Multi-label classification via incremental clustering on an evolving data stream.

NGUYEN, T.T., DANG, M.T., LUONG, A.V., LIEW, A. W.-C., LIANG, T. and MCCALL, J.

2019

Multi-Label Classification via Incremental Clustering on Evolving Data Stream

Tien Thanh Nguyen[1], Manh Truong Dang[1], Anh Vu Luong[2], Alan Wee-Chung Liew[2], Tiancai Liang[3], John McCall[1]

[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] GRGBanking Technology Co., Ltd, China

Abstract: With the advancement of storage and processing technology, an enormous amount of data is collected on a daily basis in many applications. Nowadays, advanced data analytics have been used to mine the collected data for useful information and make predictions, contributing to the competitive advantages of companies. The increasing data volume, however, has posed many problems to classical batch learning systems, such as the need to retrain the model completely with the newly arrived samples or the impracticality of storing and accessing a large volume of data. This has prompted interest on incremental learning that operates on data streams. In this study, we develop an incremental online multi-label classification (OMLC) method based on a weighted clustering model. The model is made to adapt to the change of data via the decay mechanism in which each sample's weight dwindles away over time. The clustering model therefore always focuses more on newly arrived samples. In the classification process, only clusters whose weights are greater than a threshold (called mature clusters) are employed to assign labels for the samples. In our method, not only is the clustering model incrementally maintained with the revealed ground truth labels of the arrived samples, the number of predicted labels in a sample are also adjusted based on the Hoeffding inequality and the label cardinality. The experimental results show that our method is competitive compared to several well-known benchmark algorithms on six performance measures in both the stationary and the concept drift settings.

Keywords: multi-label classification, incremental learning, online learning, clustering, data stream, concept drift

1. Introduction

Nowadays, an enormous amount of data is collected on a daily basis, from mobile devices to social networking sites. Data is growing at an astonishing rate according to IDC Research, at a compound annual growth rate of 42% through to 2020. This means that 90% of the data in the whole world has been created over the past two years. In recent years, 'big data' is one of the most popular terms mentioned in the media.

Realizing that data is a hidden resource, many companies have invested heavily in advanced data analytics to mine customer and sales data for useful information in order to gain a competitive edge. For example, about 35% of Amazon.com's revenue is generated by its recommendation system. However, the tremendous growth in the volume of data has also posed many challenges to classical machine learning systems. First, batch learning performed on large volume of data is sometime impractical, resulting in more and more data not been processed [1]. In addition, learning models trained on existing data become outdated with the appearance of new data. Periodic re-training using the accumulated data can only be a temporary solution, and resulted in huge resource consumption [2, 3]. Learning models that are maintained incrementally by integrating new information acquired from newly arrived data are, therefore, more practical [3].

In this paper, we introduce an incremental online learning method for the multi-label classification (MLC) problem. The MLC problem arises from many real-world applications where an entity is described by multiple terms or having multiple semantic meanings. As a generalization of the multi-class classification problem, the task of MLC is to assign a set of labels to an object to express its semantics. In the literature, there are MLC algorithms for both the batch [4] and stream settings [5, 6]. In this study, we derive a clustering based MLC algorithm based on an online clustering algorithm adopted from [7] to solve the MLC problem. The proposed method can adapt to concept drift in the data stream by focusing more on the newly arrived samples using a decay mechanism [8]. In addition, the label distributions in the K closest clusters are used during the MLC to predict a set of labels for a newly arrived sample. In contrast to [5, 9] where an update of the number of labels assigned to each arrived sample is performed only after a fixed set of samples are received, our algorithm performs continuous update of the number of labels for each arrived sample.

The main contributions of this paper are:

- An online learning algorithm using the clustering model is proposed for the MLC problem.
- The clusters in our algorithm evolve with time, giving higher attention to more recent samples than older samples through a weight decay mechanism.
- A novel approach for learning the number of predicted labels for MLC based on the Hoeffding inequality and the label cardinality is proposed.
- An empirical demonstration that our method is competitive to several well-known benchmark algorithms in both the stationary and concept drift settings.

The paper is organized as follows. In Section 2, we briefly review several well-known learning algorithms for MLC in the supervised learning and stream learning settings. In section 3, we describe the proposed online MLC algorithm based on the online clustering model. The setting for experimental studies is described in section 4. Section 5 presents the detailed experimental results and discussion. Finally, Section 6 provides the conclusions and suggestions for further study.

TABLE.1. SUMMARY OF MAIN NOTATION

Notation	**Description**
$\boldsymbol{\mathcal{D}}$	The stream of data
m_C	A mature cluster
im_C	An immature cluster
$m\_\boldsymbol{\mathcal{C}} = \{m\_C\}$	The set of all mature clusters
$\mathbf{c} = (c_j)\ j = 1, \dots, d$	The center of a cluster
r	The radius of a cluster
θ	The boundary of a cluster
λ	Decay control parameter
$\mathbf{x} = (x_j)\ j = 1, \dots, d$	A sample
$\mathbb{Y}_{\mathrm{x}} = \{y\}$	The label set of $\mathbf{x}$
$\widehat{\mathbb{Y}}_{\mathrm{x}}$	The predicted label set for $\mathbf{x}$
$\boldsymbol{\mathcal{Y}}$	The label set
W	The mature weight of a cluster
W_0	The threshold of mature weight
$\mathbf{p} = (p(l))\ l = 1, \dots, \lvert\boldsymbol{\mathcal{Y}}\rvert$	The label distribution of a cluster
$\mathcal{K}(\mathbf{x})$	K-nearest mature clusters of $\mathbf{x}$
h	The number of predicted labels
z	Label cardinality

2. Background

2.1. Multi-label classification

Let $\mathbb{X}$ denote the input space and $\mathbb{Y} = \{y_i | i = 1, \dots, M\}$ denote the label set. The purpose of a multi-label learning task is to search for a mapping function f 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 a generalization of the traditional multi-class classification problem in which each sample is associated with only a single label.

MLC algorithms can often be categorized into two approaches: problem transformation and algorithm adaptation [4]. In the first category, the MLC problem is transformed into some well-established learning problems such as binary classification. Two common approaches in this category are the Binary Relevance (BR) and Classifier Chains (CC) approaches, where a multi-label task is transformed into M binary classification tasks. The difference between BR and CC is that BR treats the labels independently in the learning process as each binary classification task is associated with a label in the label set. Meanwhile, CC creates the new training set for each binary problem by appending each instance with binary values that indicate which of the previous labels were assigned to that sample [4]. CC therefore has the advantage over BR of addressing label correlation. In practice, an ensemble of CC classifiers are generated via random orders over the label space instead of using a single CC to overcome the issue of label ordering in the chain. Several methods have also been introduced to improve CC's effectiveness, such as replacing binary values by probabilistic outputs [10] and using recurrent neural networks focusing only on positive labels as an extension of probabilistic CC [11]. Kumar et al. [12] improved PCC by using beam search, a classical heuristic search algorithm, so that instead of evaluating on 2^M possible labellings, only bM combinations need to be assessed (b is the beam width). The search also integrate to the learning algorithm to obtain the best order of labels. Ghamrawi and McCallum [13] modeled the dependencies between labels by constructing a graphical model to parameterize the pairwise relationships of feature-label, label-label, and feature-label-label triple. The Label Powerset (LP) is another popular method in this category which treats each different combination of labels as a single label [14]. Although LP can capture the label correlations in the learning model, it has high-complexity in training due to the exponential increase of the number of label combinations with the number of labels. LP is also unable to predict the label combinations that do not appear in the training set [6]. Read et al. [15] proposed the Pruned Set (PS) method which removes the samples belonging to the infrequent label sets to reduce the number of label combinations. Other MLC approaches considered the subsets of labels in a random way such as Random k-Labelsets [14], or in a deterministic way like in dependency network [16].

The algorithm adaptation approach is a group of methods that are adapted from multi-class classification algorithms to solve the MLC problem. Several methods can be mentioned, for example, k-Nearest Neighbor for MLC [17], Support Vector Machine for MLC [18], and Decision Tree for MLC [19].

In the era of big data, recent research on multi-label learning mainly focuses on dealing with large-scale multi-label data, especially on data with a large label set and data that come in the form of a stream. Ubaru and Mazumdar [20] used group testing and coding techniques to compress the label set to reduce the label dimension. Kapoor et al. [21] used compressed sensing in the Bayesian framework for label dimension reduction. SVD techniques was used to project the label vector onto a low dimensional space to reduce its dimension [22]. Besides, the performance of MLC systems can be enhanced 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 [23] and scalable relevance evaluation feature selection that measures feature dependency [24]. Several expansions of MLC for multi-dimensional classification (MDC, also known as multi-output classification) which is viewed as a general case of MLC where each label can take a number of discrete values. This setting increases the search space of the chain-sequences, resulting in costly testing time. Read et al. [25] proposed the classifier trellis to effectively capture the label correlation in MDC by sequentially placing the labels to the pre-defined trellis structure for the underlying graphical model. Read et al. [26] introduced double Monte Carlo optimization technique to search for the best classifier chain in the training phase and best label vector for the test sample for MDC.

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

The large volumes and the rapid growth of data have posed many challenges for traditional offline machine learning systems. First, it is often impractical or even infeasible for a learning algorithm to learn on the entire dataset at once. The newly arrived data also often make the model learned on the old data outdated, causing the degradation of the system performance. Although we can re-train the learning model with the arrival of new data, the re-training process on the continuously arriving data will consume much more time and resource. Incremental learning methods in which the learning model can be updated on-the-fly from the data stream are therefore highly desirable.

The characteristics of data stream present unique challenges to the design of learning algorithms. Bifet and Gavaladà [27] defined four characteristics of learning on data stream: (1) the model must be ready to make prediction on any sequentially arriving samples, (2) there are potentially infinitely many samples which need to be processed with finite resources (time and memory), (3) the samples need not be statistically stationary (the appearance of concept drift), and (4) samples can only be processed one at a time, and can only be inspected once before it is discarded. In this study, we aim to develop an incremental OMLC method in which the current learning model trained on the old data is used to predict unlabelled data. Only samples where true label can be revealed is used to update the learning model. As a result, both prediction and training are considered in the proposed method. We also address the challenges of data stream's characteristics in designing the proposed method.

One of the earliest approaches that solves the MLC problem in the stream context is the batch-based incremental method [28]. In this method, an ensemble of BR-based classifiers is generated by learning the BR on each sequence of same-size-chunks. The outputs of these classifiers are concatenated to the original feature space as the meta-data which is learned by another BR to obtain the meta-classifier. Based on the dynamic classifier ensemble approach, the classifiers are weighted on each test sample via their performance on the test sample's neighbors obtained from the latest chunk. A similar approach was introduced by Wang et al. [29] in which the data stream is divided into many fixed numbers of chunks.

Multi-label K Nearest Neighbor method [17] is then used to learn on these chunks to generate the ensemble of classifiers. These classifiers are also weighted and the weights are incrementally maintained based on the newly arrived chunks. Despite the ability to operate in both the stationary and concept drift settings, these methods face the memory-fill-up problem so they cannot satisfy the time and memory constraints of stream learning [5].

Read et al. [5] adapted the Hoeffding tree [30], a well-known member of the decision tree family, for MLC problem on data stream. In their method, the arrived samples are temporarily kept and the Hoeffding bound is used to determine how many samples are needed to achieve a certain level of confidence for tree splitting. The PS classifier [15] was used to prune the label combination at each leaf node when the buffer of arrived samples is full at the node. That framework was also combined with ADWIN [31] to form a new algorithm named EaHTps which can handle concept drift. The Pruned Set-based label combination module of EaHTps was improved by Shi et al. [32] in which the new frequent label combinations are dynamically recognized to update the set of label combinations. Xioufis et al. [9] used BR to solve the MLC by transforming the multi-label task into several binary classification tasks. Concept drift was handled by maintaining two variable-size windows per label for positive and negative samples. Compared to the single window approach for each label, that method can oversample the positive sample by adding the previous positive samples to the associated window and undersample the negative sample by keeping only the most recent ones. Shi et al. [33] created the super-label which is a set of class labels grouped based on their dependencies. The generated super-labels are treated as new class labels 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. These windows are resized when the difference exceeds a pre-defined threshold. Osojnik et al. [6] applied multi-target regression to multi-label learning on data stream. In their approach, the MLC problem and the multi-target regression problem are transformed back and forth as the solution is obtained by transforming the multi-target regression problem back to the MLC problem to obtain the predicted labels. The four multi-target tree-based methods introduced in [6], however, are only focused on learning stationary concept. A comparative study between several MLC methods for data stream such as BR with different learning algorithms and multi-label Hoeffding tree (with PS and Naïve Bayes at the leaves) was made by Karponi and Tsoumakas [34]. However, the experiments were conducted on just one reduced dataset, and therefore the conclusion is not convincing.

3. Proposed method

3.1. The online clustering model

The change in data often makes the learning model that is built on old data inconsistent with the new data, therefore it is crucial to adapt a learning model to concept drift. In the literature, there are two main strategies to deal with concept drift, namely, using a sliding window or a decay function (also called weighted samples [8]) [31]. In the first strategy, the windows are maintained that keep the most recent samples inside while discarding the old samples outside. Several methods use a single window of a fixed size or of variable size. For example, in ADWIN, a well-known concept drift handling method, the 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 [31]. Other research such as [32, 33] use two adjustable windows to represent old and new samples. Xioufis et al. [9] used two windows per label to capture the positive and negative groups of samples. The decay function approach, meanwhile, use weights to mitigate or strengthen the importance of samples based on their age, i.e., the older sample is less important than the new one [35]. The learning model, therefore, can adapt to the changes that appear in the new data.

In this study, we follow the decay function approach to develop an incremental online MLC method that adapts to the changes of data in the data stream. Our approach aggregates the information from incoming samples into clusters based on their proximity with each other as well as their time of arrival. We require that the weight of each sample decreases gradually over time. The incremental MLC, therefore, focuses more on the new samples than the old ones. Here the weight of each sample is decayed *exponentially* with time t via the fading function $f(t) = 2^{-\lambda t}$ where $\lambda > 0$ is the parameter that controls the decay rate [7]. From these data points and their weights, we build the online clustering model for the incremental online MLC method. First, we define the online clustering model:

Definition 1 (Mature cluster) [7]: A mature cluster C (m_C) at time t for the group of close points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$, $\mathbf{x}_i = (x_{i1}, \dots, x_{id})$ with time stamps $T_1, T_2, \dots, T_n$ is defined as the 3-tuple $\{\mathbf{CF^1}, \mathbf{CF^2}, W\}$ in which the weight $W = \sum_{i=1}^{n} f(t - T_i) > W_0$, W_0 is a pre-defined mature threshold, $\mathbf{CF^1} = (CF_j^1)$, $CF_j^1 = \sum_{i=1}^{n} f(t - T_i)\, x_{ij}$ is the weighted linear sum of the j^{th} feature, , $\mathbf{CF^2} = (CF_j^2)$, $CF_j^2 = \sum_{i=1}^{n} f(t - T_i)\, x_{ij}^2$ is the weighted squared sum of the j^{th} feature. The center of m_C is $\mathbf{c} = (c_j)$, $c_j = \frac{CF_j^1}{W}$, the radius of m_C is $r = \max_{j=1,\dots,d}\left\{\sqrt{\frac{CF_j^2}{W} - \left(\frac{CF_j^1}{W}\right)^2}\right\} \leq \theta$

Definition 2 (Immature cluster) [7]: A immature cluster C (im_C) at time t for the group of close points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$, $\mathbf{x}_i = (x_{i1}, \dots, x_{id})$ with time stamps $T_1, T_2, \dots, T_n$ is defined as $\{\mathbf{CF^1}, \mathbf{CF^2}, W\}$. The definitions of $\mathbf{CF^1}, \mathbf{CF^2}, W, \mathbf{c}$, and r are the same as those of m_C, however $W \leq W_0$.

