# Handling Tail-label Problems in Multi-label Classification Using in-situ Synthetic Data Generation

A Dissertation Submitted in fulfillment of the requirements for the award of the Degree of

**MASTER OF SCIENCE**

**IN**

**COMPUTER SCIENCE (BIG DATA ANALYTICS)**

**Submitted by:**

Pradyumna Kumar Sahoo (2018MSBDA016)

**Under the Guidance of:**

Dr. Manas Patra and Dr. Vikas Kumar

Department of Data Science and Analytics

School of Mathematics, Statistics and Computational Science

Central University of Rajasthan

May 2020

# CERTIFICATE

This is to certify that the Master of Science thesis entitled **Handling Tail-label Problems in Multi-label Classification Using in-situ Synthetic Data Generation** is a bonfide work done by **Mr. Pradyumna Kumar Sahoo** bearing enrollment number **2018MSBDA016** in the 4th semester at Central University of Rajasthan during the year 2019-20 in the partial fulfilment of the award of **Masters of Computer Science Big Data Analytics** from Central University of Rajasthan.

| | |
|---|---|
| **Dr. Manas Patra** | **Dr. Vikas Kumar** |
| Associate Professor, HOD | Assistant Professor |
| Dept. of Data Science and Analytics, | Dept. of Data Science and Analytics, |
| Central University of Rajasthan | Central University of Rajasthan |

# DECLARATION

I hereby declare that the thesis entitled **Handling Tail-label Problems in Multi-label Classification Using in-situ Synthetic Data Generation** submitted for the M.Sc. Computer Science Big Data Analytics degree is my original work conducted under the guidance of **Dr. Manas Patra**, Associate Professor and **Dr. Vikas Kumar**, Assistant Professor, Department of Data Science and Analytics.

I also certify that to the work mentioned in the thesis has been done by me and has not been submitted for the award of any degree either in this university or in any other university without proper citation.

**Pradyumna Kumar Sahoo**
Enrollment No.: - 2018MSBDA016
Department of Data Science & Analytics,
School of Mathematics, Statistics and Computational Science,
Central University of Rajasthan

# ABSTRACT

Labels with very few instances to learn from, called Tail-labels, is a critical problem faced in multilabel classification that have garnered recent attention among researchers. Thus, to improve learning, oversampling algorithm MLSMOTE is employed on tail instances to generate new labeled instances in-situ for our base classifier LLSF-DL which is retrained on the modified dataset. This provides a standalone solution for learning of the classifier taking care of class imbalance along the way.

# ACKNOWLEDGEMENTS

Firstly, I would like to express my devotion towards the almighty for making this happen. I am really grateful to my supervisors **Dr. Manas Patra** and **Dr. Vikas Kumar** for their continuous support, guidance and friendly behavior throughout my thesis work. I really look forward to work with them in future.

I am grateful to my parents for their love and unconditional affection, supporting me throughout my life, and during this thesis in particular.

I want to thank all my friends for their spending time listening, encouraging and providing valuable inputs during my thesis. I would like to specially mention my lab partner, Ruchi, for helping me with various aspects of the experiment and implementation.

Furthermore, I am thankful for the faculty members of Department of Data Science who gave me valuable knowledge all these years, enduring my inquiries, helping me setup my workspace and providing all logistical support.

Pradyumna Kumar Sahoo

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Problem Definition

To create a hybrid algorithm out of MLSMOTE and LLSF-DL that can handle tail labels in multi-label classification using synthetic data generation.

## 1.2 Thesis Overview

We have worked on tail label problem in multi-label classification using Synthetic data generation approach which have shown promising results. SMOTE has been front-runner in dealing with class imbalance in multi-class classification. Hence its multi-label extension, MLSMOTE has been preferred as the main synthetic data generation technique. We have created for the first time, a hybrid algorithm using MLSMOTE to boost prediction accuracy of tail labels in current State of the art multi-label classifier (LLSF-DL).

## 1.3 Hardware Specification

1. **RAM:** Minimum: 4 GB
   Recommended: 8 GB

2. **Processor**: Any Intel or AMD x86-64 processor
3. **Disk space:** Minimum: 2 GB of HDD space for MATLAB only, 4-6 GB for a typical installation. A full installation of all MathWorks products may take up to 22 GB of disk space.

## 1.4 Software Specification

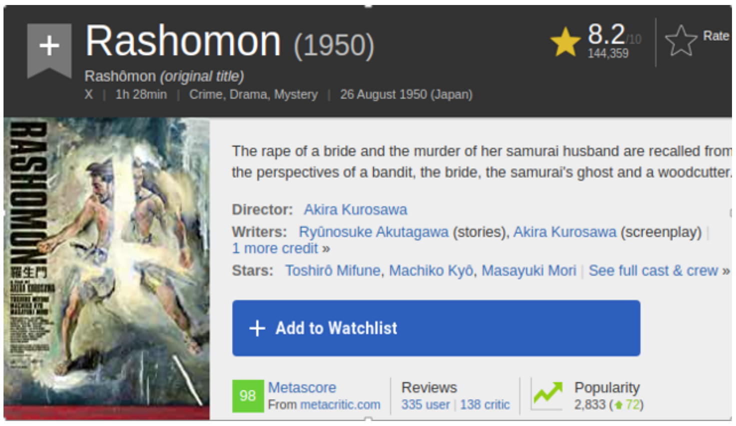
Operating system :Windows 10/*nix system
Matlab 2018a

# Chapter 2

# Literature survey

Among various other fields that comes under supervised machine learning, Classification is a premiere field that occupies a large portion of research. Classification answers the question of assigning class(es) to the given input instance which can be anything from vectors to images to audio to text [12][13][2][15][7][14] [17][16][18]. The output is discrete here. Diving deeper, classification is further divided into various sub-fields based upon the restrictions on class assignment towards an instance.

To showcase the major classification schemes,here is an example in fig 2.1.



Rashomon (1950)

Rashômon *(original title)*

X | 1h 28min | Crime, Drama, Mystery | 26 August 1950 (Japan)

8.2/10 144,359

Rate

The rape of a bride and the murder of her samurai husband are recalled from the perspectives of a bandit, the bride, the samurai's ghost and a woodcutter.

**Director:** Akira Kurosawa

**Writers:** Ryûnosuke Akutagawa (stories), Akira Kurosawa (screenplay) | 1 more credit »

**Stars:** Toshirô Mifune, Machiko Kyô, Masayuki Mori | See full cast & crew »

Add to Watchlist

98 Metascore From metacritic.com

Reviews 335 user | 138 critic

Popularity 2,833 (72)

Figure 2.1: IMDB Roshomon page.

**Binary Classification (BC):**An instance can belong one or the other class out of only two classes but not both. Ex: One of the legendary movie, Rashomon can be classified as $20^{th}$ century film or not.

**Multiclass Classification (MCC):** An instance can belong only one class out of many classes. Ex: Rashomon can be classified into one of the group of ratings i.e. 7+,8+,9+ etc.

