Synthetic Oversampling of Multi-Label Data based on Local Label Distribution

Bin Liu ✉ and Grigorios Tsoumakas

School of Informatics
Aristotle University of Thessaloniki
Thessaloniki 54124, Greece
{binliu,greg}@csd.auth.gr

Abstract. Class-imbalance is an inherent characteristic of multi-label data which affects the prediction accuracy of most multi-label learning methods. One efficient strategy to deal with this problem is to employ resampling techniques before training the classifier. Existing multi-label sampling methods alleviate the (global) imbalance of multi-label datasets. However, performance degradation is mainly due to rare subconcepts and overlapping of classes that could be analysed by looking at the local characteristics of the minority examples, rather than the imbalance of the whole dataset. We propose a new method for synthetic oversampling of multi-label data that focuses on local label distribution to generate more diverse and better labeled instances. Experimental results on 13 multi-label datasets demonstrate the effectiveness of the proposed approach in a variety of evaluation measures, particularly in the case of an ensemble of classifiers trained on repeated samples of the original data.

Keywords: Multi-label learning · Class-imbalance · Synthetic oversampling · Local label distribution · Ensemble methods

1 Introduction

In multi-label data, each example is typically associated with a small number of labels, much smaller than the total number of labels. This results in a sparse label matrix, where a small total number of positive class values is shared by a much larger number of example-label pairs. From the viewpoint of each separate label, this gives rise to class imbalance, which has been recently recognized as a key challenge in multi-label learning [6, 7, 11, 18, 31].

Approaches for handling class imbalance in multi-label data can be divided into two categories: a) reducing the imbalance level of multi-label data via resampling techniques, including synthetic data generation [5, 6, 7, 8], and b) making multi-label learning methods resilient to class imbalance [11, 18, 31]. This work focuses on the first category, whose approaches can be coupled with any multi-label learning method and are therefore more flexible.

Existing resampling approaches for multi-label data focus on class imbalance at the global scale of the whole dataset. However, previous studies of class imbalance in binary and multi-class classification [19, 20] have found that the distribution of class values in the local neighbourhood of minority examples, rather than the global imbalance level, is the main reason for the difficulty of a classifier to recognize the minority class. We hypothesize that this finding is also true, and even more important to consider, in the more complex setting of multi-label data, where it has not been examined yet.

Consider for example the 2-dimensional multi-label datasets (a) and (b) in Fig.1 concerning points in a plane. The points are characterized by three labels, concerning the shape of the points (triangles, circles), the border of the points (solid, none) and the color of the points (green, red). These datasets have the same level of label imbalance. Yet (b) appears much more challenging due to the presence of sub-concepts for the triangles and the points without border and the overlap of the green and red points as well as the points with solid and no border.

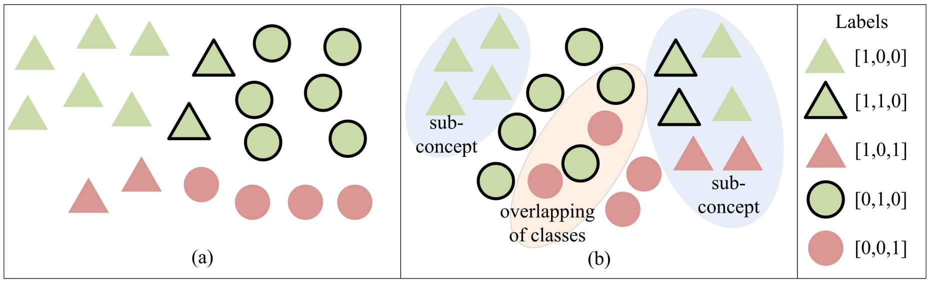

Fig. 1. Two 2-dimensional multi-label datasets (a) and (b) concerning points in a plane characterized by three labels. On the right we see the five different label combinations that exist in the datasets.

This work proposes a novel multi-label synthetic oversampling method, named MLSOL, whose seed instance selection and synthetic instance generation processes depend on the local distribution of the labels. This allows MLSOL to create more diverse and better labelled synthetic instances. Furthermore, we consider the coupling of MLSOL and other resampling methods with a simple but flexible ensemble framework to further improve its performance and robustness. Experimental results on 13 multi-label datasets demonstrate the effectiveness of the proposed sampling approach, especially its ensemble version, for three different imbalance-aware evaluation metrics and six different multi-label methods.

The remainder of this paper is organized as follows. Section 2 offers a brief review of methods for addressing class imbalance in multi-label data. Then, our approach is introduced in Section 3. Section 4 presents and discusses the experimental results. Finally, Section 5 summarizes the main contributions of this work.

2 Related Work

A first approach to dealing with class imbalance in the context of multi-label data is to utilize the resampling technique, which is applied in a pre-processing step and is independent of the particular multi-label learning algorithm that will be subsequently applied to the data. LP-RUS and LP-ROS are two twin sampling methods, of which the former removes instances assigned with the most frequent labelset (i.e. particular combination of label values) and the latter replicates instances whose labelset appears the fewest times [4].