From the definition 1 and 2, a cluster (mature or immature) is a set of data point bounded by the pre-defined threshold θ. The center of a cluster is the weighted average of all data points inside the cluster whereas the radius is the maximum value among all standard deviations of the d features. We illustrate an example of the cluster evolution from immature to mature cluster on the EURON dataset in Fig 1. In this example we set $\lambda = 0.25, W_0 = 2,$ and $\theta = 0.495$. At time $t = 0$, a cluster is generated with a newly arrived sample $\boldsymbol{x}_{47}$ with time stamp 0. This is an immature cluster because its weight is 1 (equal to the weight of $\boldsymbol{x}_{47}$). At $t = 2$, a new sample $\boldsymbol{x}_{298}$ arrives to the cluster with the weight 1 and time stamp 2. Meanwhile, the weight of $\boldsymbol{x}_{47}$ is $1 \times 2^{-0.25\times 2} = 0.7071$. As the total weight of this cluster is smaller than W_0, the cluster is still an immature cluster. At $t = 3$, we have a new sample $\boldsymbol{x}_{351}$ which arrives with time stamp 3. The weights of $\boldsymbol{x}_{47}$ and $\boldsymbol{x}_{298}$ are 0.5946 and 0.8409, respectively. That makes the weight of the cluster greater than W_0 and the cluster becomes a mature cluster. It is noted that in all cases, the radius of the cluster must be smaller than the pre-defined maximum threshold of radius $\theta = 0.495$.
In this model, the mature cluster is trusted more than the immature cluster so that only the mature clusters will be used during the classification process. It is noted that definitions 1 and 2 is for unsupervised learning, i.e. the samples do not have labels. For MLC, each sample in a cluster is associated with a set of labels. We extend the mature and immature cluster for MLC as:

Definition 3 (Cluster for MLC): A cluster C (mature or immature) at time t for the group of close points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$, $\mathbf{x}_i = (x_{i1}, \dots, x_{id})$ with time stamps $T_1, T_2, \dots, T_n$ is defined as the 4-tuple $\{\mathbf{CF^1}, \mathbf{CF^2}, W, \mathbf{p}\}$ in which $\mathbf{CF^1}, \mathbf{CF^2}, W, \mathbf{c}$, and r are defined as in Definition 1 and $\mathbf{p} = \big(p(l)\big)\ l = 1, \dots, |\boldsymbol{\mathcal{Y}}|$ is the label distribution of the points.

In this definition, along with the components $\mathbf{CF^1}, \mathbf{CF^2}$, and W, we store the label distribution as the fourth component of each cluster. The distribution is approximated by the label frequency of all data points inside:

$$p(l) = \frac{\sum_{\mathbf{x}\in C}[\![l \in Y_{\mathbf{x}}]\!]}{|C|} \quad (1)$$

in which $[\![.]\!]$ returns 1 if the predicate holds and 0 otherwise.

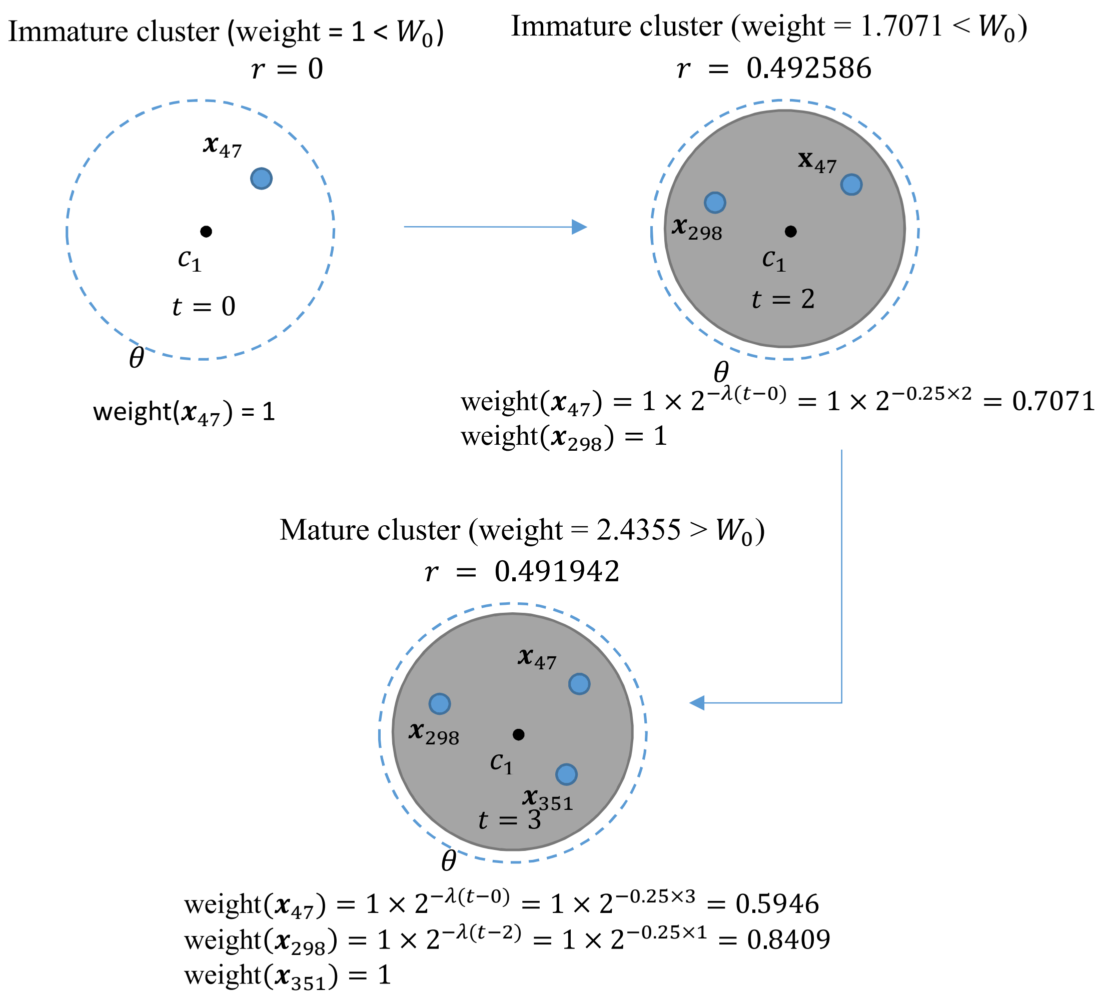

Immature cluster (weight = 1 < W_0)
$r = 0$
$\boldsymbol{x}_{47}$
c_1
$t = 0$
θ
weight($\boldsymbol{x}_{47}$) = 1
Immature cluster (weight = 1.7071 < W_0)
$r = 0.492586$
$\mathbf{x}_{47}$
$\boldsymbol{x}_{298}$
c_1
$t = 2$
θ
weight($\boldsymbol{x}_{47}$) $= 1 \times 2^{-\lambda(t-0)} = 1 \times 2^{-0.25\times2} = 0.7071$
weight($\boldsymbol{x}_{298}$) $= 1$
Mature cluster (weight = 2.4355 > W_0)
$r = 0.491942$
$\boldsymbol{x}_{47}$
$\boldsymbol{x}_{298}$
c_1
$\boldsymbol{x}_{351}$
$t = 3$
θ
weight($\boldsymbol{x}_{47}$) $= 1 \times 2^{-\lambda(t-0)} = 1 \times 2^{-0.25\times3} = 0.5946$
weight($\boldsymbol{x}_{298}$) $= 1 \times 2^{-\lambda(t-2)} = 1 \times 2^{-0.25\times1} = 0.8409$
weight($\boldsymbol{x}_{351}$) $= 1$

Fig.1. An illustration of cluster evolution on the EURON dataset

3.2. The model update

From the arrived data points, we incrementally learned the clustering model to solve the MLC problem. In this section, we introduce a method to construct the clustering structure for the MLC problem. The incremental process that maintains the clustering structure include:

- ***Update the cluster's components***: Due to the decay mechanism over the data points, the components of a cluster are updated over time, even if it does not receive any new data. In this work, we follow the update equations introduced in [7]. First, when the new data point $\mathbf{x} = (x_j)$ (its weight is 1) is merged into a cluster C, the new value of the first three components are computed as:

$$\left\{\left(CF_j^1\right)^{new}, \left(CF_j^2\right)^{new}, W^{new}\right\} = \left\{\left(CF_j^1\right)^{old} + x_j, \left(CF_j^2\right)^{old} + x_j^2, W^{old} + 1\right\} \quad (2)$$

Whereas, after the time interval δt, if the cluster C does not receive any new points, its components will be decayed due to the fact that the weight of all data points decreases by a factor of $2^{-\lambda\delta t}$ in the interval:

$$\left\{\left(CF_j^1\right)^{new}, \left(CF_j^2\right)^{new}, W^{new}\right\} = \left\{2^{-\lambda\delta t}\left(CF_j^1\right)^{old}, 2^{-\lambda\delta t}\left(CF_j^2\right)^{old}, 2^{-\lambda\delta t}W^{old}\right\} \tag{3}$$

Based on the new values of these components, the new center and radius are updated respectively:

$$\left(c_j\right)^{new} = \frac{\left(CF_j^1\right)^{new}}{W^{new}} \tag{4}$$

$$r^{new} = \max_{j=1,\dots,d}\left\{\sqrt{\frac{\left(CF_j^2\right)^{new}}{W^{new}} - \left(\frac{\left(CF_j^1\right)^{new}}{W^{new}}\right)^2}\right\} \tag{5}$$

The label frequency of the cluster that the data point is merged to is updated by using the ground truth labels if available. The incremental update of the label frequency is given by:

$$p^{new}(l) = \frac{p^{old}(l) + [\![l \in \mathbb{Y}_{\mathbf{x}}]\!]}{|C|+1} \tag{6}$$

- ***An immature cluster can become a mature cluster***: When an immature cluster receives a new data point, its weight is increased by 1 (see Eqn. (2)). When the weight of the immature cluster is greater than the pre-defined mature threshold W_0, this cluster will become a mature cluster.
- ***A new immature cluster can appear***: When a new data point cannot be merged into any cluster as it makes the radius of the cluster exceed the boundary threshold θ, it becomes the first data point of a new immature cluster. In this case, the new cluster only has one data point which is also the center of this cluster. The radius of this cluster is 0 and the weight is 1.
- ***A mature cluster can become an immature cluster***: Over time, if a mature cluster does not get any new samples, its weight will gradually decrease as a consequence of the weight decay of all data points inside the cluster. The mature cluster will become an immature cluster when its weight is smaller than or equal to the threshold W_0. In this case, we need to check the weight of all mature clusters to detect the change. Assume that after the time span T_s, the weight of the mature cluster is $2^{-\lambda T_s}W$ $(W > W_0)$ (see Eqn. 2), and a mature cluster become an immature cluster, we obtain the inequality $2^{-\lambda T_s}W_0 < 2^{-\lambda T_s}W \leq W_0$. Thus, the minimal time span for a mature cluster to become an immature cluster is $T_s = \left\lceil \frac{1}{\lambda}\left(\log\left(\frac{W_0}{W_0-1}\right)\right)\right\rceil$ which is computed from the equation $2^{-\lambda T_s}W_0 + 1 = W_0$. Hence, we only periodically check the mature clusters for every time period T_s [7].

Algorithm 1 summarized the update process to build the incremental online clustering model for the MLC problem. For each arrived sample, we try to merge it into the nearest mature cluster. If the new radius is still smaller than the boundary of a cluster, the sample is successfully merged into this mature cluster (step

2-3). The three components, as well as the cluster center and radius, are updated using the features and weight of the new member (step 4). In contrast, if the new sample makes the mature cluster exceed the boundary, we try to merge this sample to the nearest immature cluster (step 7-9). In this case, we check the weight of the immature cluster that the sample merged into. If the new weight is larger than the pre-defined mature threshold, the immature cluster will become a mature cluster (step 10). If the sample cannot be merged into any cluster as it makes the cluster's radius exceeds the boundary threshold, we build a new immature cluster for this sample (step 14-15). To complete the update process, we incrementally update the label frequency component of the cluster that the sample is merged into (step 18-20). Finally, we periodically check all clusters in the mature list. If there exist any clusters with weights smaller than or equal to the mature threshold, these clusters will become immature clusters (step 23-25).

Algorithm 1: Incremental clustering model for MLC

```
Input:    Arrived sample $\mathbf{x}$ at the current time $t$
Output    The updated incremental learning model
1         Try to merge $\mathbf{x}$ into the nearest $m\_C$
2         If (the new $r$ (of $m\_C$) $\leq \theta$)
3             Merge x into $m\_C$
4             Update $r$, $\mathbf{c}$, and $W_{m\_C}$ of $m\_C$
5         Else
6             Try to merge $\mathbf{x}$ into the nearest $im\_C$
7             If (the new $r$ (of $im\_C$) $\leq \theta$)
8                 Merge x into $im\_C$
9                 Update new $r$, $\mathbf{c}$, and $W_{im\_C}$ of of $im\_C$
10                If ($W_{im\_C} > W_0$)
11                    $im\_C \rightarrow mC$
12                End
13            Else
14                Create a new immature cluster using $\mathbf{x}$
15                Compute $r$, $\mathbf{c}$, and $W$ for the new cluster
16            End
17        End
18        For each label $i$ in $\mathcal{Y}$
19            Update label frequency by (6)
20        End
```

```
21          Set $T_{\mathbf{x}} = \left\lceil \frac{1}{\lambda}\left(\log\left(\frac{W_0}{W_0-1}\right)\right)\right\rceil$
22          If($t$ mod $T_{\mathbf{x}}$)=0
23              For each $m\_C$ in $m\_\mathcal{C}$
24                  If($W_{m\_C} \leq W_0$)
25                      $m\_C \rightarrow im\_C$
26                  End
27              End
28          End
```

3.3. The classification process

Based on the constructed clustering model, we predict the labels for each newly arrived sample. As mentioned above, we trust the mature cluster more than the immature cluster so we only use the information of the mature clusters to assign labels for each newly arrived sample. In this study, we develop a classification method based on the similarity between the sample and the mature clusters. First, we measure the distances between the sample and all mature clusters. The top K mature clusters with the shortest distance are selected as the K-nearest neighbors of that sample. We then use the label frequencies of these K nearest neighbors to compute the posterior probability that the sample belongs to a class label. The top h labels associated with the largest posterior probabilities are assigned to the sample as the prediction result. In detail, for sample **x**, let $\mathcal{K}(\mathbf{x})$ represent the set of its K-nereast neighbors in $m\_\boldsymbol{\mathcal{C}}$. Generally, the similarity between the sample and the mature cluster m_C is measured with the Euclidean distance between the samples and the clusters' center.

$$\mathcal{K}(\mathbf{x}) = \{m\_\mathcal{C}_i, 1 \leq i \leq K | i \in \arg \text{smallest\_k}\{d(\mathbf{x}, \mathbf{c}_i)\}\} \tag{7}$$

Here smallest_k$(\cdot)$ returns the K shortest distances between **x** and the mature clusters in $m\_\boldsymbol{\mathcal{C}}$ and $d(\mathbf{x}, \mathbf{c}_i)$ is the distance between **x** and the cluster center $\mathbf{c}_i$. We calculate the posterior probability that the sample belongs to a label l as the sum of the products of the weight of $m\_\mathcal{C}_i$ and label frequencies of label l among all mature clusters in $\mathcal{K}(\mathbf{x})$:

$$P(l|\mathbf{x}) \sim \frac{\mathbf{1}}{\boldsymbol{k}} \sum_{m\_\mathcal{C}_i \in \mathcal{K}(\mathbf{x})} p_{m\_\mathcal{C}_i}(l) W_{m\_\mathcal{C}_i} \tag{8}$$

