Multi-Label Learning with Emerging New Labels

Yue Zhu, Kai Ming Ting, and Zhi-Hua Zhou, *Fellow, IEEE*

Abstract—In a multi-label learning task, an object possesses multiple concepts where each concept is represented by a class label. Previous studies on multi-label learning have focused on a fixed set of class labels, i.e., the class label set of test data is the same as that in the training set. In many applications, however, the environment is dynamic and new concepts may emerge in a data stream. In order to maintain a good predictive performance in this environment, a multi-label learning method must have the ability to detect and classify instances with emerging new labels. To this end, we propose a new approach called Multi-label learning with Emerging New Labels (`MuENL`). It has three functions: classify instances on currently known labels, detect the emergence of a new label, and construct a new classifier for each new label that works collaboratively with the classifier for known labels. In addition, we show that `MuENL` can be easily extended to handle sparse high dimensional data streams by simply reducing the original dimensionality, and then applying `MuENL` on the reduced dimensional space. Our empirical evaluation shows the effectiveness of `MuENL` on several benchmark datasets and `MuENLHD` on the sparse high dimensional Weibo dataset.

Index Terms—Multi-label learning; incremental learning; emerging new labels; learnware

1 Introduction

In traditional supervised learning, one instance is associated with a single label. Yet, in many applications, one instance may possess multiple labels. For example, a scene image is usually annotated with several tags [3]; a document may hold multiple topics [17]; and a piece of music may belong to different genres [26].

Multi-label learning is the learning paradigm to handle such kind of data, and has attracted much attention in recent years [31]. Previous studies on multi-label learning have focused on a fixed set of class labels. That is, they assume that the test data have the same set of class labels as that of the training data. In many real-world data mining tasks, however, the environment is dynamic. We study a dynamic scenario in which new labels may emerge together with known labels in an observed instance of a data stream.

In a dynamic environment, a learning system must be able to reuse a previously learned model as well as to adapt the model to the changing environment [32]. In the multi-label learning setting, the system must be able to revise a pre-trained model as new instances are observed; and new classifiers are established for all emerging new labels. These demands are non-trivial, and no existing systems in the literature can meet these demands, as far as we know.

Under the dynamic multi-label learning setting, we assume that instances arrive in a data stream, and no ground truths for class labels are available in the data stream at all times, except for the initial training set. This can be regarded as a special weakly supervised learning [33]. As a result, detecting and modeling new labels are the key challenges. Specifically, the most difficult part is to detect instances with any new label. Since we do not have any prior knowledge of the new label and it almost always co-occurs with some known labels, it is very difficult to separate instances with new labels from those with the known labels only.

Moreover, because the detection is not perfect, the error will accumulate as more and more new labels emerge in a data stream. Thus, the environment demands robust models in order to maintain a high detection and prediction performances continuously in a data stream, which is also a challenging task.

To meet all the above challenges, we propose a novel Multi-label learning with Emerging New Labels (`MuENL`) approach to address the dynamic multi-label learning problem. `MuENL` consists of three components: (1) A classifier is built to optimize both the pairwise label ranking loss and the classification loss on the known labels; (2) a specially designed detector based on both the input features and predicted label attributes; and (3) a classifier updating process that incorporates detected new labels to produce a robust classifier which can tolerate detection errors, and remodels the detector for each new label identified.

The central idea of this paper is to regard instances with emerging new labels as outliers to the norm—instances of known labels seen thus far. This admits outlier detection methods to be used in the dynamic multi-label learning problem. We show that the idea works in practice.

In addition to addressing the core challenges in dynamic multi-label learning problem, we also propose an extension to deal with sparse high dimensional data streams. In a social network site such as Weibo, users post short-text messages with diverse topics. In a bag-of-word representation, each dimension represents a word in a dictionary of thousands of words. Each message is thus a sparse representation of a high dimensional space. We show that `MuENL` can be easily extended to `MuENLHD` to handle sparse high dimensional data streams.

The contribution of this work is summarized as follows:

- Formalizing the dynamic multi-label learning problem, which is different from previous multi-label learning settings: new labels may emerge with arriving new instances.

- *Y. Zhu and Z.-H. Zhou are with National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China.*
E-mail: {zhuy, zhouzh}@lamda.nju.edu.cn
- *K.M. Ting is with School of Engineering and Information Technology, Federation University, Australia.*
E-mail: kaiming.ting@federation.edu.au

- Proposing the MuENL approach to address the dynamic multi-label learning problem. It has an accurate detector for new labels and a robust classifier for both known and new labels. The time complexity of each component is analyzed. We also examine one way to handle multiple new labels which emerge at the same time using a meta label.
- Extending MuENL to MuENLHD to handle sparse high dimensional data streams.
- Extensive empirical studies are conducted to validate the effectiveness of our approach.

The rest of paper is organized as follows: Section 2 describes several related works on multi-label learning, incremental learning and outlier detection. Section 3 introduces the problem formulation and the MuENL approach. Section 4 extends MuENL to MuENLHD to handle sparse high dimensional data streams. Section 5 analyses the time complexity. Section 6 and Section 7 describe the experimental studies on (dense) low/medium dimensional data streams and sparse high dimensional data streams, respectively. The conclusions are given in the last section.

2 Related Work

Multi-label learning can be divided into three main categories based on the order of label correlations [31]. For the first-order strategy, none of the label correlations are considered. BR [3], for example, trains a classifier for each label independently. For the second-order strategy, pairwise label relations are taken into account. In this strategy, CLR [7] transforms the multi-label learning problem into a pairwise label ranking problem. For the high-order strategy, a label is assumed to be influenced by all other labels. CC [20], for instance, transforms the multi-label learning problem into a chain of binary classification, where the ground-truth labels are successively encoded into the feature space. All the above multi-label learning approaches assume that the class label set is fixed and do not admit new labels. As such, they cannot handle the dynamic multi-label learning problem we investigated in this paper.