Instead of considering whole labelset, several sampling methods alleviate the imbalance of the dataset in the individual label aspect, i.e. increasing the frequency of minority labels and reducing the number of appearances of majority labels. ML-RUS and ML-ROS simply delete instances with majority labels and clone examples with minority labels, respectively [7]. MLeNN eliminates instances only with majority labels and similar labelset of its neighbors in a heuristic way based on the Edited Nearest Neighbor (ENN) rule [5]. To make a multi-label dataset more balanced, MLSMOTE randomly selects instance containing minority labels and its neighbors to generate synthetic instances which are associated with labels that appear more that half times of the seed instance and its neighbors according to *Ranking* strategy [6].

REMEDIAL tackles the concurrence of labels with different imbalance level in one instance, of which the level is assessed by $SCUMBLE$, by decomposing the sophisticated instance of into two simpler examples, but may introduce extra confusions into the learning task, i.e. there are several pairs of instances with same features and disparate labels [8]. The REMEDIAL could be either a standalone sampling method or the prior part of other sampling techniques, i.e. RHwRSMT combines REMEDIAL with MLSMOTE [9].

Apart from resampling methods, another group of approaches focuses on multi-label learning method handling the class-imbalance problem directly. Some methods deal with the imbalance issue of multi-label learning via transforming the multi-label dataset to several binary/multi-class classification problems. COCOA converts the original multi-label dataset to one binary dataset and several multi-class datasets for each label, and builds imbalance classifiers with the assistance of sampling for each dataset [31]. SOSHF transforms the multi-label learning task to an imbalanced single label classification assignment via cost-sensitive clustering, and the new task is addressed by oblique structured Hellinger decision trees [11]. Besides, many approaches aims to modify current multi-label learning methods to handle class-imbalance problem. ECCRU3 extends the ECC resilient to class imbalance by coupling undersampling and improving of the exploitation of majority examples[18]. Apart from ECCRU3, the modified models based on neural network [26, 16, 23], SVM [3], hypernetwork [24] and BR [10, 12, 25, 28] have been proposed as well. Furthermore, other strategies, such as representation learning [17], constrained submodular minimization [29] and balanced pseudo-label [30], have been utilized to address the imbalance obstacle of multi-label learning as well.

3 Our Approach

We start by introducing our mathematical notation. Let $\mathcal{X} = \mathbb{R}^d$ be a d-dimensional input feature space, $L = \{l_1, l_2, ..., l_q\}$ a label set containing q labels and $\mathcal{Y} = \{0, 1\}^q$ a q-dimensional label space. $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) | 1 \leqslant i \leqslant n\}$ is a multi-label training data set containing n instances. Each instance $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ consists of a feature vector $\boldsymbol{x}_i \in \mathcal{X}$ and a label vector $\boldsymbol{y}_i \in \mathcal{Y}$, where y_{ij} is the j-th element of $\boldsymbol{y}_i$ and $y_{ij} = 1(0)$ denotes that l_j is (not) associated with i-th instance. A multi-label method learns the mapping function $h : \mathcal{X} \to \{0, 1\}^q$ and (or) $f : \mathcal{X} \to \mathbb{R}^q$ from D that given an unseen instance $\boldsymbol{x}$, outputs a label vector $\hat{\boldsymbol{y}}$ with the predicted labels of and (or) real-valued vector $\hat{\boldsymbol{fy}}$ corresponding relevance degrees to $\boldsymbol{x}$ respectively.

We propose a novel Multi-Label Synthetic Oversampling approach based on the Local distribution of labels (MLSOL). The pseudo-code of MLSOL is shown in Algorithm 1. Firstly, some auxiliary variables, as the weight vector $\boldsymbol{w}$ and type matrix $\boldsymbol{T}$ used for seed instance selection and synthetic examples generation respectively, are calculated based on the local label distribution of instances (line 3-6 in Algorithm 1). Then in each iteration, the seed and reference instances are selected, upon which a synthetic example is generated and added into the dataset. The loop (line 7-12 in Algorithm 1) would terminate when expected number of new examples are created. The following subsections detail the definition of auxiliaries as well as strategies to pick seed instances and create synthetic examples.

Algorithm 1: MLSOL

```
   input : multi-label data set: $D$, percentage of instances to generated: $P$,
           number of nearest neighbour: $k$
   output: new data set $D'$
1  $GenNum \leftarrow |D| * P$ ;                 /* number of instances to generate */
2  $D' \leftarrow D$ ;
3  Find the $k$NN of each instance ;
4  Calculate $\boldsymbol{C}$ according to Eq.(1) ;
5  Compute $\boldsymbol{w}$ according to Eq.(3) ;
6  $\boldsymbol{T} \leftarrow$ InitTypes($\boldsymbol{C}, k$) ;           /* Initialize the type of instances */
7  while $GenNum > 0$ do
8      Select a seed instance $(\boldsymbol{x}_s, \boldsymbol{y}_s)$ from $D$ based on the $\boldsymbol{w}$;
9      Randomly choose a reference instance $(\boldsymbol{x}_r, \boldsymbol{y}_r)$ from $kNN(\boldsymbol{x}_s)$ ;
10     $(\boldsymbol{x}_c, \boldsymbol{y}_c) \leftarrow$ GenerateInstance $((\boldsymbol{x}_s, \boldsymbol{y}_s), T_s, (\boldsymbol{x}_r, \boldsymbol{y}_r), T_r)$;
11     $D' \leftarrow D' \cup (\boldsymbol{x}_c, \boldsymbol{y}_c)$ ;
12     $GenNum \leftarrow GenNum - 1$ ;
13 return $D'$ ;
```

3.1 Selection of Seed Instances

We sample seed instances with replacement, with the probability of selection being proportional to the minority class values it is associated with, weighted by the difficulty of correctly classifying these values based on the proportion of opposite (majority) class values in the local neighborhood of the instance.