Let $\widehat{\mathbb{Y}}_{\mathrm{x}}$ denote the predicted label set for **x**. We obtain $\widehat{\mathbb{Y}}_{\mathrm{x}}$ by getting the labels associated with the h-largest values of these posterior probabilities:

$$\widehat{\mathbb{Y}}_{\mathrm{x}} = \{l \in \mathcal{Y} | P(l|\mathbf{x}) \text{ in top } h\} \tag{9}$$

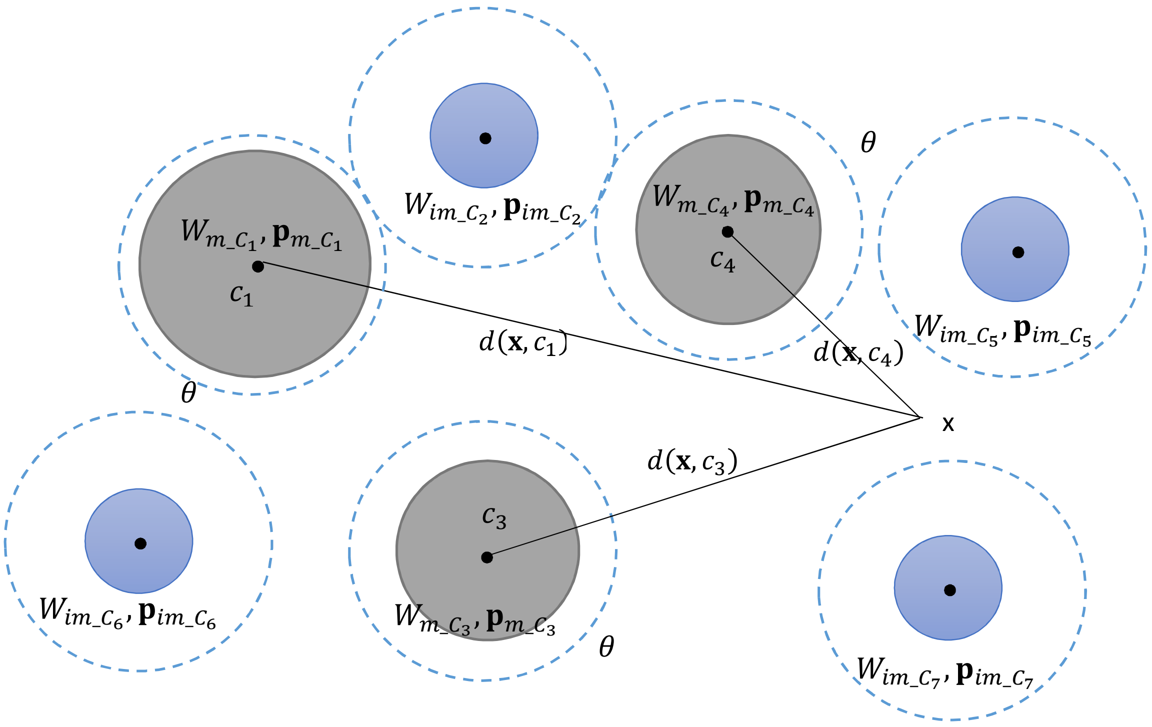

$W_{im\_C_2}, \mathbf{p}_{im\_C_2}$
$W_{m\_C_1}, \mathbf{p}_{m\_C_1}$
c_1
$W_{m\_C_4}, \mathbf{p}_{m\_C_4}$
c_4
θ
$W_{im\_C_5}, \mathbf{p}_{im\_C_5}$
$d(\mathbf{x}, c_1)$
$d(\mathbf{x}, c_4)$
θ
x
$d(\mathbf{x}, c_3)$
c_3
$W_{im\_C_6}, \mathbf{p}_{im\_C_6}$
$W_{m\_C_3}, \mathbf{p}_{m\_C_3}$
θ
$W_{im\_C_7}, \mathbf{p}_{im\_C_7}$

Fig.2. The classification process based on the clustering model

The proposed MLC algorithm is different to the K-nearest neighbor approach [17] and weighted K-nearest neighbor [36] approach in label assignment, although they all used the K-nearest neighbors for label assignment. In weighted K-nearest neighbor [36], each class label is assigned to the sample via the decision function computed from the posterior probability that the sample belongs to the considered label (positive label) or not (negative label) in the K neighboring samples. In our method, clustering is performed, then each mature cluster is treated as a single point having its mature degree and the distribution of label frequencies. Instead of selecting a label based on the decision function computed from the label's posterior probability of positive or negative label, we select h labels associated with the largest posterior probability, where h is learned from the data sequence. In our method, the number of predicted labels is updated adaptively based on the revealed ground truth labels of the arrived samples. The update procedure for h will be introduced in the next section.

The proposed classification process is summarized in Algorithm 2.

```
Algorithm 2: Predict labels based on weights of clusters
Input:      Sample $\mathbf{x}$, the set of mature cluster $m\_\boldsymbol{C}$, $h$, $K$
Output      Predicted labels for $\mathbf{x}$
1           For each $m\_C$ in $m\_\boldsymbol{C}$
```

```
2            Compute $d(\mathbf{x}, \mathbf{c})$
3        End
4        Select $\mathcal{K}(\mathbf{x})$ by (7)
5        Initialize $P(l|\mathbf{x}) = 0\ \forall l \in \boldsymbol{\mathcal{Y}}$
6        For each $l \in \boldsymbol{\mathcal{Y}}$
7            For each $m\_C$ in $\mathcal{K}(\mathbf{x})$
8                Compute $P(l|\mathbf{x})$ by (8)
9            End
10       End
11       Return $\widehat{\mathbb{Y}}_{\mathrm{x}}$ by (9)
```

3.4. Label set learning

We predict the h labels associated with the top h-posterior probabilities for the arrived sample. Very often, the number of labels to be learned is fixed beforehand. In general, however, the number of predicted labels for each sample should be flexible and depends on the sample itself. In this study, we propose a method to adjust the number of predicted labels by using the Hoeffding inequality [37] and the label cardinality.

The label cardinality of a dataset is the average number of labels per sample in the dataset. It is independent of the number of labels M, and naturally can be used to quantify the number of predicted labels h for each sample. In this study, not only the learning model but also the number of predicted labels are learned with the arrival of data samples. 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 as determined by the Hoeffding inequality.

In probability theory, the 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 [37]. We restate the result of the Hoeffding inequality as the theoretical basis of our approach: If X_i $i = 1, \dots, N$ are independent random variables and if $a_i \le X_i \le b_i$, $S = \sum_{i=1}^{N} X_i$, $\bar{X} = S/N$, $\mu = \mathbb{E}[\bar{X}]$ 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) $\mathrm{P}\{|\bar{X} - \mu| \ge \varepsilon\} \le 2\exp\left\{\frac{-2N^2\varepsilon^2}{\sum_{i=1}^{N}(a_i - b_i)^2}\right\}$

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. Applying the Hoeffding inequality with the note that $0 \leq |Y_i| \leq L$ for all $i = 1, \dots, N$, and L is the number of distinct labels of the data stream, we have:

$$\mathrm{P}\{|\bar{z} - \mathbb{E}[\bar{z}]| \geq \varepsilon\} \leq 2\exp\left\{\frac{-2N^2\varepsilon^2}{NL^2}\right\} = 2\exp\left\{\frac{-2N\varepsilon^2}{L^2}\right\} \tag{10}$$

in which the label cardinality as the average number of labels of the stream is given by:

$$\bar{z} = \frac{1}{N}\sum_{i=1}^{N}|Y_i| \tag{11}$$

Denoting the right hand of Eqn. (10) by δ, ε is computed as:

$$\varepsilon = \sqrt{\frac{L^2 \ln(2/\delta)}{2N}} \tag{12}$$

Eqn. (10) becomes:

$$\mathrm{P}\{|\bar{z} - \mathbb{E}[\bar{z}]| \leq \varepsilon\} \geq 1 - \delta \tag{13}$$

If $|h - \bar{z}| > \varepsilon$, based on the inequality $|h - \mathbb{E}[\bar{z}]| = |(h - \bar{z}) - (\mathbb{E}[\bar{z}] - \bar{z})| \geq |h - \bar{z}| - |\mathbb{E}[\bar{z}] - \bar{z}|$ and combine with Eqn. (12), we have:

$$\mathrm{P}\{|h - \mathbb{E}[\bar{z}]| > 0\} \geq 1 - \delta \tag{14}$$

That means h is different from $\mathbb{E}[\bar{z}]$ with a probability of at least $1 - \delta$ if $|h - \bar{z}| > \varepsilon$. 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 $\bar{z}$. In this case, we update h by $h = round(\bar{z})$.

Fig 3 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 the sequence in which 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:

$$\bar{z}_{N-1} = \frac{1}{N-1}\sum_{i=1}^{N-1}|Y_i| \tag{15}$$

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

$$\bar{z} = \frac{1}{N}\sum_{i=1}^{N}|Y_i| = \frac{1}{N}\left(\sum_{i=1}^{N-1}|Y_i| + |Y_N|\right) = \frac{1}{N}\left((N-1)\bar{z}_{N-1} + |Y_N|\right) \tag{16}$$

For each arrived sample, we compute the current label cardinality $\bar{z}$ based on the size of the true label $|Y_N|$ and the previous label cardinality given by (15). We then check whether the difference between $\bar{z}$ and h is greater than the threshold ε, and the value of h used for the next sample will be re-calculated by the rounded value of current $\bar{z}$. If the update condition is met, we reset $\bar{z} = 0$, $L = 0$, and $N = 0$ to begin a period with the new value of h.

In the literature, we found only three incremental MLC thresholding methods to determine the size of the predicted label set. 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 predicted labels, and is then re-calculated via sample adjustment [38] or batch adjustment [5, 9]. In [38], Read et al. performed small adjustment on the threshold on a per-sample basis depending on the predicted and the actual label cardinality. Xioufis et al. [9] by contrast incrementally adjusted 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. In [5], Read et al. incrementally adjusted a threshold for all labels on each batch to ensure that the predicted label cardinality approximates the true label cardinality. Compared to the above methods, our approach is significantly different and is more flexible since the number of samples used to adjust h is not fixed but is based on using a condition derived from the Hoeffding bound.

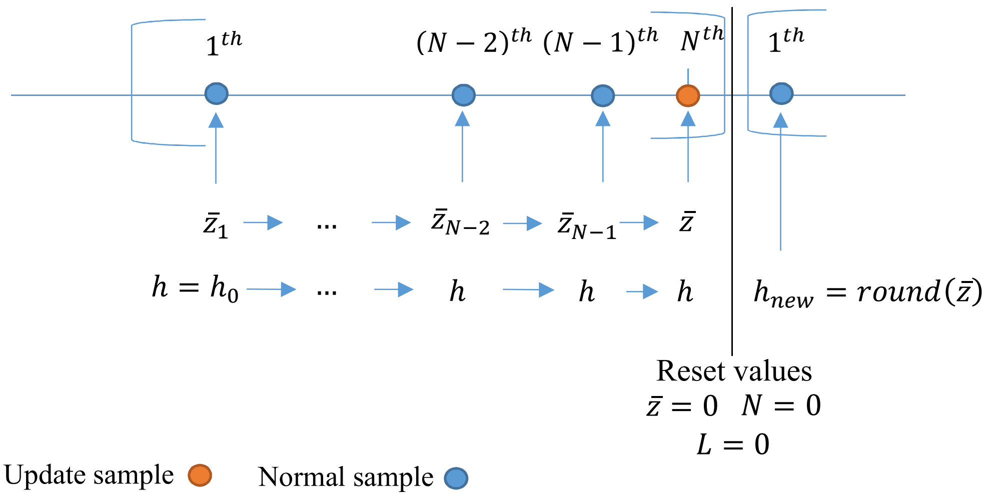


Fig.3. Illustration of adjusting the number of predicted labels

As mentioned before, we developed the incremental MLC method via the online learning paradigm. By this way, we performed the following three steps on each arrived sample **x**:

- *Predict labels*: The current learning model is used to predict the label set $\hat{\mathbb{Y}}_x$ for **x**.
- *Compute the update condition*: We update the learning model if the set of true labels $\mathbb{Y}_x$ of **x** can be revealed for the environment.
- *Update learning model*: If the update condition is satisfied, the new classification model is obtained by updating from the previous one based on sample **x** and $\mathbb{Y}_x$.

The incremental learning algorithm based on the clustering model for OMLC is summarized as:

Algorithm 3: Incremental online multi-label classification based on a clustering approach

```
Input:     Data sequence $\mathcal{D}$, $K, \varepsilon, W_0, \lambda$
Output     Predict labels for each arrived samples
1          For each arrived sample $\mathbf{x}$ from $\mathcal{D}$
2              $h = h_0$;
3              Obtain $\widehat{\mathbb{Y}}_{\mathbf{x}}$ by Algorithm 2
4              If ($\mathbb{Y}_x$ can be revealed from the environment)
5                  Update clustering structure by Algorithm 1
6              End
7              Compute $\bar{z}$ by (16)
8              Compute $\varepsilon$ by (12)
9              If ($|\bar{z} - h| > \varepsilon$)
10                 Update $h = round(\bar{z})$
11                 $\bar{z} = 0, N = 0, L = 0$
12             End
13         End
```

4. Experimental Studies

4.1. Datasets

To evaluate the performance of the benchmark algorithms and the proposed method, we selected 5 popular multi-label datasets [5, 6] for the stationary setting and generated 12 datasets for the concept drift setting, respectively. All these datasets were treated as sequential datasets by processing them in the order they were collected.

[1] We generated a synthetic dataset named *SynRTG* using the Random Tree Generator (RTG) in MOA library (http://moa.cms.waikato.ac.nz). The RTG constructs a decision tree by randomly choosing attributes to split and assigning a random class label to each leaf. Once the tree is built, new data points are generated by assigning uniformly distributed random values to attributes which are then used to determine the class label via the tree. We generated a 5-level tree to create *SynRTG*. To generate the concept drift for *SynRTG* (named *SynRTG-drift*), all generation schemes used in our work are initialized as binary generators with parameters as in Read et al. [5]. For a dataset with N generated samples, the drifts are centred at the $(N/4)^{th}$, $(N/2)^{th}$, and $(3N/4)^{th}$ sample, extending over N/1000, N/100, and N/10 samples, respectively. In the first drift, only 10% of label dependencies are changed. In the second drift, the underlying concepts

are changed and more labels are associated on average with each sample (a higher label cardinality). In the third drift, 20% of label dependencies are changed. We also generated the '*gradual*' drift version for the five multi-label datasets (named *ENRON-drift, IMDB-drift, OHSUMED-drift, SLASHDOT-drift, and TMC2007-drift*) by concatenating half of the samples from the original data and the other half generated by RTG. The gradual concept drift is formed by randomly choosing S samples around the joint from the original data and the synthetic data to transition the concept drift. We also generated the '*break*' drift for the five multi-label datasets (named *ENRON-break, IMDB-break, OHSUMED-break, SLASHDOT-break, TMC2007-break)* using the similar scheme mentioned above except with $S = 0$ (Fig 4). The details of the experimental datasets are described in Table 2.

