge e_i. The fitness is calculated as the average similarity between multi-label hyperedge labels and the labels of matching training examples, and the higher the similarity the bigger the fitness. From this point, the fitness of a multi-label hyperedge can also be viewed as its generalization ability to some extent. In the MLHN replacement step, for each multi-label hyperedge e_i its fitness is calculated, a new multi-label hyperedge is generated from the training example from which e_i was generated, and if the fitness of new multi-label hyperedge is bigger than that of e_i, then e_i is replaced by the new multi-label hyperedge.

$$fitness(e_i) = \frac{1}{|G|} \sum_{(\mathbf{x}_j, \mathbf{y}_j) \in G} \frac{|\{k \mid y_{jk} = y_{ik}, 1 \le k \le m\}|}{m} \tag{12}$$

Given a training data set $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n) \mid 1 \le n \le N\}$, the task of the first stage learning is to learn the values of the weight matrix $\mathbf{W}$ of MLHN from $\mathcal{D}$. In this paper, the first stage learning is executed by minimizing the error function shown in Eq. (13) and Eq. (14) in a gradient descent manner. In Eq. (14), $P^*(y_{ni} = 1 \mid \mathbf{x}_n)$ is the target probability which is 1.0 if label i is in the label set of training example $\mathbf{x}_n$, and $P^*(y_{ni} = 1 \mid \mathbf{x}_n) = 0.0$ otherwise. Given a training example $(\mathbf{x}_n, \mathbf{y}_n)$, it is classified by current MLHN firstly, and the weight vectors of multi-label hyperedges are updated according to the classification results. The updating of weight vector is shown in Eq. (15), Eq. (16), Eq. (17), and Eq. (18), where w_{ji} is the weight of label i in multi-label hyperedge e_j, η is the learning rate which is set to be 0.01 in this paper, y_{ji} is the value of label i in e_j.

$$Err(\mathbf{W}) = \sum_{n=1}^{N} Err_n(\mathbf{W}) \tag{13}$$

$$Err_n(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^{m} \left[P^*(y_{ni} = 1 \mid \mathbf{x}_n) - P(y_{ni} = 1 \mid \mathbf{x}_n)\right]^2 \tag{14}$$

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji} \tag{15}$$

$$\Delta w_{ji} = -\eta \frac{\partial Err_n(\mathbf{W})}{\partial w_{ji}} \tag{16}$$

$$\frac{\partial Err_n(\mathbf{W})}{\partial w_{ji}} = -[P^*(y_{ni} = 1 \mid \mathbf{x}_n) - P(y_{ni} = 1 \mid \mathbf{x}_n)] \frac{\partial P(y_{ni} = 1 \mid \mathbf{x}_n)}{\partial w_{ji}} \tag{17}$$

$$\frac{\partial P(y_{ni} = 1 \mid \mathbf{x}_n)}{\partial w_{ji}} = (-1)^{1-y_{ji}} P(y_{ni} = 1 \mid \mathbf{x}_n)(1 - P(y_{ni} = 1 \mid \mathbf{x}_n)) \tag{18}$$

Through the first-stage learning, we obtain a MLHN model which can generate basic predictions for all labels. In order to address the class-imbalance problem in multi-label learning, we initialize a TSMLHN model based on the prediction results obtained from MLHN on the training data. For the sake of easy understanding, we present an example to show how the TSMLHN model is initialized. Suppose we are given a training multi-label data set with totally five labels, i.e. $\mathcal{Y} = \{y_1, y_2, y_3, y_4, y_5\}$, and these labels are divided into two groups, i.e. common labels $\{y_1, y_2\}$ and imbalanced labels $\{y_3, y_4, y_5\}$. Consider a training example $(\mathbf{x}, \mathbf{y})$, $\mathbf{y} = [y_1 = 1, y_2 = 1, y_3 = 0, y_4 = 1, y_5 = 1]$, it is firstly classified by the MLHN in the first stage and the predicted vector is $\hat{\mathbf{y}} = [\hat{y}_1 = 1, \hat{y}_2 = 1, \hat{y}_3 = 0, \hat{y}_4 = 0, \hat{y}_5 = 0]$. For each and every multi-label hyperedge e_j that matches this training sample, if its label value of y_1 is 1, i.e. $y_{j1} = 1$, add label y_4 and label y_5 to the correlated label set $\mathbf{cy}_j$,

set the corresponding correlated label weights cw_{j14} and cw_{j15} to be 0. The same operation is executed if its label value on y_2 is also 1, i.e. $y_{j2} = 1$. The classification of a training example using MLHN follows the classification method in first stage shown in **Algorithm 1** (from step 1 to step 16).

In learning process of the second stage, the learned weight matrix $\mathbf{W}$ of MLHN in the first stage remains unchanged, while the correlated label weights of multi-label hyperedges are updated to minimized the prediction errors on imbalanced labels. The error function for imbalanced labels $ImErr(\mathbf{cw})$ is shown in Eq. (19) and Eq. (20), where cw_{jki} is the correlated label weight of the common label k for the imbalanced label i in multi-label hyperedge e_j. The updating of correlated label weights is shown in Eq. (21) and Eq. (22). The two-stage learning process of TSMLHN is presented in **Algorithm 2**.

$$ImErr(\mathbf{cw}) = \sum_{n=1}^{N} ImErr_n(\mathbf{cw}) \tag{19}$$

$$ImErr_n(\mathbf{cw}) = \frac{1}{2} \sum_{i \in ImLabel} [P^*(y_{ni} = 1 \mid \mathbf{x}_n) - P(y_{ni} = 1 \mid \mathbf{x}_n)]^2 \tag{20}$$

$$cw_{jki} \leftarrow cw_{jki} + \Delta cw_{jki} \tag{21}$$

$$\Delta cw_{jki} = \eta[P^*(y_{ni} = 1 \mid \mathbf{x}_n) - P(y_{ni} = 1 \mid \mathbf{x}_n)] \\ (1 - P(y_{ni} = 1 \mid \mathbf{x}_n))P(y_{ni} = 1 \mid \mathbf{x}_n) \tag{22}$$

Algorithm 2. Two-stage learning process of TSMLHN.
Input: training data $\mathcal{D}$; TSMLHN model: $H^0 = (V, E, \mathbf{W})$; training times: T; learning rate: η; common label set: *BLabel*; imbalanced label set: *ImLabel*.
Output: learned TSMLHN model: H^*
Process:
Initialize TSMLHN in the first-stage.
Replacement.
First-stage learning:
for $t = 1$ to T do
  for $n = 1$ to N do
    classify the n-th training example $(\mathbf{x}_n, \mathbf{y}_n)$ using H^{t-1} (from step 1 to step 16 in **Algorithm 1**)
    get predicted label vector $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]$
    extract multi-label hyperedges that match with $\mathbf{x}_n$ into U
    for $i = 1$ to m do
      if $\hat{y}_i \neq y_{ni}$
        for each $e_j \in U$
          update weight $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ (see Eq. (15))
        end for
      end if
    end for
  end for
  $H^t \leftarrow H^{t-1} + \Delta H^{t-1}$
end for
Initialize TSMLHN in the second stage:
for $n = 1$ to N do
  classify training example $(\mathbf{x}_n, \mathbf{y}_n)$ using H^T (from step 1 to step 16 in **Algorithm 1**)
  get predicted label vector $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]$
  extract multi-label hyperedges that match with $\mathbf{x}_n$ into U
  for each $i \in ImLabel$ do
    if $y_{ni} = 1$ and $\hat{y}_i = 0$ do
      for each $e_j \in U$ do
        add i to correlated label $\mathbf{cy}_j$ of e_j if i is not in $\mathbf{cy}_j$
        for each $k \in BLabel$ do
          if the value of y_{jk} of e_j is 1 do
            set the correlated label weight cw_{jki} to be 0
          end if
        end for
      end for
    end if
  end for
end for
Second-stage learning:
for $t = 1$ to T do
  for $n = 1$ to N do
    classify training example $(\mathbf{x}_n, \mathbf{y}_n)$ (from step 1 to step 34 in **Algorithm 1**)
    get predicted results $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]$
    extract multi-label hyperedges that match $\mathbf{x}_n$ into U
    for each $i \in ImLabel$ do
      if $\hat{y}_i \neq y_{ni}$ do
        for each $e_j \in U$ do
          for each $k \in BLabel$ do
            update weight $cw_{jki} \leftarrow cw_{jki} + \Delta cw_{jki}$ (see Eq. (21))
          end for
        end for
      end if
    end for
  end for
end for

3.4. Computational complexity

In this section, we discuss the computational complexity of training and testing of TSMLHN. The training process of TSMLHN consists of two stages. In the first stage of TSMLHN learning, we train a MLHN model for all labels, the computational complexity of training MLHN is $O(N^2 + ckN + ckmN)$ [21], where N is the number of training examples, c is a constant integer that controls the size of MLHN, k is the number of nearest neighbors, m is the number of labels. In the second stage of TSMLHN learning, correlations between common labels and imbalanced labels are utilized to improve the learning performance of imbalanced labels, the computational complexity of the second stage learning is $O((qm - q^2)N)$, where q is the number of common labels. Thus, the overall computational complexity of TSMLHN training is $O(N^2 + ckN + ckmN + (qm - q^2)N)$. Given a test data set with M examples, the computational complexity for testing is $O(MN + ckM + mM + qM)$.