**Multi-label Classification (MLC):** An instance can belong one or more than one classes at the same time where each class is binary in nature. Ex:Roshomon belongs to Crime, Drama, Mystery genre concurrently. Here the genre act as classes which are referred to as labels. The movie may belong to more than one label but the labels are essentially binary i.e. the movie either belongs to crime genre or it does not.

**Multidimensional Classification (MDC):** It is an extension of MLC where the labels are not necessarily binary. Ex: The movie's labels represents whether the movie is a $20^{th}$ century movie, who are the actors, which genres it belongs to, which month it has been released. The month itself being a label consists of 12 different sub-levels. This mixture of different objectives contributes towards the multidimensional aspect.

Now, discussion shall be steered towards multilabel classification. Formally, a multilabel dataset (**MLD**) has target label matrix $Y \in \{0,1\}^{m*l}$ and feature matrix $X \in \mathbb{R}^{m*d}$ where m, l, d denotes total instances, features and labels present respectively. Each instance $x_i \in X$ is a d-dimensional real vector. Similarly, the labelset associated with $x_i$ i.e. $y_i \in Y$ is also l-dimensional vector with each label having a discrete binary value {0,1}.

Multi-label classification is difficult because as labels increases, output space exponentially scales up. Thus, distributions of class labels in the data need not be uniform, producing imbalanced datasets. Naturally, the class labels do not have equal representation. Labelset-wise, when there are relatively large number of instances associated with some labelsets, this causes label skew. Label-wise, imbalance can be **inter-label** or **intra-label**. For each label, when instances belonging to a certain label outnumbers other labels, this constitute inter-label imbalance. Similarly, when positive instances are very few as compared to negative instances for each label or vice-versa, this is denoted as intra-label imbalance. These intra-label imbalanced labels are famously called **Tail-labels**.The name has been stuck with it because they lie towards the end of the label frequency plot as shown in the Figure 3.1.These notoriously remains one of the main issues that plagues various classification algorithms. These tail-labels have very few discriminating instances to train the classifier on. Thus, inferences drawn from classification over such scarce dataset are mostly either inaccurate or can not be trusted. This happens because most classifiers aim to reduce some global misclassification cost but they also inevitably penalize classification of the tail-labels along the way.

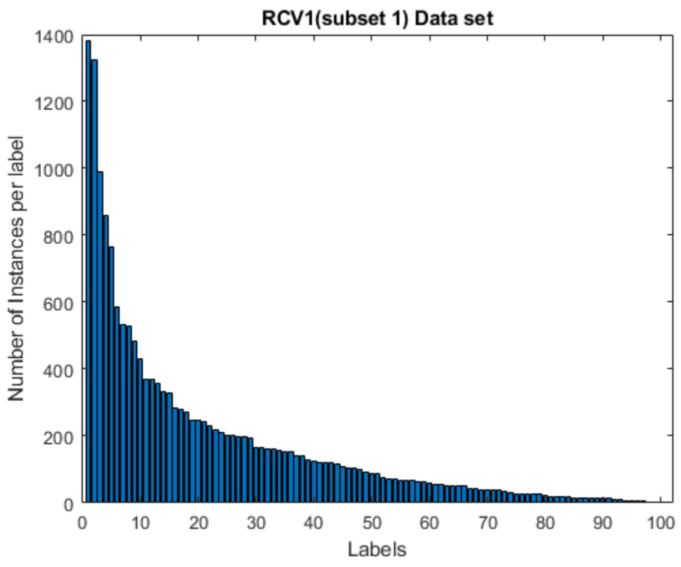


Figure 2.2: Label Frequency Plot

*Selecting (**LLSF-DL**)[10] classifier is based on the fact that it learns higher-order label correlations along with underlying second-order counterparts by selecting features strongly influencing labels as well as exploring the shared label space among class-dependent labels.*

For imbalanced classification, three most common approaches to deal with tail-labels are: **Re-sampling, cost sensitive classification, algorithmic adaptations**[4].*Data re-sampling* balances labels' distributions by either removing most frequent label(s) instances, known as *under-sampling* or adding new instances of the least frequent one, called *over-sampling.* This approach is independent of the classification algorithms. *Cost-sensitive classification* tweaks the objective function of the constrained optimisation to provide equal footing for tail-labels. *Algorithmic adaptions* includes transformative methods that converts the task at hand to a multiclass task as well as hybrid methods that mostly consist of ensembles among various other things.

Among synthetic data generation techniques, Synthetic Minority Oversampling Technique (**SMOTE**)[6] stands apart from rest of the methods as SOTA in the field. One of the extension for multi-label samples is **MLSMOTE**[5] which is relevant in our case. It separates tail-instances based upon imbalance ratio. And then generates new synthetic sample in the feature space over the lines joining a reference point and its neighbours. The labelsets are then assigned using majority voting among the reference point and its neighbours.

*For the first time, a hybrid algorithm based on LLSF-DL and MLSMOTE has been proposed which is expected to handle all kind of tail-labels by leveraging best of both the techniques.*

From this chapter onward, the thesis has been presented in following manner-

1. Chapter 2 provides the necessary background and contains proposed work.
2. Chapter 3 provides experiment design and analysis.
3. Chapter 4 concludes with some remarks on future works. In the end, appendices are given for better understanding of various items.

## 2.1 Previous Work

Multi-label classifiers fall into two categories: problem transforming type (**PTT**) that adapt data to algorithm and algorithm adaption type (**ATT**) that adapt algorithm to data. We knowingly ignore ranking-based classifiers.

On the basis of capturing label correlation capacity, the algorithms belong to three categories : First-order(no overall correlation captured)(**FO**), Second-order(label pair-wise correlation captured)(SO), Higher-order(labelset-wise correlation captured)(**HO**). These catagories are not mutually exclusive at all,an algorithm can capture both higher order as well as other correlations. We will be using the above abbreviation in pairs suggesting both their types and correlation capturing capabilities. For example, PTT-FO stands for algorithm adaption type first order algorithm. The following discussion on MLC algorithms is heavily influenced from a tutorial by Eva Gibaja and Sebastián Ventura[8].

Some PTT algorithms are mainly based on Binary relevance (**BR**) methods reducing multilabel classification into l-binary classification, l being total labels present. Some of the premiere examples are : Binary relevance algorithm which is the base **PTT-FO** classifier, Classifier Chain (**CC**), Probabilistic Classifier Chain (**PCC**,)Meta Binary Relevance (**2BR**).The main strategy they follow is One-vs-rest (**OvR**).Some PTT-SO classifiers are based on based One vs One (**OvO**) strategy. This later group contains mainly Ranking based classifiers which we are ignoring to be relevant to our cause. Many **PTT-HO** type use label powerset transformation which reduces MLC problem into a multiclass problem.Some algorithms in this field are ensemble based, like RAndom k-labELsets method (**RAkEL**), Probilistic Classifier chains ensemble (**EPCC**) and Ensemble of Pruned Sets (**EPS**) algorithm are SOTA. Some **PTT-HO** algorithms employ **BR** methods to gain optimal condition lying between strong independence assumption of BR and very few instance in label powersets to get better results. Hence they come under hybrid methods like **ChiDep** and their ensemble counterparts ChiDep Ensemble (**CDE**).

