Multi-label classification via labels correlation and one-dependence features on data stream

Tien Thanh Nguyen , Thi Thu Thuy Nguyen , Anh Vu Luong , Quoc Viet Hung Nguyen , Alan Wee-Chung Liew , Bela Stantic

PII: S0031-3203(19)30012-3
DOI: https://doi.org/10.1016/j.patcog.2019.01.007
Reference: PR 6766

To appear in: *Pattern Recognition*

Received date: 19 May 2018
Revised date: 12 December 2018
Accepted date: 4 January 2019

Highlights:

- A Bayesian-based multi-label online learning method for multi-label data stream classification is proposed.
- Our method not only learns the label correlation from each arrived sample but also dynamically determines the number of predicted labels based on Hoeffding inequality and the label cardinality.
- Our method can also handle missing values and concept drifts in the data stream effectively.
- Extensive comparative experiments with the state-of-the-art algorithms validate the superior performance of our method.

Multi-label classification via labels correlation and one-dependence features on data stream

Tien Thanh Nguyen[1], Thi Thu Thuy Nguyen[2], Anh Vu Luong[3], Quoc Viet Hung Nguyen[2], Alan Wee-Chung Liew[2*], Bela Stantic[2]

[1]School of Computing Science and Digital Media, Robert Gordon University, Aberdeen, Scotland, UK

[2] School of Information and Communication Technology, Griffith University, Australia

[3] School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, Vietnam

[*]Corresponding Author: Alan Wee-Chung Liew

Email: a.liew@griffith.edu.au

Abstract: Many batch learning algorithms have been introduced for offline multi-label classification (MLC) over the years. However, the increasing data volume in many applications such as social networks, sensor networks, and traffic monitoring has posed many challenges to batch MLC learning. For example, it is often expensive to re-train the model with the newly arrived samples, or it is impractical to learn on the large volume of data at once. The research on incremental learning is therefore applicable to a large volume of data and especially for data stream. In this study, we develop a Bayesian-based method for learning from multi-label data streams by taking into consideration the correlation between pairs of labels and the relationship between label and feature. In our model, not only the label correlation is learned with each arrived sample with ground truth labels but also the number of predicted labels are adjusted based on Hoeffding inequality and the label cardinality. We also extend the model to handle missing values, a problem common in many real-world data. To handle concept drift, we propose a decay mechanism focusing on the age of the arrived samples to incrementally adapt to the change of data. The experimental results show that our method is highly competitive compared to several well-known benchmark algorithms under both the stationary and concept drift settings.

Keywords: multi-label classification; multi-label learning; online learning; data stream; concept drift; label correlation; feature dependence.

1. Introduction

In recent years, multi-label classification (MLC) has generated a great deal of interest in the machine learning community [1]. This problem naturally arises in many real-world applications where objects are described by multiple semantic meanings. As a generalization of multi-class classification, the task of MLC is to assign a set of correct labels to an object to express its semantics. MLC has been applied in many applications such as automatic image and video annotation, bioinformatics, and web mining [1].

In the past, most research on MLC is focused on the batch learning paradigm in which the model is learned on a given training set. With the development of sensing and storage technologies, large volumes of data are being generated continuously from sources such as social networks, email, blogs, RSS feeds, IoT, etc. This has posed many challenges to batch learning systems with issues such as expensive retraining when dealing with newly arrived samples or difficulty with learning on a very large volume of data at once [2-4]. The research on incremental learning is therefore much needed with the continual growth of data, especially with data stream.

Bifet and Gavaladà [5] defined four characteristics of learning on streaming data: (1) the model must be ready to make prediction on any sequentially arriving instances, (2) there are potential infinitely many instances which need to be processed under finite resources (time and memory), (3) the instances need not be statistically stationary (the appearance of concept drift), and (4) instances can only be processed one at a time, and can only be inspected once before it is discarded. These characteristics set the requirements for the design of learning systems for multi-label streaming data. In online learning, the learning model can be updated with each arrived sample (instance-incremental) or with each data chunk (batch-incremental). Read et al. [6] showed the disadvantages of the batch-incremental over the instance-incremental learning due to the need to store arriving instances in batch and the delay in waiting for the batch to fill up.

In the literature, various MLC algorithms were introduced under both the batch and stream settings [1, 6, 7]. One of the most common approaches solving MLC is the technique of problem transformation where the MLC task is transformed into several well-established learning tasks, for example, into binary classification tasks [1, 7]. Although several methods in this category have attracted the attention of researchers due to their conceptual simplicity, the ignorance of label correlation in their transformation has caused degradation in the prediction performance [1, 8]. In this study, we adopted the problem transformation approach due to its simplicity but also taking into account label correlation in the multi-label prediction process. Here we exploit the label correlation by considering the semantic relationship a label has with the previously assigned labels on that sample. For example, if a sample has previously been assigned with the label 'indoor', labels like 'table' and 'chair' should have more chance of been assigned to the sample than labels like 'car' and 'grass'. Since we focus on working in the streaming context, this label correlation exploitation approach will be maintained with the incoming of each sample. In addition, in previous work [6, 9], the number of labels to be assigned to each arrived sample is only updated after a fixed set of samples are obtained. An efficient and flexible mechanism to learn this value from the streaming data is expected to improve the quality of the prediction.

The main contributions of this paper are:

- A Bayes learning model for online MLC that considers the pairwise label correlation.
- A novel method to infer the number of labels for an incoming sample.
- A decay method that allows incremental adaptation to concept drift.
- An empirical demonstration that our method is better than several well-known MLC algorithms under both stationary and concept drift settings.

The paper is organized as follow. In Section 2, we briefly introduce several well-known learning algorithms for MLC. In Section 3, we present the proposed multi-label Bayes learning algorithm that considers label correlation. We also describe the model prediction and update process, dealing with missing values, learning the number of predicted labels, and adaptation to concept drift. In Section 4, we describe the experimental setup and the benchmark algorithms used in the study. In Section 5, we present the experimental results and discussions. In Section 6, we draw some conclusions and suggest further studies.

2. Background

2.1. **Multi-label classification**

MLC is the generalization of multi-class classification in which samples are assigned with multiple labels simultaneously [10]. Formally, let $\mathbb{X}$ denote the input space and $\mathbb{Y} = \{y_i | i = 1, \dots, M\}$ denote the label set. The task of multi-label learning is to derive a function f mapping from input space $\mathbb{X}$ to output space $2^{\mathbb{Y}}$ so that each sample $\mathbf{x} \in \mathbb{X}$ is assigned with a subset of the output space. This is in contrast to the traditional multi-class classification in which the output space is simply the label set $\mathbb{Y}$, which means each sample is associated with only a single label. In this section, we briefly review some well-known multi-label algorithms, focusing on those which have been extended to deal with streaming data.

MLC algorithms can often be categorized into two approaches: problem transformation and algorithm adaption [1]. In the first category, the MLC problem is transformed into well-established learning problems, such as binary classification problem. The most common approach in the problem transformation category is the Binary Relevance (BR) technique where a multi-label task is transformed into M independent binary classification tasks, each of which is associated with a label in the label set [11]. Although BR provides a simple and straightforward approach to solve the multi-label problem, treating label independently could reduce the system performance due to the neglect of label dependency [1, 8].

Classifier Chains (CC) is another popular method in this category, which has been used as the baseline in many comparisons. CC also learns M binary classifiers like in BR but in CC, the new training set for each binary problem is created by appending each instance with binary values that indicate which of the previous labels were assigned to that instance [12]. In practice, several CC classifiers generated with random orders over the label space are trained in the form of the ensemble to enhance the performance. CC has the advantage of addressing label correlation, however, because of the dependency in the chains, CC cannot be implemented in parallel. Several methods have also been introduced to improve CC's effectiveness, such as replacing binary values by probabilistic outputs [13], finding good chain sequence by Monte Carlo method [14], and using recurrent neural network focusing only on positive labels as an extension of probabilistic CC [15]. Other popular approach, which takes into account label correlations is Label Powerset (LP) in which each different combination of labels is considered to be a single label [16]. The drawbacks of this method are the high complexity in training due to the exponential increase of the number of label combinations with the number of labels, and the inability to predict the label combinations that do not appear in the training set [1]. Other MLC approaches considered the subsets of labels in a random way such as Random k-Labelsets [16, 17], or in a deterministic way like dependency network [18]. Read et al. [19] proposed Pruned Set method to reduce the number of label combinations by removing the samples belonging to specific infrequent labelsets.

The second category, i.e., algorithms adaption, is the group of methods which are modified from multi-class classification algorithms to handle the MLC problem. Several methods can be mentioned, for example, k-Nearest Neighbor for MLC [20], Rank-SVM adapting kernel techniques [21], Decision Tree for MLC [22, 23], and Naïve Bayes for MLC [24].

Recently, with the advances in data collect and storage technology, large volumes of data with diverse nature has emerged. The recent research on multi-label learning in the era of big data therefore mainly focus on dealing with large-scale multi-label (ML) data, especially on data with the large label set, high dimensionality and data in the stream. Several strategies have emerged. First, to handle large label sets, the two strategies, namely compression and projection, have been used in MLC to reduce the label dimension. Ubaru and Mazumdar [25] used group testing and coding techniques to compress the label set. Kapoor et al. [26] used compressed sensing in the Bayesian framework for label dimension reduction. Label vector can also be projected onto a low dimensional space to reduce its dimension. Several techniques have been introduced for the label vector projection, for example, SVD [27] and canonical correlation analysis [28]. Second, feature selection for multi-label data has been proposed since performance can be improved by selecting an optimal subset of features to learn the MLC model. Some examples of feature selection method for MLC are feature rank-based stream feature selection method [29] and scalable relevance evaluation feature selection that measures feature dependency [30]. Third, dimensionality reduction methods have been proposed to reduce the high dimensionality of data [31]. Zhang and Zhou [32] performed ML dimensionality reduction by maximizing the dependence between the feature and the associated labels. ML sparse coding conducts a label sparse coding-based subspace learning algorithm to reduce data dimensionality [33].

2.2. **Multi-label classification for data stream**

Streaming data has created many challenges for traditional offline machine learning systems. First, storing increasingly large volumes of data in the limited memory as well as learning the entire data at once is often impractical or even infeasible. Moreover, offline algorithms require re-training when new data is available so that they are not applicable to the situation where data is arriving continuously, and the prediction model must be obtained before all data are available. Hence, learning frameworks that can deal with streaming data have become increasingly popular [2].

An approach that solves the MLC problem in the stream context was the batch-based incremental method [34] in which the learning model is learned on a sequence of same-size-chunks. In detail, the Binary Relevance (BR) method learns on each chunk to obtain the ensemble of classifiers, and these classifiers' outputs are concatenated to the original feature space as the meta-data which is learned by a second BR. For each test sample, the weights for the classifiers are determined via the dynamic classifier ensemble approach [35] i.e. based on the performance of the classifiers on the test sample's neighbors in the latest chunk. In a similar paradigm, Wang et al. [36] generated an ensemble of classifiers by using the multi-label K Nearest Neighbor method [20] on the fixed number of chunks obtained from the data stream. Each classifier is weighted, and the weights are updated with the newly arrived chunks. Although these two methods can handle both stationary data and concept drift in multi-label learning, they cannot satisfy the time and memory constraints of stream learning as they face the memory-fill-up problem [6].

For the algorithm adaption paradigm, Read et al. [6] proposed the incremental multi-label decision tree which is the adaptation of Hoeffding tree for multi-label data stream. The Hoeffding tree [37] is a well-known member of decision tree family developed to solve streaming data classification problem. In the Hoeffding tree algorithm, Hoeffding bound is used in tree induction to determine how many examples are needed to achieve a certain level of confidence for tree splitting. The multi-label Hoeffding tree was incorporated with Pruned Set classifier [19] to prune the label combination at each leaf node when the buffer of arrived samples in this node is full. That framework was also combined with ADWIN Bagging [38] (namely EaHTps) to deal with the problem of change detection and adaptation. Shi et al. [39] proposed the improvement for EaHTps by dynamically recognizing the new frequent label combinations and updating the set of label combination.

Recently, Xioufis et al. [9] employed BR to solve the MLC by transforming the multi-label task into several binary classification tasks. In their method, concept drift was handled by maintaining two variable-size windows per label for positive and negative examples. Shi et al. [40] used the Apriori algorithm and the EM algorithm to divide the set of class labels into different subsets in which the labels are grouped based on their dependencies. These subsets are treated as new class labels to be used to annotate each arrived sample. To handle concept drift, the authors first measured the distribution of features and new class labels by a multi-label entropy approach. The change of the distribution was then monitored by the difference of the entropy measure between the two windows that keep the old and the recent samples. Once the changes happened, i.e. the difference exceeds the pre-defined threshold, these windows are resized. Osojnik et al. [7] applied multi-target regression to multi-label learning on the stream data but only focused on learning the stationary concept. Karponi and Tsoumakas [41] compared several multi-label data stream classification methods implemented in the MULAN and MEKA library [42, 43] such as Binary Relevance with different learning algorithms and multi-label Hoeffding tree with Pruned Set and Naïve Bayes at the leaves. The comparison was conducted on just one reduced dataset, therefore, the results are not convincing.

3. The proposed approach

3.1. **Problem analysis**

In this section, we formulate a learning model of the MLC problem for data streams by taking into account label correlation. Specifically, we assign a class label to each arrived sample in which the present assignment process depends on the previously predicted label set in the previous steps. Here we aim to take advantage of the information we obtained earlier, i.e., the dependence of the labels to the previously predicted labels, for the next prediction.