For each instance x_i we first retrieve its k nearest neighbours, $kNN(x_i)$. Then for each label l_j we compute the proportion of neighbours having opposite class with respect to the class of the instance and store the result in the matrix $\boldsymbol{C} \in \mathbb{R}^{n \times q}$ according to the following equation, where $[\![\pi]\!]$ is the indicator function that returns 1 if π is true and 0 otherwise:

$$C_{ij} = \frac{1}{k} \sum_{\boldsymbol{x}_m \in kNN(\boldsymbol{x}_i)} [\![y_{mj} \neq y_{ij}]\!] \tag{1}$$

The values in $\boldsymbol{C}$ range from 0 to 1, with values close to 0 (1) indicating a safe (hostile) neighborhood of similarly (oppositely) labelled examples. A value of $C_{ij} = 1$ can further be viewed as a hint that x_i is an outlier in this neighborhood with respect to l_j.

The next step is to aggregate the values in $\boldsymbol{C}$ per training example, x_i, in order to arrive at a single sampling weight, w_i, characterizing the difficulty in correctly predicting the *minority* class values of this example. A straightforward way to do this is to simply sum these values for the labels where the instance contains the minority class. Assuming for simplicity of presentation that the value 1 corresponds to the minority class, we arrive at this aggregation as follows:

$$w_i = \sum_{j=1}^{q} C_{ij} [\![y_{ij} = 1]\!] \tag{2}$$

There are two issues with this. The first one is that we have also taken into account the outliers. We will omit them by adding a second indicator function requesting C_{ij} to be less than 1. The second issue is that this aggregation does not take into account the global level of class imbalance of each of the labels. The fewer the number of minority samples, the higher the difficulty of correctly classifying the corresponding minority class. In contrast, Equation 2 treats all labels equally. To resolve this issue, we can normalize the values of the non-outlier minority examples in $\boldsymbol{C}$ so that they sum to 1 per label, by dividing with the sum of the values of all non-outlier minority examples of that label. This will increase the relative importance of the weights of labels with fewer samples. Addressing these two issues we arrive at the following proposed aggregation:

$$w_i = \sum_{j=1}^{q} \frac{C_{ij} [\![y_{ij} = 1]\!] [\![C_{ij} < 1]\!]}{\sum_{i=1}^{n} C_{ij} [\![y_{ij} = 1]\!] [\![C_{ij} < 1]\!]} \tag{3}$$

3.2 Synthetic Instance Generation

The definition of the type of each instance-label pair is indispensable for the assignment of appropriate labels to the new instances that we shall create. Inspired by [19], we distinguish minority class instances into four types, namely safe (SF), borderline (BD), rare (RR) and outlier (OT), according to the proportion of neighbours from the same (minority) class:

- $SF : 0 \leqslant C_{ij} < 0.3$. The safe instance is located in the region overwhelmed by minority examples.
- $BD : 0.3 \leqslant C_{ij} < 0.7$. The borderline instance is placed in the decision boundary between minority and majority classes.
- $RR : 0.7 \leqslant C_{ij} < 1$, and only if the type of its neighbours from the minority class are RR or OT. Otherwise there are some SF or BD examples in the proximity, which suggests that it could be rather a BD. The rare instance, accompanied with isolated pairs or triples of minority class examples, is located in the majority class area and distant from the decision boundary.
- $OT : C_{ij} = 1$. The outlier is surrounded by majority examples.

For the sake of uniform representation, the type of majority class instance is defined as majority (MJ). Let $\boldsymbol{T} \in \{SF, BD, RR, OT, MJ\}^{n \times q}$ be the type matrix and T_{ij} be the type of y_{ij}. The detailed steps of obtaining $\boldsymbol{T}$ are illustrated in Algorithm 2.

Once the seed instance $(\boldsymbol{x}_s, \boldsymbol{y}_s)$ has been decided, the reference instance $(\boldsymbol{x}_r, \boldsymbol{y}_r)$ is randomly chosen from the k nearest neighbours of the seed instance. Using the selected seed and reference instance, a new synthetic instance is generated according to Algorithm 3. The feature values of the synthetic instance $(\boldsymbol{x}_c, \boldsymbol{y}_c)$ are interpolated along the line which connects the two input samples (line 1-2 in Algorithm 3). Once $\boldsymbol{x}_c$ is confirmed, we compute $cd \in [0, 1]$, which indicates whether the synthetic instance is closer to the seed ($cd < 0.5$) or closer to the reference instance ($cd > 0.5$) (line 3-4 in Algorithm 3).