In case of ATT, most MLC algorithms are basically extensions of their multi-class formats. Thus we can categorize these algorithms based on their multi-class counterparts. The label correlation capturing from here on is based upon the individual algorithms and their implicit strategy. Decision tree based methods used in a hierarchical fashion as well as in the ensemble setting like **ML-C4.5** and Predictive Clustering Trees (**PCT**) are some of the examples. Some Support Vector Machine (**SVM**) based also exist that are applied using One vs Rest approach. The major portion of SVM based MLC algorithms are ranking based. Instance based MLC algorithms derive their power from considering nearest neighbours for each instance and base their prediction upon Maximum APosteriori principle. Some major algorithm are multi-label k-nearest neighbor (**ML-kNN**), Instance-Based Learning by Logistic Regression (**IBLR**) and a BR hybrid, **BrkNN**. Artificial neural networks (**ANN**) are a hot topic now and their entry into MLC started with Multi-label Multiclass Perceptron (**MMP**). Backpropagation for Multi-label Learning (**BP-MLL**) and Multi-label Radial Basis Function (**ML-RBF**) were later proposed as an adaptation for simple feed-forward neural networks for MLC which is known as **MLRBFnet**. Generative models have been utilised too for MLC. Mixture models were the first to be used as early as 1999. Parametric Mixture Models **(PMM)** were later introduced. Similarly, Conditional Random Fields **(CRFs)** for MLC and Multi-label Naive Bayes **(MLNB)** are some of the big players in this field. Associative methods like Multiclass, multi-label associative classification **(MMAC)** and Evolutionary methods like Multi-label Ant-Miner **(MuLAM),GEP-MLC, GC** are some novel methods that are commendable.Ensemble strategy is also used.It can be homogeneous grouped(ensemble of same classifiers) or heterogeneous grouped(ensemble of different classifiers). Some major ensemble methods are **AdaBoost-MH, AdaBoost-MR**, Model-Shared Subspace Boosting **(MSSBoost)**. Some homogeneous grouped methods prevalent in MLC are Random Forest of Predictive Clustering Tress **(RF-PCT)** and Random Forest of ML-C4.5. Structured forest of Hellinger Decision Trees **(sHHDT)** deserves special mention because of their ability of addressing the problem of class imbalance in MLC.

LLSF_DL thus,is a ATT-HO-SO MLC algorithm that models both labelset correlation alongwith pairwise label correlation. Selecting this particular algorithm has been based upon the search for an optimal algorithm having properties discussed in Proposed work chapter.

The above two paragraphs contain many pointers towards major MLC algorithms. Now, focusing on the imbalance problem, we shall discuss some major approaches on basis of their influence at different levels: *algorithm-level, data-level & hybrid methods*[11]. Algorithm level approach modifies existing MLC algorithms to prevent bias towards majority items(features,labels,intra-label) and evolve them for skewed data distributions. Modification of datasets to balance distributions across features and/or labels takes place via data level methods. If deemed necessary, it also removes difficult samples. The hybrid approach tries to leverage advantages of both approaches mentioned above. The major algorithms using these approaches fall into three categories : *Data Pre-processing,Post Prediction-processing,Special-purpose Learning Methods[3].*

We will be focusing on Data pre-processing approaches. They modify the dataset changing the data distribution. Under this approach, all algorithms can again be classified on the basis of their strategy: *Distribution change, Data-space weighting.* The former class can be sub-divided into *Stratified sampling, Synthetic data generation and Combination of both.* Stratified sampling algorithms use various strategies like data cleaning approaches, random under/over-sampling, distance methods, clustering algorithms and/or evolutionary algorithms. The original dataset is modified by adding instances,removing instances or both.Multi-label Random Under-Sampling **(ML-RUS)** and Multi-label Random Over-Sampling **(ML-ROS)** have been extensively studied along with label powerset transformation like LP-Random Under-Sampling **(LP-RUS)** as well as LP- Random Over-Sampling **(LP-ROS)**[4].

Synthetic data generation algorithms generates new data and append them to the existing data.Then the enlarged dataset is used for further tasks under supervised learning purview. SMOTE is one of the hallmark techniques that rules this field even after many years of its publication. An extension of SMOTE for multi-label datasets,

**MLSMOTE**, has been used for our purpose of at-place synthetic data generation.In our setting, this technique runs for each instance in which the tail label exist, and then SMOTE is ran multiple times to generate new samples.

Various pointers has been referenced for further investigation into the developments. We have briefly presented an overview of works that has been done related to our individual algorithms. In the following chapters, we have individually discussed some topics that are essential for our algorithm development, aggregating all these developments into our algorithm in the end portion.

### 2.1.1 LLSF-DL

Learning Label-Specific Features and Class-Dependent Labels (LLSF-DL) [10] classifier is one of the SOTA algorithm for MLC. To gain better understanding, here we will be decomposing the LLSF-DL into its fundamental components first.

**Label-specific features:** In MLC, instances can have various labels associated with it. The features of those instances are obviously responsible for their labels. But the question is - which subset of features out of all are responsible for the individual labels when the instance has multiple ones? Thus, the features can be partitioned as per their role towards guiding instances towards a specific labels. These are called Label-specific features. Correlated labels share the same label-specific features. For example, suppose we have $X$ with features $[F_1, F_2, F_3]$ with labels $[L_1, L_2, L_3]$. If we know that $F_1$ and $F_2$ features are responsible for label $L_1$ just like $F_2$ and $F_3$ for label $L_2$, then $X_1$ and $X_2$ constitute L1-specific features while $F_2$ and $F_3$ constitute L2-specific features. If $L_1$ and $L_2$ are found to be correlated, they share the same label-specific features, i.e. $F_2$ here.

**Class-dependent labels:** In real world, labels are known to be not completely uncorrelated. Thus, a subset of labels can influence another subset as well as individual labels. For example, the Roshomon movie has genre labels – crime,drama,mystery. We usually find movies that are genre Crime associated with mystery. We seldom find crime and musical genre together. Similarly, we commonly find a movie tagged with crime and mystery also have drama elements in it. The former example was related to pair-wise correlations (*second order*) while the later one depicts label correlation between a subset of labels with an individual one (*higher order*)[19]. In both the cases, those label or label subset that get influenced by other label or label subsets are called class-dependent labels.

Modeling these label correlations when the label-space grows exponentially with the number of labels, is computationally heavy and practically less scalable. The usual problems faced are: error propagation and unnecessary dependencies. This is because of the assumption that all examples share the global label correlations. Hence, removing unnecessary label dependencies reduces error propagation ab initio.