Denote $\hat{Y}_{\mathbf{x}}$ as the previously assigned label set for a sample **x**, the probability to assign another label for **x** conditioned on $\hat{Y}_{\mathbf{x}}$ is given by:

$$\mathrm{P}\left(y_m|\mathbf{x},\hat{Y}_{\mathbf{x}}\right)=\frac{\mathrm{P}(\mathbf{x},\hat{Y}_{\mathbf{x}}|\mathrm{y}_m)\mathrm{P}(y_m)}{\mathrm{P}(\mathbf{x},\hat{Y}_{\mathbf{x}})}\sim\mathrm{P}\left(\mathbf{x},\hat{Y}_{\mathbf{x}}|y_m\right)\mathrm{P}(y_m)\text{ for each } y_m\in\mathbb{Y}\text{ and } y_m\notin\hat{Y}_{\mathbf{x}} \tag{1}$$

We assume that **x** and each of the labels in $\hat{Y}_{\mathbf{x}}$ are conditionally independence given y_m. The left side of (1) becomes:

$$\mathrm{P}\left(\mathbf{x},\hat{Y}_{\mathbf{x}}|y_m\right)\mathrm{P}(y_m)=\mathrm{P}(\mathbf{x}|y_m)\prod_{i,\hat{y}_i\in\hat{Y}_{\mathbf{x}}}\mathrm{P}(\hat{y}_i|y_m)\,\mathrm{P}(y_m) \tag{2}$$

So we have:

$$\mathrm{P}\left(y_m|\mathbf{x},\hat{Y}_{\mathbf{x}}\right)\sim\mathrm{P}(\mathbf{x},y_m)\prod_{i,\hat{y}_i\in\hat{Y}_{\mathbf{x}}}\mathrm{P}(\hat{y}_i|y_m) \tag{3}$$

A new label is assigned to **x** by getting the label that maximizes $\mathrm{P}\left(y_m|\mathbf{x},\hat{Y}_{\mathbf{x}}\right)$:

$$\mathrm{P}\left(\hat{\mathrm{y}}|\mathbf{x},\hat{Y}_{\mathbf{x}}\right)=\max_{m=1,\dots,M}\mathrm{P}\left(y_m|\mathbf{x},\hat{Y}_{\mathbf{x}}\right)\sim\max_{m=1,\dots,M}\mathrm{P}(\mathbf{x},y_m)\prod_{i,\hat{y}_i\in\hat{Y}_{\mathbf{x}}}\mathrm{P}(\hat{y}_i|y_m) \tag{4}$$

The predicted label $\hat{\mathrm{y}}$ is added to the label set $\hat{Y}_{\mathbf{x}}$ as $\hat{Y}_{\mathbf{x}}=\hat{Y}_{\mathbf{x}}\cup\hat{\mathrm{y}}$. The new label set $\hat{Y}_{\mathbf{x}}$ is then used to learn another label for **x**. Two questions that arise from (4) are:

- How to estimate $\mathrm{P}(\mathbf{x},y_m)$ and $\mathrm{P}(\mathrm{y}_i|\mathrm{y}_m)$ given the limited memory and processing time.
- How to learn the set of labels $\hat{Y}_{\mathbf{x}}$ for each sample **x** in the data stream.

3.2. **Label correlation estimation**

To compute $\mathrm{P}\left(y_m|\mathbf{x},\hat{Y}_{\mathbf{x}}\right)$ in (4), we first estimate the conditional distribution between y_i and y_m. Based on Bayesian theorem, we have:

$$\mathrm{P}(\mathrm{y}_i|\mathrm{y}_m)=\frac{\mathrm{P}(\mathrm{y}_i,\mathrm{y}_m)}{\mathrm{P}(\mathrm{y}_m)} \tag{5}$$

This can be estimated from the data as:

$$\frac{\mathrm{P}(\mathrm{y}_i,\mathrm{y}_m)}{\mathrm{P}(\mathrm{y}_m)}=\frac{F(\mathrm{y}_i,\mathrm{y}_m)}{F(\mathrm{y}_m)} \tag{6}$$

in which $F(\mathrm{y}_m)$ and $F(\mathrm{y}_i,\mathrm{y}_m)$ are the counts of samples having label y_m, and the pair of labels y_i and y_m, respectively. As only a 2-dimensional label co-occurrence table is needed to store the frequencies, the space complexity is $\binom{M}{2}$.

In an online learning setting, the sequential arrival of samples and the sparsity of labels mean that obtaining a reliable estimate of the (full) joint distribution of labels is usually not practical. In addition, the correlation between some labels can be very weak since **x** can be described by multiple concepts that are uncorrelated. Although the conditional independence assumption between labels in the predicted label set $\hat{Y}_{\mathbf{x}}$ can at first seems overly-simplistic, it offers a practical solution to the issue of label sparsity while still allowing the exploitation of the pairwise correlation between labels.

3.3. **Joint distribution estimation**

If we assume that each feature x_i of **x** is conditionally independent to every other feature x_j given the class label y_m, $\mathrm{P}(\mathbf{x}, y_m)$ is then given by:

$$\mathrm{P}(\mathbf{x}, y_m) = \mathrm{P}(\mathbf{x}|y_m)\mathrm{P}(y_m) = \mathrm{P}(y_m)\prod_{i=1}^{D}\mathrm{P}(x_i|y_m) \tag{7}$$

The likelihood $\mathrm{P}(x_i|y_m)$ in turn is computed based on the assumption about the distribution of each feature x_i given class label y_m, such as $\mathcal{N}\left(\mu_{mi}, \sigma_{mi}^2\right)$ in which the parameters μ_{mi} and σ_{mi}^2 are computed from the training observations. Clearly, the joint distribution is simple to compute since the class label pior $\mathrm{P}(y_m)$ and the likelihood $\mathrm{P}(x_i|y_m)$ are all univariate distributions. The drawback of this approach is that the conditional independence assumption does not generally hold in practice. Although it has been observed that the violation of the conditional independence assumption does not have significant impact in practice [44], there are approaches that could relax the assumption while still keeping the simplicity and efficiency of the Naïve Bayes method [45-49].

In this study, we use a method called Aggregating One-Dependence Estimators (AODE) to estimate the joint distribution $\mathrm{P}(\mathbf{x}, y_m)$ [46]. The idea of AODE is that each feature only depends on one other feature (called parent feature), and the joint distribution is estimated by aggregating the results among all parent features. In detail, for a feature x_i, we have:

$$\mathrm{P}(y_m, \mathbf{x}) = \mathrm{P}(y_m, x_i)\mathrm{P}(\mathbf{x}|y_m, x_i) \tag{8}$$

To estimate $\mathrm{P}(y_m, \mathbf{x})$, instead of using the prior $\mathrm{P}(y_m)$ and the likelihood $\mathrm{P}(\mathbf{x}|y_m)$ like in Naïve Bayes, we consider a given parent feature x_i in the conditional probability $\mathrm{P}(\mathbf{x}|y_m, x_i)$ and the joint distribution $\mathrm{P}(y_m, x_i)$ as in (8). Since (8) holds for every parent features, it will hold for the mean over all D features in **x**. In detail, suppose that each feature x_i can take different discrete values v_i. Upon receiving an observation $\mathbf{x} = (x_1 = v_1, \dots, x_D = v_D)$, the joint distribution is aggregated by:

$$\mathrm{P}(y_m, \mathbf{x}) = \frac{\sum_{i, 1 \le i \le D}\mathrm{P}(y_m, x_i = v_i)\mathrm{P}(\mathbf{x}|y_m, x_i = v_i)}{D} \tag{9}$$

Since we are considering the dependence between a feature and its parent feature, the conditional probability $\mathrm{P}(\mathbf{x}|y_m, x_i = v_i)$ is given by:

$$\mathrm{P}(\mathbf{x}|y_m, x_i = v_i) = \prod_{j=1}^{D}\mathrm{P}\left(x_j = v_j|y_m, x_i = v_i\right) \tag{10}$$

So equation (9) becomes:

$$\mathrm{P}(y_m, \mathbf{x}) = \frac{\sum_{i,1\le i\le D} \mathrm{P}(y_m, x_i = v_i) \prod_{j=1}^{D} \mathrm{P}(x_j = v_j | y_m, x_i = v_i)}{D} \quad (11)$$

Moreover, the number of terms involved in the aggregation in (11) can be reduced by excluding terms where the number of samples with parent feature value $x_i = v_i$, i.e. $F(x_i = v_i)$ are less than a threshold T. Equation (11) becomes:

$$\mathrm{P}(y_m, \mathbf{x}) = \frac{\sum_{i,1\le i\le D \wedge F(x_i = v_i)\ge T} \mathrm{P}(y_m, x_i = v_i) \prod_{j=1}^{D} \mathrm{P}(x_j = v_j | y_m, x_i = v_i)}{|\{i: 1\le i\le D \wedge F(x_i = v_i)\ge T\}|} \quad (12)$$

In (12), if $\nexists i: 1 \le i \le D \wedge F(x_i = v_i) \ge T$, AODE defaults to the Naïve Bayes given in (7). In the experiment, we set $T = 1$ as suggested in [50].

In this study, we use Laplace smoothing for $\mathrm{P}(x_j = v_j | y_m, x_i = v_i)$ and $\mathrm{P}(y_m, x_i = v_i)$. The estimates are given by:

$$\mathrm{P}(y_m) = \frac{F(y_m)+1}{N+M} \quad (13)$$

$$\mathrm{P}(y_m, x_i = v_i) = \frac{F(y_m, x_i = v_i)+1}{N+M\times n_i} \quad (14)$$

$$\mathrm{P}(x_j = v_j | y_m, x_i = v_i) = \frac{\mathrm{P}(x_j = v_j, y_m, x_i = v_i)}{\mathrm{P}(y_m, x_i = v_i)} = \frac{F(y_m, x_i = v_i, x_j = v_j)+1}{F(y_m, x_i = v_i)+n_j} \quad (15)$$

in which $F(.)$ is the frequency count of the term, for example $F(y_m, x_i = v_i, x_j = v_j)$ is the count of the number of samples having class label y_m, x_i having value v_i, and x_j having value v_j; n_i and n_j are the number of distinct values for feature x_i and x_j, respectively; N is the number of arrived sample. It is noted that we only need a 3-dimensional table to store the frequencies $F(y_m, x_i = v_i, x_j = v_j)$ for the estimates in (13)-(15). Therefore, the space complexity of AODE is $\binom{\mathrm{M}(\sum_{\mathrm{i=1}}^{\mathrm{D}} \mathrm{n_i})}{2}$ or $\binom{\mathrm{MDn}}{2}$ in which n is the average number of values for the features. As the proposed method requires discrete valued data, in this study, each continuous feature is discretized into 5 bins (i.e. 5 distinct values for each feature) via the incrementally fixed bins discretization method in the MOA library [51].

While keeping the simplicity and efficiency of Naïve Bayes method, by considering the 1-dependence among features, AODE can provide a better estimate of $\mathrm{P}(\mathbf{x}, y_m)$ than Naïve Bayes where the 'naïve' condition is too strict and is often violated in practice. Moreover, the estimate of the joint distribution is obtained by aggregating multiple models associated with each features. Since many research [52-60] have shown that aggregating multiple models can improve the classification accuracy, the 1-dependence method is expected to enhance the performance of the system.

3.4. Model Update

We follow the update mechanism of some incremental instance-based MLC methods in which every arrived sample and its revealed true labels are used to update the learning model [6]. Meanwhile, the samples with unknown true labels would only be predicted.

After making the prediction, if the true label set Y of the arrived sample $\mathbf{x} = (x_1 = v_1, \dots, x_D = v_D)$ can be revealed from the environment, we update the current learning model. In this study, we propose a 2-step update process involving the update for the label dependence and for the label-feature-value relationship. As mentioned in section 3.2, we use a table to store the frequency of the co-occurrence of each label pair. For a pair of y_m and y_i in Y, we update its dependence by increasing its co-occurrence entry in the table as:

$$F(\mathrm{y}_m, \mathrm{y}_i) = F(\mathrm{y}_m, \mathrm{y}_i) + 1 \tag{16}$$

Second, based on the true labels Y and the value of each feature $x_i = v_i$ $i = 1, \dots, D$, we update the label-feature-value relationship by increasing the frequency of the term $(y_m, x_i = v_i,\ x_j = v_j)$. This can be done by updating the associated entries of the 3-dimensional table as:

$$F\big(y_m, x_i = v_i, x_j = v_j\big) = F\big(y_m, x_i = v_i, x_j = v_j\big) + 1 \tag{17}$$

By doing the 2-step updates, we not only maintain the label dependence based on the true labels of each arrived sample but also continue to learn the label-feature-value relationship for the joint distribution. It is noted that the time complexity of each entry update is $\mathcal{O}(1)$ since we can directly access the entries in the label co-occurrence table and the label-feature-value relationship table.