4. Experimental study

In this section, we firstly introduce the multi-label data sets used in our experiments, followed by multi-label evaluation

Table 1: Characteristics of multi-label data sets used in our experiments

data set	$\|D\|$	$dim(D)$	$L(D)$	$LC(D)$	$min(ImR)$	$max(ImR)$	$avg(ImR)$	domain
CAL500	502	68	174	26.044	1.041	99.400	22.345	music
emotions	593	72	6	1.869	1.246	3.007	2.320	music
medical	978	1449	45	1.245	2.677	977.00	328.069	text
enron	1702	1001	53	3.378	1.009	1701.000	136.867	text
scene	2407	294	6	1.074	3.561	5.613	4.662	image
yeast	2417	103	14	4.237	1.329	70.088	8.954	biology
slashdot	3782	1079	22	1.181	5.531	1248.667	112.690	text
corel5k	5000	499	374	3.522	3.464	4999.000	845.284	image
rcv1subset1	6000	47236	101	2.880	3.342	2999.00	235.580	text
bibtex	7395	1836	159	2.402	6.097	144.000	87.699	text
eurlex-sm	19348	5000	201	2.213	3.510	19347.000	2420.775	text
tmc2007	28596	49060	22	2.158	1.449	69.958	27.996	text

Table 2: Comparing algorithms used in experiments

algorithm	parameter setting	reference
Binary relevance based on SVM (BR-SVM)	based learner: linear SVM.	[3]
Multi-label k nearest neighbors (ML-kNN)	smooth factor: 1.0, number of nearest neighbors: 10	[7]
Calibrate label ranking (CLR)	base learner: linear SVM	[9]
Random k labelset (RAkEL)	base learner: linear SVM, labelset size $k = 3$, ensemble size = $2m$	[11]
Ensemble of classifier chains (ECC)	base learner: linear SVM, ensemble size = 50	[27]
Instance-based learning and logistic regression (IBLR)	number of nearest neighbors: 10	[8]
Cross-coupling aggregation (COCOA)	base learner: C4.5	[18]
Individual label random oversampling (ML-ROS)	sampling percentage: 10, base learner: IBLR	[16]
Individual label random under-sampling (ML-RUS)	sampling percentage: 10, base learner: IBLR	[16]
Imbalanced multi-label learning through synthetic instance generation (MLSMOTE)	number of nearest neighbors: 5, base learner: IBLR	[17]
Multi-label hypernetwork (MLHN)	number of nearest neighbors: 20	[21]

Table 3: Average classification results of TSMLHN and the comparing algorithms in terms of *Hamming loss* ↓.

data set	BR-SVM	ML-kNN	CLR	RAkEL	ECC	IBLR	COCOA	ML-ROS	ML-RUS	MLSMOTE	MLHN	TSMLHN
CAL500	0.138	0.140	0.140	0.138	0.142	0.232	0.174	0.234	0.234	0.231	**0.136**	0.152
emotions	0.202	0.208	0.201	**0.197**	0.198	0.204	0.207	0.204	0.208	0.221	0.205	0.204
medical	0.012	0.013	0.023	0.012	0.012	0.014	0.014	0.014	0.014	0.013	**0.011**	**0.011**
enron	0.046	0.049	0.047	0.047	0.048	0.057	0.055	0.057	0.059	0.056	**0.043**	0.054
scene	0.110	0.093	0.112	0.096	0.093	**0.087**	0.092	0.088	0.088	0.095	0.091	0.093
yeast	0.202	0.198	0.202	0.201	0.203	0.199	0.209	0.203	0.200	0.208	0.202	**0.197**
slashdot	0.041	0.038	0.054	0.043	**0.037**	0.044	0.041	0.041	0.041	0.040	0.053	0.050
corel5k	0.009	0.009	0.009	0.009	0.010	0.026	0.009	0.025	0.026	0.024	**0.008**	0.009
rcv1subset1	0.026	0.027	0.026	0.026	0.027	0.030	0.027	0.030	0.034	0.030	**0.024**	0.031
bibtex	0.013	0.013	0.013	0.013	**0.012**	0.023	0.014	0.024	0.025	0.018	0.013	0.013
eurlex-sm	0.006	0.006	0.008	0.006	**0.005**	0.007	0.006	0.007	0.007	0.007	0.006	0.006
tmc2007	0.062	0.061	0.061	0.063	0.052	0.061	0.056	0.062	0.061	0.062	0.052	**0.048**

metrics and comparing algorithms. Finally, experimental results and analysis are presented.

4.1. Data sets

In order to present comprehensive performance evaluation, a total of 12 benchmark multi-label data sets [1] have been collected for experimental studies. Table 1 shows some basic characteristics of experimental data sets. In Table 1, $|D|$ denotes the number of data examples, $dim(D)$ stands for the dimension of feature space, $L(D)$ is the number of labels, $LC(D)$ is the average number of labels associated with each data example, $min(ImR)$ is the minimum imbalanced ratio, $max(ImR)$ is the maximum imbalance ratio, $avg(ImR)$ is the average imbalance ratio over all labels. Data sets are ordered by the number of data examples. In our experiments, dimensionality reduction is performed on text data sets using the fuzzy relevance method

[1] data sets except for slashdot are available at http://mulan.sourceforge.net/datasets-mlc.html, the slashdot data set is available at http://meka.sourceforge.net/datasets

Table 4: Average classification results of TSMLHN and the comparing algorithms in terms of *Example based F-measure* ↑.

data set	BR-SVM	ML-kNN	CLR	RAkEL	ECC	IBLR	COCOA	ML-ROS	ML-RUS	MLSMOTE	MLHN	TSMLHN
CAL500	0.336	0.319	0.296	0.344	0.347	0.320	0.367	0.320	0.318	0.317	0.360	**0.388**
emotions	0.582	0.585	0.592	0.643	0.643	0.595	0.639	0.600	0.592	0.586	0.662	**0.672**
medical	0.665	0.669	0.241	0.664	0.729	0.726	0.720	0.729	0.723	0.731	**0.794**	0.754
enron	0.534	0.517	0.528	0.535	0.559	0.497	0.538	0.502	0.489	0.501	**0.565**	0.541
scene	0.613	0.673	0.622	0.690	0.695	0.681	0.658	0.679	0.672	0.672	0.750	**0.764**
yeast	0.606	0.607	0.610	0.627	0.619	0.612	0.612	0.605	0.611	0.584	0.644	**0.651**
slashdot	0.324	0.446	0.001	0.311	0.395	0.424	**0.565**	0.425	0.420	0.424	0.532	0.552
corel5k	0.068	0.059	0.037	0.066	0.161	0.102	0.186	0.104	0.103	0.106	**0.193**	0.188
rcv1subset1	0.304	0.391	0.289	0.306	0.382	0.404	0.415	0.406	0.402	0.404	0.480	**0.539**
bibtex	0.276	0.307	0.271	0.272	0.297	0.295	0.415	0.297	0.288	0.313	0.433	**0.445**
eurlex-sm	0.645	0.678	0.333	0.646	0.685	0.658	0.722	0.663	0.649	0.660	**0.742**	0.738
tmc2007	0.644	0.655	0.646	0.644	0.658	0.659	0.717	0.656	0.659	0.653	0.715	**0.732**

proposed by Lee and Jiang [36]. After dimensionality reduction, the feature space and label space of text data set are of the same size.