When labels are correlated to each other, they share the same label-specific features. Hence, weights of these features are also modeling pairwise label correlation taken into consideration by adding another correlation term R that is basically 1- C, where C is the correlation matrix of $Y$ multiplied with $W_x^T W_x$, a dot product for non-zero weights responsible for label-specific information parsing into selection of features of all labels. This product will be large for correlated labels sharing same label specific features.

$$\min_{W} \frac{1}{2}\|XW - Y\|_F^2 + \frac{\alpha}{2}Tr(RW^TW) + \beta\|W\|_1 \tag{2.1}$$

Hence, LLSF-DL augments $W_Y Y$ is stacked alongside $W_x X$ as guiding feature inside the main least square objective function. L1-regularization of weight matrix $W_y$ further introduce sparsity in label dependencies removing unnecessary label dependencies.

$$\min_{W_x, W_y} \frac{1}{2}\|XW_x + YW_y - Y\|_F^2 + \frac{\alpha}{2}Tr(RW_x^TW_x) + \beta\|W_x\|_1 + \gamma\|W_y\|_1 \tag{2.2}$$

The L1-norm makes the objective functions non-smooth, thus conventional optimization techniques will not work. Hence **accelerated proximal gradient** method is used.(see Appendix A) And the optimized LLSF-DL algorithm return us weight matrix for $X$ and $Y$, it is as follows:

**Algorithm 1** LLSF-DL

```
    Input: Feature Matrix $X \in \mathbb{R}^{m*d}$,Label Matrix $Y \in \{0,1\}^{m*l}$, Weight parameters $\alpha$, $\beta$, $\gamma$, $\rho$, Test Matrix $X_t \in \mathbb{R}^{n*d}$ ,Threshold $\tau$, tuning parameter $\kappa$.
    Output:Predicted Label Matrix $Y_t \in \{0,1\}^{n*l}$
 1: Initialization:
 2: $b_0 \leftarrow 1$, $b_1 \leftarrow b_0$, $t \leftarrow 1$
 3: $W_x^{(0)} \leftarrow (X^TX + \rho I)^{-1}X^TY$
 4: $W_x^{(1)} \leftarrow W_x^{(0)}$
 5: $W_y^{(0)} \leftarrow (Y^TY + \rho I)^{-1}Y^TY$
 6: $W_y^{(1)} \leftarrow W_y^{(0)}$
 7: $R \leftarrow \frac{Y^TY}{||Y||_2^2}$                                  ▷ Cosine similarity
 8: $W_f \leftarrow [3(||X^TX||_2^2 + ||X^TY||_2^2 + ||\alpha R||_2^2) + 2(||Y^TY||_2^2)]^{1/2}$
 9: repeat
10:     ▷ Gradient descent
11:     $W_x^{(t)} \leftarrow W_x^{(t)} + \frac{b_{t-1}-1}{b_t}(W_x^{(t)} - W_x^{(t-1)})$
12:     $W_y^{(t)} \leftarrow W_y^{(t)} + \frac{b_{t-1}-1}{b_t}(W_y^{(t)} - W_y^{(t-1)})$
13:     ▷ Regularization
14:     $G_x^{(t)} \leftarrow W_x^{(t)} - \frac{1}{L_f}[X^TXW_x + X^TYW_y - X^TY + \alpha W_xR]$
15:     $G_y^{(t)} \leftarrow W_y^{(t)} - \frac{1}{L_f}[Y^TYW_y + Y^TXW_x - Y^TY]$
16:     $W_x^{(t+1)} \leftarrow soft\_threshold(G_x^{(t)}, (\frac{\rho}{L_f}))$
17:     $W_y^{(t+1)} \leftarrow soft\_threshold(G_y^{(t)}, (\frac{\rho}{L_f}))$
18:     $b_{t+1} \leftarrow 0.5(1 + [4b_t^2 + 1]^{(1/2)})$
19:     $t \leftarrow t + 1$
20: until Stopping criteria not reached
21: $\hat{Y}_t \leftarrow LLSF[X, Y, Xt, \alpha, \beta, \rho, \tau$
22: for $i \leftarrow 1, \kappa$ do
23:     $S_t \leftarrow X_tW_x + \hat{Y}_tW_y$
24:     $\hat{Y}_t \leftarrow sign(S_t - \tau)$
25: end for
26: $Y_t \leftarrow \hat{Y}_t$
```

The soft threshold function has been defined in Appendix A. LLSF algorithm(see Appendix B) produces Weight matrix for X only, which is then fed into LLSF - predict function that gives us $\hat{Y}$. We then use the weight matrices from LLSF-DL to get our final predicted label matrix.

### 2.1.2 MLSMOTE

The main idea of Multi-label SMOTE (**MLSMOTE**)[5] is to generate new and balanced datasets for further training purpose taking the basic idea of SMOTE and generalizes towards the multi-label cases. Since, this method shares the core of SMOTE, here a basic overview of SMOTE has been provided.

**SMOTE** algorithm first of all isolates all the tail instances. . For each tail instance,among its closest neighbors, an instance is selected at random and it generates a new instance of same tail label. Further, for each minority instance, the number of neighbors and synthetic instances can be set in advance. MLSMOTE has some implicit assumptions:

1. Neighboring instances share correlations among them, particularly local correlations.
2. The randomly selected reference neighbor represents the locality in the feature space effectively.
3. Global label correlations does not influence local label space distribution.

Functioning of MLSMOTE can be decomposed into following ones:

1. Selection of tail instances
2. Nearest neighbor search
3. Feature set generation
4. Synthetic labelset assignment

Based upon the imbalance ratio per label, those labels are marked whose IRLbl > MeanIRLbl. These are the tail-labels. Instances having these tail labels, called tail