3.5. Label set learning

In this section, we introduce a method which learns the label set for each arrived sample in a sequential manner. As mentioned in section 3.1, we predict a new label for $\mathbf{x}$ based on the previously predicted labels in the set $\hat{Y}_{\mathbf{x}}$. First, starting with $\hat{Y}_{\mathbf{x}} = \emptyset$, we compute the posterior probability $\mathrm{P}(y_i|\mathbf{x})$ $i = 1, \dots, M$. The first label is assigned to $\mathbf{x}$ by getting the label that maximizes the posterior probability:

$$\mathrm{P}(\hat{y}_1|\mathbf{x}) = \max_{i=1,\dots,M} \mathrm{P}(y_i|\mathbf{x}) \tag{18}$$

The first predicted label $\hat{y}_1$ is added to the set $\hat{Y}_{\mathbf{x}} = \{\hat{y}_1\}$. In the next step, the posterior is conditioned on $\hat{y}_1$ as $\mathrm{P}(y_i|\mathbf{x}, \hat{y}_1)$. The predicted label $\hat{y}_2$ given by (19) is added to $\hat{Y}_{\mathbf{x}}$.

$$\mathrm{P}(\hat{y}_2|\mathbf{x}, \hat{y}_1) = \max_{i=1,\dots,M; y_i \neq \hat{y}_1} \mathrm{P}(y_i|\mathbf{x})\mathrm{P}(\hat{y}_1|y_i) \tag{19}$$

Suppose that we have assigned $(h-1)$ label $\hat{Y}_{\mathbf{x}} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{h-1}\}$ to $\mathbf{x}$. The new label $\hat{y}_h$ will be added to $\hat{Y}_{\mathbf{x}}$ i.e. $\hat{Y}_{\mathbf{x}} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_h\}$ by:

$$\mathrm{P}\left(\hat{y}_h | \mathbf{x}, \hat{Y}_{\mathbf{x}}\right) = \max_{i=1,\dots,M; y_i \notin \hat{Y}_{\mathbf{x}}} \mathrm{P}(y_i|\mathbf{x}) \prod_{j=1}^{h} \mathrm{P}\left(\hat{y}_j | y_i\right) \quad (20)$$

Often, the number of labels to be learned is predefined beforehand, for example, 5 labels per image are used in [61] for automatic image annotation. However, in practice, the number of labels for each sample should be dynamically determined. For data stream, the number of label for each arrived sample should be determined based on the revealed true labels. In this study, we propose a method to dynamically determine the number of predicted labels using the Hoeffding inequality [62] and the label cardinality.

The label cardinality of a dataset is the average number of labels of the samples in the dataset. Its value is independent of the number of labels M, and it can be used to quantify the number of predicted labels h for each sample [10]. Here, we introduce an approach to incrementally learn the value of h in which h is adjusted if it is different from the label cardinality by more than a certain amount. The amount in turn is determined using the Hoeffding inequality. Fig. 1 shows the proposed approach to determine the size of the predicted label set. We first initialize a value for h. After receiving $N-1$ samples in sequence in which the i^{th} sample has $|Y_i|$ true labels, the label cardinality of the stream at the $(N-1)^{th}$ sample, denoted by $\bar{z}_{N-1}$, is given by:

$$\bar{z}_{N-1} = \frac{1}{N-1} \sum_{i=1}^{N-1} |Y_i| \quad (21)$$

After predicting the N^{th} sample, the label cardinality at the N^{th} sample is updated by:

$$\bar{z}_N = \frac{1}{N} \sum_{i=1}^{N} |Y_i| = \frac{1}{N}\left(\sum_{i=1}^{N-1} |Y_i| + |Y_N|\right) = \frac{1}{N}\left((N-1)\bar{z}_{N-1} + |Y_N|\right) \quad (22)$$

We check whether the difference between $\bar{z}_N$ and h is greater than a threshold ε, the value of h used for the next sample will be re-calculated by the rounded value of the current $\bar{z}_N$. The threshold is computed based on the Hoeffding inequality as:

$$\varepsilon = \sqrt{\frac{L^2 ln(2/\delta)}{2N}} \quad (23)$$

in which N is the number of samples, δ is a given confident value, L is the total distinct labels among the true labels of the samples we received. If the update condition is met, we reset $\bar{z}_N = 0$, $L = 0$, and $N = 0$ to begin a period with the new value of h. The key idea of our approach is that we only update h if there is a certain difference between h and the current label cardinality $\bar{z}$. The application of Hoeffding inequality to quantify the number of predicted labels is presented in the Appendix.

To our knowledge, there are only three incremental MLC thresholding methods that can determine the number of predicted labels. In all of them, a label is selected if its associated confident score (e.g., posterior probability) is higher than a threshold. The threshold is initialized, and is employed to obtain the labels, and is then re-calculated via instance adjustment [63] or batch adjustment [6, 9]. On the one hand, Read [63] makes a small adjustment on the threshold on a per-instance basis depending on the predicted and actual label cardinality. Xioufis et al. [9] by contrast incrementally adjusts a threshold for each label on each fixed window of samples so that the frequency of the predicted label approximates that of the ground truth label. Read et al. [6] incrementally adjusts a threshold for all labels on each batch to ensure that the predicted label cardinality approximates the true label cardinality. Comparing to these methods, our approach is significantly different and more flexible since the number of samples used to adjust h is not fixed, i.e., h is adjusted if the update condition is satisfied.

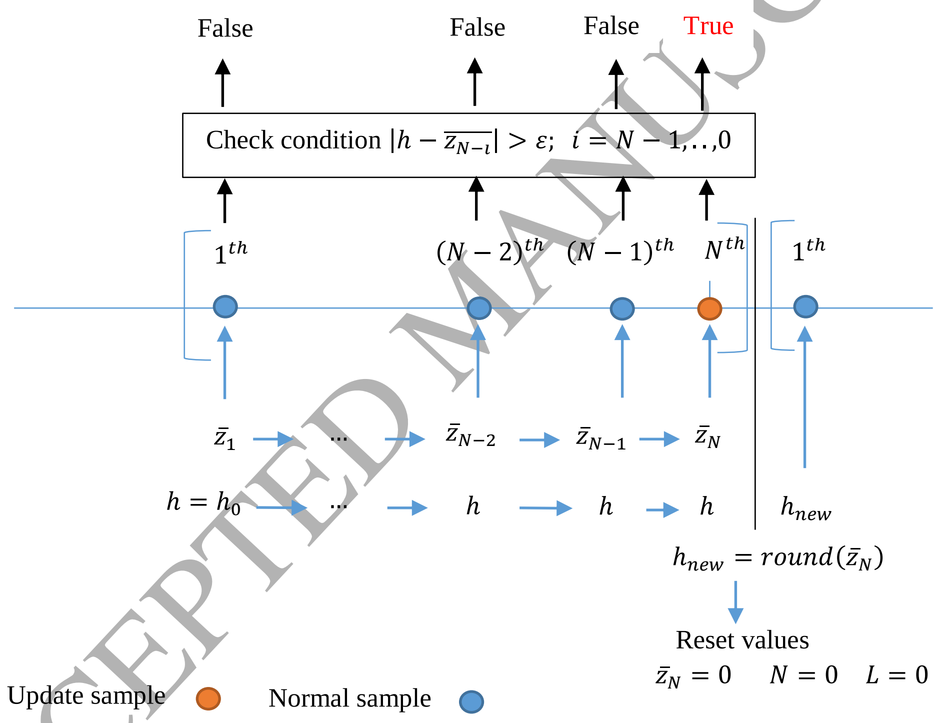


Fig.1. Illustration on adjusting the number of predicted labels

Algorithm 1: Multi-label classification via label correlation and 1-feature dependence on data stream

```
Input:   Multi-label data stream $\mathcal{D}$, $\delta$
Output:  Predicted label set for each sample
         %Initialize learning model
```

```
Initialize $h$;
Initialize frequencies of label co-occurrence table
$F(y_i, y_j) = 0$
Initialize frequencies of label-feature-value table
$F(y_m, x_i = v_i, x_j = v_j) = 0$
For each arrived sample $\mathbf{x}$ from $\mathcal{D}$
    $N = N + 1$;
    %Predict labels for $\mathbf{x}$
    $\hat{Y}_{\mathbf{x}} = \emptyset$
    While ($|\hat{Y}_{\mathbf{x}}| \leq h$)
         For each $y_m \in \mathbb{Y}$ and $y_m \notin \hat{Y}_{\mathbf{x}}$
             Compute $\mathrm{P}(y_m|\mathbf{x}, \hat{Y}_{\mathbf{x}})$ by (3) in which:
                  Probabilities $\mathrm{P}(\hat{y}_l|y_m)$ for each $\hat{y}_l \in \hat{Y}_{\mathbf{x}}$ are
                  computed by (5) and (6)
                  $\mathrm{P}(y_m, \mathbf{x})$ is computed by (11)
                  Assign label $\hat{y}$ to $\mathbf{x}$ by (4)
                  $\hat{Y}_{\mathbf{x}} = \hat{Y}_{\mathbf{x}} \cup \{\hat{y}\}$
          End
      End
      %Update the learning model
      Reveal the true labels of $\mathbf{x}$, i.e. $Y$, from the
      environment
      For each $y$ in $Y$
          Update $F(y, y_i)$ $y_i \in Y, y_i \neq y$ by (16)
          For the values of features of $\mathbf{x}$
              Update $F(y, x_i = v_i, x_j = v_j)$ by (17)
          End
      End
      %Update the number of predicted labels
      Compute $\bar{z}_N$ by (22)
      Compute $\varepsilon$ by (23)
      If ($|\bar{z}_N - h| > \varepsilon$)
           Update $h = round(\bar{z})$
           Reset $\bar{z}_N = 0, N = 0, L = 0$
```

```
                    End
          End
```

3.6. Handling of missing values

In many applications, the values of some features may be missing sometimes. For example, some data cannot be obtained due to poor weather condition, or due to the failure in data transmission process, or due to the interviewee unwilling to provide information. These attributes are marked as missing. There are methods to handle missing data such as missing data imputation from observed attributes or omitting feature vectors having missing attributes. In the former, the average value computed from the non-missing attributes is usually used to replace the missing value. In the latter, all feature vectors that contain missing attributes will be removed from the dataset. Although this approach is simple, it is impractical when only a small dataset is available. Other more sophisticated imputation methods can also be employed to impute the missing attributes [64-66].

In this work, instead of trying to impute the missing value, we expand the label-feature-value table by an "undetermined value" to account for the missing values. Specifically, for each feature, we add a new column to count its missing value (see Table 1). When an arrived sample with some missing values is used to update the table, for each missing value, we find the missing value column of the associated feature and increment the entry inside using (24). In this way, the missing value of each feature is considered as an extra "undetermined value" for that feature. Our approach allows us to avoid imputing the missing values, which could be difficult since we might not have enough data to obtain an accurate imputation under the online learning setting. It avoids the arbitrariness of assigning a value to the missing value when there is no strong evidence by assigning them instead to the category of undetermined value. The space complexity of this approach is $\binom{\mathrm{M(Dn+D)}}{2}$.

$$F(x_i = \text{Missing value}, x_j = v_j, y_m) = F(x_i = \text{Missing value}, x_j = v_j, y_m) + 1 \tag{24}$$

Table 1. The expansion of 'label-feature-value' table to handle missing values

y_m		Feature x_i			Feature x_{i+1}		
		v_i	…	Missing value of feature x_i	v_{i+1}	…	Missing value of feature x_{i+1}
x_j	…	…	…	…	…	…	…
	v_j	$F(x_i = v_i, x_j = v_j, y_m)$	…	$F(x_i = \text{Missing value}, x_j = v_j, y_m)$	$F(x_{i+1} = v_{i+1}, x_j = v_j, y_m)$	…	$F(x_{i+1} = \text{Missing value}, x_j = v_j, y_m)$
	…	…	…	…	…	…	…

3.7. Concept drift adaptation

Concept drift in data often make the model built on old data inconsistent with the new data, therefore it is crucial to make a learning model adaptive to concept drift [39]. In the literature, there are two main strategies to deal with concept drift, namely, by using a sliding window or a decay function (or called weighted samples [67]) [68]. In the first strategy, the most recent samples are kept within a sliding window while older samples are discarded. Several methods used one sliding window with either a fixed length or a variable length. For example, in ADWIN, a well-known concept drift handling method, a window will grow or shrink depending on whether data changes or not. The improved version, ADWIN2, is more efficient than the first one in time and memory consumption [68]. Other works such as [39, 40] used two adjustable windows to represent old and new samples. Xioufis et al. [9] used two windows per label to capture the positive and negative group of samples. The decay function strategy, meanwhile, uses weights to mitigate or strengthen the importance of samples based on their ages, i.e. the older sample is less important than the new one [69, 70]. The learning model, therefore, can adapt to changes in the newer data.

In this study, we use the decay function strategy to allow our model to handle concept drift. Our proposed approach includes two steps. First, we update every entry of the label co-occurrence table by decaying the frequencies $F(\mathrm{y}_i, \mathrm{y}_j)$ as:

$$F(\mathrm{y}_i, \mathrm{y}_j) = \alpha * F(\mathrm{y}_i, \mathrm{y}_j) + \mathbb{I}[y_i = y_m, y_j = y_t, y_m, y_t \in Y] \tag{25}$$

in which $\mathbb{I}[.]$ is an indicator function and $0 < \alpha < 1$ is a given decay factor. Because $\mathbb{I}[y_i = y_m, y_j = y_t, y_m, y_t \in Y] = 1$ for the pairs of labels (y_m, y_t) in Y, the frequencies $F(y_m, y_t)$ associated with these labels will increase to strengthen the label dependency. On the other hand, the co-occurrence of other pairs that do not appear in Y will be reduced since their frequencies are reduced by the decay factor α. Similarly, we propose the update in (26) for the label-feature-value table so that the joint distribution can also adapt with the changes of data through the decay factor β and the indicator function $\mathbb{I}[y = y_m, y_m \in Y, x_i = v_i, x_j = v_j]$.

$$F(y, x_i = '.', x_j = '.') = \beta * F(y, x_i = '.', x_j = '.') + \mathbb{I}[y = y_m, y_m \in Y, x_i = v_i, x_j = v_j] \qquad (26)$$

By using the decay strategy in (25) and (26), we adapt both the label correlation and the joint distribution with the age of the data, i.e., by focusing more attention on the newer samples. This decay is conducted directly on the two tables storing the label correlations and the label-feature-value relationships and does not consume additional memory throughout the adaptation process. The detail of the proposed decay strategy is given in Algorithm 2.