4.2. Evaluation metrics

In our experiments, six evaluation metrics are utilized to comprehensively evaluate the performance of multi-label learning methods, including four common multi-label evaluation metrics (i.e. *Hamming loss*, *Example based F-measure*, *Ranking loss*, and *Average precision*) and two imbalance-specific evaluation metrics (i.e. *Macro F-measure* and *Macro AUC*). Let $\mathcal{S} = \{(\mathbf{x}_i, Y_i) \mid 1 \le i \le M\}$ be a test data set with M test examples, $\mathcal{Y} = \{1, 2, 3, \ldots, m\}$ be a label set with totally m labels, and h be the learned multi-label model. Given a test example $\mathbf{x}_i$ the label set predicted by h is represented by $h(\mathbf{x}_i)$, and $f(\mathbf{x}_i, y)$ is a real-valued function that concerns the ranking quality of label y for data example $\mathbf{x}_i$. The function $r_f(\mathbf{x}_i, y)$ maps the output of $f(\mathbf{x}_i, y)$ to one element in the set $\{1, 2, 3, \ldots, m\}$ for a test example $\mathbf{x}_i$. The six evaluation metrics utilized in our experiments are presented as follows:

- *Hamming loss*

$$hloss = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{m}|h(\mathbf{x}_i)\Delta Y_i|$$

Here, $h(\mathbf{x}_i)$ is the predicted label set for test example $\mathbf{x}_i$, Y_i is the true label set of $\mathbf{x}_i$, and Δ stands for the symmetric difference between two sets. For *Hamming loss*, the smaller the better.

- *Example based F-measure*

$$F^1_{exam} = \frac{1}{M}\sum_{i=1}^{M}\frac{2|Y_i \cap h(\mathbf{x}_i)|}{|Y_i| + |h(\mathbf{x}_i)|}$$

For *Example based F-measure*, the bigger the better.

- *Ranking loss*

$$rloss = \frac{1}{M}\sum_{i=1}^{M}\frac{|L|}{|Y_i||\bar{Y}_i|}$$

Here, $L = \{(y_1, y_2) \mid f(\mathbf{x}_i, y_1) \le f(\mathbf{x}_i, y_2), (y_1, y_2) \in Y_i \times \bar{Y}_i\}$, and $\bar{Y}_i$ is the complementary set of Y_i. For *Ranking loss*, the smaller the better.

- *Average precision*

$$avepre = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{|\Re(\mathbf{x}_i, y)|}{r_f(\mathbf{x}_i, y)}$$

Here, $\Re(\mathbf{x}_i, y) = \{y' \mid r_f(\mathbf{x}_i, y') \le r_f(\mathbf{x}_i, y), y' \in Y_i\}$. For *Average precision*, the bigger the better.

- *Macro F-measure*

$$F^1_{macro} = \frac{1}{m}\sum_{i=1}^{m}\frac{2TP_i}{2TP_i + FN_i + FP_i}$$

Here, TP_i is the number of true positive predictions for label i, FN_i is the number of false negative predictions for label i, and FP_i is the number of false positive predictions for label i. For *Macro F-measure*, the bigger the better.

- *Macro AUC*

$$AUC_{macro} = \frac{1}{m}\sum_{i=1}^{m}AUC_i$$

Here, $AUC_i = \frac{|\{(\mathbf{x}^p, \mathbf{x}^n) | f(\mathbf{x}^p, y_i) \ge f(\mathbf{x}^n, y_i)), (\mathbf{x}^p, \mathbf{x}^n) \in \mathcal{P}_i \times \mathcal{N}_i\}|}{|\mathcal{P}_i||\mathcal{N}_i|}$, $\mathcal{P}_i$ is the set of positive examples of label i, and $\mathcal{N}_k$ is the set of negative examples of label i. For *Macro AUC*, the bigger the better.

4.3. Comparing algorithms and parameter setting

In this section, we introduce 11 state-of-the-art multi-label learning methods that are used for benchmark comparison. Seven well-established multi-label learning algorithms (i.e. BR-SVM, ML-*k*NN, CLR, RA*k*EL, ECC, IBLR and MLHN) and four algorithms (i.e. COCOA, ML-ROS, ML-RUS, and MLSMOTE) which are capable of dealing with class-imbalance issue in multi-label learning are included. Table 2 lists the comparing algorithms along with their parameter settings which are recommended by the author of the corresponding citation. Implementations of BR-SVM, ML-*k*NN, CLR, RA*k*EL, ECC, and

Table 5: Average classification results of TSMLHN and the comparing algorithms in terms of *Ranking loss* ↓.

data set	BR-SVM	ML-kNN	CLR	RAkEL	ECC	IBLR	COCOA	ML-ROS	ML-RUS	MLSMOTE	MLHN	TSMLHN
CAL500	0.497	**0.185**	0.196	0.442	0.206	0.442	0.204	0.445	0.441	0.398	0.198	0.193
emotions	0.282	0.179	**0.158**	0.204	0.165	0.170	0.166	0.170	0.176	0.184	0.172	0.161
medical	0.163	0.045	0.104	0.156	0.071	0.058	0.039	0.057	0.057	0.056	0.068	**0.031**
enron	0.304	0.088	0.085	0.287	0.146	0.111	**0.055**	0.109	0.114	0.109	0.114	0.103
scene	0.176	0.084	0.082	0.106	0.083	0.078	0.074	0.080	0.079	0.086	0.083	**0.070**
yeast	0.324	0.171	0.170	0.254	0.187	0.171	0.172	0.177	0.173	0.181	0.172	**0.162**
slashdot	0.288	0.113	0.265	0.276	0.158	0.111	0.093	0.112	0.112	0.111	**0.092**	0.097
corel5k	0.711	**0.126**	0.135	0.710	0.446	0.374	0.151	0.376	0.398	0.375	0.242	0.139
rcv1subset1	0.332	0.069	0.052	0.324	0.192	0.077	0.064	0.076	0.079	0.076	0.070	**0.040**
bibtex	0.368	0.121	0.059	0.366	0.317	0.190	**0.055**	0.191	0.202	0.156	0.144	0.132
eurlex-sm	0.269	0.024	0.125	0.260	0.172	0.058	**0.015**	0.056	0.062	0.056	0.044	0.017
tmc2007	0.204	0.053	0.042	0.188	0.043	0.048	0.035	0.049	0.048	0.049	0.032	**0.028**

Table 6: Average classification results of TSMLHN and the comparing algorithms in terms of *Average precision* ↑.

data set	BR-SVM	ML-kNN	CLR	RAkEL	ECC	IBLR	COCOA	ML-ROS	ML-RUS	MLSMOTE	MLHN	TSMLHN
CAL500	0.293	0.486	0.480	0.353	0.463	0.253	0.467	0.251	0.251	0.249	0.481	**0.494**
emotions	0.731	0.786	0.807	0.784	0.803	0.790	0.799	0.790	0.784	0.774	**0.812**	0.808
medical	0.682	0.841	0.435	0.698	0.820	0.814	0.860	0.815	0.808	0.818	0.854	**0.868**
enron	0.488	0.670	0.665	0.527	0.648	0.605	**0.715**	0.615	0.591	0.613	0.671	0.677
scene	0.767	0.856	0.850	0.845	0.855	0.864	0.869	0.860	0.862	0.856	0.858	**0.878**
yeast	0.666	0.757	0.758	0.714	0.758	0.758	0.759	0.752	0.756	0.746	0.761	**0.772**
slashdot	0.491	0.660	0.253	0.493	0.610	0.662	0.697	0.660	0.659	0.662	0.698	**0.704**
corel5k	0.061	0.296	0.254	0.061	0.241	0.129	0.271	0.129	0.125	0.133	**0.305**	0.282
rcv1subset1	0.321	0.615	0.605	0.340	0.527	0.587	0.434	0.589	0.579	0.588	**0.642**	0.622
bibtex	0.275	0.509	0.524	0.278	0.367	0.375	**0.623**	0.376	0.360	0.415	0.526	0.541
eurlex-sm	0.598	0.815	0.429	0.611	0.723	0.752	0.811	0.759	0.741	0.755	0.818	**0.824**
tmc2007	0.676	0.818	0.824	0.694	0.723	0.820	0.855	0.816	0.819	0.816	0.836	**0.864**

Table 7: Average classification results of TSMLHN and the comparing algorithms in terms of *Macro F-measure* ↑.