TABLE 2. THE EXPERIMENTAL DATASETS

Dataset	**# of samples**	**# of features**	**# of labels**	**Label cardinality**	***S***
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	-
ENRON-drift	3000	1001 binary	53	3.4 -> 5.0	150
IMDB-drift	200000	1001 binary	28	2.0 -> 4.0	5000
OHSUMED-drift	26000	1002 binary	23	1.7 -> 4.0	1300
SLASHDOT-drift	7000	1079 binary	22	1.2 -> 3.0	350
TMC2007-drift	56000	500 binary	22	2.2 -> 5.0	2800
SynRTG-drift	1000000	30 binary	8	1.8 -> 3.0	-
ENRON-break	3000	1001 binary	53	3.4 -> 5.0	0
IMDB-break	200000	1001 binary	28	2.0 -> 4.0	0
OHSUMED-break	26000	1002 binary	23	1.7 -> 4.0	0
SLASHDOT-break	7000	1079 binary	22	1.2 -> 3.0	0
TMC2007-break	56000	500 binary	22	2.2 -> 5.0	0
SynRTG-break	1000000	30 binary	8	1.8 -> 3.0	-

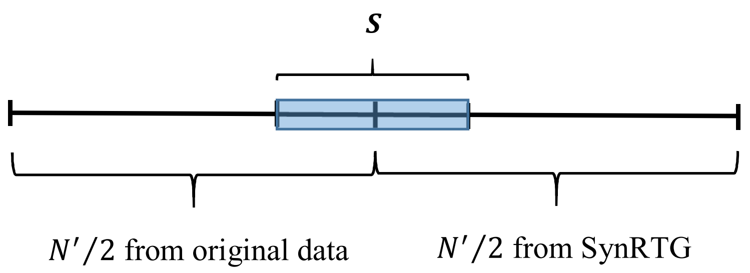


Fig.4. Structure of datasets with concept drift

4.2. Experimental Setup

For each arrived sample, we first predict the labels for each arrived sample and then use this sample with its true labels to update the learning model. For the benchmark algorithms, we use the parameters set by the MOA library. For the proposed method, there are four parameters, i.e., K, θ, W_0, and λ, that need to be considered. In the next section, we evaluate the influence of K and λ on each of the performance measures. Meanwhile, the mature weight W_0 was set to 3 in the experiment as in [7].

We set the cluster bound θ for binary data since all datasets used in the experiment have only binary features. Assume that on the feature x_j, s samples have value 1 and $n - s$ samples have value 0. The standard variation computed on n samples on this feature is given by:

$$\sigma(x_j) = \sqrt{\mathbb{E}(x_j{}^2) - \mathbb{E}(x_j)^2} = \sqrt{\frac{s}{n} - \left(\frac{s}{n}\right)^2} = \sqrt{\frac{1}{4} - \left(\frac{1}{2} - \frac{s}{n}\right)^2} \le \frac{1}{2} \tag{17}$$

Since the inequality (17) holds on all features, we have the upper bound of the radius of the cluster given by $r = \max_{j=1,\dots,d}\{\sigma(x_j)\} \le 1/2$. If the cluster bound θ is set with a value greater than or equal to 0.5, step 2 in Algorithm 1 will always be satisfied and all samples will be merged into a single mature cluster. Therefore, θ was set to 0.495.

4.3. Performance Measures and Benchmark Algorithms

In multi-label learning, each sample is associated with a set of labels. Zhang and Zhou [4] stated that the performance of MLC algorithms should be tested on a range of measures instead of only the one being optimized to ensure a fair evaluation. In our experiments, we compute six performance measures based on the predicted label $\hat{\mathbb{Y}}_{\mathbf{x}}$ and the ground truth $\mathbb{Y}_{\mathbf{x}}$ on each arrived sample $\mathbf{x}$ from the data sequence $\mathcal{D}$. 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). The details of the performance measures can be found in the Appendix. The running time including the time for prediction and update are also reported for all methods.

Since our method is applicable for both the stationary and concept drift settings, we compared the proposed method with both well-known non-adaptive and adaptive incremental MLC algorithms. For the stationary setting, the proposed method was compared with the following well-known incremental MLC algorithms: incremental Classifier Chain (denoted by iCC), incremental Pruned Set (denoted by iPS), and Majority Label Set (denoted by MLS). The base classifier for these methods is the Hoeffding tree. All these benchmark algorithms were run with the default parameters as given in the MEKA library (http://waikato.github.io/meka). We also implemented incremental Binary Relevance with SVM as base classifier using the scikit-learn library (https://scikit-learn.org) (denoted by iBR(SVM)). The proposed method was also compared with four recent incremental multi-target tree based MLC algorithms introduced in [6] including the 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 set as recommended in [6]. The algorithms iCC, iPS, MLS, iBR(SVM) were combined with the state-of-the-art adaptive method named ADWIN2 [31] to adapt to concept drift in the data. However, the four incremental multi-target-tree based MLC algorithms [6] are only introduced for the stationary concept. Therefore, we omitted them from the comparison under the concept drift setting.

4.4. Statistical Test

To assess the statistical significance of the experimental results of incremental MLC methods, Read et al. [5] and Osojnik et al. [6] used the Friedman test [39] to test the difference between the performances of multiple methods on multiple datasets. The Friedman test is preferred over the ANOVA test for the following reasons. First, the Friedman test does not assume normal distribution as in the ANOVA test. Second, 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 [40]. Here the Friedman test is used to test the null hypothesis that "all methods perform equally". If the null hypothesis is rejected, a post-hoc test is then conducted. Read et al. [5], and Osojnik et al. [6] used the Nemenyi test for all pairwise comparisons based on the rankings of algorithms on all datasets. The difference in performance of two methods is treated as statistically significant if the $p-value$ computed from the post-hoc test statistic is smaller than an adjust value of confident level computed from Nemenyi's procedure. The confident level of the test was set to 0.05.

5. Experimental Results

5.1. The influence of parameters

In this section, we examine the influence of two parameters, i.e., the number of nearest neighbors K and the decay rate λ on the 6 performance measures. The value of λ was set in the range of $\{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$ to examine its influence on the 6 performance measures. Fig. 5 and Fig. 6 present the relationship between λ and the 6 measures on the 5 datasets at $K = 3$ and $K = 10$, respectively. The up or down arrow beside each measure in Fig. 5 and 6 indicates whether a higher or lower value is better, respectively. It can be seen that the performance measures at $K = 3$ and $K = 10$ with the different values of λ are nearly similar. Therefore, we only analyze the influence of λ on the 6 measures at $K = 3$.

We can observe that the 4 measures, i.e., sample-based F1, sample-based accuracy, micro-average F1, and average precision, remain nearly constant for different values of λ for the datasets OHSUMED, IMDB, and SLASHDOT. However, for the TMC2007 dataset, the value of these 4 measures fluctuates and decreases to a minimum at $\lambda = 0.4$ before slightly increases. For the ENRON dataset, these 4 measures show a fluctuating but generally still increasing trend with the increase of λ. For the TMC2007 dataset, the macro-average F1 decreases with the increase of λ like the 4 measures mentioned above but the decrease here is more significant (decreases from 0.3 to 0.1). The ranking loss for TMC2007 increases slightly when λ is between 0.05 and 0.5. The ranking loss for the SLASHDOT and IMDB datasets also remains nearly constant for different values of λ. For the OHSUMED dataset, the ranking loss decreases to a minimum at $\lambda = 0.15$ and then fluctuates. For the ENRON dataset, the ranking loss becomes better in general with the increase of λ, reaching the best value at $\lambda = 0.5$.

Recall that the decay speed λ determines how fast the model forgets the old samples as reflected by the reduction of the sample's weight. Hence, this parameter significantly affects the clustering model. A large value of λ would cause a significant reduction of the weights of old samples, reducing the number of mature clusters that would appear. Based on the observations from Fig 5 and 6, there does not exist a common value of λ in which the proposed method would perform well on all the experimental datasets for all 6 measures. Therefore, we set $\lambda = 0.25$ when comparing the performance of the proposed method to the benchmark algorithms.

Fig. 7 presents the relationship between K and the six measures on the 5 non-drift datasets with $\lambda = 0.25$. In this study, the value of K was set to be in the range of $\{3, 5, 7, 10\}$. Generally, there is not a common trend of the performance values with the different values of K among the 5 datasets.

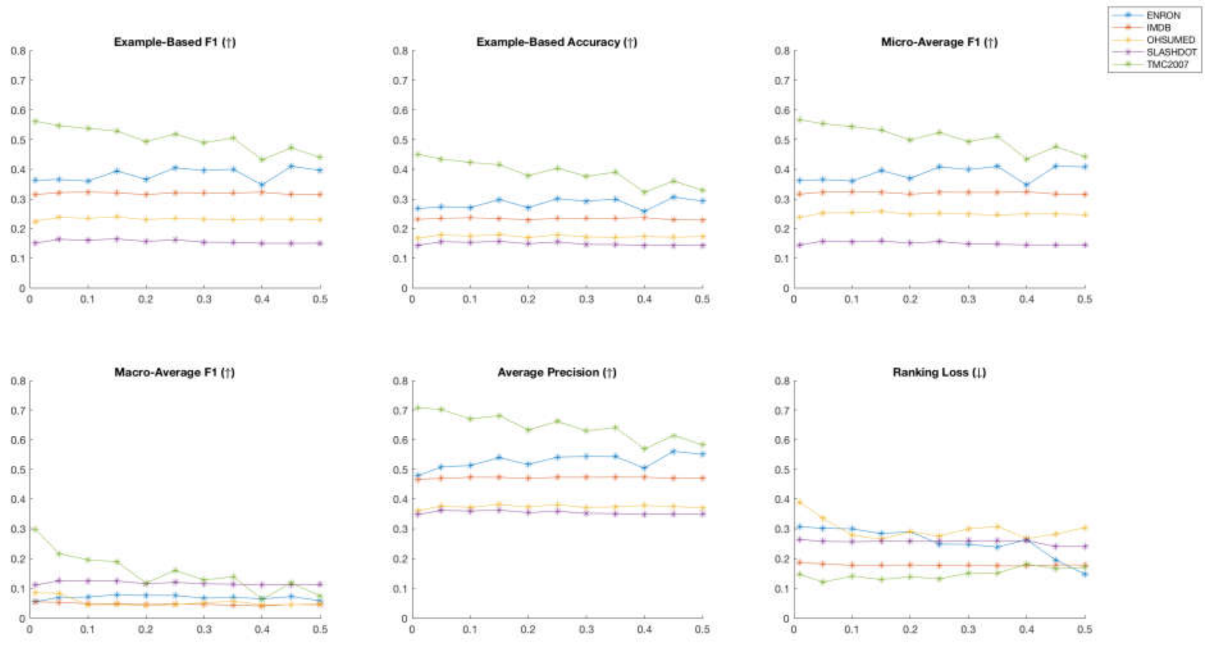


Fig.5. Performance measures (y-axis) for different values of λ (x-axis) at $K = 3$

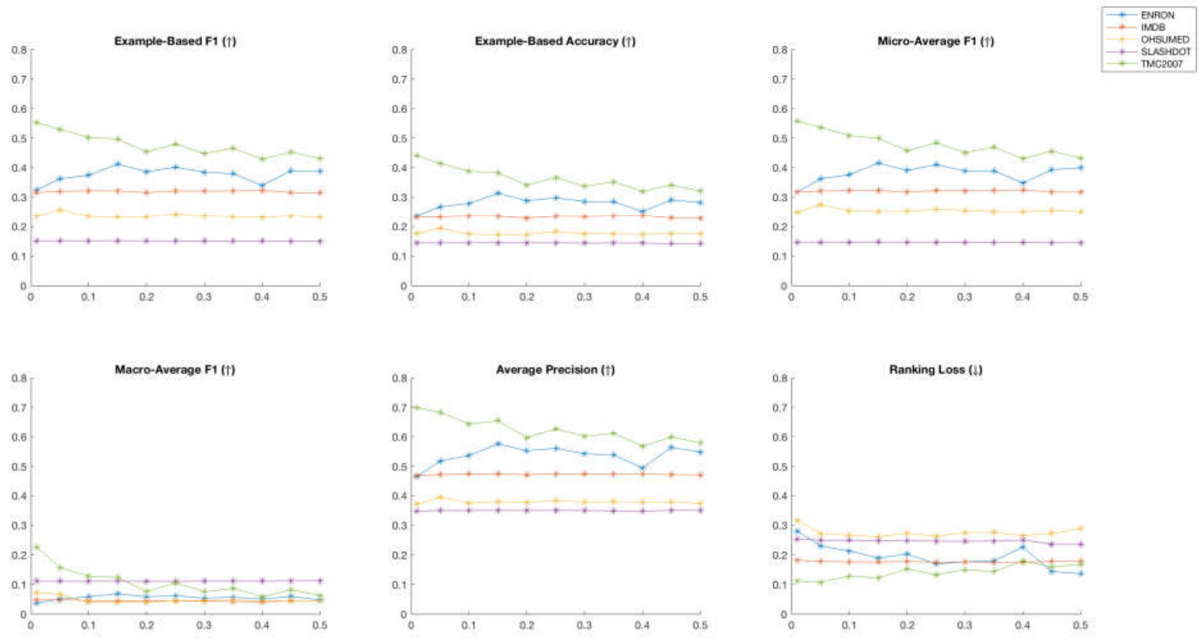


Fig.6. Performance measures (y-axis) for different values of λ (x-axis) at $K = 10$

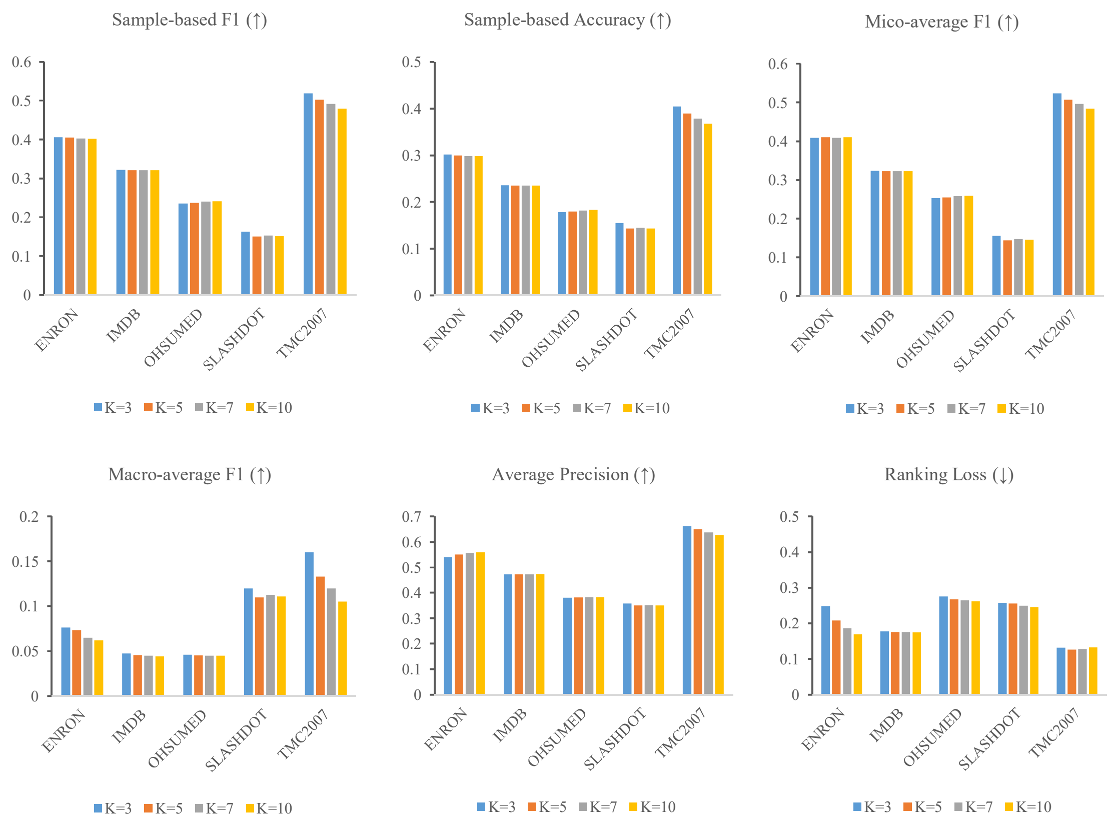
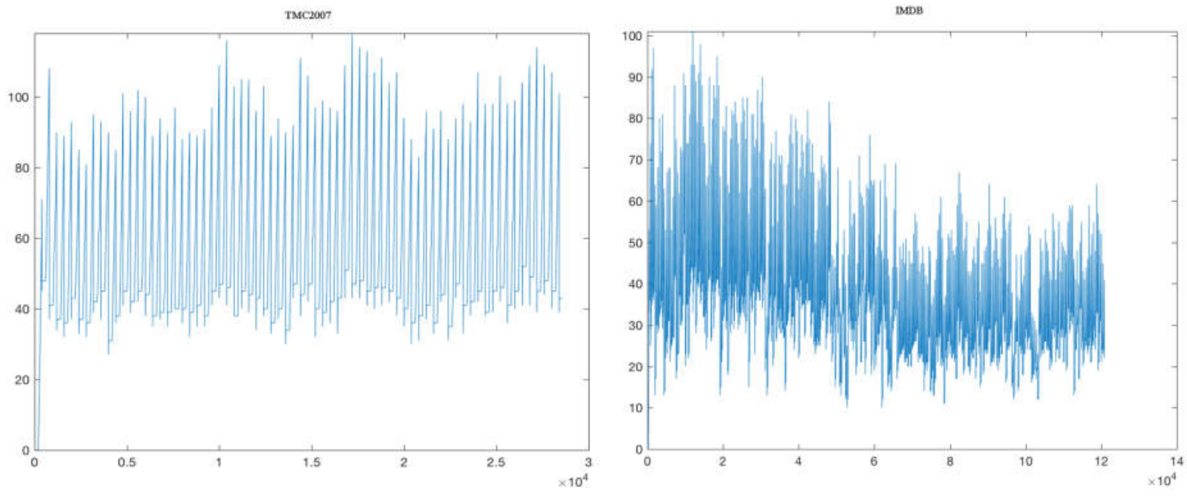