Algorithm 2: Decay strategy for concept drift adaption on data stream

```
Input    Multi-label data stream $\mathcal{D}$ with concept drift
         %Generate learning model using Algorithm 1;
         For each arrived sample $\mathbf{x}$ from $\mathcal{D}$
                $N = N + 1$;
                Predict labels for $\mathbf{x}$ using Algorithm 1;
                Update the learning model
                Reveal labels of $\mathbf{x}$, i.e. $Y$, from the environment
                For $i = 1$ to $|\mathbb{Y}|$
                    For $j = i + 1$ to $|\mathbb{Y}|$
                         Update $F(y_i, y_j)$ by (25)
                    End
                    For the values of all features
                          Update all label-feature-value
                          tables $F(y_m, x_i =., x_j =.)$ by (26)
                    End
                End
            Update the number of predicted labels: Conduct the
         similar steps in Algorithm 1;
         End
```

4. Experimental Studies

4.1. Datasets

In this study, we compared the performance of the proposed method and the benchmark algorithms under both stationary and concept drift settings. We chose 6 multi-label real-word datasets for comparison under the stationary setting. For the concept drift setting, we generated 3 synthetic datasets with different types of drift so that there were a total of 9 datasets for the evaluation. All these datasets were treated as stream data by processing them in the order that they were collected.

In detail, the first six real-world datasets are popular datasets used in multi-label stream classification experiments [6, 7]. We used MOA to generate the three datasets with concept drifts as in [6]. The synthetic data was generated based on two generation schemes in MOA [51]:

- The Random Tree Generator (RTG) constructs a decision tree by choosing attributes at random to split and assigning a random class label to each leaf. Once the tree is built, new examples are generated by assigning uniformly distributed random values to attributes which then determine the class label via the tree. We used a tree depth of 5 and create a dataset named *SynRTG.*
- The Radial Basis Function (RBF) was devised to offer an alternate complex concept type that is not straightforward to approximate using a decision tree model. This generator effectively creates a normally distributed hypersphere of examples surrounding each central point with varying densities. Drift is introduced by moving the centroids with constant speed initialized by a drift parameter. We used two central points for the dataset named *SynRBF*.

All generation schemes used by our framework were initialized as binary generators with inferred parameters from Read et al. [6]. Three concept drifts of varying types, i.e., different magnitude and extent, were introduced in *SynRTG* and *SynRBF* (named *SynRTG-drift* and *SynRBF-drift*, respectively). For N generated samples, the drifts were centered at the $(N/4)^{th}$, $(N/2)^{th}$, and $(3N/4)^{th}$ sample, and extend over N/1000, N/100, and N/10 samples, respectively. In the first drift, we changed 10% of label dependencies. In the second drift, we changed the underlying concept as well as adding more labels to each sample, leading to higher label cardinality. In the third drift, we changed 20% of label dependencies. Such types and magnitudes of drift can be found in many real-world datasets.

We generated the drift version of IMDB dataset named *IMDB-drift* by concatenating 100000 samples from the original data and 100000 samples generated by RTG. For RTG, we created a tree with 28 labels, a label cardinality of 4.0, and label dependencies of 0.25 and used it to generate the samples. Only one artificial drift was centered in the middle of *IMDB-drift*, extending over 2000 samples. The details of the experimental datasets are given in Table 2.

Table 2. The experimental datasets

Dataset	# of samples	# of features	# of labels	Label cardinality
20NG	19300	1001 binary	20	1.1
ENRON	1702	1001 binary	53	3.4
IMDB	120919	1001 binary	28	2.0
OHSUMED	13929	1002 binary	23	1.7
SLASHDOT	3782	1079 binary	22	1.2
TMC2007	28596	500 binary	22	2.2
IMDB-drift	200000	1001 binary	28	$2.0 \rightarrow 4.0$
SynRBF-drift	100000	80 numeric	25	$1.5 \rightarrow 3.5$
SynRTG-drift	1000000	30 binary	8	$1.8 \rightarrow 3.0$

4.2. Experimental Setup

Under the stream setting, there are two common evaluation procedures called holdout and predictive sequential (prequential) [71]. The holdout evaluation procedure applies the current decision model to an independent test set. Meanwhile, in the prequential procedure, for each arrived sample, we first predicted the labels and then used this sample to update the learning model. For all experiments in our studies, we used the prequential evaluation method for data streams since it is the most frequently used measure for evaluating the accuracy of a classifier [6]. During the evaluation, we performed the following steps on each arrived sample $\mathbf{x}$:

- Predict the labels: The current learning model is used to predict the label set $\hat{Y}_{\mathbf{x}}$ for $\mathbf{x}$.
- Update the learning model: If the true labels Y of $\mathbf{x}$ are available, the classification model is updated based on the sample $\mathbf{x}$ and its true label Y.

Gama et al. [71] proposed the forgetting mechanism by using a sliding window and a fading factor for the prequential error estimator in stream learning. He observed that the outputs of these two approaches are very similar and therefore the fading factor can be used to replace the sliding window due to its advantage in memory saving. Gama et al. [71] also introduced the fading factor-based McNemar statistical test to compare the accuracy of the two stream classification methods, and the fading factor-based Page-Hinkley test for change detection. To assess the statistical significance of the results of multi-label stream classification methods, Read et al. [6] and Osojnik et al. [7] used Friedman test [72, 73] to test the difference between each evaluation measure of multiple methods on multiple datasets. Friedman test is preferred over ANOVA test for the following reasons. First, Friedman test does not assume normal distribution as in ANOVA test. Besides, an important assumption of repeated-measures ANOVA is sphericity (similar to the requirement that the random variables have equal variance), which cannot be taken for granted because of the nature of learning algorithms and datasets [53, 73]. Here, Friedman test is used to test the null hypothesis that "all methods perform equally" on the datasets. If the null hypothesis is rejected, a post-hoc test is then conducted. Read et al. [6], and Osojnik et al. [7] used Nemenyi test for all pairwise comparisons based on the rankings of algorithms on all datasets. The difference in the performance of two methods is treated as statistically significant if the $p - value$ computed from the post-hoc test statistic is smaller than an adjusted value of confident level computed from the Nemenyi's procedure. The confident level of the test was set to 0.05.

4.3. **Performance Measure and Benchmark Algorithms**

In multi-label learning, each sample is associated with a set of labels, making the performance evaluation more complicated compared to single class learning. Zhang and Zhou [1] stated that the performance of multi-label algorithms should be tested on many measures instead of on the one that is being optimized to ensure a fair and honest evaluation. In our experiment, we used six measures to compare the proposed method to the benchmark algorithms. These measures are grouped into three groups: sample-based measures (accuracy and F1), label-based measures (micro F1 and macro F1), and ranking-based measures (average precision and ranking loss). It is noted that each group of measures expresses different aspects of the performance, e.g., how good the classification process is over different samples in sample-based measures or over different labels in label-based measures (The details of the performance measures can be found in the Appendix). Besides, the running time, including the time for prediction and model update, was also reported to evaluate the time efficiency of all methods.

Since our algorithm can be used under both stationary and concept drift settings, we compared with both well-known non-adaptive and adaptive multi-label stream classification algorithms. First, the proposed method was compared with the following well-known incremental MLC algorithms: incremental Binary Relevance (denoted by iBR), incremental Classifier Chain (denoted by iCC), incremental Pruned Set (denoted by iPS), Rank Threshold (denoted by RT), and Majority Label Set (denoted by MLS). The base classifier for these methods is the Hoeffding tree. All these benchmark algorithms used the default parameters given in the MEKA library [42]. The proposed method was also compared with four very recent multi-target tree based MLC algorithms for stream data introduced in [7], including multi-target model trees (denoted by iSOUP-MT), multi-target regression tree (denoted by iSOUP-RT), online Bagging for iSOUP-MT (denoted by iSOUP-EBMT), and online Bagging for iSOUP-RT (denoted by iSOUP-EBRT). The parameters for these methods were as in [7].

Five methods, namely, iBR, iCC, iPS, RT, and MLS, were combined with the state-of-the-art adaptive method named ADWIN2 [68] to handle the changes of data. As the four multi-target-tree based MLC algorithms [7] can only handle stationary concept, we did not include them in the comparison experiment under the concept drift setting.

5. **Results and Discussions**

5.1. ***Evaluation under the stationary setting***

5.1.1. ***Influence of confident value***

The confident value δ in the Hoeffding inequality (23) determines whether to update the number of predicted labels h. In this study, it was set to be in the range of $\{0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3\}$. Our goal is to examine the influence of $\boldsymbol{\delta}$ on the 6 performance measures. Fig 2 presents the relationship between $\boldsymbol{\delta}$ and the six measures on the six datasets. Some interesting observations can be made. First, there is no different in all 6 measures for different values of δ on 4 datasets, i.e. 20NG, OHSUMED, IMDB, and TMC 2007. It means that the value of δ does not affect the update of the number of predicted labels on these datasets.

On the SLASHDOT dataset, δ has a slight influence on 3 measures: sample-based F1, sample-based accuracy, and micro-average F1. A similar trend can be observed on the 3 measures, i.e. the performance value reduces to a minimum at $\delta = 0.01$ and then slightly increase with the increase of δ. Meanwhile, the other 3 measures remain unchanged with different values of δ.

On the ENRON dataset, a similar pattern is observed on the four measures: sample-based (F1 and Accuracy) and label-based (micro F1 and macro F1). These performance measures have the same value for the first four values of δ before increasing at $\delta = 0.1$. After that, they remain unchanged at $\delta = 0.2$ and 0.3.

It is noted that the value of δ will affect the up2date of the number of predicted labels and therefore affects some performance measures. If δ is too small, h is less likely to be updated. As a result, the learning model might predict too few or too many labels for a sample. In contrast, a large value of δ might cause more frequent update for h, making the model less stable. Based on the observations from Fig 2, we set $\delta = 0.1$. In the next section, we will compare the performance of the proposed method using $\delta = 0.1$.

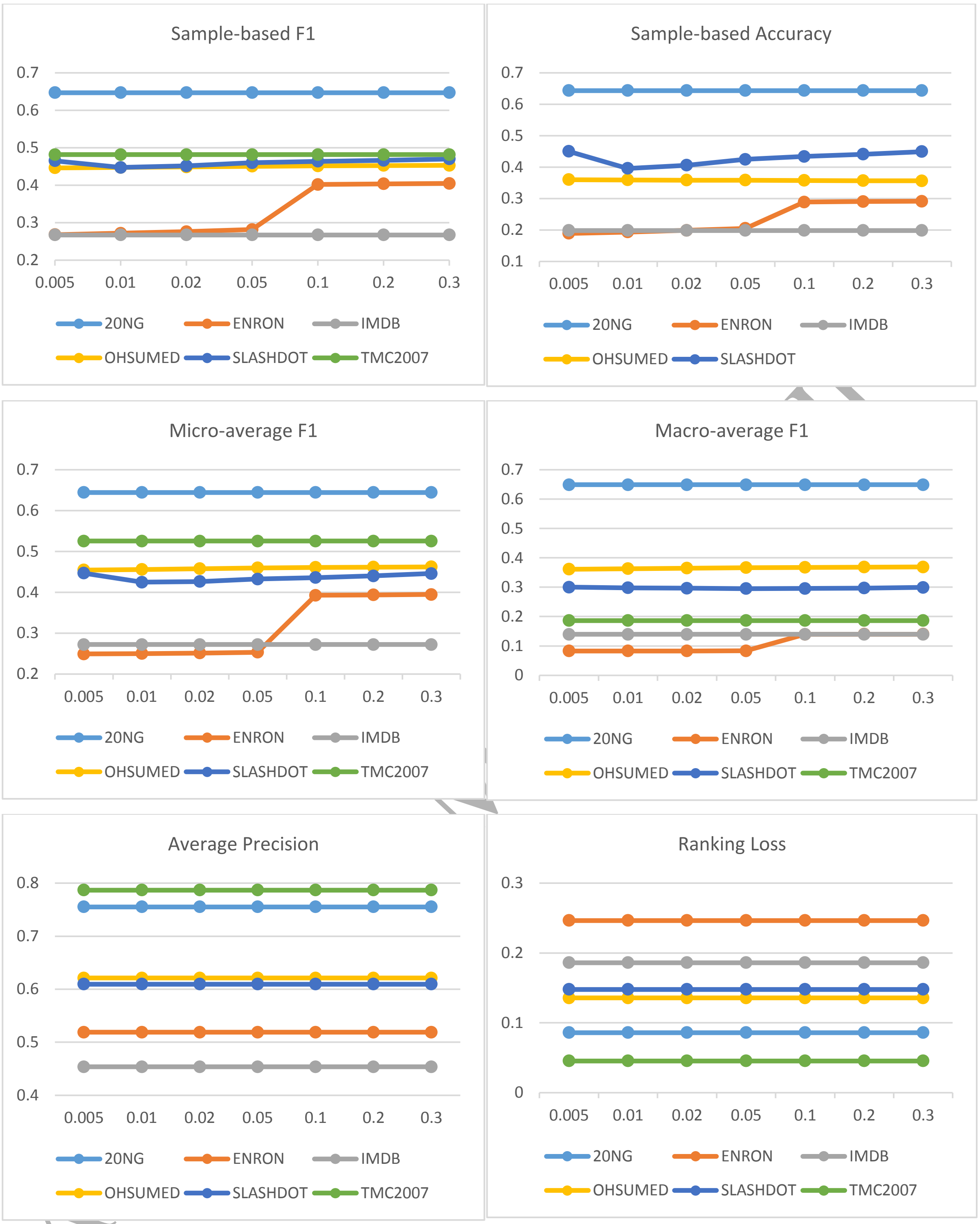
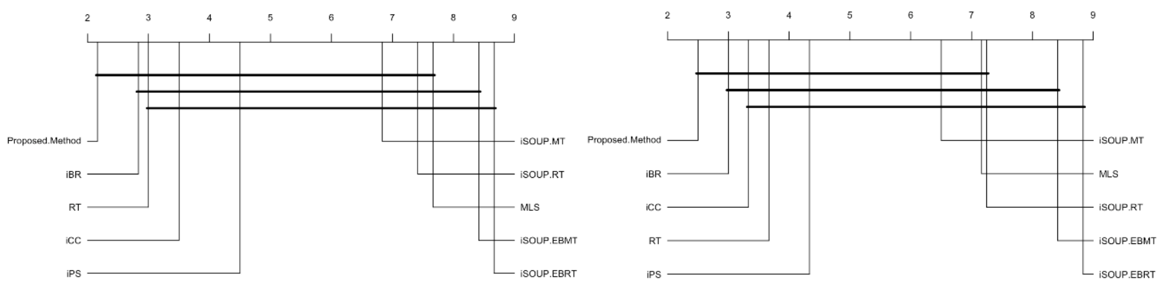


Fig.2. Influence of confident value δ (x-axis) to performance measures (y-asix) on six datasets

5.1.2. ***Results on sample-based measures***

Table S1 in the supplement material shows the two sample-based measures, i.e. F1 and accuracy, of the benchmark algorithms and the proposed method. The up/down arrow beside the measure indicates that a higher/lower value is preferred for that measure. The P-values computed based on these rankings with Friedman test are 1.698e-05 and 1.1087E-04 for F1 and accuracy, respectively, therefore we rejected the null hypotheses that the performances of all methods are equal. From the Nemenyi significance test results shown in Fig 3, the proposed method is better than iSOUP-EBRT and iSOUP-EBMT for both F1 and accuracy. Meanwhile, there is no statistical difference in the pairwise comparison between the proposed method and the other benchmark algorithms.