With respect to label assignment, we employ a scheme considering the labels and types of the seed and reference instances as well as the location of the synthetic instance, which is able to create informative instances for difficult minority class labels without bringing in noise for majority labels. For each label l_j, y_{cj} is set as y_{sj} (line 6-7 in Algorithm 3) if y_{sj} and y_{rj} belong to the same class. In the case where y_{sj} is majority class, the seed instance and the reference example should be exchanged to guarantee that y_{sj} is always the minority class (line 9-11 in Algorithm 3). Then, θ, a threshold for cd is specified based on the type of the seed label, T_{sj} (line 12-16 in Algorithm 3), which is used to determine the instance (seed or reference) whose labels will be copied to the synthetic example. For SF, BD and RR, where the minority (seed) example is surrounded by several majority instances and suffers more risk to be classified wrongly, the cut-point of label assignment is closer to the majority (reference) instance. Specifically, $\theta = 0.5$ for SF represents that the frontier of label assignment is in the midpoint between seed and reference instance, $\theta = 0.75$ for BD denotes that the range of minority class extends as three times as large than the majority

Algorithm 2: InitTypes

```
input  : The matrix storing proportion of kNNs with opposite class for each
         instance and each label: $\boldsymbol{C}$, number of nearest neighbour: $k$
output: types of instances $\boldsymbol{T}$
for $i \leftarrow 1$ to $n$ do /* $n$ is the number of instances */
    for $j \leftarrow 1$ to $q$ do /* $q$ is the number of labels */
        if $y_{ij} = majority\ class$ then
            $T_{ij} \leftarrow MJ$ ;
        else /* $y_{ij}$ is the minority class */
            if $C_{ij} < 0.3$ then $T_{ij} \leftarrow SA$ ;
            else if $C_{ij} < 0.7$ then $T_{ij} \leftarrow BD$ ;
            else if $C_{ij} < 1$ then $T_{ij} \leftarrow RR$ ;
            else $T_{ij} \leftarrow OT$ ;
repeat/* re-examine $RR$ type */
    for $i \leftarrow 1$ to $n$ do
        for $j \leftarrow 1$ to $q$ do
            if $T_{ij} = RR$ then
                foreach $\boldsymbol{x}_m$ in $kNN(\boldsymbol{x}_i)$ do
                    if $T_{mj} = SF$ or $T_{mj} = BD$ then
                        $T_{ij} \leftarrow BD$;
                        break ;
until no change in $\boldsymbol{T}$;
return $\boldsymbol{T}$ ;
```

class, and $\theta > 1 \geqslant cd$ for RR ensures that the generated instance is always set as minority class regardless of its location. With respect to OT as a singular point placed at majority class region, all possible synthetic instances are assigned the majority class due to the inability of an outlier to cover the input space. Finally, y_{cj} is set as y_{sj} if cd is not larger than θ, otherwise y_{cj} is equal to y_{rj} (line 17-20 in Algorithm 3).

Compared with MLSMOTE, MLSOL is able to generate more diverse and well-labeled synthetic instances. As the example in Figure 2 shows, given a seed instance, the labels of the synthetic instance are fixed in MLSMOTE, while the labels of the new instance change according to its location in MLSOL, which avoids the introduction of noise as well.

3.3 Ensemble of Multi-Label Sampling (EMLS)

Ensemble is a effective strategy to increase overall accuracy and overcome overfitting problem, but has not been leveraged to multi-label sampling approaches. To improve the robustness of MLSOL and current multi-label sampling methods, we propose the ensemble framework called EMLS where any multi-label sampling approach and classifier could be embedded. In EMLS, M multi-label

```
Algorithm 3: GenerateInstance
input : seed instance: $(\boldsymbol{x}_s, \boldsymbol{y}_s)$, types of seed instance: $T_s$, reference instance:
        $(\boldsymbol{x}_r, \boldsymbol{y}_r)$, types of reference instance: $T_r$
output: synthetic instance: $(\boldsymbol{x}_c, \boldsymbol{y}_c)$
for $j \leftarrow 1$ to $d$ do
    $x_{cj} \leftarrow x_{sj} + \texttt{Random}(0,1) * (x_{rj} - x_{sj})$ ;    /* Random(0,1) generates a
      random value between 0 and 1 */
$d_s \leftarrow dist(x_c, x_s)$, $d_r \leftarrow dist(x_c, x_r)$ ; /* $dist$ return the distance between 2
  instances */
$cd \leftarrow d_s/(d_s + d_r)$ ;
for $j \leftarrow 1$ to $q$ do
    if $y_{sj} = y_{rj}$ then
        $y_{cj} \leftarrow y_{sj}$ ;
    else
        if $T_{sj} = MJ$ then /* ensure $y_{sj}$ being minority class */
            $s \longleftrightarrow r$ ;  /* swap indices of seed and reference instance */
            $cd \leftarrow 1 - cd$ ;
        switch $T_{sj}$ do
            case $SF$ do $\theta \leftarrow 0.5$ ; break ;
            case $BD$ do $\theta \leftarrow 0.75$ ; break ;
            case $RR$ do $\theta \leftarrow 1 + 1e - 5$ ; break ;
            case $OL$ do $\theta \leftarrow 0 - 1e - 5$ ; break ;
        if $cd \leqslant \theta$ then
            $y_{cj} \leftarrow y_{sj}$ ;
        else
            $y_{cj} \leftarrow y_{rj}$ ;
return $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ ;
```