Fig.7. Performance measures for different values of *K* nearest neighbors



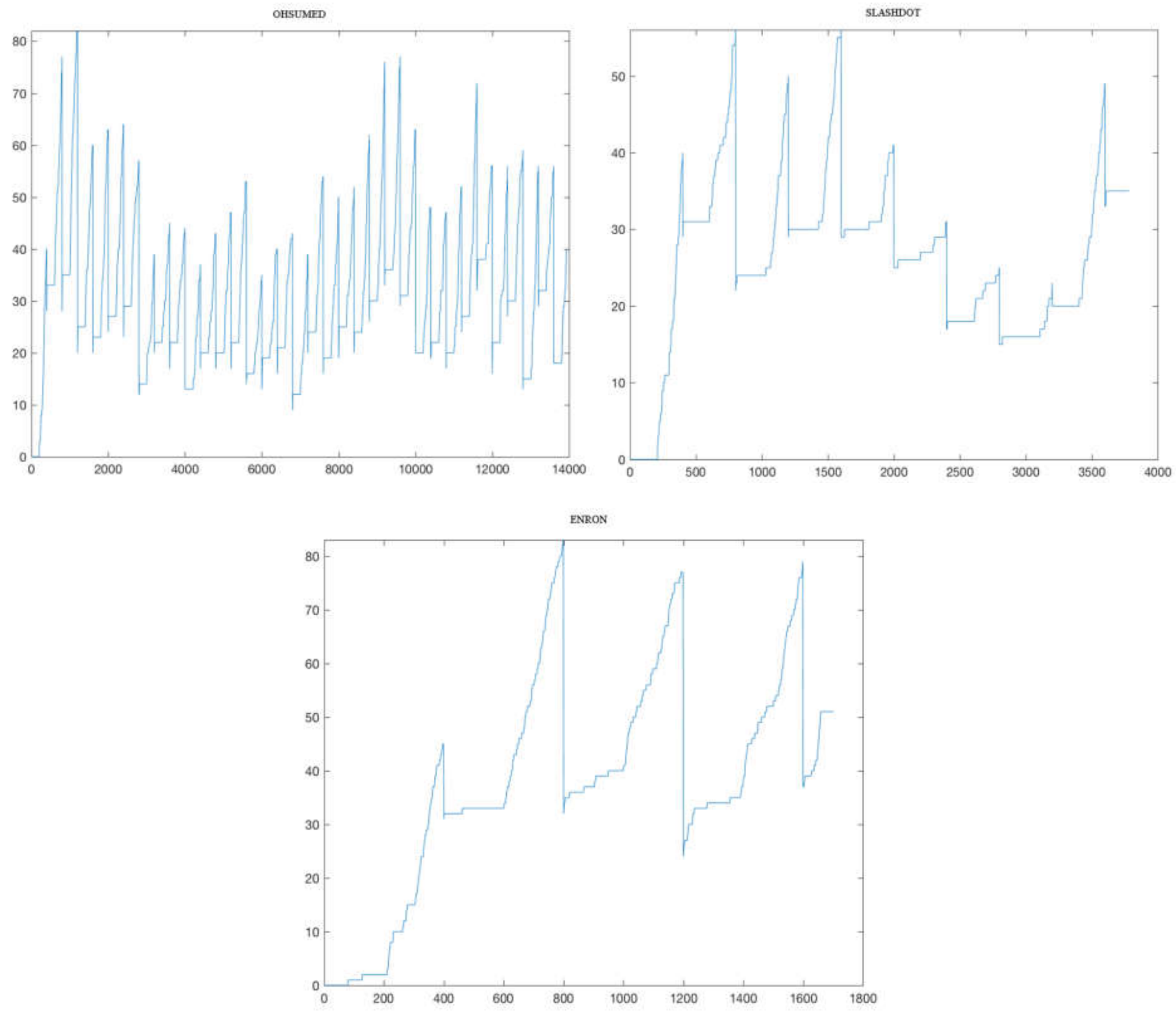


Fig.8. Number of mature clusters (y-axis) generated on the sequence of samples (x-axis) under the stationary setting

In detail, on the ENRON dataset, the different values of K almost do not affect the first three measures, and Macro-average F1 obtains the best value at $K = 3$ whereas average precision and ranking loss obtain the best value at $K = 10$. On the IMDB dataset, all six measures almost remain unchanged with different values of K. On the OHSUMED dataset, K has little influence on 5 measures except for ranking loss in which it obtains the best value at $K = 10$. On the SLASHDOT dataset, the proposed method obtains the best value at $K = 3$ for the first 5 performance measures. In contrast, it preforms the poorest on ranking loss at $K = 3$. Finally, on the TMC2007 dataset, on the first 5 measures, the performance of the proposed method reduces with the increase of K. Meanwhile, it performs the best at $K = 5$ for the ranking loss. In the next section, we will compare the performance of the proposed method using $K = 3$.

Fig 8 shows the number of mature clusters generated versus the sequence of arriving samples. It can be seen that the number of mature clusters fluctuates significantly because of the appearance of new mature clusters as well as the change of mature clusters to immature clusters.

5.2. Comparison with weighted KNN

In this section, we compare the effectiveness of the proposed method to the weighted KNN method [36], where we learn the number of labels h for a sample from the data stream and [36] uses a decision function to decide whether a label should be included in the label set of a sample. The experimental results on 5 non-drift datasets are shown in Fig 9. Comparison is only done on the non-drift datasets as the weighted KNN method is a batch learning method and is not designed to handle concept drift in data. Clearly, our method is better than weighted KNN on all datasets for the 4 performance measures: sample-based F1, sample-based accuracy, macro-average F1 and mico-average F1. In particular, our approach is significantly better on 3 datasets IMDB, OHSUMED, and SLASHDOT. For instance, on the IMDB dataset, the sample-based F1 of our method is 0.3220 while that of weighted KNN is only 0.0055. For the average-prediction and ranking loss, our method is only very slightly worse than the weighted KNN method. The result is particularly noteworthy since, in general, batch learning algorithms have better performance than online learning algorithms due to having information available all at once.

The success of the proposed classification method compared to the weighted KNN method originates from the effectiveness of our approach in predicting the number of labels h for each sample. The number of predicted labels h is learned from the true labels of the arrived samples. Here we assigned the top h labels associated with the top h values of the posterior probability to the sample. In our approach, the value of h is adaptive since the number of samples used to adjust h is not fixed, i.e., h is adjusted if the adjust condition based on Hoeffding inequality is satisfied.

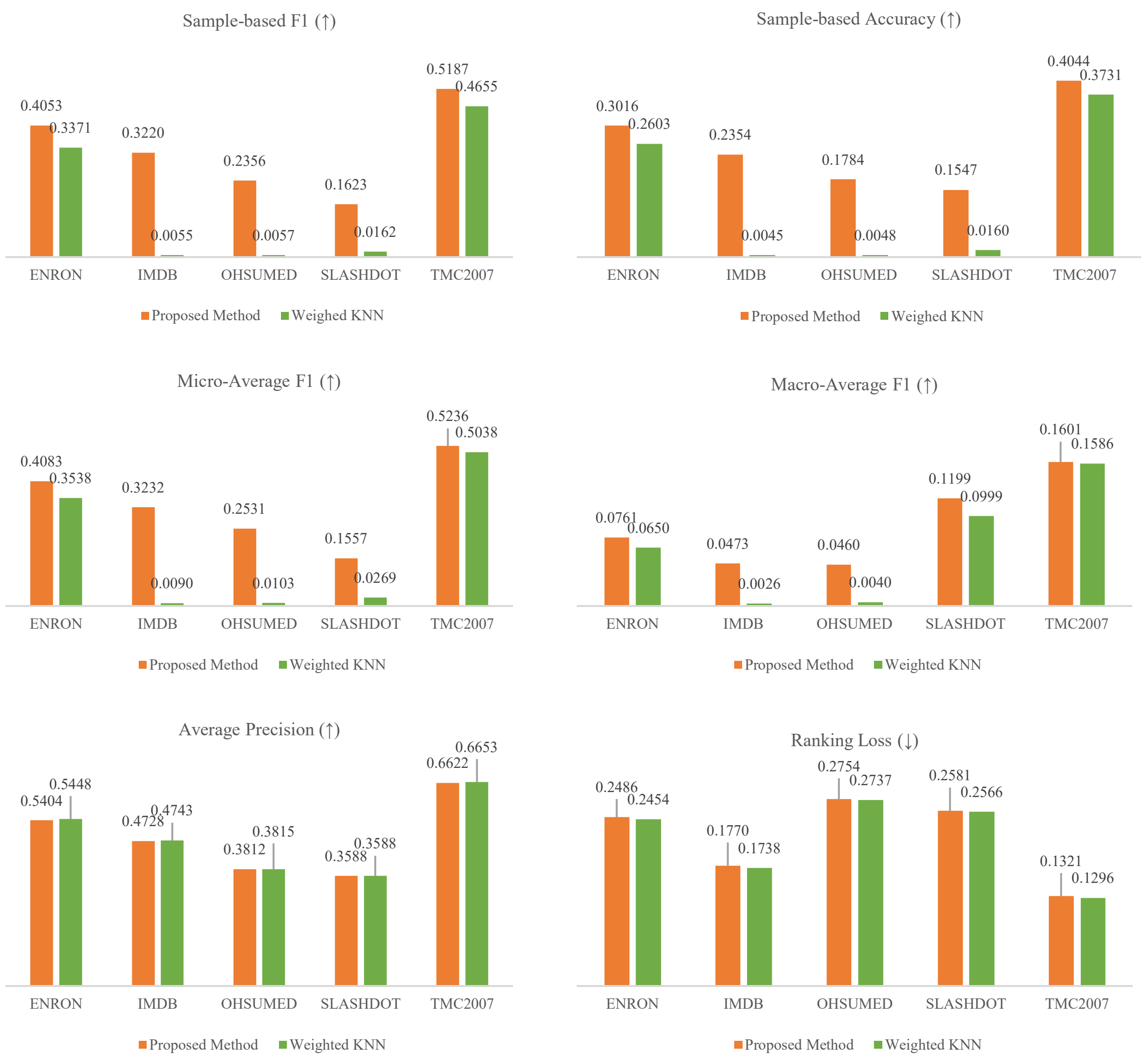


Fig.9. Performance of the proposed classification method and weighted KNN

5.3. Comparative study under the stationarity setting

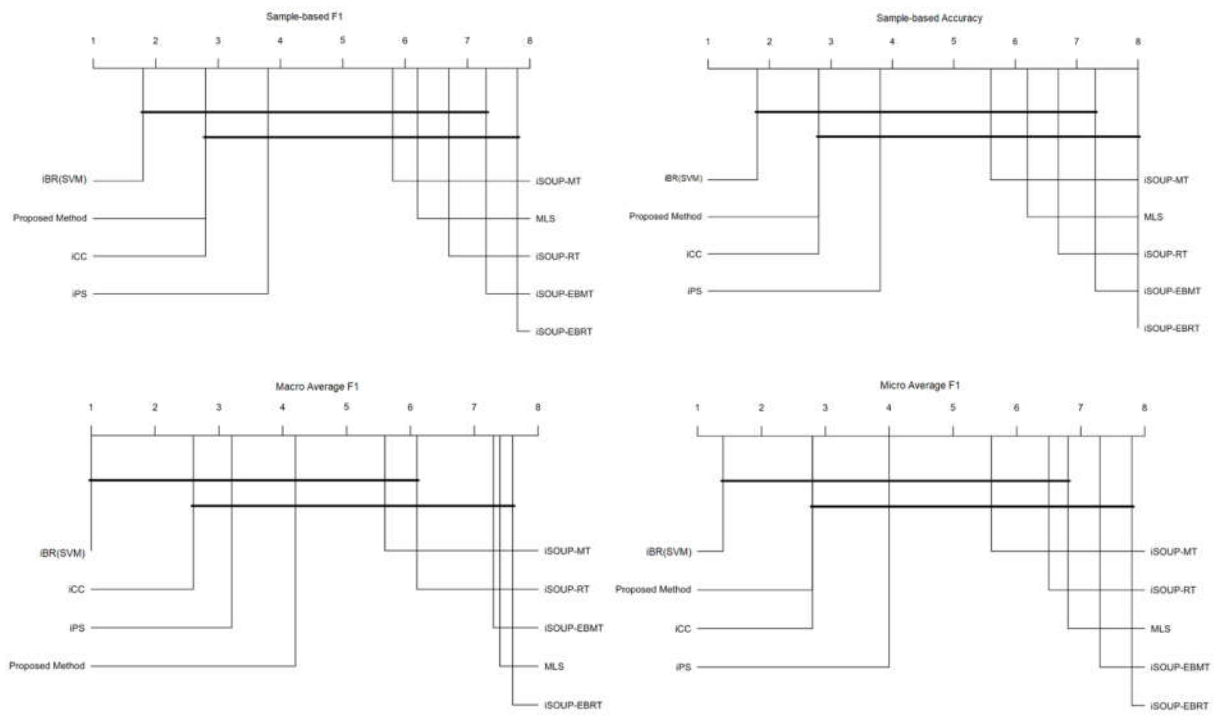
5.3.1. *Results on sample-based measures*

Table S3 in the supplement material shows the sample-based F1 and sample-based accuracy of the benchmark algorithms and the proposed method. The P-values computed based on the rankings with the Friedman test are 9.3060E-4 and 7.3760E-4 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 10, there is a statistical difference in the pairwise comparison between iBR(SVM) and iSOUP-EBRT.