In detail, the proposed method ranks first for both measures (2.17 for F1 and 2.5 for accuracy) followed by iBR (2.83 for F1 and 3 for accuracy). On 3 datasets, i.e. 20NG, OHSUMED, and SLASHDOT, the proposed method obtains the best results which are far better than the benchmark algorithms. iBR ranks first on two datasets: ENRON and TMC 2007. Meanwhile, iCC, iPS, and RT perform at the average level and rank in the middle. The 4 multi-target tree-based methods and MLS clearly are not suitable for some datasets like 20NG, OHSUMED, and SLASHDOT since their performance are significantly poorer than those of the other methods.



**Proposed Method > { iSOUP-EBRT, iSOUP-EBMT}; iBR > iSOUP-EBRT*

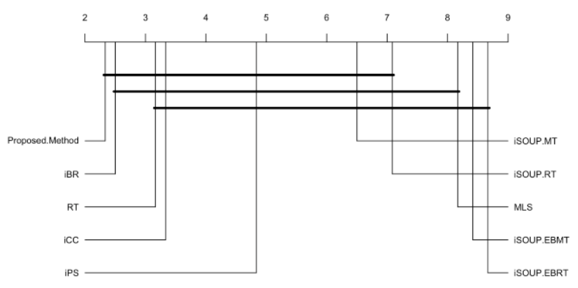
**Proposed Method > { iSOUP-EBRT, iSOUP-EBMT}; iBR > iSOUP-EBRT*

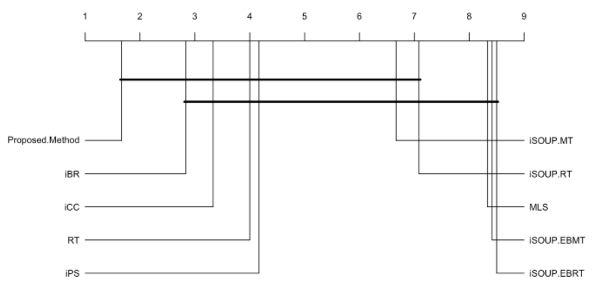
Fig.3. Nemenyi test for sample-based F1 (left) and accuracy (right)

5.1.3. ***Results on label-based measures***

We compared the proposed method to the benchmark algorithms based on the label-based measures. In this case, all P-values for the 6 label-based measures computed by Friedman test are smaller than 0.05. Hence, we rejected the null hypotheses based on Friedman test and conducted the post-hoc test for all pairwise comparisons among the ten methods.

On the micro F1 measures, our method ranks first among 10 methods (rank value 2.33), followed by iBR (rank value 2.50) and RT (rank value 3.17) (see Table S2 in the supplement material). Our method is better than iSOUP-EBRT, iSOUP-EBMT, and MLS based on Nemenyi test. The results on the macro F1 measures show even better performance for the proposed method, obtaining the best results on five datasets: 20NG, ENRON, IMDM, OHSUMED, and SLASHDOT. The four multi-target tree-based methods and MLS continue to perform poorly on 4 datasets: 20NG, IMDB, OHSUMED, and SLASHDOT.





**Proposed Method > { iSOUP-EBRT, iSOUP-EBMT, MLS}; iBR > {iSOUP-EBRT, iSOUP-EBMT}*

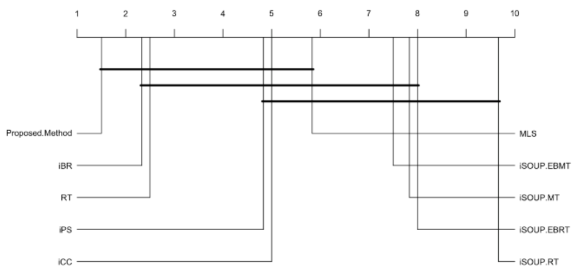
**Proposed Method > { iSOUP-EBRT, iSOUP-EBMT, MLS}*

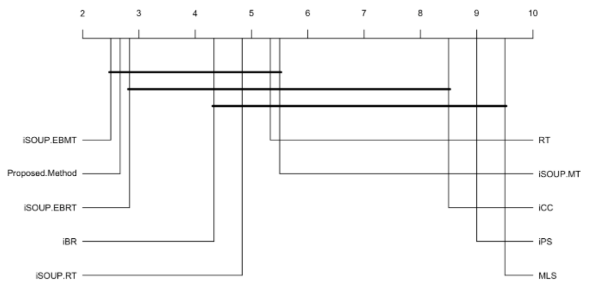
Fig.4. Nemenyi test for label-based micro F1 (left) and macro F1 (right)

5.1.4. *Results on ranking-based measures*

The performance of the benchmark algorithms and the proposed method for the average precision and ranking loss a2re shown in Table S3 in the supplement material. We again conducted the Nemenyi post-hoc test and reported the results in Fig 5 for the pairwise comparison. The four multi-target tree-based methods perform poorly on all 6 datasets for average precision, and are worse than the proposed method using the Nemenyi test. Although RT obtains the best results on EURON and IMDB dataset, the difference between its average precision and that of the proposed method are only about 2.5%. The proposed method meanwhile significantly outperforms the benchmark algorithms on 20NG, OHSUMED, SLASHDOT, and TMC2007.

Surprisingly, the two ensembles iSOUP-EBMT and iSOUP-EBRT perform quite well on all datasets for the ranking loss measure. Even iSOUP-EBMT's rank value is slightly higher than the proposed method (2.5 vs. 2.67). The ranking losses of iBR and RT are average while iCC, iPS, and MLS have poor results (rank values are 8.5, 9 and 9.5 respectively). Based on the Nemenyi test, iCC, iPS, and MLS are worse than iSOUP-EBMT and the proposed method, while iPS and MLS are worse than iSOUP-EBRT.





**Proposed Method > { iSOUP-RT, iSOUP-EBRT, iSOUP-MT, iSOUP-EBMT}; iBR > iSOUP-RT; RT > iSOUP-RT*

**iSOUP-EBMT > {MLS, iPS, iCC}; Proposed Method > {MLS, iPS, iCC}; iSOUP-EBRT > {MLS, iPS}*

Fig.5. Nemenyi test for ranking-based average precision (left) and ranking loss (right)

5.1.5. ***Discussions***

Our experiments show that the performance measures vary significantly across the different methods. For the macro-average F1, for example, iBR obtains very poor performance value on SLASHDOT compared to that of the proposed method and its counterpart iCC (0.0767 vs. 0.2957 and 0.2589). These variations were also observed in [6] in which they hypothesized that the variations might be due to the many unaccounted or partially accounted effects of the multi-label data stream. These effects might arise, for example, from the multi-label aspect of the problem, the threshold used, the use of single-label classifier, and algorithm parameter setting.

Two ensemble algorithms, i.e., iSOUP-EBRT and iSOUP-EBMT, turn up to be poor incremental MLC methods, losing to the proposed method on 5 performance measures except for the ranking loss. iSOUP-MT and iSOUP-RT also generally have poor performance values with respect to all the measures. In fact, multi-target tree-based methods are considerably dependent on the threshold used. In these methods, both the target and the instance space of a MLC problem are first transformed into the associated spaces of a multi-target regression problem. The solution of the multi-target regression problem is then transformed back to the MLC problem to obtain the predicted labels. Here, the threshold is used twice, i.e., once in converting muti-target prediction to multi-label prediction and then in updating the regressor. It is noted that the sample-based measures and label-based measures are threshold-based measures. Therefore, multi-target tree-based methods could underperform on these measures if a sub-optimal threshold is used. Moreover, the label correlation is ignored in the transformations which also negatively affect the method’s performance.

MLS is, unsurprisingly, a weak MLC algorithm, and usually ranks at the bottom. Since MLS is the simplest multi-label classifier which assigns the most common label set from the training data for all test instances, it is an uncompetitive incremental classifier for most performance measures. MLS, however, is the fastest MLC algorithm because of its simple learning mechanism.

iCC, iPS, and RT, meanwhile, have average performance with middle ranking in all measures. iCC and iPS are two well-known MLC which take into account the label correlation in the learning model. RT transforms a MLC problem into a multi-class classification problem by duplicating multi-label samples and then assigning only one label for each copy to generate the new single-label data. iBR is a competitive incremental classifier among the benchmark algorithms as it ranks second on 5 performance measures. However, the variations in performance values between iBR and our methods are very significant on some datasets. These four methods are threshold-based approaches that select the predicted labels from the prediction scores. In this paper, we adjusted the threshold via batch-based approach on the predicted and true label cardinality as in (Read et al., 2012). The dependence on choosing the appropriate threshold and the less flexible adjustment for the number of predicted labels per instance could have caused the under-performance of these methods on some datasets.

The proposed method ranks first on five performance measures except for the ranking loss. The post-hoc test shows that our method is better than iSOUP-EBMT and iSOUP-EBRT for all measures, is better than MLS for the ranking loss, micro-F1, and macro-F1, is better than iSOUP-MT and iSOUP-RT for the average precision, and is better than iPS and iCC for the ranking loss. On some datasets like 20NG, SLASHDOT, and OHSUMED, the proposed method achieves significantly better performance compared to the2 benchmark algorithms. There are several reasons for the outstanding performance of the proposed method. First, the label correlation is taken into account in the label prediction process, in which a new label is assigned with consideration to the previously assigned labels in the previous steps. Moreover, the prediction process for each sample stops if the label set obtains enough number of predicted labels. The proposed method, as a result, is a threshold-free incremental multi-label classifier. Besides, the number of predicted labels is learned based on the Hoeffding inequality to ensure that each sample is assigned with an adequate set of label. Finally, the learning model improves itself by not only considering the label correlation but also updating the label-feature-value relationship with the arrival of each sample that has a ground truth label set.

5.2. The advantage of using AODE method

In section 3.3, we used the AODE method to estimate the joint probability $\mathrm{P}(\mathbf{x}, y_m)$. By considering the dependency between one feature and only one other feature (called parent feature), the joint distribution was estimated by aggregating the results among all parent features. In this section, we compared the effectiveness of using AODE method and the traditional Naïve Bayes method (Equation 7) with the proposed framework. For the Naïve Bayes method, we used Gaussian distribution $\mathcal{N}(\mu_{mi}, \sigma_{mi}^2)$ to approximate the distribution of the likelihood $\mathrm{P}(x_i|y_m)$ in (7). We referred the update equations for mean and standard variation to [74]. The experimental results of the proposed method using AODE and Naïve Bayes method on 6 real-world datasets are shown in Fig 6.

Clearly, the proposed method using the AODE is better than that using the Naïve Bayes on the 6 datasets for the six measures except for the ranking loss on the ENRON dataset. Specifically, the AODE-based method significantly outperforms the Naïve Bayes-based method, better by more than 14% and 20% on the sample-based measures, the micro-averaged F1, and the average precision on the IMDB and TMC2007 dataset, respectively. Meanwhile, on the ENRON dataset, the AODE-based method is about 7% poorer than the Naïve Bayes-based method for the ranking loss. This is the only case the AODE-based method is worse than the Naïve Bayes-based method. This demonstrates the advantage of using the AODE method compared to using the Naïve Bayes method.

As mentioned in Section 3.3, instead of assuming conditional independence between a pair of features given a class label like the Naïve Bayes method, the AODE method considers the 1-dependence among the features which is more realistic in practice. Moreover, in AODE, we aggregate the multiple models associated with each feature to obtain the estimation for the joint distribution. Aggregating multiple models in an ensemble system can usually improve the classification accuracy [52-60]. These are the reasons why the AODE-based method performs better than the Naïve Bayes-based method.