**Algorithm 2** Pseudo-code of MLSMOTE algo

```
    Input: Feature matrix $X$, Label matrix $Y$, Nearest neighbors $K$

 1: $Z \leftarrow$ Concatenate the $X$ and $Y$          ▷ Data set for oversampling
 2: $L \leftarrow$labels In Dataset($Z$)
 3: Calculate $MeanIR$ for $Z$ by Eq 3.2
 4: for each $Labels$ in $L$ do
 5:     $IRLbl_{label} \leftarrow$ calculate Imbalance ratio for each Label($D$, label)
 6:     if $IRLbl_{label} > MeanIR$ then
 7:         ▷ Minority labels samples
 8:         $Bag \leftarrow$ get All Instances Of Label(labels)
 9:         for each $selected$ in $Bag$ do
10:             $dist \leftarrow$ calculate Distance($selected$, $Bag$)
11:             sort Smaller To Largest($dist$)
12:             ▷ Neighbor set selection
13:             $Neighbors \leftarrow$ get Head Items($dist, k$)
14:             $refNeigh \leftarrow$ get Random Neighbor($Neighbors$)
15:             ▷ Generation of feature and label set
16:             $Synthetic\_sample \leftarrow$ SYNTHETIC($selected, refNeigh, neighbors$)
17:             $Z \leftarrow Z + Synthetic\_sample$
18:         end for
19:     end if
20: end for
21: ▷ function for the new synthetic sample
22: function SYNTHETIC($selected, refNeigh, neighbors$)
23:     $Synthetic\_sample \leftarrow new\_Sample$          ▷ Create new empty instance
24:     ▷ Assignment of feature set
25:     for each $feature$ in$Synthetic\_sample$ do
26:         $diff \leftarrow refNeigh.feat - selected.feat$
27:         $offset \leftarrow diff * randonInterval(0, 1)$
28:         $value \leftarrow selected.feat + offset$
29:     end for
30:     ▷ Assignment of label set
31:     $label\_Counts \leftarrow counts(selected.labels)$
32:     $label\_Counts+ \leftarrow counts(neighbors.labels)$
33:     $Labels \leftarrow label\_Counts > (k + 1) * 0.5$
34:     $Synthetic\_sample.Labels \leftarrow Labels.$
35:     return $Synthetic\_sample$
36: end function
```

instances, are selected for further functions. This relieves us from setting any manual threshold for deciding whether any label is a tail-label or not.

For each tail instance, the k- nearest neighbours are found using distance obtained between that tail instance and remaining ones. Feature space can have mixed attributes, hence, distance for numerical features is found using euclidean distance while for the categorical features, Value Difference Metric (VDM) is used. Randomly one of the neighbour is picked which acts as the reference neighbor.

The features of the new synthetic instance is determined by the distance in-between the tail instance under consideration and the reference neighbour. The distance is multiplied with a random noise belonging to [0,1],this gives us an offset. This offset is then added to the features of the tail instance and we get the features of the synthetic instance. This approach is mostly used for numerical features while nominal features are decided simply by majority.

Labelset of the new instance is decided using ranking. For each label, frequency is recorded among the reference instance and its neighbours. The new instance is assigned the labelset existing across half or more of the instances.

## 2.2 Proposed Work

Tail labels present in multi-label dataset are quite a challenge in multi-label classification. Our algorithm join forces between two of the SOTA methods for tackling imbalanced multi-label classification – LLSF_DL and MLSMOTE.

An optimal algorithm should have the following capabilities for handling tail labels:

1. The algorithm can model higher order labelset correlations as well as pairwise labels correlation for better labelset learning.

2. The algorithm should avoid error propagation meanwhile discard unnecessary label dependencies.

3. For efficient learning, only label-specific features should be used for responsible labels prediction.

4. The algorithm should also capture imbalance among labels.

5. Also, the algorithm should provide better generalization over base models for both tail labels as well as for all labels.

6. The algorithm should be scalable in terms of handling extremely large labelspace (Extreme Multi-label Classification) and time-efficient.

The proposed algorithm has most of these capabilities.

**Algorithm 3**

**Input:** Training Matrix $X \in \mathbb{R}^{m*d}$,Label Matrix $Y \in \{0,1\}^{m*l}$,Test Matrix $X_t \in \mathbb{R}^{n*d}$ , Weight parameters $\alpha$, $\beta$, $\gamma$, $\rho$, Threshold $\tau$, tuning parameter $\kappa$, drop_majority $\in \{true, false\}$, no of nearest neighbours $a$

**Output:**Predicted Label Matrix $\hat{Y_t} \in \{0,1\}^{n*l}$

```
1:  Find $MeanIR$ by Eq 3.2
2:  $Y_{tail\_labels} \leftarrow$ all tail labels for which $IRLbl \geq MeanIR$.
3:  repeat
4:      if $drop\_majority == True$ then
5:          $Y \leftarrow Y_{tail\_label}$
6:      end if
7:      $X_{new}, Y_{new} \leftarrow MLSMOTE([X, Y], a)$
8:      $X \leftarrow X \cup X_{new}$
9:      $Y \leftarrow Y \cup Y_{new}$
10:     $\hat{Y} \leftarrow LLSF - DL(X, Y, X_t, \alpha, \beta, \gamma, \rho, \tau, \kappa)$
11:     return $\hat{Y}$
12: until Stopping condition reached
```

Our algorithm uses the feature selection strategy of LLSF_DL for assigning labels specific features to prediction of the associated labels. The L1-regularisation introduced in the former algorithm zeroes out weights of unnecessary labels while prediction for uncorrelated labels, hence, prevents error propagation. Global as well as local correlations among features a well as labels are captured explicitly using correlation terms in the objective function directly. Our algorithm has overall time- complexity O( $l$ + iter( $l_{tail} * (l_{tail} * |x_{tail}|^2) + t * (d^2 l + dl^2 + l^3)))$ which is approximately equal to O(max$(d^2 l, l^3))$. Here, $l$ stands for number of labels, *iter* for number of iterations, $l_{tail}$ for number of tail labels, $|x_{tail}|$ for number of tail instances.

# Chapter 3

# Experiment Design

## 3.1 Requirement Specification

**Hyper-parameter Selection:** In our experiment, we have used the default hyper-parameter range as set by the authors. For MLSMOTE, for the reference point, the number of nearest neighbours to be selected is kept as it is,i.e. 5. Similarly, for LLSF_DL. $\alpha, \beta, \gamma$ have been searched in range $\{4^{-5}, 4^{-4}, 4^{-3}, ..., 4^{5}\}$,$\rho$ in $\{0.1, 1, 10\}$,threshold is kept 0.5 and the tuning parameter $\kappa$ as 3. The hyper-parameters of our model could not be tuned using Grid-search on our commodity hardware. We believe our model can achieve even better performance with carefully tuned hyper-parameter set.