In detail, iBR(SVM) ranks first for both measures (1.8 for both F1 and accuracy), followed by the proposed method and iCC (2.8 for both F1 and accuracy). On dataset IMDB, the proposed method obtains the best results for both F1 and accuracy. iBR(SVM) ranks first on three datasets SLASHDOT, EURON, and OHSUMED. Meanwhile, the four multi-target tree-based methods and MLS obtain poor results on the experimental datasets, ranking at the bottom positions. Especially on two datasets SLASHDOT and IMDB, the four multi-target tree-based methods are significantly poorer than the other methods.

5.3.2. *Results on label-based measures*

For the two label-based measures, i.e., micro-average F1 and macro-average F1, the P-values for the label-based measures computed by the Friedman test are 4.0171E-4 and 2.1023E-4, respectively. Hence, we rejected the null hypotheses and conducted the post-hoc test for all pairwise comparisons among all the methods. On the micro F1 measure, iBR(SVM) ranks first (rank value 1.4), our method and iCC rank second (rank value 2.8), followed by iPS (rank value 4) and iSOUP-MT (rank value 5.6) (see Table S4 in the supplement material). Although on the macro F1 measure, the proposed method is worse than iCC and iPS (rank value 4.2 compared to 2.6 of iCC and 3.2 of iPS), our method continues to obtain better results on ENRON and IMDB. The four multi-target tree-based methods continue to perform poorly on IMDB and SLASHDOT.



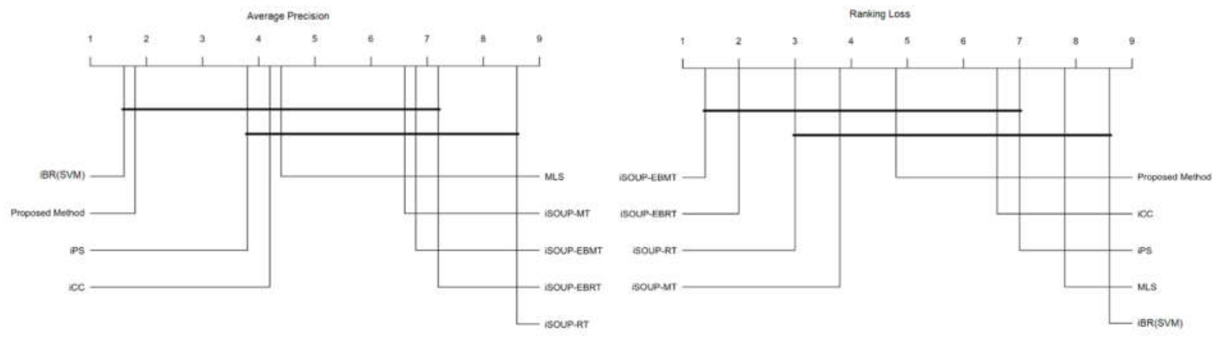


Fig.10. Nemenyi test for the six measures under the stationary setting

5.3.3. *Results on ranking-based measures*

The performance of the benchmark algorithms and the proposed method for the average precision and ranking loss are shown in Table S5 in the supplement material. We again conducted the Nemenyi post-hoc test and reported the results in Fig 10 for the pairwise comparison. The iSOUP-RT method performs the worst among all methods for the average precision measure, and Nemenyi test shows that it is worse than the proposed method and iBR(SVM) for the average precision measure. For the average precision measure, the proposed method is ranked the second (rank value 1.8) after iBR(SVM) (rank value 1.6), while iPS and iCC are ranked the third and fourth, respectively. The four multi-target tree-based methods continue to perform poorly for the average precision measure, ranking at the four bottom positions.

Surprisingly, the four multi-target tree-based methods perform well on all datasets for the ranking loss measure. The ranking loss of the proposed method is in the middle while iCC, iPS, and MLS obtain poor results (rank values are 6.6, 7, and 7.4 respectively). iBR(SVM) is the poorest among all methods (rank value 8.6). Based on the Nemenyi test, MLS and iBR(SVM) are worse than iSOUP-EBMT and iSOUP-EBRT.

5.4. Comparative study under the concept drift setting

Table S6, S7, and S8 in the supplement material show the experimental results of all methods for the six measures under the concept drift setting. Based on the Nemenyi test, the proposed method is better than MLS on 5 measures except for the sample-based accuracy (Fig 11). The proposed method is also better than iCC on the average precision and better than iBR(SVM) on the ranking loss, while there is no statistical difference between ours and iPS.

In detail, the proposed method ranks first on 2 measures, i.e., average precision and ranking loss, and ranks second on two label-based measures. iPS is a competitive MLC method in handling concept drift which ranks first on 2 measures, i.e., two sample-based measures and ranks second on 3 measures, i.e., micro-average F1, average precision and ranking loss. Although iBR(SVM) ranks first on two label-based measure, it performs the poorest for the ranking loss. MLS meanwhile is the weakest among all methods on 5 measures.

Unsurprisingly, the benchmark algorithms iPS and iCC obtain better performance on SynRTG-drift and SynRTG-break than the proposed method for the two sample-based measures, i.e., micro-average F1, and average precision. The SynRTG-drift and SynRTG-break are generated based on the decision tree generator, and in the experiment, we used the Hoeffding tree (an incremental decision tree) as the base classifier of these two methods. The performance difference between our algorithm and iCC and iPS for these two synthetic datasets, however, are not statistically significant for the average precision and ranking loss whereas the proposed method outperforms all benchmark algorithms for the macro-average F1.

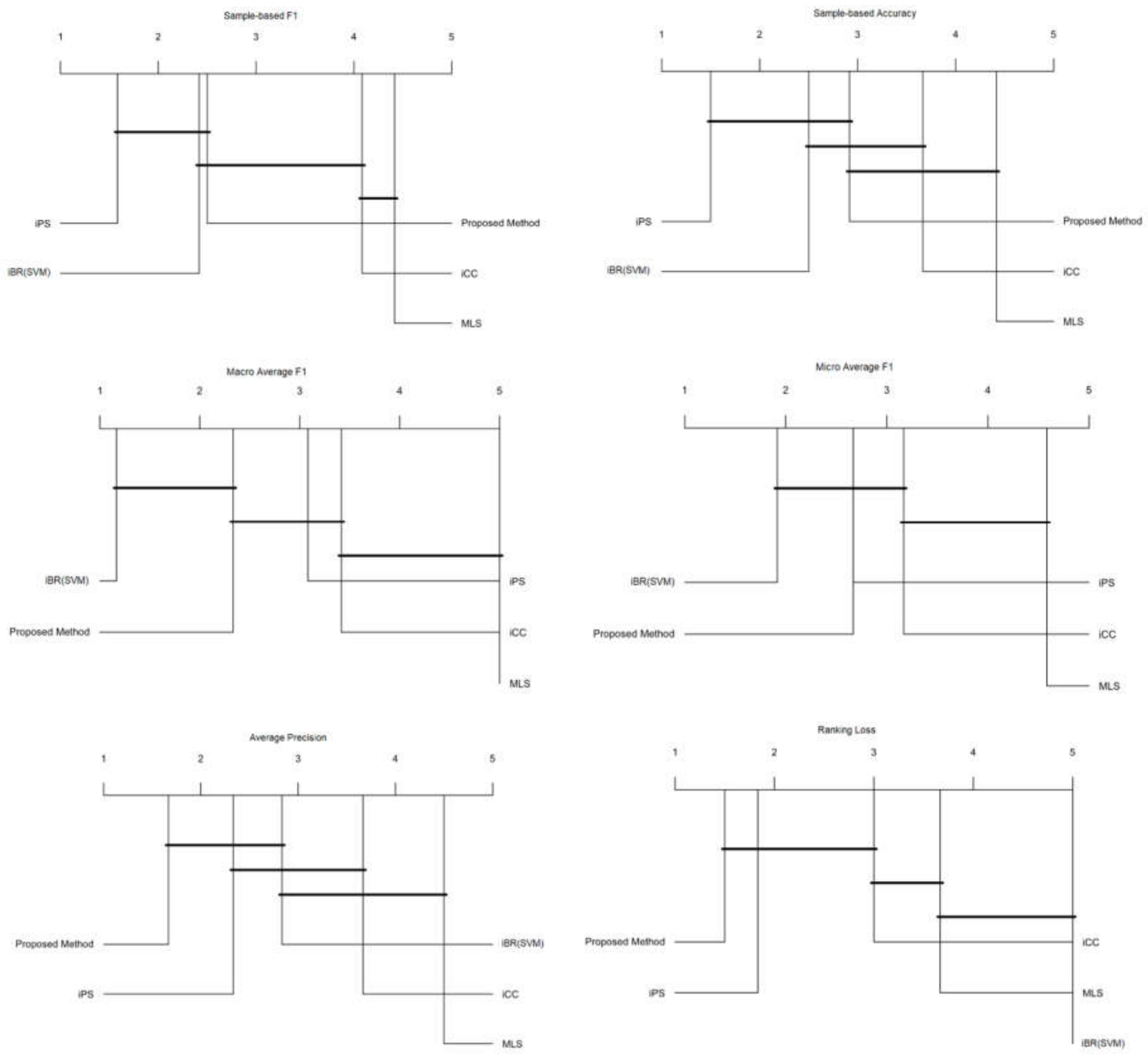


Fig.11. Nemenyi test for the six measures under the concept drift setting

5.5. Discussion

The four multi-target tree-based methods and MLS perform poorly on the experimental datasets. While MLS ranks at the bottom on 5 measures except for the average precision, the four multi-target tree-based methods only perform well for the ranking loss measure. In fact, multi-target tree-based methods are highly threshold-dependent. In these methods, 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. MLS method, meanwhile, is the simplest multi-label classifier which assigns the most common label set from the training data for all test samples. It, therefore, is an uncompetitive incremental classifier for most of the performance measures.

iCC and iPS, especially iPS, are competitive MLC methods for all measures except the ranking loss. Because of taking into account the label correlation in the learning model, these methods can choose suitable labels to assign to each arrived sample, resulting in good MLC performance. In fact, iCC and iPS compare the prediction scores to a threshold to select the predicted labels. In this paper, we followed (Read et al., 2012) by adjusting the threshold via a batch-based approach on the predicted and true label cardinality. The dependence on choosing the appropriate threshold and the less flexible adjustment for the number of predicted labels per sample could have caused the under-performance of these methods on some datasets. iBR(SVM) meanwhile is a very high-performing MLC methods in both settings because of the fact that SVM is a state-of-the-art supervised learning algorithm. This method ranks first on 5 measures except for the ranking loss in the non-concept drift setting and ranks first on the 2 label-based measures and in the concept drift setting. However, it under-performs compared to the proposed method for the ranking loss.

The proposed method is competitive to the other benchmark algorithms like iBR(SVM) and iPS. In the concept drift setting, it ranks first on the two ranking based measures and ranks second on the two label-based measures. Although iBR(SVM) has higher ranking than the proposed method on 5 measures on the non-concept drift setting, the proposed method is a better choice as it always attain high ranks and does not rank last on any measures. In the proposed method, the clustering model is maintained properly with the arrival of each sample based on its ground truth label set. The classification stage, therefore, can benefit when computing the posterior probability that a sample belongs to a class label based on the weight of the mature clusters. The weight of each cluster is decayed exponentially with time so that the model can be adaptive to the concept drift. Moreover, the number of predicted labels is learned based on the Hoeffding inequality and the label cardinality to ensure that each sample is assigned with an adequate set of labels. The proposed method, therefore, performed well in our experiments.

6. Conclusions

In this paper, we have introduced an incremental online learning method to solve the online MLC problem. In detail, we aggregate information in the arriving samples through a clustering process that takes into account the sample's time of arrival to compute the sample' weight. Each cluster's weight is then computed from the weights of the samples inside. The clustering process also builds up a distribution of labels in each cluster that would be later used for MLC. To handle concept drift, we proposed a decay mechanism on the sample's weight so that the influence of old samples is reduced over time while attention is put on the new ones. The classification process on each sample is conducted by using the mature clusters and their weights to compute the posterior probability that the sample belongs to a class. For MLC, the h labels associated with the top h classes of posterior probabilities (computed from the cluster's label distributions) are selected to assign labels for the sample. The number of predicted labels h is determined adaptively in our algorithm. Specifically, given a confidence level, we conduct the adjustment if there is a certain difference between h and the current label cardinality $\bar{z}$. The difference needed for the update is computed based on the Hoeffding inequality. The clustering model and the number of predicted labels are updated on-the-fly with the arrived samples and their ground truth labels. Due to the incremental learning nature of our algorithm, the incoming samples do not need to be stored once they are processed.

The proposed method and the benchmark algorithms were evaluated on five popular multi-label datasets in the stationary setting, and twelve multi-label datasets in the concept drift setting. The experimental results showed that our method is highly competitive compared to several benchmark algorithms, especially under the concept drift setting. The proposed method is high desirable in practice as it always maintains the high ranks and does not rank last on any measures

The performance of the proposed method can be enhanced by integrating label correlation [4] in the learning model. For MLC problems, label correlation is an important factor that can enhance the prediction quality. For example, if a sample has 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'. By considering the semantic relationship between the labels, we can choose the proper set of labels in the prediction process. In the streaming context, moreover, the label correlation will need to be updated with the incoming of each sample. Several methods capturing the label correlation such as classifier trellis [25] and graphical model for feature-label-label relationship triple [13] can be combined with the proposed method to enhance the performance of MLC task. Moreover, the proposed method can be expanded to handle the more general learning paradigm like MDC [25]. These will be our future works.

References

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

[2] T.T.T. Nguyen, T.T. Nguyen, A.W.C. Liew, S.L. Wang, Variational inference based Bayes online classifiers with concept drift adaptation, Pattern Recognition.81 (2018), 280-293.

[3] X.C. Pham, M.T. Dang, S.V. Dinh, S. Hoang, T.T. Nguyen, A.W.C. Liew, Learning from data stream based on Random Projection and Hoeffding Tree Classifier, in Proceeding of Digital Image Computing: Techniques and Applications, 2017.

[4] M.L. Zhang, Z.H. Zhou, A review on Multi-Label Learning Algorithms, IEEE Transactions on Knowledge and Data Engineering. 26(8) (2014), 1819-1837.

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

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

[7] F. Cao, M. Ester, W. Qian, A. Zhou, Density-Based Clustering over an Evolving Data Stream with Noise, in Proceedings of the SIAM International Conference on Data Mining, 2006.

[8] 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.

[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] 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. 279–286.

[11] 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.

[12] A. Kumar, S. Vembu, A. Menon, C. Elkan, Beam search algorithms for multi-label learning, Machine Learning 92 (2013), 65-89.

[13] N. Ghamrawi, A. McCallum, Collective multi-label classification, in: CIKM '05: 14th ACM International Conference on Information and Knowledge Management, ACM Press, New York, NY, USA, 2005, pp. 195-200.

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

[15] 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.

[16] Y. Guo, S. Gu, Multi-label classification using conditional dependency networks, in Proceeding of the International Joint Conference on Artificial Intelligence, 2011, pp 1300-1305.

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

[18] X. Li, Y. Guo, Active Learning with Multi-Label SVM Classification, in Proceeding of the International Joint Conference on Artificial Intelligence, 2013, pp. 1479-1485.

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

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

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

[22] 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.

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

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

[25] J. Read, L.Martino, P. Olmos, D. Luengo, Scalable Multi-Output Label Prediction: From Classifier Chains to Classifier Trellises, Pattern Recognition.48(6) (2015), 2096-2109.

[26] J. Read, L. Martino, D. Luengo, Efficient Monte Carlo Methods for Multi-Dimensional Learning with Classifier Chains, Pattern Recognition.47(3) (2014), 1535-1546.