learning models are trained and each model is built upon a re-sampled dataset generated by a multi-label sampling method with various random seed. There are many random operations in existing and proposed multi-label learning sampling methods [7, 6], which guarantees the diversity of training set for each model in the ensemble framework via employing different random seed. Then the bipartition threshold of each label is decided by maximizing F-measure on training set, as COCOA [31] and ECCRU3 [18] do. Given the test example, the predicting relevant scores is calculated as the average output relevant degrees obtained from M models, and the labels whose relevance degree is larger than the corresponding bipartition threshold are predicted as "1", and "0" otherwise.

3.4 Complexity Analysis

The complexity of searching kNN of input instances is $O(n^2d + n^2k)$. The complexity of computing C, $\boldsymbol{w}$ and T is $O(knq)$, $O(nq)$ and $O(nq)$, respectively. The complexity of creating instances is $O(nP(n + d))$ where nP is the number of

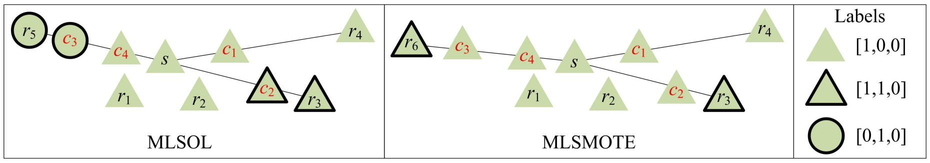

Fig. 2. An example of MLSOL excelling MLSMOTE. s is the seed instance, r_* are candidate reference instances (kNNs of s), and c_* are possible synthetic examples. The synthetic instances created by MLSMOTE are associated with unique label vector ([1,0,0]) decided by predominant kNNs, while MLSOL assigns labels to new examples according to its location. The two sampling approaches are identical if the synthetic instance (c_1 and c_4) is near the instance whose labels are same with majority kNNs or seed instance, otherwise (c_2 and c_3) the MLSMOTE introduces noise while MLSOL could tackle it by copying the labels of nearest instance to the new example.

generated examples. The overall complexity of MLSOL is $O(n^2d + n^2k + nkq)$, of which the kNN searching is the most time-consuming part.

Let's define $\Theta_t(n, d, q)$ and $\Theta_p(d, q)$ the complexity of training and prediction of multi-label learning method respectively, and $\Theta_s(n, d, q)$ the complexity of a multi-label sampling approach. The complexity of EMLS is $O(M\Theta_p(d, q))$ for prediction and $O\left(M\left(\Theta_s(n, d, q) + \Theta_t(n, d, q) + n\Theta_p(d, q)\right)\right)$ for training.

4 Empirical Analysis

4.1 Setup

Table 1 shows detailed information for the 13 benchmark multi-label datasets, obtained from Mulan's repository[1], that are used in this study. Besides, in textual data sets with more than 1000 features we applied a simple dimensionality reduction approach that retains the top 10% (bibtex, enron, medical) or top 1% (rcv1subset1, rcv1subset2, yahoo-Arts1, yahoo-Business1) of the features ordered by number of non-zero values (i.e. frequency of appearance). Besides, we remove labels only containing one minority class instance, because when splitting the dataset into training and test sets, there may be only majority class instances of those extremely imbalanced labels in training set.

Four multi-label sampling methods are used for comparison, namely the state-of-the-art MLSMOTE [6] and RHwRSMT [9] that integrates REMEDIAL [8] and MLSMOTE, as well as their ensemble versions, called EMLSMOTE and ERHwRSMT respectively. Furthermore, the base learning approach without employing any sampling approach, denoted as Default, is also used for comparing. For all sampling methods, the number of nearest neighbours is set to 5 and the Euclidean distance is used to measure the distance between the examples. In MLSOL, the sampling ratio is set to 0.3. In RHwRSMT, the threshold for

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

Table 1. The 16 multi-label datasets used in this study. Columns n, d, q denote the number of instances, features and labels respectively, LC the label cardinality, $MeanImR$ the average imbalance ratio of labels, where imbalance ratio of a label is computed as the number of majority instances divided by the number of minority instance of the label.

Dataset	Domain	n	d	q	LC	$MeanImR$
bibtex	text	7395	183	159	2.402	87.7
cal500	music	502	68	174	26	22.3
corel5k	image	5000	499	347	3.517	522
enron	text	1702	100	52	3.378	107
flags	image	194	19	7	3.392	2.753
genbase	biology	662	1186	24	1.248	78.8
medical	text	978	144	35	1.245	143
rcv1subset1	text	6000	472	101	2.88	236
rcv1subset2	text	6000	472	101	2.634	191
scene	image	2407	294	6	1.074	4.662
yahoo-Arts1	text	7484	231	25	1.654	101
yahoo-Business1	text	11214	219	28	1.599	286
yeast	biology	2417	103	14	4.237	8.954

decoupling instance is set to $SCUMBLE$. For MLSMOTE and RHwRSMT, the label generation strategy is *Ranking*. The ensemble size is set to 5 for all ensemble methods. In addition, six multi-label learning methods are employed as base learning methods, comprising four standard multi-label learning methods (BR [2], MLkNN [32], CLR [13], RAkEL [27]), as well as two state-of-the-art methods addressing the class imbalance problem (COCOA [31] and ECCRU3 [18]).