data set	BR-SVM	ML-kNN	CLR	RAkEL	ECC	IBLR	COCOA	ML-ROS	ML-RUS	MLSMOTE	MLHN	TSMLHN
CAL500	0.064	0.055	0.049	0.067	0.108	0.154	0.153	**0.173**	0.171	0.171	0.122	0.157
emotions	0.605	0.588	0.625	0.664	0.657	0.630	0.665	0.633	0.633	0.618	0.676	**0.679**
medical	0.426	0.383	0.207	0.428	0.448	0.419	0.465	0.452	0.405	0.443	0.423	**0.505**
enron	0.119	0.123	0.124	0.119	0.142	0.152	**0.223**	0.174	0.151	0.168	0.151	0.178
scene	0.680	0.710	0.686	0.733	0.722	0.738	0.725	0.735	0.730	0.723	0.719	**0.756**
yeast	0.329	0.356	0.340	0.363	0.369	0.373	0.428	0.383	0.370	0.380	0.419	**0.471**
slashdot	0.329	0.389	0.109	0.336	0.378	0.424	0.461	0.422	0.421	0.425	0.402	**0.482**
corel5k	0.075	0.077	0.063	0.075	0.089	0.055	0.106	0.046	0.053	0.051	0.081	**0.113**
rcv1subset1	0.188	0.237	0.194	0.186	0.228	0.250	0.298	0.249	0.247	0.248	0.301	**0.380**
bibtex	0.124	0.175	0.129	0.125	0.167	0.197	0.343	0.200	0.194	0.217	0.269	**0.351**
eurlex-sm	0.365	0.401	0.167	0.365	0.432	0.344	0.529	0.386	0.329	0.361	0.462	**0.566**
tmc2007	0.411	0.503	0.433	0.414	0.424	0.526	0.646	0.540	0.527	0.527	0.511	**0.658**

IBLR are obtained from Mulan [37] which provides an open-source Java code of various multi-label learning algorithms. The source code of the COCOA [2] method is provided by authors of [18]. For TSMLHN, the learning rate η is set to be 0.01, the number of nearest neighbors is set to be 20, and the threshold θ used to divide labels into common labels and imbalanced labels is set for each data set, respectively. The approach to the parameter setting of TSMLHN is presented later (in section 4.4.3). In detail, for multi-label data sets with low average imbalance ratio (i.e. emotions, scene, and yeast) the threshold θ is set to be 2.0, while the value of θ for multi-label data sets with high average imbalance ration (i.e. CAL500, medical, enron, slashdot, corel5k, rcv1subset1, bibtex, eurlex-sm, and tmc2007) is set to be 10.

[2]available at http://www.cse.seu.edu.cn/PersonalPage/zhangml/

4.4. Experiments

In order to assess the benefits of TSMLHN, an extensive experimental study was conducted. Firstly, we present the classification results of TSMLHN and the comparing algorithms over 12 multi-label data sets. Secondly, we compare the effectiveness of each learning stage of TSMLHN, as two learning stages are included in TSMLHN model. Thirdly, we compare the effectiveness of TSMLHN with that of the comparing algorithms. Fourthly, we compare the runtime efficiency of TSMLHN and the comparing algorithms. Finally, the parameter setting and parameter sensitivity of TSMLHN are discussed.

Table 8: Average classification results of TSMLHN and the comparing algorithms in terms of *Macro AUC* ↑.

data set	BR-SVM	ML-kNN	CLR	RAkEL	ECC	IBLR	COCOA	ML-ROS	ML-RUS	MLSMOTE	MLHN	TSMLHN
CAL500	0.504	0.514	0.553	0.510	0.529	0.510	0.548	0.509	0.509	0.512	0.537	**0.558**
emotions	0.729	0.808	0.825	0.801	0.825	0.828	0.835	0.829	0.824	0.813	0.831	**0.859**
medical	0.644	0.764	0.808	0.648	0.711	0.755	**0.830**	0.749	0.753	0.755	0.723	0.771
enron	0.535	0.633	**0.695**	0.539	0.579	0.640	0.668	0.642	0.638	0.642	0.624	0.659
scene	0.796	0.923	0.916	0.881	0.909	0.937	0.938	0.934	0.935	0.925	0.913	**0.957**
yeast	0.563	0.670	0.644	0.601	0.630	0.687	0.702	0.683	0.681	0.680	0.679	**0.723**
slashdot	0.599	0.752	0.678	0.609	0.705	0.771	**0.811**	0.749	0.760	0.769	0.755	0.802
corel5k	0.508	0.576	**0.703**	0.510	0.541	0.554	0.608	0.558	0.549	0.592	0.554	0.612
rcv1subset1	0.562	0.758	**0.908**	0.572	0.652	0.736	0.892	0.731	0.734	0.741	0.838	0.897
bibtex	0.549	0.789	**0.916**	0.552	0.619	0.809	0.914	0.809	0.805	0.843	0.861	0.908
eurlex-sm	0.638	0.850	**0.903**	0.634	0.732	0.821	0.898	0.812	0.803	0.832	0.802	0.893
tmc2007	0.659	0.881	0.901	0.675	0.695	0.909	**0.919**	0.906	0.908	0.906	0.886	0.915

4.4.1. Experimental results

In this section, we present the classification results of TSMLHN and the comparing algorithms. In our experiments, on each data set 50 percent data examples are randomly selected to be the training data and the rest 50 percent data examples are used as test data. The sampling process is repeated ten times and the average classification results across ten train/test trails are reported. The experimental results in terms of six evaluation metrics are reported in Tables 3, 4, 5, 6, 7, and 8, respectively, where the best result on each data is shown in boldface and the symbol "↓" indicates the lower the better while "↑" means the higher the better.

Table 9: Summary of *Wilcoxon signed-rank test* for TSMLHN and MLHN

evaluation metric	p-value	TSMLHN vs. MLHN
Hamming loss	0.5703	TSMLHN ≈ MLHN
Example based F-measure	0.3906	TSMLHN ≈ MLHN
Ranking loss	0.0020	TSMLHN ≻ MLHN
Average precision	0.1821	TSMLHN ≈ MLHN
Macro F-measure	4.8828×10^{-4}	TSMLHN ≻ MLHN
Macro AUC	4.8828×10^{-4}	TSMLHN ≻ MLHN

4.4.2. Effectiveness of each learning stage of TSMLHN

In our proposed TSMLHN model, we introduce a two-stage learning algorithm into the learning process of MLHN to deal with the class-imbalance problem. In the first stage, a multi-label hypernetwork (MLHN) is trained. In the second stage, correlations between common labels and imbalanced labels are explicitly utilized to improve the learning performance of imbalanced labels. In order to verify the effectiveness of each stage, we compare the classification performance of TSMLHN with that of the MLHN. In Tables 3, 4, 5, 6, 7, and 8, the results of MLHN represent the classification results of the first stage learning of TSMLHN. In order to verify whether TSMLHN has statistically significant advantages over MLHN, *Wilcoxon signed-rank test* at significant level 0.05 is employed. Table 9 summarizes the *Wilcoxon signed-rank test* on the classification results of TSMLHN and MLHN in terms of six evaluation metrics, where "TSMLHN ≻ MLHN" means TSMLHN significantly outperforms MLHN while "TSMLHN ≈ MLHN" means TSMLHN and MLHN obtain similar performance. From Table 9, we observe that TSMLHN significantly outperforms MLHN in terms of *Ranking loss* and two imbalance-specific evaluation metrics (i.e. *Macro F-measure* and *Macro AUC*) while obtains similar performance in terms of three common multi-label evaluation metrics (i.e. *Hamming loss*, *Example based F-measure*, and *Average precision*). From the experimental results shown above, it is reasonable to draw a conclusion that the two-stage learning of TSMLHN can significantly improve the learning performance of MLHN for imbalanced labels.