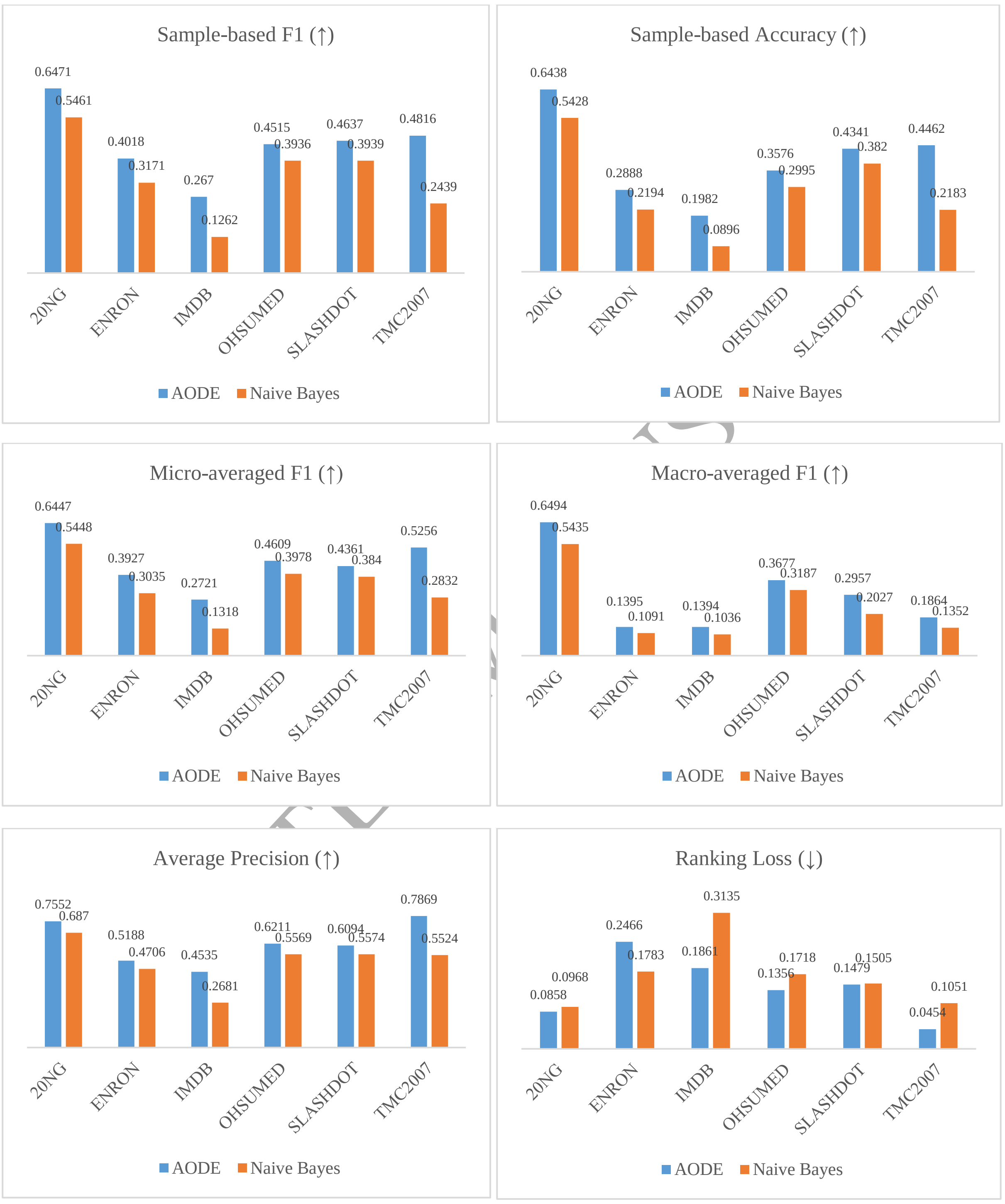
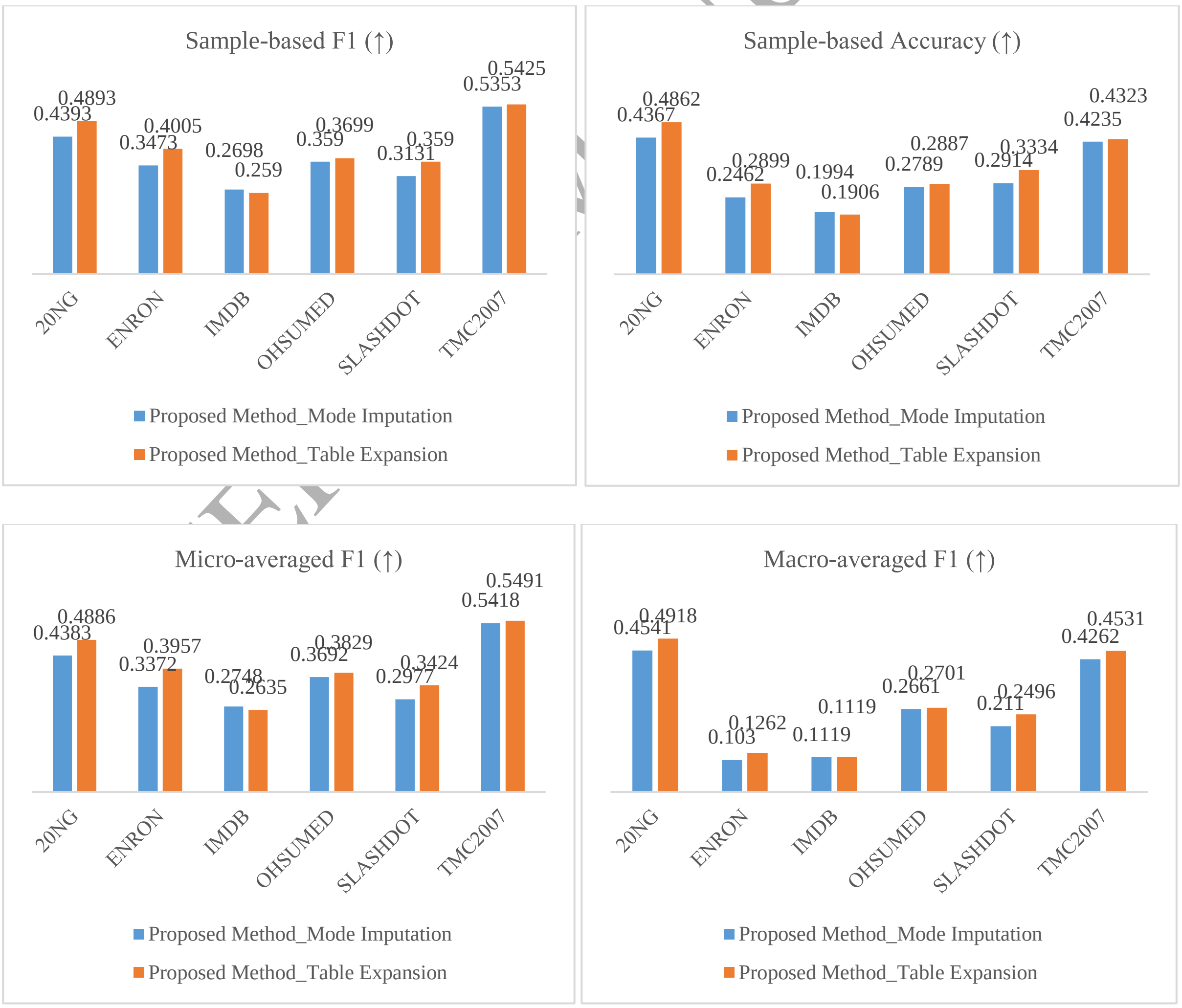


Fig.6. Comparison of the proposed method using AODE and Naïve Bayes method

5.3. Evaluation on data stream having missing values

As mentioned in Section 3.6, we treat the missing value of each feature as an extra "undetermined value". The label-feature-value table is expanded by D columns to store the count of missing values associated with all features. In this section, we compare the effectiveness of this approach to the traditional data imputation method in which the missing value is replaced by the mode value of each feature. We simulated the missing values by removing a number of values from randomly selected features on each arrived sample. The learning model was then updated via (24). The experimental results on 6 real-world datasets are shown in Fig 7.

The proposed approach is better than mode imputation on 5 datasets except for IMDB based on the six measures. Although on IMDB, our approach is worse than mode imputation, the difference between the two approaches is small. Meanwhile, our approach is significantly better on some datasets such as ENRON (around 6% better on average) and SLASHDOT (nearly 4% better on average). This shows the effectiveness of our approach compared to the traditional imputation approach despite its simplicity.



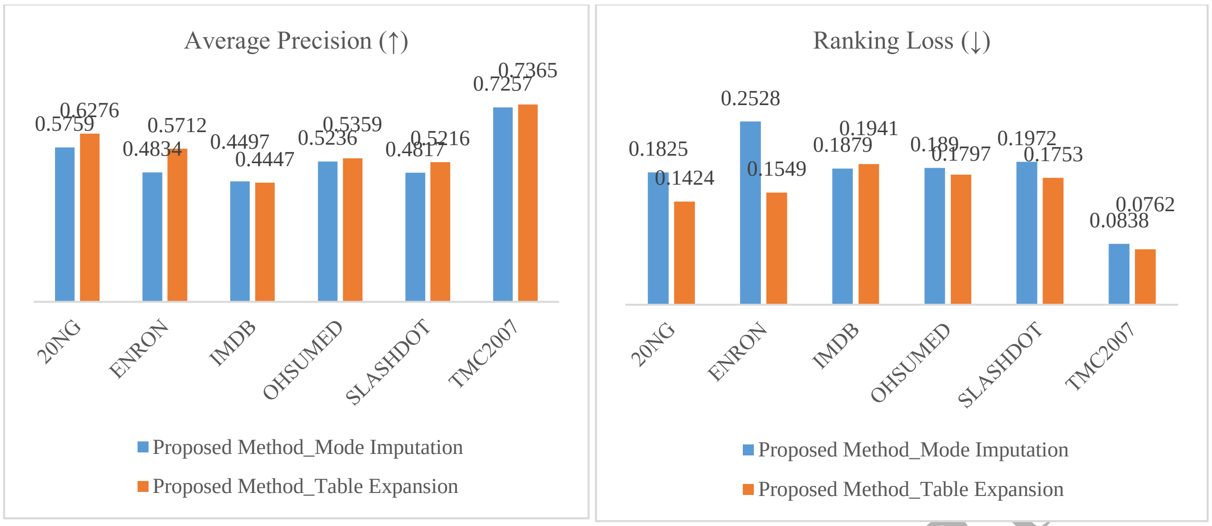


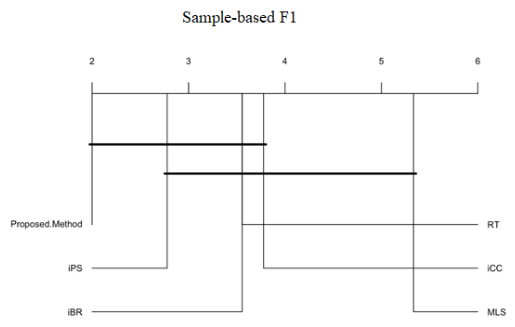
Fig.7. Comparison between mode imputation and 'label-feature-value' table expansion for missing values

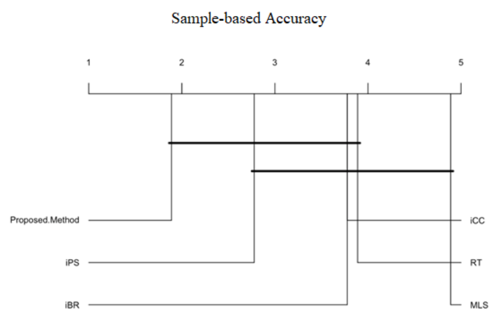
5.3. Comparison under concept drift

As mentioned above, we evaluated the benchmark algorithms and the proposed method on nine datasets including six real-world datasets and three synthetic datasets with concept drift. The decay parameters in (25) and (26) were set to $\alpha = \beta = 0.9$ as in [70].

Table S4, S5, and S6 in the supplement material show the experimental results of all methods with respect to the six measures under the concept drift setting. Once again, the proposed method is better than the benchmark algorithms, ranking first on five among six measures. Only on ranking loss our method is ranked third after iBR and RT. Similar to the stationary setting, the proposed method obtains significantly better performance values than the benchmark algorithms on some real datasets such as 20NG, IMDM, OHSUMED, and SLASHDOT. Below we focus our discussion on the three synthetic datasets with concept drift.

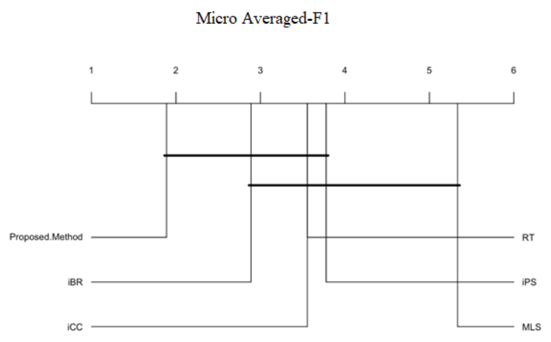
For sample-based F1, sample-based accuracy, and micro F1, our method is better than the benchmark algorithms on SynRBF-drift and IMDM-drift. For macro F1, our method is outstanding on all three datasets. For ranking-based measures, our method obtains the best results on SynRBF-drift. Unsurprisingly, the benchmark algorithms iPS, iCC, RT, and iBR obtain better performance on SynRTG-drift than the proposed method. The SynRTG-drift is generated based on the decision tree, and in the experiment, we used the Hoeffding tree (an incremental decision tree) as the base classifier of these methods.