Three widely used imbalance aware evaluation metrics are leveraged to measure the performance of methods, namely macro-averaged F-measure, macro-averaged AUC-ROC (area under the receiver operating characteristic curve) and macro-averaged AUCPR (area under the precision recall curve). For simplicity, we omit the "macro-averaged" in further references to these metrics within the rest of this paper.

The experiments were conducted on a machine with 4×10-core CPUs running at 2.27 GHz. We apply 5×2-fold cross validation with multi-label stratification [22] to each dataset and the average results are reported. The implementation of our approach and the scripts of our experiments are publicly available at Mulan's GitHub repository[2]. The default parameters are used for base learners.

4.2 Results and Analysis

Detailed experimental results are listed in the supplementary material[3] of this paper. The statistical significance of the differences among the methods par-

[2] https://github.com/tsoumakas/mulan/tree/master/mulan

[3] https://intelligence.csd.auth.gr/publications/ecml-pkdd-2019-supplementary.pdf

ticipating in our empirical study is examined by employing the Friedman test, followed by the Wilcoxon signed rank test with Bergman-Hommel's correction at the 5% level, following literature guidelines [14, 1]. Table 2 shows the average rank of each method as well as its significant wins/losses versus each one of the rest of the methods for each of the three evaluation metrics and each of the six base multi-label methods. The best results are highlighted with bold typeface.

We start our discussion by looking at the single model version of the three resampling methods. We first notice that RHwRSMT achieves the worst results and that it is even worse than no resampling at all (default), which is mainly due to the additional bewilderment yielded by REMEDIAL, i.e. there are several pairs of instances with same features and disparate labels. MLSOL and MLSMOTE exhibit similar total wins and losses, especially in AUCPR, which is considered as the most appropriate measure in the context of class imbalance [21]. Moreover, the wins and losses of MLSOL and MLSMOTE are not that different from no resampling at all. This is particularly true when using a multi-label learning method that already handles class imbalance, such as COCOA and ECCRU3, which is not surprising.

We then notice that the ensemble versions of the three multi-label resampling methods outperform their corresponding single model versions in all cases. This verifies the known effectiveness of resampling apporaches in reducing the error, in particular via reducing the variance component of the expected error [15]. Ensembling enables MLSMOTE and MLSOL to achieve much better results compared to no resampling and it even helps RHwRSMT to do slightly better than no resampling.

Focusing on the ensemble versions of the three resampling methods we notice that EMLSOL achieves the best average rank and the most significant wins without suffering any significant loss in all 18 different pairs of the 6 base multi-label methods and the 3 evaluation measures, with the exception that MLSMOTE with MLkNN as base learner achieves best average rank in terms of F-measure. EMLSMOTE comes second in total wins and losses in most cases, while ERHwRSMT does much worse than EMLSMOTE.

An interesting observation here is that while MLSOL and MLSMOTE have similar performance, MLSOL benefitted much more than MLSMOTE from the ensemble approach. This happens because randomization plays a more important role in MLSOL than in MLSMOTE. MLSOL uses weighted sampling for seed instance selection, while MLSMOTE takes all minority samples into account instead. This allows EMLSOL to create more diverse models, which achieve greater error correction when aggregated.

5 Conclusion

We proposed MLSOL, a new synthetic oversampling approach for tackling the class-imbalance problem in multi-label data. Based on the local distribution of labels, MLSOL selects more important and informative seed instances and generates more diverse and well-labeled synthetic instances. In addition, we employed

Table 2. Average rank of the compared methods using 6 base learners in terms of three evaluation metrics. A_1, A_2, EA_1 and, EA_2 stands for MLSMOTE, RHwRSMT, EMLSMOTE and ERHwRSMT, respectively. The parenthesis (n_1/n_2) indicates the corresponding method is significantly superior to n_1 methods and inferior to n_2 methods based on the Wilcoxon signed rank test with Bergman-Hommel's correction at the 5% level.