4.4.3. Effectiveness of TSMLHN and the comparing algorithms

In order to verify the effectiveness of TSMLHN in addressing class-imbalance problem in multi-label learning, we compare the classification performance of TSMLHN with that of the comparing algorithms. Since a number of algorithms are compared over a number of experimental data sets, we employ the *Friedman test* [38] to present performance analysis among the comparing algorithms systematically. Given J comparing algorithms and K data sets, let r_k^j be the rank of j-th algorithm on the k-th data set, $R_j = \frac{1}{K}\sum_{k=1}^{K} r_k^j$ be the average rank for the j-th algorithm over K data sets, and the null hypothesis be that all algorithms have "equal" performance. The Friedman statistic F_F is calculated as following:

$$F_F = \frac{(K-1)\chi_F^2}{K(J-1)-\chi_F^2},$$

$$\chi_F^2 = \frac{12K}{J(J+1)}\left[\sum_{j=1}^{J} R_j^2 - \frac{J(J+1)^2}{4}\right]$$

Table 10: Summary of the Friedman statistics F_F ($J = 11, K = 12$) and critical value in terms of each evaluation metric

evaluation metric	F_F	critical value ($\alpha = 0.05$)
Hamming loss	3.463	1.918
Example based F-measure	13.129	
Ranking loss	25.617	
Average precision	18.754	
Macro F-measure	16.776	
Macro AUC	33.050	

Since TSMLHN has been compared with MLHN in previous

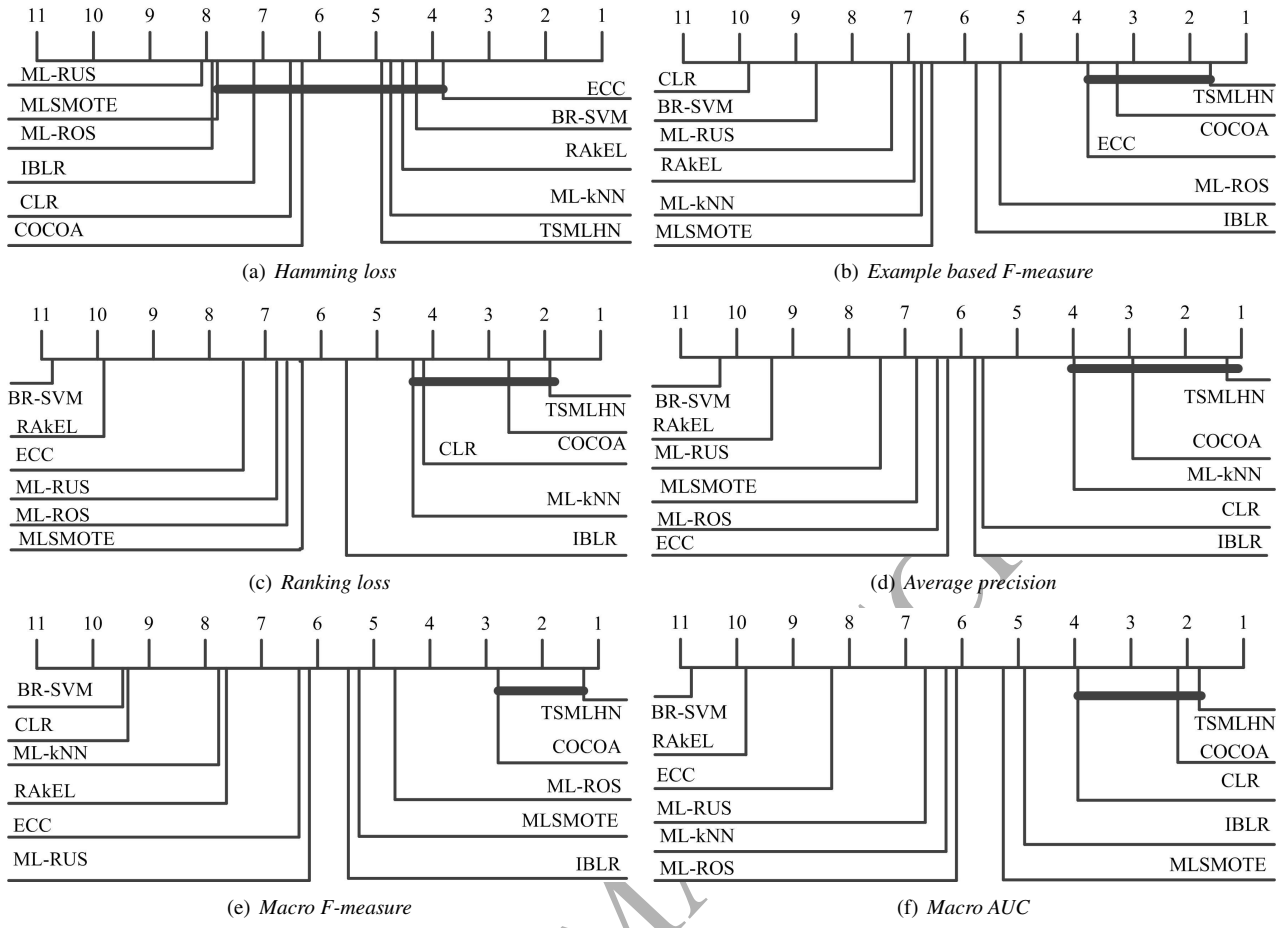


Figure 4: Average ranks diagram of TSMLHN and the comparing algorithm in terms of each evaluation metric.

section, in the section, we compare the classification effectiveness of TSMLHN with that of the comparing algorithms except for MLHN. Thus, the number of comparing algorithms is 11 (i.e. J=11) and the number of data sets is 12 (i.e. K=12). Table 10 summarizes the Friedman statics and corresponding critical value on each evaluation metric. From Table 10, we can see that the null hypothesis that all algorithms have "equal" performance is rejected in terms of each evaluation metric. Consequently, a *post-hoc* test of *Friedman test* is needed to further analysis the relative performance among comparing algorithms. In this paper, the *Bonferroni-Dunn test* is used as the *post-hoc* test of *Friedman test*. In detail, we treat our proposed TSMLHN as the control algorithm, and the performance difference between TSMLHN and one comparing algorithm is significant if their average ranks differ at least one *CD* (*critical difference*). The calculation is *CD* is shown as following:

$$CD = q_\alpha \sqrt{\frac{J(J+1)}{6K}}$$

For *Bonferroni-Dunn test*, we have $q_\alpha = 2.773$ at significance level $\alpha = 0.05$ and thus CD is $3.428(J = 11, K = 12)$. Fig. 4 plots the average ranks diagram of TSMLHN and the comparing algorithm, where a thick line is connected to TSMLHN if the difference between the average ranks of TSMLHN and the corresponding comparing algorithm is less than one *CD*, no line is connected otherwise, and the best algorithm is located at the rightmost side.

Since TSMLHN is aimed at addressing the class-imbalance in multi-label learning, we proceed with experiments to show how TSMLHN works under different levels of imbalance ratios. Specifically, we firstly select four data sets (enron, corel5k, rcv1subset1, and eurlex-sm) with high average imbalance ratio. Given one multi-label data set, for the sake of simplicity, we only consider classification performance of labels whose imbalance ratio is less than 50, and group the imbalance ratios of the these labels into five intervals, i.e. $I_1 = [1, 5], I_2 = [5, 10], I_3 = [10, 15], I_4 = [15, 25]$ and $I_5 = [25, 50]$. Average performance of TSMLHN over class labels in each interval is compared with that of the comparing algorithms. On each data set, let A_k, B_k denote the average performance of TSMLHN and a comparing algorithm over the labels falling into interval I_k, respectively. A performance gain $C_k = [(A_k - B_k)/B_k] \times 100\%$ is computed to reflect the relative performance between TSMLHN and the comparing algorithms within interval I_k. Fig. 5 and Fig. 6 illustrate the performance gain of TSMLHN over the comparing algorithms in terms of *Macro F-measure* and *Macro AUC*, respectively.