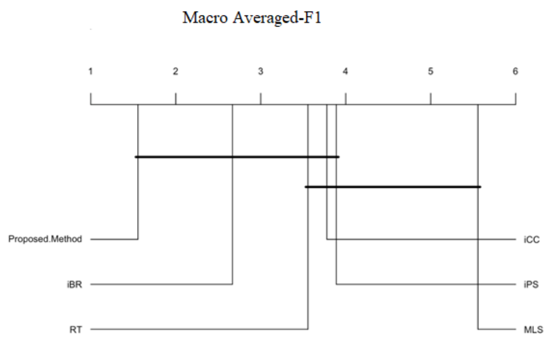




*Proposed Method > MLS

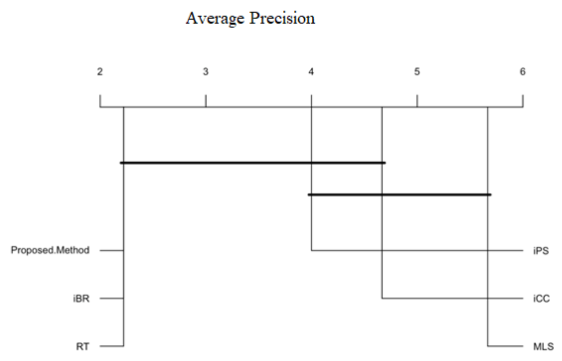
*Proposed Method > MLS

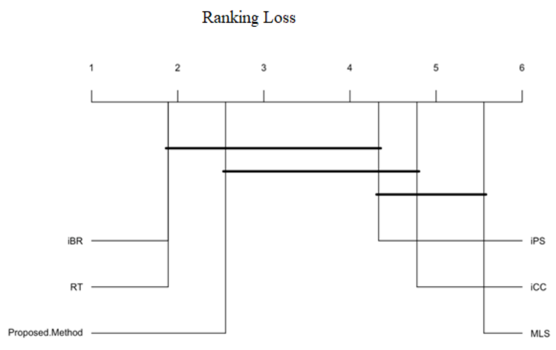




Proposed Method > MLS

Proposed Method > MLS; iBR > MLS





**Proposed Method > MLS;
iBR > MLS; RT > MLS*

**Proposed Method > MLS;
iBR > {MLS,iCC}; RT > {MLS, iCC}*

Fig.8. Nemenyi test for the six measures under concept drift setting

5.4. Time complexity

The complexity of the learning process of the proposed method on a sample $(\mathbf{x}, Y)$ is $\mathcal{O}(\max(hMD^2 + hD, |Y|^2 + |Y|D^2))$ in which $\mathcal{O}(hMD^2 + hD)$ is the time complexity of the classification process and $\mathcal{O}(|Y|^2 + |Y|D^2)$ is the time complexity of the update process. In detail, the time complexity of classification process includes $\mathcal{O}(hMD^2)$ time complexity to compute $\mathrm{P}(\mathbf{x}, y_m)$ in (12) and $\mathcal{O}(hD)$ time complexity to compute $\mathrm{P}(\mathrm{y}_i|\mathrm{y}_m)$ in (5). The time complexity of the model update process includes $\mathcal{O}(|Y|^2)$ to update the label correlations based on ground truth label set Y and $\mathcal{O}(|Y|D^2)$ to update the label-feature-value relationships.

In comparison to the benchmark algorithm, the time complexity of iBR and iCC is $\mathcal{O}\big(M \times \mathcal{O}(base\_classifier)\big)$ in which $\mathcal{O}(base\_classifier)$ is the time complexity of the base classifier we used for the binary classification tasks [12]. In this study, we used Hoeffding tree as the base classifier in iBR and iCC. The time complexity of Hoeffding tree is $\mathcal{O}(\max(clD \times max(n_i), lD)) = \mathcal{O}\big(clD \times max(n_i)\big)$ where $\mathcal{O}\big(clD \times max(n_i)\big)$ is the time complexity of the update proces, $\mathcal{O}(lD)$ is the time complexity of the classification process, l is maximum depth of the Hoeffding tree, c is the number of class labels (in this case $c = 2$ because Hoeffding tree is used to solve the binary classification), $max(n_i)$ is the maximum number of values per feature [74]. Therefore, the time complexity of iBR and iCC is $\mathcal{O}\big(MclD \times max(n_i)\big)$. Clearly, the time complexity of our method is more dependent on the dimension of data D than the iBR and iCC.

Fig 9 shows the running time (in seconds) for the proposed method and benchmark algorithms summarized on three selected datasets, i.e., SynRTG-drift, TMC2007, and 20NG, which have 30, 500, and 1001 features, respectively. All source codes were implemented on a PC with Intel Core i7 with 2.5 GHz processor and 16G RAM. On high dimensional datasets such as 20NG and TMC2007, the proposed method takes significantly longer running time than the benchmark algorithms. On SynRTG-drift, a one million samples dataset with only 30 features, although our method runs slower than the benchmark algorithms, the difference between ours and the benchmark algorithms is less significant compared to that of 20NG and TMC2007. We noted this disadvantage of the proposed method although the running time is still within practical limit.

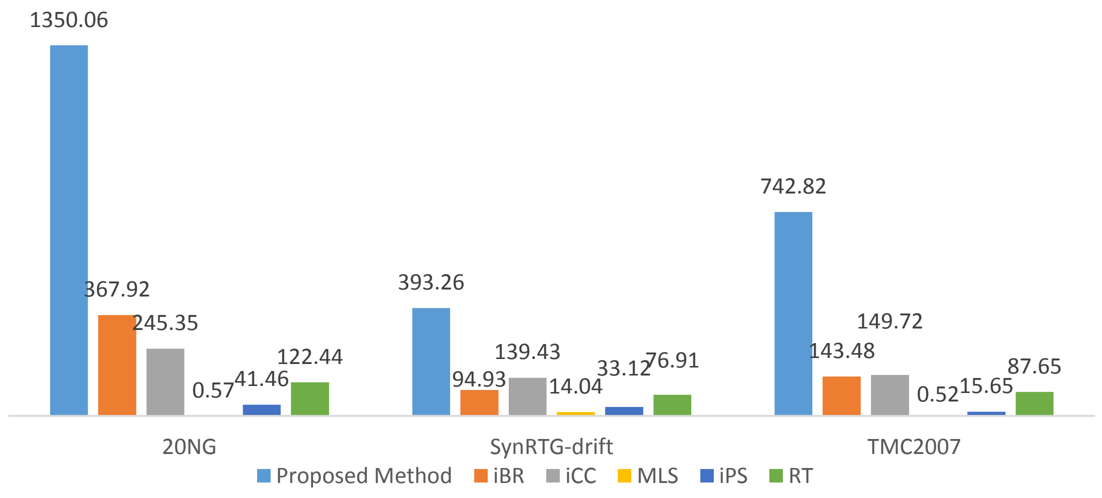


Fig.9. The running time (in seconds) of the proposed method and the benchmark algorithms on three selected datasets under concept drift setting

6. Conclusion

In this paper, we have introduced an incremental method for MLC of streaming data. We formulated the incremental instance-based MLC under a Bayes framework in which our algorithm predicts the set of labels for each arrived sample by taking into account the pairwise label correlation. We also proposed a mechanism to adjust the number of predict labels for each sample based on the Hoeffding inequality and a way to effectively handle missing values in the arriving samples. To adapt to concept drift, we proposed a decay mechanism on the label correlation and the 'label-feature-value' relationship which effectively reduces the influence of old samples and puts more attention on newer samples.

We performed extensive comparative study between the proposed method and several state-of-the-art benchmark algorithms on six real-world multi-label datasets under the stationary context, and three multi-label synthetic datasets under the concept drift setting. The experimental results showed that our method is better than the benchmark algorithms, ranking first in five out of six performance measures under both the stationary and concept drift settings. On some datasets like 20NG, SLASHDOT, and OHSUMED, the performance of our algorithm is far better than the benchmark algorithms. Besides, our approach to handle missing values is also more effective than the traditional mode imputation approach.

Although our algorithm has shown very good performance, it has high time complexity compared to existing algorithms. As noted, the time complexity of our algorithm is mainly depended on the dimension of the data. An incremental dimension reduction or feature selection method will be needed to decrease the running time of the proposed algorithm. This will be our future work.

Appendix

Hoeffding inequality and the number of predicted labels

In probability theory, Hoeffding inequality provides a bound on the probability that the sum of independent random variables will deviate from its expected value by more than a certain amount. We restate the result of the Hoeffding inequality [62] as the theoretical basis of our approach: If X_i $i = 1, \dots, N$ are independent random variable and if $a_i \le X_i \le b_i$, $S = \sum_{i=1}^{N} X_i$, $\bar{X} = S/N$, $\mu = E[\bar{X}]$ then for $\varepsilon > 0$

a) $\mathrm{P}\{\bar{X} - \mu \ge \varepsilon\} \le \exp\left\{\frac{-2N^2\varepsilon^2}{\sum_{i=1}^{N}(a_i - b_i)^2}\right\}$ and $\mathrm{P}\{\mu - \bar{X} \ge \varepsilon\} \le \exp\left\{\frac{-2N^2\varepsilon^2}{\sum_{i=1}^{N}(a_i - b_i)^2}\right\}$

b) $\mathrm{P}\{|\bar{X} - \mu| \ge \varepsilon\} \le 2\exp\left\{\frac{-2N^2\varepsilon^2}{\sum_{i=1}^{N}(a_i - b_i)^2}\right\}$

In this study, we apply Hoeffding inequality to select the number of labels predicted for each sample. Assume that we have a stream with N arrived samples $(\mathbf{x}_i, Y_i)$ $i = 1, \dots, N$, in which the i^{th} sample has $|Y_i|$ labels. The label cardinality as the average number of labels of the stream is given by:

$$\bar{z}_N = \frac{1}{N}\sum_{i=1}^{N}|Y_i| \tag{A1}$$

We denote $\bar{z} \coloneqq \bar{z}_N$ to keep the notation simple. Applying the Hoeffding inequality with the note that $0 \le |Y_i| \le L$ for all $i = 1, \dots, N$, and L is the number of distinct labels of the data stream, we have:

$$\mathrm{P}\{|\bar{z} - E[\bar{z}]| \ge \varepsilon\} \le 2\exp\left\{\frac{-2N^2\varepsilon^2}{NL^2}\right\} = 2\exp\left\{\frac{-2N\varepsilon^2}{L^2}\right\} \tag{A2}$$

Denote the right hand of (A2) by δ, ε is computed as:

$$\varepsilon = \sqrt{\frac{L^2 \ln(2/\delta)}{2N}} \tag{A3}$$

So (A2) becomes:

$$\mathrm{P}\{|\bar{z} - E[\bar{z}]| \le \varepsilon\} \ge 1 - \delta \tag{A4}$$

If $|h - \bar{z}| > \varepsilon$, based on the inequality $|h - E[\bar{z}]| = |(h - \bar{z}) - (E[\bar{z}] - \bar{z})| \ge |h - \bar{z}| - |E[\bar{z}] - \bar{z}|$ and combine with (A4), we have:

$$\mathrm{P}\{|h - E[\bar{z}]| > 0\} \ge 1 - \delta \tag{A5}$$

That means h is different from $E[\bar{z}]$ with a probability of at least $1 - \delta$ if $|h - \bar{z}| > \varepsilon$. In this case, we update h by $h = round(\bar{z})$ and reset $\bar{z}$, N, and L to begin the new period with a new value of h until the next update occurs.

References

[1] M.-L. Zhang, Z.-H. Zhou, A review on Multi-Label Learning Algorithms, IEEE Transactions on Knowledge and Data Engineering, Volume: 26, Issue: 8, Aug. 2014.

[2] T.T. Nguyen, T.T.T. Nguyen, X.C. Pham, A.W.-C. Liew, J.C. Bezdek, An ensemble-based online learning algorithm for streaming data, preprint arXiv: 1704.07938, 2017.

[3] T.T.T. Nguyen, T.T. Nguyen, X.C. Pham, A.W.-C. Liew, Y. Hu, T. Liang, C.-T. Li, A Novel Online Bayes Classifier, in Proceeding of Digital Image Computing: Techniques and Applications, 2016.

[4] T.T.T. Nguyen, A.W.-C. Liew, T.T. Nguyen, S. Wang, A novel Bayesian framework for Online Imbalanced Learning, in Proceeding of Digital Image Computing: Techniques and Applications, 2017.

[5] A. Bifet, R. Gavaldà, Adaptive learning from evolving data streams, In Advances in Intelligent Data Analysis VIII (pp. 249–260). Springer, 2009.

[6] J. Read, A. Bifet, G. Holmes, B. Pfahringer, Scalable and efficient multi-label classification for evolving data stream, Machine Learning. 88, 2012, 243-272.

[7] A. Osojnik, P. Panov, S. Dzeroski, Multi-label classification via multi-target on data streams, Machine Learning. 106, 2017, 745-770.

[8] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, X. Geng, Binary relevance for multi-label learning: an overview, Front. Comput. Sci., 2017, 1-12.

[9] E.S. Xioufis, M. Spiliopoulou, G. Tsoumakas, I. Vlahavas, Dealing with Concept Drift and Class Imbalance in Multi-Label Stream Classification, in Proceeding of 22nd International Conference on Artificial Intelligence, 2011.

[10] G. Tsoumakas, K. Ioannis, Multi-label classification: An overview, in Int J Data Warehousing and Mining. 2007, 1–13.

[11] M. R. Boutell, J. Luo, X. Shen, C. M. Brown, Learning multi-label scene classification, Pattern Recognition, vol. 37, no. 9, pp. 1757–1771, 2004.

[12] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, in Lecture Notes in Artificial Intelligence 5782, W. Buntine, M. Grobelnik, and J. Shawe-Taylor, Eds. Berlin: Springer, 2009, pp. 254–269.