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

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

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

[36] J. Xu, Multi-Label Weighted k-Nearest Neighbor Classifier with Adaptive Weight Estimation, International Conference on Neural Information Processing, 2011, pp 79-88

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

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

[39] 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.

[40] 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.

Appendix

The performance measures

In this paper, the comparisons of the proposed method and the benchmark algorithms are based on six well-known performance measures: sample-based F1/accuracy, label-based micro-averaged F1/macro-averaged F1, and ranking-based average precision/ranking loss. We briefly describe each measure supposing that N examples $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ are received with ground truth label sets $\mathbb{Y}_1, \mathbb{Y}_2, \mathbb{Y}_N$ and prediction label sets $\widehat{\mathbb{Y}}_1, \widehat{\mathbb{Y}}_2, \dots, \widehat{\mathbb{Y}}_N$, respectively.

Sample-based measures

Sample-based measures evaluate the performance of a MLC algorithm on a per-sample basis. They are calculated for each sample and then averaged over all of them.

Sample-based accuracy is the average proportion of label values correctly classified out of the total number (predicted and true) of labels:

$$\text{Sample} - \text{based accuracy} = \frac{1}{N}\sum_{t=1}^{N}\frac{|\mathbb{Y}_t \cap \hat{\mathbb{Y}}_t|}{|\mathbb{Y}_t \cup \hat{\mathbb{Y}}_t|} \quad \text{(A1)}$$

The sample-based F1 is the harmonic mean of sample-based precision and recall:

$$\text{Sample} - \text{based F1} = 2/\left(\frac{1}{\text{Ex.Precision}} + \frac{1}{\text{Ex.Recall}}\right) \quad \text{(A2)}$$

Here,

$$\text{Ex. Precision} = \frac{1}{N}\sum_{t=1}^{N}\frac{|\mathbb{Y}_t \cap \hat{\mathbb{Y}}_t|}{|\hat{\mathbb{Y}}_t|} \quad \text{(A3)}$$

$$\text{Ex. Recall} = \frac{1}{N}\sum_{t=1}^{N}\frac{|\mathbb{Y}_t \cap \hat{\mathbb{Y}}_t|}{|\mathbb{Y}_t|} \quad \text{(A4)}$$

The greater the sample-based accuracy and F1 of an MLC algorithm (with an optimal value of 1), the better its classification performance over different samples.

Label-based measures

Label-based measures evaluate the performance of a MLC algorithm on a per-label basis. They are calculated for each label and then averaged over all of them. Definitions of many of the label-based measures are based on four basic quantities named true positive (TP), true negative (TN), false positives (FP) and false negative (FN), which are calculated as follows for label $l \in \boldsymbol{\mathcal{Y}}$:

$$TP_l = \left|\left\{\mathbf{x}_t \middle| l \in \mathbb{Y}_t \wedge l \in \hat{\mathbb{Y}}_t, 1 \le t \le N\right\}\right| \quad \text{(A5)}$$

$$TN_l = \left|\left\{\mathbf{x}_t \middle| l \notin \mathbb{Y}_t \wedge l \notin \hat{\mathbb{Y}}_t, 1 \le t \le N\right\}\right| \quad \text{(A6)}$$

$$FP_l = \left|\left\{\mathbf{x}_t \middle| l \notin \mathbb{Y}_t \wedge l \in \hat{\mathbb{Y}}_t, 1 \le t \le N\right\}\right| \quad \text{(A7)}$$

$$FN_l = \left|\left\{\mathbf{x}_t \middle| l \in \mathbb{Y}_t \wedge l \notin \hat{\mathbb{Y}}_t, 1 \le t \le N\right\}\right| \quad \text{(A8)}$$

The value of F1 can be obtained in the form of macro-averaged or micro-averaged:

$$\text{Macro} - \text{average F1} = \frac{1}{M}\sum_l F1(TP_l, TN_l, FP_l, FN_l) \quad \text{(A9)}$$

$$\text{Micro} - \text{average F1} = F1\left(\sum_l TP_l, \sum_l TN_l, \sum_l FP_l, \sum_l FN_l\right) \quad \text{(A10)}$$

Here,

$$F1(TP, TN, FP, FN) = \frac{2TP}{2TP + FP + FN} \quad \text{(A11)}$$

Clearly, the greater the micro F1 and macro F1 (with an optimal value of 1), the better the predictive performance over different labels obtained by the learner.

Ranking-based measures

Ranking-based measures analyze the confidence outputs $f(\mathbf{x}_t, l) \in \mathbb{R}, l \in \boldsymbol{\mathcal{Y}}$ of a MLC methods directly, i.e. independent of the thresholding procedure. For $\mathbf{x}_t$, $rank_f(\mathbf{x}_t, l)$ returns the rank of l in $\boldsymbol{\mathcal{Y}}$ based on the descending order induced from $f(\mathbf{x}_t,\cdot)$. That means label l is considered to be ranked higher than l', i.e. $rank_f(\mathbf{x}_t, l) \leq rank_f(\mathbf{x}_t, l')$ if $f(\mathbf{x}_t, l) > f(\mathbf{x}_t, l')$.

Ranking loss evaluates the fraction of reversely ordered label pairs when an irrelevant label is ranked higher than a relevant label:

$$\text{Ranking Loss} = \frac{1}{N}\sum_{t=1}^{N} \frac{1}{|\mathbb{Y}_t||\overline{\mathbb{Y}}_t|} |\{(l, l')|f(\mathbf{x}_t, l) \leq f(\mathbf{x}_t, l'), (l, l') \in \mathbb{Y}_t \times \overline{\mathbb{Y}}_t\}| \tag{A12}$$

where $\overline{\mathbb{Y}}_t$ is the complementary set of $\mathbb{Y}_t$ in $\boldsymbol{\mathcal{Y}}$. Small values of ranking loss are desired.

Average precision evaluates the average fraction of labels ranked above a particular label $l \in \mathbb{Y}_t$ which actually are in $\mathbb{Y}_t$.

$$\text{Avegrage Precision} = \frac{1}{N}\sum_{t=1}^{N} \frac{1}{|\mathbb{Y}_t|} \sum_{l \in \mathbb{Y}_t} \frac{\left|\left\{l' \middle| rank_f(\mathbf{x}_t, l') \leq rank_f(\mathbf{x}_t, l), l' \in \mathbb{Y}_t\right\}\right|}{rank_f(\mathbf{x}_t, l)} \tag{A13}$$

Average precision reaches the maximum value of 1 when f ranks the labels for all samples perfectly so that there is no sample $\mathbf{x}_t$ for which a label not in $\mathbb{Y}_t$ has higher rank than a label in $\mathbb{Y}_t$.

Supplement Material

Paper: Multi-Label Classification via Incremental Clustering on Evolving Data Stream

Tien Thanh Nguyen et al.

**The up/down arrow beside the measure indicates that a higher/lower value is preferred for that measure.*

TABLE S1. PERFORMANCE MEASURES FOR DIFFERENT VALUES OF K NEAREST NEIGHBORS

Sample-based F1(↑)				
	K=3	K=5	K=7	K=10
ENRON	0.4053	0.4049	0.4023	0.4013
IMDB	0.3220	0.3212	0.3212	0.3213
OHSUMED	0.2356	0.2375	0.2403	0.2415
SLASHDOT	0.1623	0.1504	0.1527	0.1515
TMC2007	0.5187	0.5026	0.4914	0.4797
Sample-based Accuracy (↑)				
	K=3	K=5	K=7	K=10
ENRON	0.3016	0.2999	0.2983	0.2981
IMDB	0.2354	0.2348	0.2349	0.2350
OHSUMED	0.1784	0.1795	0.1819	0.1827
SLASHDOT	0.1547	0.1432	0.1447	0.1439
TMC2007	0.4044	0.3888	0.3781	0.3668
Micro-average F1 (↑)				
	K=3	K=5	K=7	K=10
ENRON	0.4083	0.4105	0.4083	0.4101
IMDB	0.3232	0.3224	0.3226	0.3226
OHSUMED	0.2531	0.2551	0.2577	0.2591
SLASHDOT	0.1557	0.1447	0.1479	0.1463
TMC2007	0.5236	0.5073	0.4963	0.4845
Macro-average F1 (↑)				
	K=3	K=5	K=7	K=10
ENRON	0.0761	0.0733	0.0651	0.0618
IMDB	0.0473	0.0457	0.0447	0.0440
OHSUMED	0.0460	0.0451	0.0448	0.0449
SLASHDOT	0.1199	0.1097	0.1125	0.1109
TMC2007	0.1601	0.1331	0.1198	0.1054
Average Precision (↑)				

	K=3	K=5	K=7	K=10
ENRON	0.5404	0.5507	0.5575	0.5597
IMDB	0.4728	0.4729	0.4730	0.4734
OHSUMED	0.3812	0.3820	0.3827	0.3836
SLASHDOT	0.3588	0.3502	0.3507	0.3506
TMC2007	0.6622	0.6500	0.6385	0.6269
Ranking Loss (↓)				
	K=3	K=5	K=7	K=10
ENRON	0.2486	0.2077	0.1859	0.1691
IMDB	0.1770	0.1758	0.1752	0.1747
OHSUMED	0.2754	0.2676	0.2650	0.2626
SLASHDOT	0.2581	0.2554	0.2488	0.2460
TMC2007	0.1321	0.1262	0.1279	0.1323

TABLE S2. PERFORMANCE MEASURES AND RANKINGS OF THE PROPOSED CLASSIFICATION METHOD AND WEIGHTED K-NN

	Sample-based F1(↑)		Micro-average F1 (↑)		Average Precision (↑)	
	Proposed Method	Weighed KNN	Proposed Method	Weighted KNN	Proposed Method	Weighted KNN
ENRON	0.4053 (1)	0.3371 (2)	0.4083 (1)	0.3538 (2)	0.5404 (2)	0.5448 (1)
IMDB	0.3220 (1)	0.0055 (2)	0.3232 (1)	0.0090 (2)	0.4728 (2)	0.4743 (1)
OHSUMED	0.2356 (1)	0.0057 (2)	0.2531 (1)	0.0103 (2)	0.3812 (2)	0.3815 (1)
SLASHDOT	0.1623 (1)	0.0162 (2)	0.1557 (1)	0.0269 (2)	0.3588 (1)	0.3588 (2)
TMC2007	0.5187 (1)	0.4655 (2)	0.5236 (1)	0.5038 (2)	0.6622 (2)	0.6653 (1)
Averaged Rank	1	2	1	2	1.8	1.2
	Sample-based Accuracy (↑)		Macro-average F1 (↑)		Ranking Loss (↓)	
	Proposed Method	Weighed KNN	Proposed Method	Weighted KNN	Proposed Method	Weighted KNN
ENRON	0.3016 (1)	0.2603 (2)	0.0761 (1)	0.0650 (2)	0.2486 (2)	0.2454 (1)
IMDB	0.2354 (1)	0.0045 (2)	0.0473 (1)	0.0026 (2)	0.177 (2)	0.1738 (1)
OHSUMED	0.1784 (1)	0.0048 (2)	0.0460 (1)	0.0040 (2)	0.2754 (2)	0.2737 (1)
SLASHDOT	0.1547 (1)	0.0160 (2)	0.1199 (1)	0.0999 (2)	0.2581 (2)	0.2566 (1)
TMC2007	0.4044 (1)	0.3731 (2)	0.1601 (1)	0.1586 (2)	0.1321 (2)	0.1296 (1)
Averaged Rank	1	2	1	2	2	1

TABLE S3. THE SAMPLE-BASED MEASURES AND RANKINGS OF THE BENCHMARK ALGORITHMS AND THE PROPOSED METHOD UNDER STATIONARY SETTING

Sample-based F1(↑)									
	Proposed Method	iSOUP-MT	iSOUP-RT	iSOUP-EBRT	iSOUP-EBMT	iCC	MLS	iPS	iBR (SVM)
ENRON	0.4053 (2)	0.3296 (4)	0.2411 (8)	0.2530 (7)	0.3221 (5)	0.3933 (3)	0.2021 (9)	0.2778 (6)	0.4792 (1)
IMDB	0.3220 (1)	0.0227 (6)	0.0031 (8)	0.0008 (9)	0.0037 (7)	0.0915 (5)	0.2483 (2)	0.2420 (3)	0.2175 (4)
OHSUMED	0.2356 (4)	0.1767 (6)	0.1829 (5)	0.1280 (8)	0.1156 (9)	0.2425 (3)	0.1432 (7)	0.3316 (2)	0.3914 (1)
SLASHDOT	0.1623 (3)	0.0049 (6)	0.0003 (7.5)	0.0000 (9)	0.0003 (7.5)	0.2425 (2)	0.1484 (4)	0.1438 (5)	0.4100 (1)
TMC2007	0.5187 (4)	0.4303 (7)	0.4335 (5)	0.4307 (6)	0.4175 (8)	0.6146 (1)	0.2166 (9)	0.5696 (3)	0.5942 (2)
Averaged Rank	2.8	5.8	6.7	7.8	7.3	2.8	6.2	3.8	1.8
Sample-based Accuracy (↑)									
	Proposed Method	iSOUP-MT	iSOUP-RT	iSOUP-EBRT	iSOUP-EBMT	iCC	MLS	iPS	iBR (SVM)
ENRON	0.3016 (2)	0.2438 (4)	0.1797 (8)	0.1887 (7)	0.2379 (5)	0.2979 (3)	0.1680 (9)	0.2246 (6)	0.3659 (1)
IMDB	0.2354 (1)	0.0187 (6)	0.0026 (8)	0.0007 (9)	0.0031 (7)	0.0710 (5)	0.2081 (2)	0.2032 (3)	0.1606 (4)
OHSUMED	0.1784 (4)	0.1563 (6)	0.1611 (5)	0.1143 (8)	0.1035 (9)	0.2110 (3)	0.1270 (7)	0.2902 (2)	0.3190 (1)
SLASHDOT	0.1547 (3)	0.0049 (6)	0.0003 (7.5)	0.0000 (9)	0.0003 (7.5)	0.2110 (2)	0.1458 (4)	0.1407 (5)	0.3706 (1)
TMC2007	0.4044 (4)	0.3448 (6)	0.3479 (5)	0.3439 (7)	0.3317 (8)	0.5185 (1)	0.1786 (9)	0.4804 (3)	0.4929 (2)
Averaged Rank	2.8	5.6	6.7	8	7.3	2.8	6.2	3.8	1.8

TABLE S4. THE LABEL-BASED MEASURES AND RANKINGS OF THE BENCHMARK ALGORITHMS AND THE PROPOSED METHOD UNDER STATIONARY SETTING