## 3.2 Flowcharts

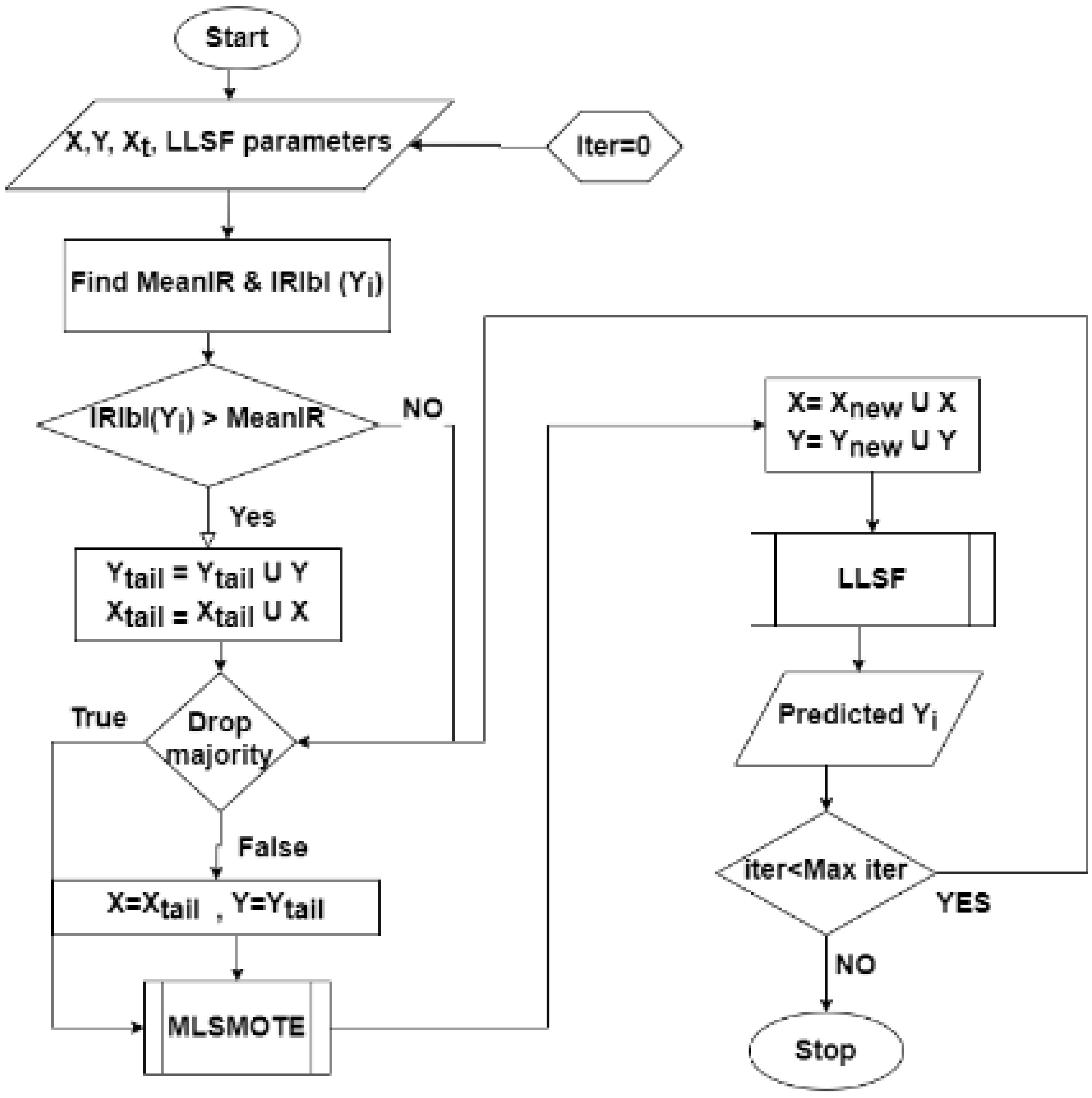


Figure 3.1: Flow chart of algorithm

## 3.3 Test Criteria

### 3.3.1 Metrics

**Imbalance Metrics:** For convenience, we represent a multi-label dataset $D$ with $Y$ labels. $Y_i$ represents $i_{th}$ label.

1. **Imbalance Ratio per Label (IRLbl):** This is the ratio of the label $y$ with the majority label . Most frequent label has value 1 while other labels have values >1. Higher imbalance is reflected by a higher IRLbl value. Total instances present is given by $|D|$, $Y_i$ being $i^{th}$ instance in label matrix $Y$, $y$ being any label and $y^{'}$ is the predicted label.

$$IRLbl(y) = \frac{\max\limits_{y' \in Y}(\sum_{i=1}^{|D|} h(y', Y_i))}{\sum_{i=1}^{|D|} h(y, Y_i)} \qquad h(y, Y_i) = \begin{cases} 1, & y \in Y_i \\ 0, & y \notin Y_i \end{cases} \tag{3.1}$$

2. **Mean Imbalance Ratio (MeanIR):**

   Represents the average imbalance in $D$.

$$MeanIR = \frac{1}{|Y|} \sum_{y=Y_1}^{Y_{|y|}} (IRLbP(y)) \tag{3.2}$$

3. **Coefficient of Variation of IRLbl (CVIR):**

   Indicates differences in imbalance level for all labels. Higher this difference, higher the CVIR value

$$CVIR = \frac{IRLbl\sigma}{MeanIR} \qquad IRLbl\sigma = \sqrt{\sum_{y=Y_1}^{Y_{|y|}} \frac{(IRLbl(y) - MeanIR)^2}{|Y| - 1}} \tag{3.3}$$

For label-wise evaluation, IRLbl is an useful measure. At labelset-wise, different labelsets can have same MeanIR value. Thus, both MeanIR and CVIR are used to decide whether an MLD is imbalanced or not. Any $D$ with a MeanIR value $> 1.5$ (in average, majority label instances exist 0.5 times more than of minority label instances) and CVIR value $> 0.2$ ( $\pm 0.2$ variance ) are termed imbalanced. For datasets with CVIR value $< 0.5$ and MeanIR less than 1.5, re-sampling hardly works [4].

**Multi-label Classification Metrics:** Most of the MLC metrics are extension of their binary counterparts. These metrics fall into two categories:

1. **Example-based:** assess the difference between the true and predicted labelsets.These metrics evaluates performance separately for each test instance, and the averaged value across the test set is returned.

   **Hamming loss:** Measures the average of misclassification frequency for instance-label pairs. $\triangle$ denotes symmetric difference.

   $$\text{Hamming loss} = \frac{1}{|Y|}\sum_{i=1}^{|Y|}\frac{1}{l}[y^{'}_{i}\triangle Y_i] \tag{3.4}$$

   **Subset Accuracy**: Reports the percentage of predicted label sets that contain an error. Both incorrect and partially correct predictions are not considered as exact match is required.

   $$\text{Subset Accuracy} = \frac{1}{|Y|}\sum_{n=1}^{|Y|}1(y^{'}_{n} = y_n] \tag{3.5}$$

2. **Label-based:** Here, for each label,performance is measured separately but we can take average in macro/micro across all labels. Suppose B is a binary metric (e.g., precision & recall, accuracy, or F1-score),$q$ be total labels present, true positive($tp$),true negative($tn$), false positive($fp$), false negative($fn$). Then, for each label,wherever one metric is computed and the values are averaged over all labels,this is the macro-averaged score. By contrast, whenever predictions for all instances are considered together and averaged across all labels, this is micro-averaged score.

   $$B_{macro} = \frac{1}{q}\sum_{i=1}^{q}B(tp_i, fp_i, tn_i, fn_i) \tag{3.6}$$

$$B_{micro} = B(\sum_{i=1}^{q} tp_i, \sum_{i=1}^{q} fp_i, \sum_{i=1}^{q} tn_i, \sum_{i=1}^{q} fn_i, ) \tag{3.7}$$

For every instance,equal weightage is assigned in micro-averaged scores and performance among frequent labelsets overshadow all others'. Similarly,for each label,equal weightage is assigned in macro-averaged score irrespective of its frequency and performance on tail-labels has a greater influence.