Based on above experimental results, following observations

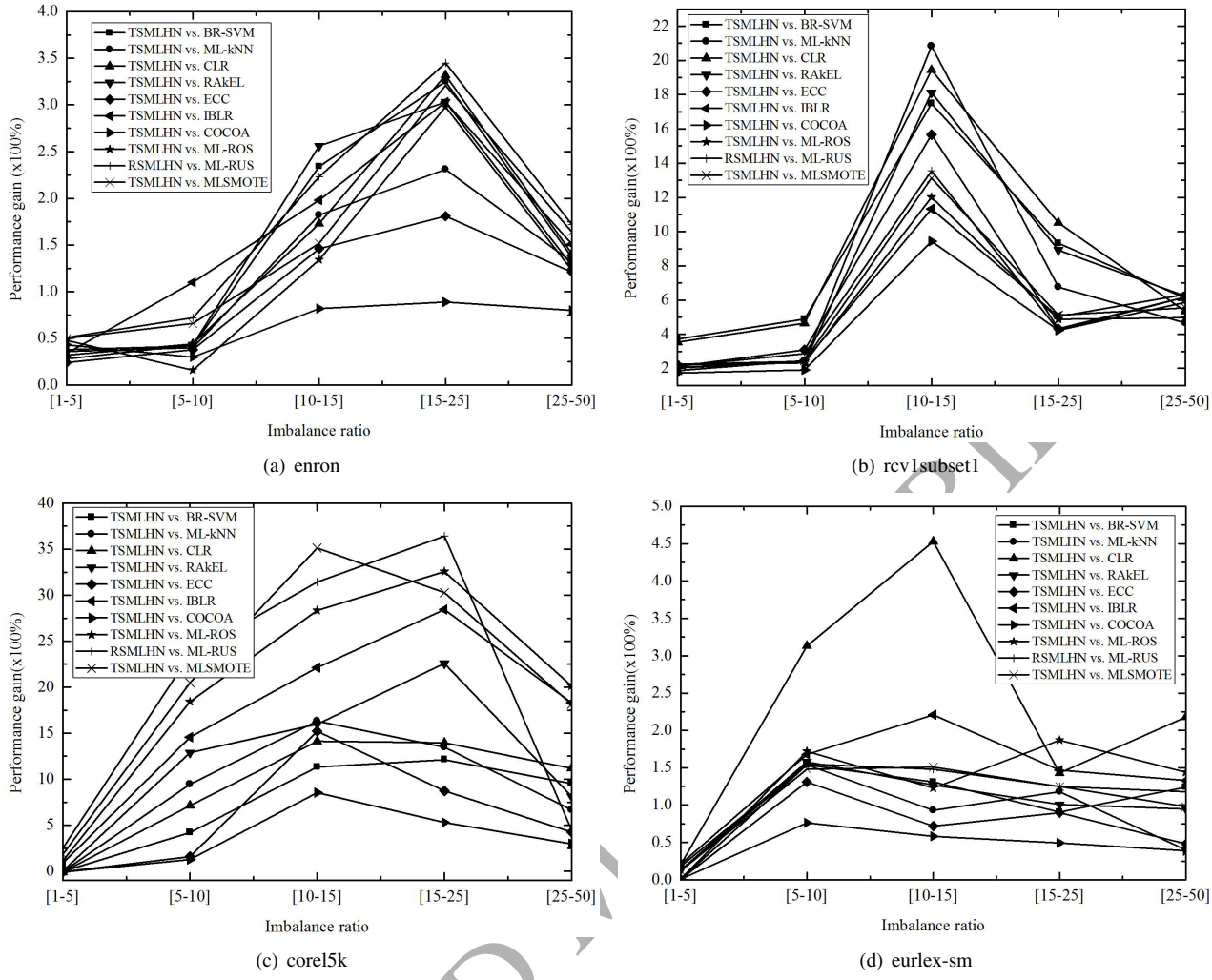


Figure 5: Performance gain C_k in terms of *Macro F-measure* between TSMLHN and the comparing algorithms under different levels of imbalance ratio I_k.

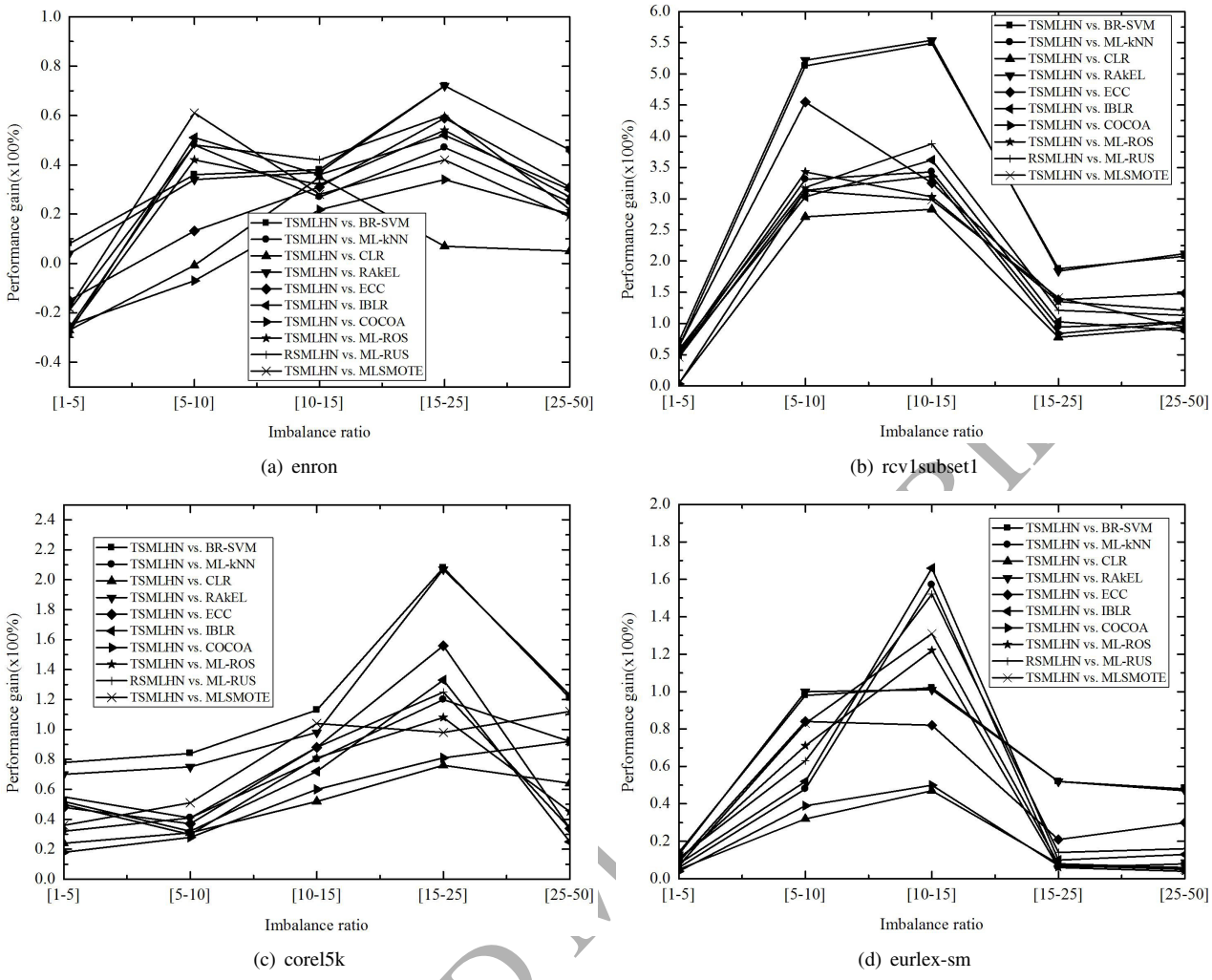
can be made:

- The performance of TSMLHN and the comparing algorithms does not show significant difference in terms of *Hamming loss* (see Fig. 4(a) and Table 3).
- As shown in Fig. 4, TSMLHN outperforms most of the classic multi-label learning algorithms (i.e. BR-SVM, ML-*k*NN, CLR, RA*k*EL, IBLR, and ECC) in terms of three common evaluation metrics (i.e. *Example based F-measure*, *Ranking loss*, and *Average precision*) and two class-imbalance sensitive evaluation metrics (i.e. *Marco F-measure* and *Macro AUC*) while obtains superior or at least comparable performance against the classic multi-label learning algorithms in terms of *Hamming loss*.
- As shown in Fig. 4, TSMLHN significantly outperforms three re-sampling technique based class-imbalance sensitive multi-label learning algorithms in terms of *Macro F-measure* and *Macro AUC*.
- As shown in Fig. 4, although no significant difference is shown on the performance of TSMLHN and COCOA, TSMLHN achieves superior performance against COCOA in terms of all evaluation metrics (TSMLHN is on the right side of COCOA).
- TSMLHN ranks *1st* in 83.3% cases on all data sets and ranks *1st* in 85.7% cases on high imbalanced data sets (i.e. data sets whose *avg(ImR)* is bigger than 100) in terms of *Macro F-measure* (see Table 7).
- TSMLHN achieves high performance gain across relatively high imbalance levels ($I_2 = [5, 10], I_3 = [10, 15], I_4 = [15, 25]$), however, the advantages of TSMLHN over the comparing algorithms are not so significant when the imbalance level becomes extremely high (Fig. 5 and Fig. 6).

To summarize, TSMLHN achieves competitive performance against other well-established multi-label learning algorithms and class-imbalance specific multi-label learning algorithms.