Micro-average F1 (↑)									
	Proposed Method	iSOUP-MT	iSOUP-RT	iSOUP-EBRT	iSOUP-EBMT	iCC	MLS	iPS	iBR (SVM)
ENRON	0.4083 (2)	0.3374 (4)	0.2251 (8)	0.2385 (7)	0.3270 (5)	0.3990 (3)	0.1427 (9)	0.2547 (6)	0.4818 (1)
IMDB	0.3232 (1)	0.0350 (6)	0.0057 (7)	0.0012 (9)	0.0056 (8)	0.1263 (5)	0.2379 (3)	0.2309 (4)	0.2398 (2)
OHSUMED	0.2531 (4)	0.2325 (6)	0.2399 (5)	0.1724 (7)	0.1564 (8)	0.3206 (3)	0.1374 (9)	0.3297 (2)	0.4233 (1)
SLASHDOT	0.1557 (3)	0.0084 (6)	0.0004 (7.5)	0.0000 (9)	0.0004 (7.5)	0.3206 (2)	0.1411 (4)	0.1379 (5)	0.4408 (1)
TMC2007	0.5236 (4)	0.4651 (6)	0.4732 (5)	0.4642 (7)	0.4484 (8)	0.6346 (1)	0.1988 (9)	0.5731 (3)	0.6101 (1)
Averaged Rank	2.8	5.6	6.5	7.8	7.3	2.8	6.8	4	1.4
Macro-average F1 (↑)									
	Proposed Method	iSOUP-MT	iSOUP-RT	iSOUP-EBRT	iSOUP-EBMT	iCC	MLS	iPS	iBR (SVM)
ENRON	0.0761 (2)	0.0364 (5)	0.0199 (8)	0.0217 (7)	0.0340 (6)	0.0440 (4)	0.0090 (9)	0.0464 (3)	0.1881 (1)
IMDB	0.0473 (2)	0.0113 (6)	0.0023 (7)	0.0004 (9)	0.0019 (8)	0.0452 (3)	0.0245 (5)	0.0277 (4)	0.1356 (1)
OHSUMED	0.0460 (8)	0.1210 (5)	0.1269 (4)	0.0745 (6)	0.0617 (7)	0.2589 (2)	0.0135 (9)	0.1478 (3)	0.3229 (1)
SLASHDOT	0.1199 (4)	0.0041 (6)	0.0002 (7.5)	0.0000 (9)	0.0002 (7.5)	0.2589 (2)	0.1030 (5)	0.1267 (3)	0.3788 (1)
TMC2007	0.1601 (5)	0.1503 (6)	0.1605 (4)	0.1228 (7)	0.1110 (8)	0.4456 (2)	0.0232 (9)	0.1905 (3)	0.5082 (1)
Averaged Rank	4.2	5.6	6.1	7.6	7.3	2.6	7.4	3.2	1

TABLE S5. THE RANKING-BASED MEASURES AND RANKINGS OF THE BENCHMARK ALGORITHMS AND THE PROPOSED METHOD UNDER STATIONARY SETTING

Average Precision (↑)									
	Proposed Method	iSOUP-MT	iSOUP-RT	iSOUP-EBRT	iSOUP-EBMT	iCC	MLS	iPS	iBR (SVM)
ENRON	0.5404 (1)	0.1131 (6)	0.1023 (9)	0.1024 (8)	0.1125 (7)	0.3709 (3)	0.2409 (5)	0.2934 (4)	0.4560 (2)
IMDB	0.4728 (1)	0.1986 (5)	0.1901 (8)	0.1907 (7)	0.1972 (6)	0.1841 (9)	0.3003 (3)	0.2959 (4)	0.3028 (2)
OHSUMED	0.3812 (3)	0.1846 (7)	0.1806 (9)	0.1836 (8)	0.1848 (6)	0.3684 (4)	0.2700 (5)	0.4220 (2)	0.4576 (1)
SLASHDOT	0.3588 (3)	0.1586 (6)	0.1529 (9)	0.1585 (7)	0.1565 (8)	0.3684 (2)	0.2912 (4)	0.2843 (5)	0.5083 (1)
TMC2007	0.6622 (1)	0.1992 (9)	0.2001 (8)	0.2121 (6)	0.2016 (7)	0.6423 (3)	0.2860 (5)	0.5953 (4)	0.6429 (2)
Averaged Rank	1.4	5.6	7.6	6.2	5.8	3.2	3.4	2.8	1.6
Ranking Loss (↓)									
	Proposed Method	iSOUP-MT	iSOUP-RT	iSOUP-EBRT	iSOUP-EBMT	iCC	MLS	iPS	iBR (SVM)
ENRON	0.2486 (5)	0.1208 (4)	0.1181 (2)	0.1183 (3)	0.1165 (1)	0.3803 (6)	0.5095 (9)	0.4594 (7)	0.5022 (8)
IMDB	0.1770 (4)	0.1878 (5)	0.1737 (3)	0.1708 (2)	0.1705 (1)	0.4934 (8)	0.4514 (6)	0.4532 (7)	0.7158 (9)
OHSUMED	0.2754 (5)	0.2254 (4)	0.2163 (3)	0.2024 (1)	0.2110 (2)	0.3652 (7)	0.4376 (8)	0.3350 (6)	0.5688 (9)
SLASHDOT	0.2581 (5)	0.2202 (2)	0.2216 (4)	0.2206 (3)	0.2185 (1)	0.3652 (6)	0.3678 (7)	0.3732 (8)	0.5420 (9)
TMC2007	0.1321 (5)	0.1220 (4)	0.1158 (3)	0.1012 (1)	0.1132 (2)	0.2254 (6)	0.5848 (9)	0.2826 (7)	0.3728 (8)
Averaged Rank	4.8	3.8	3	2	1.4	6.6	7.8	7	8.6

TABLE S6. THE SAMPLE-BASED MEASURES AND RANKINGS OF THE BENCHMARK ALGORITHMS AND THE PROPOSED METHOD UNDER THE CONCEPT DRIFT SETTING

Sample-based F1 (↑)					
	Proposed Method	iCC	MLS	iPS	iBR (SVM)
ENRON-drift	0.2312 (3)	0.2073 (4)	0.1082 (5)	0.2376 (2)	0.2607 (1)
ENRON-break	0.2329 (2)	0.2075 (4)	0.1070 (5)	0.2325 (3)	0.2617 (1)
IMDB-drift	0.2293 (1)	0.0577 (5)	0.1325 (4)	0.1741 (2)	0.1628 (3)
IMDB-break	0.2299 (1)	0.0467 (5)	0.1323 (4)	0.1753 (2)	0.1632 (3)
OHSUMED-drift	0.2040 (3)	0.1203 (5)	0.1254 (4)	0.2308 (1)	0.2153 (2)
OHSUMED-break	0.2040 (3)	0.1201 (4)	0.0740 (5)	0.2343 (1)	0.2169 (2)
SLASHDOT-drift	0.1447 (3)	0.0222 (5)	0.0991 (4)	0.1469 (2)	0.2288 (1)
SLASHDOT-break	0.1440 (3)	0.0213 (5)	0.0992 (4)	0.1482 (2)	0.2337 (1)
SynRTG-drift	0.4453 (4)	0.4763 (2)	0.4490 (3)	0.5000 (1)	0.3817 (5)
SynRTG-break	0.4465 (3)	0.4765 (2)	0.3469 (5)	0.5021 (1)	0.3826 (4)
TMC2007-drift	0.3450 (2)	0.3124 (4)	0.1142 (5)	0.3522 (1)	0.3374 (3)
TMC2007-break	0.3457 (2)	0.3124 (4)	0.1084 (5)	0.3574 (1)	0.3392 (3)
Averaged Rank	2.50	4.08	4.42	1.58	2.42
Sample-based Accuracy (↑)					
	Proposed Method	iCC	MLS	iPS	iBR (SVM)
ENRON-drift	0.1698 (3)	0.1582 (4)	0.0904 (5)	0.1837 (2)	0.1952 (1)
ENRON-break	0.1710 (3)	0.1585 (4)	0.0894 (5)	0.1804 (2)	0.1959 (1)
IMDB-drift	0.1616 (1)	0.0484 (5)	0.1099 (4)	0.1385 (2)	0.1148 (3)
IMDB-break	0.1620 (1)	0.0361 (5)	0.1097 (4)	0.1396 (2)	0.1151 (3)
OHSUMED-drift	0.1461 (3)	0.1038 (4)	0.1007 (5)	0.1918 (1)	0.1607 (2)
OHSUMED-break	0.1459 (3)	0.1037 (4)	0.0652 (5)	0.1945 (1)	0.1620 (2)
SLASHDOT-drift	0.1216 (3)	0.0182 (5)	0.0916 (4)	0.1261 (2)	0.1913 (1)
SLASHDOT-break	0.1206 (3)	0.0174 (5)	0.0918 (4)	0.1272 (2)	0.1949 (1)
SynRTG-drift	0.3370 (4)	0.3866 (2)	0.3577 (3)	0.4137 (1)	0.2812 (5)
SynRTG-break	0.3388 (3)	0.3870 (2)	0.2868 (4)	0.4169 (1)	0.2826 (5)
TMC2007-drift	0.2575 (3)	0.2622 (2)	0.0935 (5)	0.2852 (1)	0.2595 (3)
TMC2007-break	0.2581 (3)	0.2626 (2)	0.0894 (5)	0.2894 (1)	0.2613 (3)
Averaged Rank	2.92	3.67	4.42	1.5	2.5

TABLE S7. THE LABEL-BASED MEASURES AND RANKINGS OF THE BENCHMARK ALGORITHMS AND THE PROPOSED METHOD UNDER THE CONCEPT DRIFT SETTING

Micro average-F1 (↑)					
	Proposed Method	iCC	MLS	iPS	iBR (SVM)
ENRON-drift	0.2289 (4)	0.2618 (2)	0.0898 (5)	0.2324 (3)	0.2811 (1)
ENRON-break	0.2303 (4)	0.2623 (2)	0.0893 (5)	0.2320 (3)	0.2820 (1)
IMDB-drift	0.2165 (1)	0.0616 (5)	0.1181 (4)	0.1499 (3)	0.1698 (2)
IMDB-break	0.2172 (1)	0.0625 (5)	0.1182 (4)	0.1502 (3)	0.1703 (2)
OHSUMED-drift	0.2052 (2)	0.1427 (4)	0.1211 (5)	0.1981 (3)	0.2140 (1)
OHSUMED-break	0.2052 (2)	0.1424 (4)	0.0709 (5)	0.2020 (3)	0.2152 (1)
SLASHDOT-drift	0.1381 (3)	0.0344 (5)	0.0826 (4)	0.1506 (3)	0.2038 (1)
SLASHDOT-break	0.1374 (3)	0.0328 (5)	0.0827 (4)	0.1524 (3)	0.2108 (1)
SynRTG-drift	0.4659 (3)	0.4960 (2)	0.4641 (4)	0.5204 (1)	0.4115 (5)
SynRTG-break	0.4674 (3)	0.4963 (2)	0.4099 (5)	0.5228 (1)	0.4127 (4)
TMC2007-drift	0.3208 (3)	0.3721 (1)	0.1062 (5)	0.3162 (4)	0.3253 (2)
TMC2007-break	0.3213 (3)	0.3717 (1)	0.1046 (5)	0.3195 (4)	0.3273 (2)
Averaged Rank	2.67	3.17	4.58	2.67	1.92
Macro average-F1 (↑)					
	Proposed Method	iCC	MLS	iPS	iBR (SVM)
ENRON-drift	0.0834 (2)	0.0412 (4)	0.0081 (5)	0.0621 (3)	0.1407 (1)
ENRON-break	0.0871 (2)	0.0412 (4)	0.0083 (5)	0.0588 (3)	0.1416 (1)
IMDB-drift	0.1204 (2)	0.0299 (4)	0.0183 (5)	0.0506 (3)	0.1399 (1)
IMDB-break	0.1211 (2)	0.0331 (4)	0.0184 (5)	0.0466 (3)	0.1404 (1)
OHSUMED-drift	0.1211 (4)	0.1302 (2)	0.0141 (5)	0.1272 (3)	0.1831 (1)
OHSUMED-break	0.1211 (4)	0.1297 (3)	0.0118 (5)	0.1380 (2)	0.1844 (1)
SLASHDOT-drift	0.0960 (2)	0.0317 (4)	0.0096 (5)	0.0757 (3)	0.1740 (1)
SLASHDOT-break	0.0950 (2)	0.0307 (4)	0.0096 (5)	0.0730 (3)	0.1820 (1)
SynRTG-drift	0.3598 (1)	0.3376 (4)	0.2982 (5)	0.3441 (3)	0.3592 (2)
SynRTG-break	0.3614 (1)	0.3394 (4)	0.2792 (5)	0.3464 (3)	0.3601 (2)
TMC2007-drift	0.2049 (3)	0.2315 (2)	0.0193 (5)	0.1687 (4)	0.2591 (1)
TMC2007-break	0.2059 (3)	0.2314 (2)	0.0220 (5)	0.1737 (4)	0.2610 (1)
Averaged Rank	2.33	3.42	5	3.08	1.17

TABLE S8. THE RANKING-BASED MEASURES AND RANKINGS OF THE BENCHMARK ALGORITHMS AND THE PROPOSED METHOD UNDER THE CONCEPT DRIFT SETTING

Average Precision (↑)					
	Proposed Method	iCC	MLS	iPS	iBR (SVM)
ENRON-drift	0.3324 (1)	0.2488 (4)	0.1811 (5)	0.2558 (3)	0.2837 (2)
ENRON-break	0.3323 (1)	0.2493 (4)	0.1806 (5)	0.2534 (3)	0.2838 (2)
IMDB-drift	0.3524 (1)	0.1904 (5)	0.2346 (4)	0.2464 (2)	0.2440 (3)
IMDB-break	0.3527 (1)	0.1827 (5)	0.2348 (4)	0.2470 (2)	0.2444 (3)
OHSUMED-drift	0.3290 (2)	0.2826 (4)	0.2573 (5)	0.3296 (1)	0.3025 (3)
OHSUMED-break	0.3301 (2)	0.2823 (4)	0.2368 (5)	0.3316 (1)	0.3029 (3)
SLASHDOT-drift	0.3112 (2)	0.2343 (5)	0.2528 (4)	0.2720 (3)	0.3431 (1)
SLASHDOT-break	0.3104 (2)	0.2327 (5)	0.2529 (4)	0.2728 (3)	0.3468 (1)
SynRTG-drift	0.5997 (3)	0.6053 (2)	0.5837 (4)	0.6178 (1)	0.5539 (5)
SynRTG-break	0.6011 (3)	0.6054 (2)	0.5768 (4)	0.6192 (1)	0.5553 (5)
TMC2007-drift	0.4738 (1)	0.4418 (2)	0.2611 (5)	0.4191 (4)	0.4274 (3)
TMC2007-break	0.4734 (1)	0.4418 (2)	0.2644 (5)	0.4227 (4)	0.4291 (3)
Averaged Rank	1.67	3.67	4.5	2.33	2.83
Ranking Loss (↓)					
	Proposed Method	iCC	MLS	iPS	iBR (SVM)
ENRON-drift	0.4848 (3)	0.4440 (2)	0.5125 (4)	0.4315 (1)	0.7262 (5)
ENRON-break	0.4868 (3)	0.4431 (2)	0.5123 (4)	0.4331 (1)	0.7265 (5)
IMDB-drift	0.3727 (1)	0.5091 (4)	0.4895 (3)	0.4823 (2)	0.8224 (5)
IMDB-break	0.3723 (1)	0.5086 (4)	0.4895 (3)	0.4829 (2)	0.8221 (5)
OHSUMED-drift	0.4064 (1)	0.4443 (3)	0.4666 (4)	0.4192 (2)	0.7685 (5)
OHSUMED-break	0.4047 (1)	0.4445 (3)	0.4776 (4)	0.4177 (2)	0.7671 (5)
SLASHDOT-drift	0.4147 (1)	0.4620 (4)	0.4450 (3)	0.4353 (2)	0.7519 (5)
SLASHDOT-break	0.4159 (1)	0.4630 (4)	0.4450 (3)	0.4350 (2)	0.7474 (5)
SynRTG-drift	0.3872 (2)	0.3883 (3)	0.4157 (4)	0.3704 (1)	0.6176 (5)
SynRTG-break	0.3861 (2)	0.3882 (3)	0.4191 (4)	0.3683 (1)	0.6033 (5)
TMC2007-drift	0.3290 (1)	0.3728 (2)	0.5530 (4)	0.4045 (3)	0.6502 (5)
TMC2007-break	0.3301 (1)	0.3732 (2)	0.5526 (4)	0.4009 (3)	0.6483 (5)
Averaged Rank	1.5	3	3.67	1.83	5