The uneven weightage assigned to each labels by Macro-averaged F1 score and subset accuracy allows tail labels to influence the metrics easily. Hence, these two shall be our optimal metric while others will be satisfying metrics.

## 3.4 Testing Process

For validation of our algorithm, we have used standard MLDs. In the Dataset description Table 3.1, we have provided metadata like tail labels and domain along with number of features, labels and instances respectively. Datasets that we have used can be downloaded from mulan[1].

Table 3.1: Data Sets Statistics.

| Data sets | Instances | Features | Domain | Labels | Tail Labels |
|---|---|---|---|---|---|
| Emotions | 593 | 72 | music | 6 | 4 |
| Genbase | 662 | 1186 | biology | 27 | 8 |
| Rcv1 | 6000 | 944 | text | 101 | 24 |
| Recreation | 5000 | 606 | text(web) | 22 | 7 |

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

# Chapter 4

# Result and Analysis

We wanted to know whether there are any explicit improvements in the classification of tail labels. Hence, we divided our experiment into two parts,namely, for overall data and for tail instances. In the former approach, we have used the given whole Y matrix while in the later part, we have trained the classifier only using the tail labels directly.

The results have been showcased in this chapter.

Table 4.1: Experimental results on different evaluation metrics

**HAMMING LOSS**

| Dataset | For overall data | | | For minority instances | | |
|---|---|---|---|---|---|---|
| | Iter - 1 | Iter - 2 | Iter - 3 | Iter - 1 | Iter - 2 | Iter - 3 |
| Emotions | 0.2215 | 0.2313 | 0.2473 | 0.1981 | 0.2066 | 0.2175 |
| Genbase | 0.0024 | 0.0031 | 0.0032 | 0.0029 | 0.0032 | 0.0032 |
| RCV1 | 0.0454 | 0.0507 | 0.0606 | 0.0028 | 0.0042 | 0.0073 |
| Recreation | 0.056 | 0.0568 | 0.0596 | 0.0127 | 0.0164 | 0.0185 |

**SUBSET ACCURACY**

| Dataset | For overall data | | | For minority instances | | |
|---|---|---|---|---|---|---|
| | Iter - 1 | Iter - 2 | Iter - 3 | Iter - 1 | Iter - 2 | Iter - 3 |
| Emotions | 0.2514 | 0.2446 | 0.2108 | 0.3931 | 0.4049 | 0.393 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Genbase | 0.9515 | 0.938 | 0.9425 | 0.9788 | 0.9758 | 0.9758 |
| RCV1 | 0.022 | 0.0168 | 0.0093 | 0.9415 | 0.9218 | 0.8733 |
| Recreation | 0.301 | 0.2756 | 0.2168 | 0.9226 | 0.9048 | 0.8916 |

**LABEL-BASED MACRO F1-MEASURE**

| Dataset | For overall data | | | For minority instance | | |
|---|---|---|---|---|---|---|
| | Iter - 1 | Iter - 2 | Iter - 3 | Iter - 1 | Iter - 2 | Iter - 3 |
| Emotions | 0.6548 | 0.6496 | 0.6241 | 0.5686 | 0.6082 | 0.6353 |
| Genbase | 0.9744 | 0.9673 | 0.966 | 0.2967 | 0.3017 | 0.3017 |
| RCV1 | 0.3751 | 0.352 | 0.2871 | 0.0026 | 0.0161 | 0.0125 |
| Recreation | 0.2932 | 0.3033 | 0.2413 | 0.121 | 0.1462 | 0.1256 |

**LABEL-BASED MICRO F1-MEASURE**

| Dataset | For overall data | | | For minority instances | | |
|---|---|---|---|---|---|---|
| | Iter - 1 | Iter - 2 | Iter - 3 | Iter - 1 | Iter - 2 | Iter - 3 |
| Emotions | 0.7937 | 0.7746 | 0.7445 | 0.577 | 0.6075 | 0.6302 |
| Genbase | 0.9872 | 0.9872 | 0.986 | 0.6514 | 0.6533 | 0.6533 |
| RCV1 | 0.4918 | 0.4655 | 0.3912 | 0.0245 | 0.0503 | 0.0582 |
| Recreation | 0.4557 | 0.4243 | 0.3422 | 0.1556 | 0.1552 | 0.1443 |

We have particularly selected the above four metrics to get better idea regarding the performance of the classifier. The first two metrics focus on instance-wise prediction performance while the last two focus on label wise prediction performance. Out of all these metrics, label based macro F1- measure is very sensitive towards rare labels. Hence, we have chosen this metric as our optimal metric while all others act as satisfying metrics.

The algorithm have performed marginally well in modelling tail labels for smaller datasets. We have the best results from Emotions data set. Increment in the label based macro f1-measure for all datasets shows that we are on the right track.

However, for overall dataset, our algorithm performs poorly in our optimal metric. With each iteration, there is a decrease in performance. Main reasons for such performance are:

1. The base classifier LLSF_DL needs to be modified further to improve its prediction capacities.

2. Poor hyper-parameter tuning. We have set the values randomly. Grid-search seemed unfeasible for an algorithm with time complexity O(max($d^2l, l^3$) ) over commodity hardware.

3. The generated data may not be of high quality. It is because MLSMOTE generates new instances on the basis of features and after that the labels are assigned simply using majority voting.Thus, samples are generated based upon the marginal distribution P(X|D). But these does not model inherent distribution of data which is a joint distribution P(X,Y|D).

4. There is no check and balance mechanism in-between data generation part and classification. Hence their assembly is performing poorly. To site a case, the MLSMOTE algorithm keeps generating new samples without any check implicitly effected by the classifier, hence the sample distribution of the overall dataset keeps getting skewed.This further introduces imbalances.

# Chapter 5

# Conclusion & Future Work

Though our algorithm is capable of handling tail labels, the experiments show that more improvements are needed. The algorithm has increased label based macro F1-measure for all datasets. It is very effectively dealing with smaller datasets. The drawbacks in designing an standalone algorithm for all type of tail labels has proved to be an eye-opener. Further, each component of algorithm - data generation and classification are to be worked upon in order to avoid the previously stated drawbacks. At the best, the SOTA classifier -LLSF_DL alone is merely 38% - 40% accurate (rcv1,recreation) in predicting the labels. Hence, our prediction accuracy also suffers. Similarly, the data generator should model the joint distribution of X and Y. Further, hyperparameter tuning should be carried on specialised hardwares. We believe these pointers for future works can improve our models' predictions.

Extending the research,the future course of action are being figured out. Application of Generative Adversarial Networks to generate samples are showing great results in better quality data generation. Similarly, recent papers on multilabel classifiers have performed multilabel classification by using non-sparse label specific features, give even better predictions. Both the aspects are being evaluated before incorporating them.