4.4.4. *Efficiency of TSMLHN and the comparing algorithms*

In order to verify the efficiency of TSMLHN in class-imbalanced multi-label learning, we compare the training and



(a) enron (b) rcv1subset1

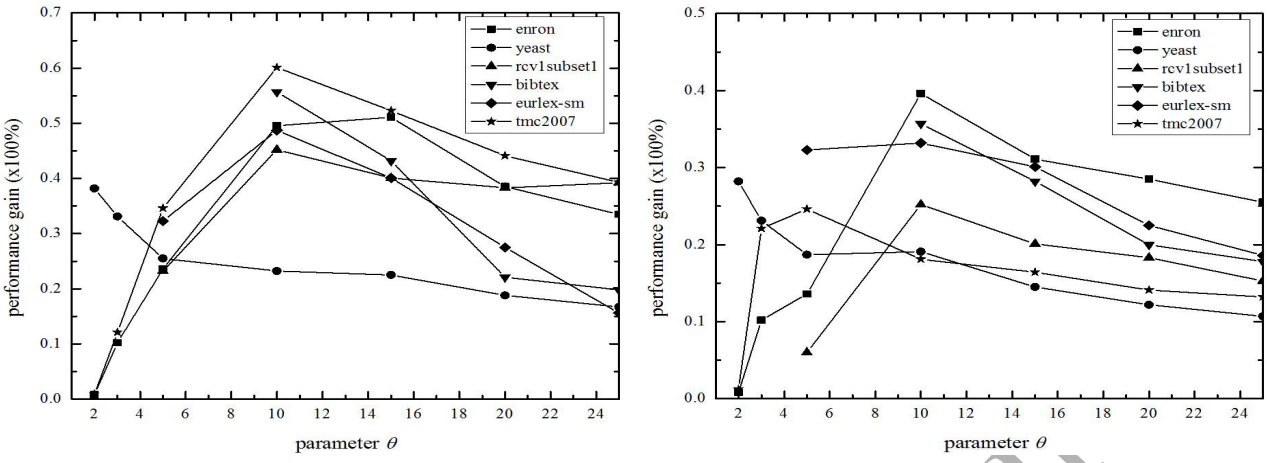
(c) corel5k (d) eurlex-sm

Figure 6: Performance gain C_k in terms of *Macro AUC* between TSMLHN and the comparing algorithms under different levels of imbalance ratio I_k.

Table 11: Training and testing time (in seconds) of TSMLHN and four comparing algorithms

data set		ML-RUS	ML-ROS	MLSMOTE	COCOA	TSMLHN
CAL500	training time	8.096	11.107	274.638	214.855	**2.437**
	testing time	0.547	0.563	0.593	3.051	**0.085**
enron	training time	**3.416**	4.386	4.166	105.928	5.143
	testing time	**0.687**	0.703	0.695	2.314	0.843
yeast	training time	**1.575**	2.262	2.075	313.011	5.830
	testing time	1.608	1.717	1.686	1.999	**0.745**
corel5k	training time	378.76	464.904	485.912	7088.140	**72.437**
	testing time	24.462	26.834	26.241	53.466	**8.867**
bibtex	training time	246.981	198.165	1439.262	3867.742	**165.753**
	testing time	**11.857**	13.186	19.653	34.945	12.480
eurlex-sm	training time	721.231	1090.044	1326.967	23154.955	**553.815**
	testing time	**59.435**	68.954	97.348	113.532	64.368

testing runtime efficiency of TSMLHN and the comparing algorithms. Specifically, we compare the training and testing time of TSMLHN to that of four class-imbalance specific multi-label learning algorithms (i.e. ML-RUS, ML-ROS, MLSMOTE and COCOA) on six data sets (i.e CAL500, enron, yeast, corel5k, bibtex and eurlex-sm), on each data set 50 percent of the data examples are selected as training data and the rest are used as testing data. Table 11 reports the training and testing time(in seconds) of TSMLHN and four comparing algorithms. The least training and testing time on each data set is marked in



(a) performance gain in terms of *Macro F-measure* (b) performance gain in terms of *Macro AUC*

Figure 7: Performance gain C_k between TSMLHN and MLHN in the first stage under different setting of parameter θ.

bold face.

From Table 11, we can get following observations:

- Among the comparing algorithms, COCOA is the most time-consuming, especially when the number of labels is large (see the training and testing time on corel5k, bibtex and eurlex-sm in Table 11). The efficiency of COCOA suffers from its model complexity as it builds one binary predictive model for each label and several multi-class imbalance learners.

- TSMLHN performs similarly to the re-sampling based methods (i.e. ML-RUS, ML-ROS and MLSMOTE) in terms of testing time. However, TSMLHN is more efficient than re-sampling based methods on data sets with a large number of labels in terms of training time (see training time on CAL500, corel5k, bibtex and eurlex-sm in Table 11). The efficiency of re-sampling strategy is related to two factors, i.e the computational complexity of base learner and the time used for re-sampling. As IBLR is used as the base learner for re-sampling strategy in this paper, the runtime efficiency benefits from the model simplicity of lazy learning. However, when both the size of training data and the number of labels become large a lot of time is consumed for re-sampling, as data examples are independently re-sampled for each imbalance label.

- TSMLHN is efficient and scalable because it builds only one prediction model for all labels and do not need to re-sample the training data. The runtime efficiency of TSMLHN benefits from its linear computational complexity with respect to the number of labels (see Section 3.4).

To summarize, TSMLHN is efficient in addressing class-imbalance in multi-label learning, especially when the number of labels becomes large.

4.4.5. Parameter setting and sensitivity analysis

In our proposed TSMLHN, θ is an important parameter. In this section, we discuss how to set the value of θ and study the parameter sensitivity of TSMLHN with respect to θ. For the setting of parameter θ, on each data set, we randomly select 10 percent of the training data to form a validation set, and confine the possible values of θ to an interval [1, 25] with step 1, for each value in the interval, we compare the classification performance of TSMLHN with that of MLHN in the first stage in terms of *Macro F-measure* and *Macro AUC*. Specifically, let A_k denote the classification performance of TSMLHN and B_k denote the classification performance of MLHN in the first stage on the validation data under parameter setting θ_k, a performance gain $C_k = [(A_k - B_k)/B_k] \times 100\%$ is calculated to indicate how many performance improvements can be obtained by utilizing the TSMLHN method. Finally, the value that produces the highest performance gain is selected as the parameter setting of θ. Fig. 7 illustrates the performance gain of TSMLHN over MLHN in the first stage under different settings of θ on six multi-label data sets (i.e. enron, yeast, rcv1subset1, bibtex, eurlex-sm and tmc2007). We note that in most cases the performance gain of TSMLHN grows as the value of θ increases but when the value of θ becomes high (i.e. $\theta = 10$ in Fig. 7) the performance gain drops with the increasing of θ. We think that this phenomenon can be explained by the characteristic of TSMLHN that if θ is set to be too small many common labels are treated as imbalanced labels while if θ is set to be too big many imbalanced labels are viewed as common labels. In both cases, the correlations between common labels and imbalanced are not fully utilized.

5. Conclusions

In this paper, we propose a TSMLHN method to deal with the class-imbalance problem in multi-label learning. In TSMLHN, class labels are divided into two groups, i.e. common labels and imbalanced labels based on their imbalance ratios, and the correlations between common labels and imbalanced labels are used to improve the learning performance of imbalanced labels. Experimental results on a variety of multi-label data sets demonstrate that TSMLHN achieves competitive results against

state-of-the art multi-label learning approaches in terms of both common multi-label evaluation metrics and imbalance-specific evaluation metrics. Through experimental studies, we also note that TSMLHN significantly outperforms the comparing algorithms over moderately imbalanced labels while obtain competitive performance against the comparing algorithms over extremely imbalanced labels. This may indicate that label correlations are beneficial to addressing the class-imbalance problem in multi-label scenario but it might be insufficient to use correlations among labels merely to learn extremely imbalanced labels. In the future, we will focus on combining label correlations with other strategies (e.g. re-sampling techniques) to improve the learning performance of extremely imbalanced labels.

Acknowledgments

This work was supported by INHA UNIVERSITY Research Grant.

References

[1] I. Katakis, G. Tsoumakas, I. Vlahavas, Multi-label text classification for automated tag suggestion, in: Proceedings of the ECML/PKDD-08 Workshop on Discovery Challenge, 2008, pp. 75-83. doi:10.1.1.183.2636.

[2] J.Y. Jiang, S.C. Tsai, S.J. Lee, FSKNN: Multi-label text categorization based on fuzzy similarity and k nearest neighbors, Expert System with Application 39 (3) (2012) 2813-2821. http://dx.doi.org/10.1016/j.eswa.2011.08.141.

[3] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognition 37 (9) (2004) 1757-1771. http://dx.doi.org/10.1016/j.patcog.2004.03.009.