F-measure							
Base Method	Default	A_1	A_2	MLSOL	EA_1	EA_2	EMLSOL
BR	5.19(1/4)	3.73(2/2)	7.00(0/6)	4.23(2/2)	2.08(5/1)	4.38(1/2)	**1.38(6/0)**
MLkNN	5.81(1/5)	4.92(2/4)	7.00(0/6)	4.27(3/3)	**1.62(5/0)**	2.69(4/2)	1.69**(5/0)**
CLR	5.04(1/3)	4.5(1/3)	7.00(0/6)	4.15(1/3)	2.58**(4/0)**	2.62**(4/0)**	**2.12(4/0)**
RAkEL	5.04(1/4)	3.88(2/2)	7.00(0/6)	3.5(2/2)	2.46(5/1)	4.69(1/2)	**1.42(6/0)**
COCOA	3.58(1/0)	4.42(1/1)	6.35(0/5)	5.23(0/1)	3.27(1/0)	3.31(1/0)	**1.85(3/0)**
ECCRU3	3(2/0)	4.58(1/1)	6.31(0/5)	5.46(0/2)	3.35(1/0)	3.46(1/1)	**1.85(4/0)**
Total	7/16	9/13	0/34	8/13	21/2	12/7	**28/0**
AUC-ROC							
Base Method	Default	A_1	A_2	MLSOL	EA_1	EA_2	EMLSOL
BR	5.23(1/3)	4.65(1/3)	6.27(0/6)	3.46(3/1)	2.81(4/1)	4.58(1/2)	**1.00(6/0)**
MLkNN	4.69(1/1)	3.73(2/2)	6.35(0/6)	4.23(2/1)	2.58(3/1)	5.35(1/4)	**1.08(6/0)**
CLR	4.35(0/1)	4.77(0/2)	5.58(0/3)	5.00(0/2)	2.85(4/1)	4.08(1/2)	**1.38(6/0)**
RAkEL	4.38(2/4)	3.73(3/2)	6.77(0/6)	3.54(3/2)	2.54(5/1)	6.00(1/5)	**1.04(6/0)**
COCOA	5.23(0/1)	4.73(0/1)	5.42(0/1)	4.54(0/1)	3.42(0/1)	3.65(0/1)	**1.00(6/0)**
ECCRU3	4.73(0/1)	4.23(0/2)	5.73(0/3)	5.46(0/1)	2.65(3/1)	4.12(1/2)	**1.08(6/0)**
Total	4/11	6/12	0/25	8/8	19/6	5/16	**36/0**
AUCPR							
Base Method	Default	A_1	A_2	MLSOL	EA_1	EA_2	EMLSOL
BR	4.81(1/2)	3.85(1/2)	6.46(0/6)	4.15(1/2)	2.46(5/1)	5.19(1/2)	**1.08(6/0)**
MLkNN	5.04(0/2)	4.5(1/2)	5.92(0/5)	4.08(1/1)	3.04(4/1)	4.42(1/2)	**1.00(6/0)**
CLR	4.15(1/1)	4.88(0/2)	5.92(0/4)	5.00(0/3)	3.00(3/1)	3.81(2/1)	**1.23(6/0)**
RAkEL	4.42(1/2)	3.92(1/2)	6.77(0/6)	3.92(1/2)	2.5(5/1)	5.46(1/2)	**1.00(6/0)**
COCOA	5.31(0/1)	4.85(0/1)	5.31(0/1)	4.62(0/1)	3.15(0/1)	3.77(0/1)	**1.00(6/0)**
ECCRU3	4.96(0/2)	4.5(0/2)	5.50(0/3)	5.04(0/2)	2.88(5/1)	4.12(1/2)	**1.00(6/0)**
Total	3/10	3/11	0/25	3/11	22/6	6/10	**36/0**

MLSOL within a simple ensemble framework, which exploits the random aspects of our approach during sampling training examples to use as seeds and during the generation of synthetic training examples.

We experimentally compared the proposed approach against two state-of-the art resampling methods on 13 benchmark multi-label datasets. The results offer strong evidence on the superiority of MLSOL, especially of its ensemble version, in three different imbalance-aware evaluation measures using six different underlying base multi-label methods.

Acknowledgements Bin Liu is supported from the China Scholarship Council (CSC) under the Grant CSC No.201708500095.

References

[1] Benavoli, A., Corani, G., Mangili, F.: Should We Really Use Post-Hoc Tests Based on Mean-Ranks? Journal of Machine Learning Research **17**, 1–10 (2016)

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

[3] Cao, P., Liu, X., Zhao, D., Zaiane, O.: Cost Sensitive Ranking Support Vector Machine for Multi-label Data Learning. In: Proceedings of the 16th International Conference on Hybrid Intelligent Systems (HIS 2016). pp. 244–255. Springer International Publishing, Cham (2017)

[4] Charte, F., Rivera, A., del Jesus, M.J., Herrera, F.: A First Approach to Deal with Imbalance in Multi-label Datasets. In: Proceedings of the 8th International Conference on Hybrid Artificial Intelligent Systems (HAIS 2013). vol. 8073 LNAI, pp. 150–160 (2013). https://doi.org/10.1007/978-3-642-40846-5_16

[5] Charte, F., Rivera, A.J., Del Jesus, M.J., Herrera, F.: MLeNN: A first approach to heuristic multilabel undersampling. In: Intelligent Data Engineering and Automated Learning – IDEAL 2014. vol. 8669 LNCS, pp. 1–9. Springer International Publishing (2014). https://doi.org/10.1007/978-3-319-10840-7_1

[6] Charte, F., Rivera, A.J., Del Jesus, M.J., Herrera, F.: MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation. Knowledge-Based Systems **89**, 385–397 (2015). https://doi.org/10.1016/j.knosys.2015.07.019

[7] Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: Addressing imbalance in multilabel classification: Measures and random resampling algorithms. Neurocomputing **163**, 3–16 (9 2015). https://doi.org/10.1016/j.neucom.2014.08.091