# Appendix A

# Accelerated Proximal Gradient (APG)[1]

It is used which optimises a sequence of quadratic convex approximations of the objective function that are easily separable. We write the objective function as follows in line to APG:

$$\min_{W \in H} [F(W) = f(W) + g(W)]. \tag{A.1}$$

$$f(\mathbf{W}_x, \mathbf{W}_y) = \frac{1}{2}\|\mathbf{X}\mathbf{W}_x + \mathbf{Y}\mathbf{W}_y - \mathbf{Y}\|_F^2 + \frac{\alpha}{2}\mathrm{Tr}(\mathbf{R}\mathbf{W}_x^T\mathbf{W}_x) \tag{A.2}$$

$$g(\mathbf{W}_x, \mathbf{W}_y) = \beta\|\mathbf{W}_x\|_1 + \gamma\|\mathbf{W}_y\|_1. \tag{A.3}$$

Both functions are convex. $f$ is Lipschitz differentiable but g need not be. Accelerated Proximal gradient methods are applied which optimises a sequence of separable quadratic approximations. In general,it is as follows:

$$Q_{L_f}(\mathbf{W}, \mathbf{W}^{(t)}) = f(\mathbf{W}^{(t)}) + \langle \nabla f(\mathbf{W}^{(t)}), \mathbf{W} - \mathbf{W}^{(t)} \rangle + \frac{L_f}{2}\|\mathbf{W} - \mathbf{W}^{(t)}\|_F^2 + g(\mathbf{W}). \tag{A.4}$$

Minimizing the above function with respect to W gives us the following:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} Q_{L_f}(\mathbf{W}, \mathbf{W}^{(t)}) \tag{A.5}$$

For $w \in R$ & $\varepsilon > 0$, the operation of soft-thresholding function can be defined as:

$$S_\varepsilon[w] = \begin{cases} w - \varepsilon, & \text{if } w > \varepsilon \\ w + \varepsilon, & \text{if } w < -\varepsilon \\ 0 & \text{otherwise.} \end{cases} \tag{A.6}$$

$$\mathbf{W}^{t+1} = S_\varepsilon[\mathbf{G}^{(t)}] = \arg\min_{\mathbf{W}} \varepsilon\|\mathbf{W}\|_1 + \frac{1}{2}\|\mathbf{W} - \mathbf{G}^{(t)}\|_F^2. \tag{A.7}$$

# Appendix B

# LLSF algorithm

**Algorithm 4** LLSF [9]

**Input:** Training Matrix $X \in \mathbb{R}^{m*d}$,Label Matrix $Y \in \{0,1\}^{m*l}$, Weight parameters $\alpha$, $\beta$, $\gamma$, $\rho$, Test Matrix $X_t \in \mathbb{R}^{n*d}$,Threshold $\tau$ .

**Output:** Predicted Label Matrix $Y_t \in \{0,1\}^{n*l}$

```
 1: Initialization:
 2: $b_0 \leftarrow 1$
 3: $b_1 \leftarrow b_0$
 4: $t \leftarrow 1$
 5: $W_x^{(0)} \leftarrow (X^T X + \rho I)^{-1} X^T Y$
 6: $W_x^{(1)} \leftarrow W_x^{(0)}$
 7: $R \leftarrow \frac{Y^T Y}{||Y||_2^2}$                                   ▷ Cosine similarity
 8: $W_f \leftarrow [3(||X^T X||_2^2 + ||X^T Y||_2^2 + ||\alpha R||_2^2) + 2(||Y^T Y||_2^2)]^{1/2}$
 9: repeat
10:     ▷ Calculate Gradient descent
11:     $W_x^{(t)} \leftarrow W_x^{(t)} + \frac{b_{t-1}-1}{b_t}(W_x^{(t)} - W_x^{(t-1)})$
12:     ▷ Regularization
13:     $G_x^{(t)} \leftarrow W_x^{(t)} - \frac{1}{L_f}[(X^T X W_x) + X^T Y W_y - X^T Y + \alpha W_x R]$
14:     $W_x^{(t+1)} \leftarrow soft\_threshold(G_x^{(t)}, (\frac{\rho}{L_f}))$
15:     $b_{t+1} \leftarrow 0.5(1 + [4b_t^2 + 1]^{0.5})$
16:     $t \leftarrow (t+1)$
17: until Stopping criteria not reached
18: $S_t \leftarrow X_t W_x$
19: $\hat{Y}_t \leftarrow sign(S_t - \tau)$
```

# References

[1] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 693–696, 2009.

[2] Matthew Boutell, Jiebo Luo, Xipeng Shen, and Christopher Brown. Learning multi-label scene classification. *Pattern Recognition*, 37:1757–1771, 09 2004.

[3] Paula Branco, Luís Torgo, and Rita Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys*, 49, 08 2016.

[4] F. Charte, Antonio Rivera Rivas, María José Del Jesus, and Francisco Herrera. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163, 04 2015.

[5] F. Charte, Antonio Rivera Rivas, María José Del Jesus, and Francisco Herrera. Mlsmote: Approaching imbalanced multilabel learning through synthetic instance generation. *Knowledge-Based Systems*, pages –, 09 2015.

[6] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 01 2002.

[7] Feng Kang, Rong Jin, and R. Sukthankar. Correlated label propagation with application to multi-label learning. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1719–1726, 2006.

[8] Eva Gibaja and Sebastian Ventura. A tutorial on multi-label learning. *ACM Computing Surveys*, 47, 04 2015.

[9] J. Huang, G. Li, Q. Huang, and X. Wu. Learning label specific features for multi-label classification. In *2015 IEEE International Conference on Data Mining*, pages 181–190, 2015.

[10] J. Huang, G. Li, Q. Huang, and X. Wu. Learning label-specific features and class-dependent labels for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3309–3323, 2016.

[11] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 4 2016.

[12] Andrew Mccallum. Multi-label text classi cation with a mixture model trained by em. 11 1999.

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

[14] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, and Tao Mei. Correlative multi-label video annotation. pages 17–26, 01 2007.

[15] F. Sun, J. Tang, H. Li, G. Qi, and T. S. Huang. Multi-label image categorization with sparse factor representation. *IEEE Transactions on Image Processing*, 23(3):1028–1037, 2014.

[16] Konstantinos Trochidis, Grigorios Tsoumakas, George Kalliris, and I. Vlahavas. Multi-label classification of music into emotions. volume 2011, pages 325–330, 01 2008.

[17] Xi Wang and Gita Sukthankar. Multi-label relational neighbor classification using social context features. *Proceedings of the 19th ACM SIGKDD*, pages 464–472, 01 2013.

[18] Bin Wu, Erheng Zhong, Andrew Horner, and Qiang Yang. Music emotion recognition by multi-label multi-layer multi-instance multi-view learning. *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, pages 117–126, 11 2014.

[19] M. Zhang and Z. Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.