[4] R.S. Cabral, F. De la Torre, J.P. Costeira, A. Bernardino, Matrix completion for multi-label image classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (01) (2015) 121-135. doi:10.1109/TPAMI.2014.2343234.

[5] Z. Barutcuoglu, R.E. Schapire, O.G. Troyanskaya, Hierarchical multi-label prediction of gene function, Bioinformatics 22 (7) (2006) 830-836. doi:10.1093/bioinformatics/btk048.

[6] N. Cesa-Bianchi, M. Re, G. Valentini, Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference, Machine Learning 88 (1/2) (2012) 209-241. doi:10.1007/s10994-011-5271-6.

[7] M.L. Zhang, Z.H. Zhou, ML-*k*NN: A lazy learning approach to multi-label learning, Pattern Recognition 40 (7) (2007) 2038-2048. http://dx.doi.org/10.1016/j.patcog.2006.12.019.

[8] W.W. Cheng, E. Hullermeier, Combining instance-based learning and logistic regression for multi-label classification, Machine Learning 76 (2) (2009) 211-225. doi:10.1007/s10994-009-5127-5

[9] J. Furnkranz, E. Hullermeier, E. Loza Mencia, K. Brinker, Multi-label classification via calibrate label ranking, Machine Learning 73 (2) (2008) 133-153. doi:10.1007/s10994-008-5064-8.

[10] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, in: Lecture Notes in Artificial Intelligence 5782, (2009), pp. 254-269. doi:10.1007/978-3-642-04174-7-17.

[11] G. Tsoumakas, I. Katakis, I. Vlahavas, Random *k*-labelset for multi-label classification, IEEE transaction on Knowledge and Data Engineering 23 (7) (2011) 1079-1089. doi: 10.1109/TKDE.2010.164.

[12] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321-357. doi:10.1613/jair.953.

[13] X.Y. Liu, J.X. Wu, Z.H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE transaction on System, Man, and Cybernetics-Part B: Cybernetics 39 (2) (2009) 539-550. doi:10.1109/TSMCB.2008.2007853.

[14] Y.M. Sun, A.K.C. Wong, M.S. Kamel, Classification of imbalanced data: A review, International Journal of Pattern Recognition and Artificial Intelligence 23 (4) (2009) 687-719. http://dx.doi.org/10.1142/S0218001409007326.

[15] C. Seiffert, T.M. Khoshgoftaar, J.V. Hulse, A. Napolitano, RUSBoost: A hybrid approach to alleviating class imbalance, IEEE transaction on System, Man, and Cybernetics-Part A: Systems and Humans 40 (1) (2010) 185-197. doi: 10.1109/TSMCA.2009.2029559.

[16] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, Addressing imbalance in multi-label classification: measures and random resampling algorithms, Neurocomputing, 163 (2015) 3-16. http://dx.doi.org/10.1016/j.neucom.2014.08.091.

[17] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, MLSMOTE: Approaching imbalanced multi-label learning through synthetic instance generation, Knowledge based Systems 89 (2015) 385-397. http://dx.doi.org/10.1016/j.knosys.2015.07.019.

[18] M.L. Zhang, Y.K. Li, X.Y. Liu, Towards class-imbalance aware multi-label learning, in: Proceeding of the 24th International Joint Conference on Artificial Intelligence, 2015, pp. 4041-4047.

[19] M. Fang, Y.Q. Xiao, C.J. Wang, J.Y. Xie, Multi-label classification: dealing with imbalance by combing labels, in: IEEE 26th International Conference on Tools with Artificial Intelligence, 2014, pp. 233-237. doi:10.1109/ICTAI.2014.42.

[20] K.W. Sun, C.H. Lee, X.F. Xie, MLHN: A hypernetwork model for multi-label classification, International Journal of Pattern Recognition and Artificial Intelligence 29 (6) (2015) 1550020. http://dx.doi.org/10.1142/S0218001415500202.

[21] K.W. Sun, C.H. Lee, J. Wang, Multi-label classification via co-evolutionary multi-label hypernetwork, IEEE Transaction on Knowledge and Data Engineering 28 (9) (2016) 2438-2451. doi:10.1109/TKDE.2016.2566621.

[22] M.L. Zhang, Z.H. Zhou, Multi-label learning by exploiting label dependency, in: Proceeding of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 999-1008. doi:10.1145/1835804.1835930.

[23] M.L. Zhang, Z.H. Zhou, A review on multi-label learning algorithms, IEEE Transaction on Knowledge and Data Engineering 26 (8) (2014) 1819-1837. doi: 10.1109/TKDE.2013.39.

[24] A. Clare, R.D. King, Knowledge discovery in multi-label phenotype data, in: Lecture Notes in Computer Science 2168, 2001, pp. 42-53. doi:10.1007/3-540-44794-6-4.

[25] A. Elisseeff, J. Weston, A kernel method for multi-labeled classification, in: Advances in Neural Information Processing Systems 14, 2002, pp. 681-687. doi:10.1.1.18.2423.

[26] N. Ghamrawi, A.McCallum, Collective multi-label classification, in: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, 2005, pp. 195-200. doi: 10.1145/1099554.1099591.

[27] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Machine Learning 85 (3) (2011) 333-359. doi:10.1007/s10994-011-5256-5.

[28] L. Sun, S.W. Ji, J.P. Ye, Multi-label dimensionality reduction, Chapman and Hall/CRC, 2013.

[29] M.L. Zhang, L. Wu, Lift: Multi-label learning with label-specific features, IEEE Transaction on Pattern Analysis and Machine Intelligence 37 (1) (2015) 107-120. doi:10.1109/TPAMI.2014.2339815.

[30] E.S. Xioufis, M. Spiliopoulou, G. Tsoumakas, I. Vlahavas, Dealing with concept drift and class imbalance in multi-label stream classification, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, 2011, pp. 1583-1588. doi:10.5591/978-1-57735-516-8/IJCAI11-266.

[31] B.T. Zhang, Hypernetworks: A molecular evolutionary architecture for cognitive learning and memory, IEEE Computational Intelligence Magazine 3 (3) (2008) 49-63. doi:10.1109/MCI.2008.926615.

[32] S. Kim, S.J. Kim, B.T. Zhang, Evolving hypernetwork classifiers for microRNA expression profile analysis, in: Proceeding of IEEE Congress on Evolutionary Computation, 2007, pp. 313-319. doi:10.1109/CEC.2007.4424487.

[33] C.H. Park,S.J. Kim, S. Kim, D.Y. Cho, B.T. Zhang, Use of evolutionary hypernetworks for mining prostate cancer data, in: Proceedings of the 8th Symposium on Advanced Intelligent Systems, 2007, pp. 702-706.

doi:10.1.1.63.7761.

[34] B.J. Lee, J.W. Ha, K.M. Kim, B.T. Zhang, Evolutionary concept learning from cartoon videos by multimodal hypernetworks, in: Proceeding of IEEE Congress on Evolutionary Computation, 2013, pp. 1186-1192. doi:10.1109/CEC.2013.6557700.

[35] J. Wang, P.L. Huang, K.W. Sun, B.L. Cao, R. Zhao, Ensemble of cost-sensitive hypernetworks for class-imbalance learning, in: IEEE International Conference on System, Man, and Cybernetics, 2013, pp. 1883-1888. doi:10.1109/SMC.2013.324.

[36] S.J. Lee, J.Y. Jiang, Multi-label text categorization based on fuzzy relevance clustering. IEEE Transaction on Fuzzy System 22 (6) (2014) 1457-1471. doi:10.1109/TFUZZ.2013.2294355.

[37] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, I. Vlahavas, MULAN: A Java library for multi-label learning, Journal of Machine Learning Research (12) (2011) 2411-2414. doi:10.1.1.415.6577.

[38] J. Demsar, Statistical comparison of classifiers over multiple data sets, Journal of Machine Learning Research (7) (2006) 1-30. doi:10.1.1.141.3142.

Kai Wei Sun received the B.S. and M.S. from Department of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China, in 2010 and 2013 respectively. He is currently working toward the PhD degree in the Department of Information and Communication Engineering, Inha University, Incheon, Korea. His current research interests include machine learning, data mining and pattern recognition.

Chong Ho Lee received the B.S. and M.S. degrees in Electrical Engineering from Seoul National University, Korea, in 1976 and 1978, respectively, and the PhD degree in Computer and Information Technology from Iowa State University, USA, in 1986. He is currently a professor with the Department of Information and Communication Engineering, Inha University.