[8] Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: Dealing with difficult minority labels in imbalanced mutilabel data sets. Neurocomputing **326-327**, 39–53 (2019). https://doi.org/10.1016/j.neucom.2016.08.158

[9] Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: REMEDIAL-HwR: Tackling multilabel imbalance through label decoupling and data resampling hybridization. Neurocomputing **326-327**, 110–122 (2019). https://doi.org/10.1016/j.neucom.2017.01.118

[10] Chen, K., Lu, B.L., Kwok, J.T.: Efficient Classification of Multi-label and Imbalanced Data using Min-Max Modular Classifiers. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings. pp. 1770–1775. IEEE (2006). https://doi.org/10.1109/IJCNN.2006.246893

[11] Daniels, Z.A., Metaxas, D.N.: Addressing Imbalance in Multi-Label Classification Using Structured Hellinger Forests. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence. pp. 1826–1832 (2017)

[12] Dendamrongvit, S., Kubat, M.: Undersampling Approach for Imbalanced Training Sets and Induction from Multi-label Text-Categorization Domains. In: Proceedings of the 13th Pacific-Asia International Conference on Knowledge Discovery and Data Mining (PAKDD'09). pp. 40–52 (2009). https://doi.org/10.1007/978-3-642-14640-4_4

[13] Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning **73**(2), 133–153 (2008). https://doi.org/10.1007/s10994-008-5064-8

[14] Garcia, S., Herrera, F.: An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. Journal of machine learning research **9**, 2677–2694 (2008)

[15] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer (2016)

[16] Li, C., Shi, G.: Improvement of learning algorithm for the multi-instance multi-label RBF neural networks trained with imbalanced samples. Journal of Information Science and Engineering **29**(4), 765–776 (2013)

[17] Li, L., Wang, H.: Towards Label Imbalance in Multi-label Classification with Many Labels. arXiv preprint arXiv:1604.01304 (2016)

[18] Liu, B., Tsoumakas, G.: Making Classifier Chains Resilient to Class Imbalance. In: 10th Asian Conference on Machine Learning (ACML 2018). p. 280295. Beijing (2018)

[19] Napierala, K., Stefanowski, J.: Types of minority class examples and their influence on learning classifiers from imbalanced data. Journal of Intelligent Information Systems **46**(3), 563–597 (2016)

[20] Sáez, J.A., Krawczyk, B., Woźniak, M.: Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. Pattern Recognition **57**, 164–178 (2016). https://doi.org/10.1016/j.patcog.2016.03.012

[21] Saito, T., Rehmsmeier, M.: The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS ONE (2015). https://doi.org/10.1371/journal.pone.0118432

[22] Sechidis, K., Tsoumakas, G., Vlahavas, I.: On the Stratification of Multi-label Data. In: Proc. 2011 European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 145–158. Springer Berlin Heidelberg, Athens, Greece (2011)

[23] Sozykin, K., Khan, A.M., Protasov, S., Hussain, R.: Multi-label Class-imbalanced Action Recognition in Hockey Videos via 3D Convolutional Neural Networks. In: 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). pp. 146–151 (2018)

[24] Sun, K.W., Lee, C.H.: Addressing class-imbalance in multi-label learning via two-stage multi-label hypernetwork. Neurocomputing **266**, 375–389 (2017). https://doi.org/10.1016/j.neucom.2017.05.049

[25] Tahir, M.A., Kittler, J., Yan, F.: Inverse random under sampling for class imbalance problem and its application to multi-label classification. Pattern Recognition **45**(10), 3738–3750 (2012). https://doi.org/10.1016/j.patcog.2012.03.014

[26] Tepvorachai, G., Papachristou, C.: Multi-label imbalanced data enrichment process in neural net classifier training. In: Proceedings of the International Joint Conference on Neural Networks. pp. 1301–1307 (2008). https://doi.org/10.1109/IJCNN.2008.4633966

[27] Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multilabel classification. IEEE Transactions on Knowledge and Data Engineering **23**(7), 1079–1089 (2011)

[28] Wan, S., Duan, Y., Zou, Q.: HPSLPred: An Ensemble Multi-Label Classifier for Human Protein Subcellular Location Prediction with Imbalanced Source. Proteomics **17**(17-18), 1700262 (2017). https://doi.org/10.1002/pmic.201700262

[29] Wu, B., Lyu, S., Ghanem, B.: Constrained Submodular Minimization for Missing Labels and Class Imbalance in Multi-label Learning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. pp. 2229–2236. AAAI'16, AAAI Press (2016)

[30] Zeng, W., Chen, X., Cheng, H.: Pseudo labels for imbalanced multi-label learning. In: 2014 International Conference on Data Science and Advanced Analytics (DSAA). pp. 25–31 (10 2014). https://doi.org/10.1109/DSAA.2014.7058047

[31] Zhang, M.L., Li, Y.K., Liu, X.Y.: Towards class-imbalance aware multi-label learning. In: Proceedings of the 24th International Conference on Artificial Intelligence. pp. 4041–4047 (2015)

[32] Zhang, M.L., Zhou, Z.H.: ML-KNN: A lazy learning approach to multi-label learning. Pattern recognition **40**(7), 2038–2048 (2007)