[13] K. Dembczýnski, W. Cheng, E. Hullermeier, Bayes optimal multilabel classification via probabilistic classifier chains, in Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 2010, pp. 279–2.

[14] J. Read, Martino Luca, Luengo, David Efficient, Monte Carlo methods for multi-dimensional learning with classifier chains, Pattern Recognition. Handwriting Recognition and other PR Applications. 47 (3): 1535–1546, 2014.

[15] J. Nam, E.L. Mencía, H.J. Kim, J. Furnkranz, Maximizing Subset Accuracy with Recurrent Neural Networks in Multi-Label Classification, in Proceeding of NIPS, 2017.

[16] G. Tsoumakas and I. Vlahavas, Random k-labelsets: an ensemble method for multilabel classification, in Lecture Notes in Artificial Intelligence 4701, J. N. Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladeniˇc, and A. Skowron, Eds. Berlin: Springer, 2007, pp. 406–417.

[17] G. Tsoumakas, I. Katakis, and I. Vlahavas, Random k-labelsets for multi-label classification, IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 7, pp. 1079–1089, 2011.

[18] Y. Guo, S.Gu, Multi-label classification using conditional dependency networks, in Proceeding of the 22nd IJCAI, 2011, pp 1300-1305.

[19] J. Read, B. Pfahringer, G. Holmes, Multi-label Classification using Ensembles of Pruned Sets, in Proc. of IEEE International Conference on Data Mining, 2008, pp. 995–1000.

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

[21] A. Elisseeff and J. Weston, A kernel method for multi-labeled classification, in Proceeding of 14th NIPS, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, pp. 681–687.

[22] A. Clare, R. D. King, Knowledge discovery in multi-label phenotype data, in Lecture Notes in Computer Science 2168, L. De Raedt and A. Siebes, Eds. Berlin: Springer, 2001, pp. 42–53.

[23] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multilabel classification. Machine Learning, 73(2), 185–214, 2008.

[24] M.-L. Zhang, J.M. Peña, V. Robles, Feature selection for multi-label naive Bayes classification, Information Sciences. 179(19) (2009), 3218-3229.

[25] S. Ubaru, A. Mazumdar, Multilabel Classification with Group Testing and Codes, in Proceeding of ICML, 2017.

[26] A. Kapoor, R. Viswanathan, P. Jain, Multilabel Classification using Bayesian Compressed Sensing, in Proceeding of NIPS, 2012.

[27] Y.-N. Chen, H.-T. Lin, Feature-aware label space dimension reduction for multi-label classification, In Advances in Neural Information Processing Systems, 2012, pp. 1529-1537.

[28] Y. Zhang, J.G. Schneider, Multi-label output codes using canonical correlation analysis, In AISTATS, pp. 873-882, 2011.

[29] Y. Lin, Q. Hu, J. Zhang, Z. Wu, Multi-label feature selection with stream labels, Information Sciences. 372 (2016), 256-275.

[30] J. Lee, D.-W. Kim, SCLS: Multi-label feature selection based on scalable criterion for large dataset, Pattern Recognition. 66 (2017), 342-352.

[31] X.C. Pham, M.T. Dang, S.V. Dinh, S. Hoang, T.T. Nguyen, Alan Wee-Chung Liew, Learning from Data Stream based on Random Projection and Hoeffding Tree Classifier, in Proceeding of Digital Image Computing: Techniques and Applications, 2017.

[32] Y. Zhang, Z-H. Zhou, Multi-Label Dimensionality Reduction via Dependence Maximization, ACM Transactions on Knowledge Discovery from Data. 4(3), 2010, 1-21.

[33] C. Wang, S. Yan, L. Zhang, H.-J. Zhang, Multi-label sparse coding for automatic image annotation, in Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, 1643-1650.

[34] W. Qu, Y. Zhang, J. Zhu, Q. Qiu, Mining multi-label concept-drifting data streams using dynamic classifier ensemble. In Advances in machine learning (pp. 308–321). Springer, 2009.

[35] M.T. Dang, A.V. Luong, T.-T. Vu, Q.V.H. Nguyen, T.T. Nguyen, B. Stantic, An ensemble system with random projection and dynamic ensemble selection, in Proceeding of ACIIDS, 2018.

[36] L. Wang, H. Shen, H. Tian, Weighted Ensemble Classification of Multi-label Data Streams, in J. Kim et al. (Eds.), PAKDD Part II, Lecture Note in Artificial Intelligence, pp. 551-562, 2017.

[37] P. Domingos, G. Hulten. Mining high-speed data streams. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 71–80.

[38] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New Ensemble Methods For Evolving Data Streams, in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, Paris, France, June28-July 01 2009. (pp. 139-148). New York, USA: ACM.

[39] Z. Shi, Y. Xue, Y. Wen, G. Cai, Efficient Class Incremental Learning for Multi-label classification of Evolving Data Streams, International Joint Conference on Neural Networks 2014.

[40] Z. Shi, C. Feng, Y. Wen, H. Zhao, Drift Detection for Multi-label Data Streams Based on Label Grouping and Entropy, In IEEE International Conference on Data Mining Workshop, 2014.

[41] K. Karponi, G. Tsoumakas, An Empirical Comparison of Methods for Multi-Label data Stream Classification, in P. Angelov et al. (eds.), Advances in Big Data, Advantages in Intelligent Systems and Computing 529, 2017.

[42] J. Read, P. Reutemann, B. Pfahringer, G. Holmes, MEKA: A Multi-label/Multi-target Extension to WEKA. Journal of Machine Learning Research. 17(21):1−5, 2016.

[43] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, MULAN: A Java library for multi-label learning, Journal of Machine Learning Research, vol. 12, no. Jul, pp. 2411–2414, 2011.

[44] P. Domingos, M. Pazzani, Beyond independence: Conditions for the optimality of the simple Bayesian classifier, in Proceeding of 13th ICML, 105-112, 1996.

[45] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. Machine Learning, 29 (2):131–163, 1997.

[46] G.I. Webb, J.R. Boughton, Z. Wang, Not So Naïve Bayes: Aggregating One-Dependence Estimators, Machine Learning. 58, 2005, 5-24.

[47] G. I. Webb, J. Boughton, F. Zheng, K. M. Ting, and H. Salem. Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. Machine Learning, pages 1–40, 2011.

[48] F. Zheng, G. I. Webb, P. Suraweera, and L. Zhu. Subsumption resolution: An efficient and effective technique for semi-naive Bayesian learning. Machine Learning, 87(1) (2012), 93–125.

[49] N.A. Zaidi, J. Cerquides, M.J. Carman, G.I. Webb, Alleviating Naive Bayes Attribute Independence Assumption by Attribute Weighting, Journal of Machine Learning Research 14 (2013) 1947-1988.

[50] F. Zheng, G.I. Webb, Efficient Lazy Elimination for Averaged-One Dependence Estimators. In: Proceedings of the Twenty-third International Conference on Machine Learning (ICML 2006), 1113-1120, 2006.

[51] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering, In Journal of Machine Learning Research Workshop and Conference Proceedings, Vol.11: Workshop on Applications of Pattern Analysis, 2010. URL: http://moa.cms.waikato.ac.nz (the latest release of June 2017).

[52] T.T. Nguyen, T.T.T. Nguyen, X.C. Pham, A.W.-C. Liew, A Novel Combining Classifier Method based on Variational Inference, Pattern Recognition 49(2016) 198-212.

[53] T.T. Nguyen, M.P. Nguyen, X.C. Pham, A.W.-C. Liew, Heterogeneous Classifier Ensemble with Fuzzy Rule-based Meta Learner, Information Sciences. 422, 2018, 144-160.

[54] T.T. Nguyen, A.W.-C. Liew, X.C. Pham, M.P. Nguyen, A Novel 2-Stage Combining Classifier Model with Stacking and Genetic Algorithm Based Feature Selection, in: D.-S. Huang, K.-H. Jo, L. Wang (Eds.), Intelligent Computing Methodologies, Springer International Publishing, 2014, pp. 33-43.

[55] T.T. Nguyen, A.W.-C. Liew, M.T. Tran, M.P. Nguyen, Combining Multi Classifiers Based on a Genetic Algorithm – A Gaussian Mixture Model Framework, in: D.-S. Huang, K.-H. Jo, L. Wang (Eds.), Intelligent Computing Methodologies, Springer International Publishing, 2014, pp. 56-67.

[56] T.T. Nguyen, A.W.-C. Liew, M.T. Tran, X.C. Pham, M.P. Nguyen, A novel genetic algorithm approach for simultaneous feature and classifier selection in multi classifier system, in: IEEE Congress on Evolutionary Computation (CEC), 2014, pp.1698-1705.

[57] T.T. Nguyen, A.W.-C. Liew, X.C. Pham, M.P. Nguyen, Optimization of ensemble classifier system based on multiple objectives genetic algorithm, International Conference on Machine Learning and Cybernetics (ICMLC), 2014 (Vol.1), pp. 46 – 51.

[58] T.T. Nguyen, A.W.-C. Liew, M.T. Tran, T.T.T. Nguyen, M.P. Nguyen, Fusion of Classifiers Based On A Novel 2-Stage Model, in: X. Wang, W. Pedrycz, P. Chan, Q. He (Eds.), Machine Learning and Cybernetics, Springer, 2014, pp. 60-68.

[59] T.T. Nguyen, X. C. Pham, A.W.-C. Liew, W. Pedrycz, Aggregation of classifiers: a justifiable information granularity approach, IEEE Transactions on Cybernetics, DOI: 10.1109/TCYB.2018.2821679, In Press.

[60] T.T. Nguyen, A.W.-C. Liew, M.T. Tran, M.P. Nguyen, Combining Classifiers Based on Gaussian Mixture Model Approach to Ensemble Data, in: X. Wang, W. Pedrycz, P. Chan, Q. He (Eds.), Machine Learning and Cybernetics, Springer, 2014, pp. 3-12.

[61] F. Shi, J. Wang, Z. Wang, Region-based supervised annotation for semantic image retrieval, International Journal of Electronics and Communications (AEU). 65 (2011), 929-936.

[62] W. Hoeffding, Probability Inequalities for Sums of Bounded Random Variables, Journal of the American Statistical Association. 58 (301) (1963), pp.13-30.

[63] J. Read, Scalable multi-label classification, PhD Thesis, University of Waikato, 2010.

[64] A.W.C. Liew, N.F. Law, and H. Yan, "Missing value imputation for gene expression data: computational techniques to recover missing data from available information", Briefings in Bioinformatics, Vol.12:5, Sep 2011, pp. 498-513.

[65] R. Deb, A.W.C. Liew, Missing value imputation for the analysis of incomplete traffic accident data, Information Sciences. 339 (2016), 274–289.

[66] X. Gan, A.W.C. Liew, and H. Yan, "Microarray Missing Data Imputation based on a Set Theoretic Framework and Biological Consideration", Nucleic Acids Research, 34 (2006), 1608-1619.

[67] J. Gama, P. Medas , G. Castillo, Pedro Rodrigues, Learning with Drift Detection, in Advances in Artificial Intelligence, Springer Berlin Heidelberg. 2004, vol 3171, pp. 268 – 295.

[68] A. Bifet, R. Gavaldà, Learning from Time-Changing Data with Adaptive Windowing, in Proceeding of ICDM, 2007.

[69] E. Cohen, M. Strauss, Maintaining Time-Decaying Stream Aggregates, Journal of Algorithms. 59 (1), 2006, 19-36.

[70] Y. Sun, K. Tang, L.L. Minku, S. Wang, X. Yao, Online Ensemble Learning of Data Streams with Gradually Evolved Classes, IEEE Transactions on Knowledge and Data Engineering. 28 (6), 2016, 1532-1545.

[71] J. Gama, R. Sebastiao, P. P. Rodrigues, Issues in Evaluation of Stream Learning Algorithms, in Proceeding of KDD, 2009.

[72] J. Demsar, Statistical comparisons of classifiers over multiple datasets, Journal of Machine Learning Research 7 (2006), 1–30

[73] S. Garcia, F. Herrera, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, Journal of Machine Learning Research 9 (2008), 2579–2596.

[74] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, 2001, pp 97-106.

Author Biographies

Tien Thanh Nguyen received his PhD degree in computer science from the School of Information & Communication Technology, Griffith University, Australia in 2017. He is currently a Research Fellow in the School of Computing Science and Digital Media, Robert Gordon University, Aberdeen, Scotland, UK. His research interest is in the field of machine learning, pattern recognition, and evolutionary computation. He is a member of the IEEE since 2014.

Thi Thu Thuy Nguyen is currently a PhD student at the School of Information & Communication Technology, Griffith University, Australia. She graduated from the Faculty of Mathematics, Voronezh State University, Russia in 2008. Her research interest is in the field of applied mathematics, machine learning, and pattern recognition.

Anh Vu Luong has recently completed his advanced undergraduate study at the School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, Vietnam. His research interest is in the field of machine learning and pattern recognition.

Quoc Viet Hung Nguyen is currently a Lecturer at the School of Information & Communication Technology, Griffith University, Australia. His research focuses on Data Integration, Data Quality, Information Retrieval, Trust Management, Recommender Systems, Machine Learning and Big Data Visualization, with special emphasis on web data, social data and sensor data.

Alan Wee-Chung Liew is currently an Associate Professor at the School of Information & Communication Technology, Griffith University, Australia. His research interest is in the field of medical imaging, bioinformatics, computer vision, pattern recognition, and machine learning. He serves on the technical program committee of many international conferences and is on the editorial board of several journals, including the IEEE Transactions on Fuzzy Systems. He is a senior member of the IEEE since 2005.

Bela Stantic is currently a Professor and HOS at the School of Information & Communication Technology, Griffith University, Australia. His research interest is in the area of big data analytics, databases, and bioinformatics.