Incremental learning is critical for the tasks where a frequent data update is involved or when it is desirable not to re-train the model from scratch. According to [34] , there are roughly three major incremental learning settings, i.e., example incremental learning (E-IL) [21], where new instances arrive after the learning system has been trained; attribute incremental learning (A-IL) [27], where new features may appear; and class incremental learning (C-IL) [4], [13], [30], where the class label set may be enlarged. Learning with new labels is a kind of C-IL, which has been studied under various names including zero-shot learning [15], [18] and open-set classification [14], [22]. Besides, the basic assumption of those C-IL works is that the class labels on each instance are mutually exclusive, namely, they are in a multi-class learning setting.

The dynamic multi-label learning setting with emergingnew labels is a combination of E-IL and C-IL, where a new instance may be associated with multiple new labels that co-occur with known labels. A straightforward approach to adapt C-IL under our setting is to transform the multi-label learning into the multi-class learning by converting each possible label combination into a class [24]. Unfortunately, this approach has two severe limitations. First, a new class may not correspond to a new label, but an unseen combination of known labels. Second, when the label set is large, the number of possible classes is huge. This leads to a difficult training problem, i.e., having an extremely small number of positive instances for most classes. As a result, it cannot be applied in practice.

When a part of the dynamic multi-label learning problem is converted to be an outlier detection problem, many existing methods can be applied; but not in a straightforward manner. For example, OC-SVM [23] learns a boundary for instances with known labels, and decides instances outside the boundary as outliers; iForest [16] predicts instances located in a sparse region as outliers. However, under the multi-label setting, a new label may co-occur with known labels, which makes it difficult to separate instances with new labels only from instances with known labels only.

Recently, Fu [6] has proposed a transductive multi-label zero-shot learning. However, in a transductive setting, all test instances are assumed to be available during the training and all the new labels are assumed to be known. As a result, it cannot be applied in our setting—new instances successively arrive, and we do not know when one or more new labels may occur; or the total number of new labels may occur in one time period.

Another line of related works is multi-label learning with missing labels. Many approaches aim to recover missing labels by exploiting low rank structure or label correlations [1], [12], [29], [35]. In their settings, the labels, which are missing in some instances, are within the observed class label set. In other words, they assume no previously unseen labels; and the missing labels cannot be recovered if they have never been observed during training. Therefore, these approaches cannot deal with novel labels in our setting.

3 The MuENL Approach

The dynamic multi-label learning problem we studied in this paper faces the following challenges: (A) detecting instances with emerging new labels which are also associated known labels; (B) building a robust classifier for both the new and known labels. In this section, we propose an approach called MuENL to handle the dynamic multi-label learning problem that addresses the two challenges.

3.1 Problem Formulation

In open dynamic multi-label learning, we have an initial labeled training set, then unlabeled data successively arrive in a streaming fashion. Let $\mathcal{X}$ denote the feature space and define $X_0 = [\boldsymbol{x}_{-n+1}, \cdots, \boldsymbol{x}_{-1}, \boldsymbol{x}_0]^\top \subseteq \mathcal{X}$ as the observed n labeled instances in the initial labeled training set. The arriving data stream has an unlabeled instance $\boldsymbol{x}_t$ observed at time t. Let $X_t, t \in \{1, 2, \cdots, T\}$ be the accessible data trunk at time t.

We denote by $\boldsymbol{v}_0 = \{1, 2, \cdots, \ell\}$ the known class label set [1] at $t = 0$, which is the union of all distinct labels appearing in the initial training set. Let $\boldsymbol{v}_t$ be the class label set at time t, and its maximum indexed label is ℓ, which equals to the length of $\boldsymbol{v}_t$. When a new label is to be converted to

[1] The number in $\boldsymbol{v}$ represents the index of the label names.

a known label at time t, the class label set will be enlarged with a new label $\ell' = \ell + 1$: $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} \cup \{\ell'\}$ [2]; otherwise, it will not change: $\boldsymbol{v}_t = \boldsymbol{v}_{t-1}$. Let $Y_0 = [\boldsymbol{y}_{-n+1}, \cdots, \boldsymbol{y}_{-1}, \boldsymbol{y}_0] \in \{-1, 1\}^{\ell \times n}$ be the known label matrix of X_0 in the initial training set, and $\boldsymbol{y}_t = [y_{t,1}, \cdots, y_{t,\ell}] \in \{-1, 1\}^{\ell}$ be the label vector of the newly arrived instance $\boldsymbol{x}_t$, where $y_{t,j} = 1$ suggests that $\boldsymbol{x}_t$ holds the j-th label; $y_{t,j} = -1$ otherwise. Note that none of the elements in $\boldsymbol{y}_t$ are observed, since $\boldsymbol{x}_t$ is unlabeled for $t > 0$. The dynamic multi-label learning problem is defined as:

Definition 1. Dynamic Multi-Label Learning: *Given X_t and Y_0, the task is to learn a function set $\mathcal{H}_t = [h_{t,1}, h_{t,2}, \cdots, h_{t,\ell}]$, where $h_{t,j} : \mathcal{X} \to \{-1, 1\}^{\ell}$ represents the classifier for j-th class label, $j \in \{1, 2, \cdots, \ell\}$, at time $t \in \{1, 2, \cdots, T\}$. For each $\boldsymbol{x}_t$, $\hat{\boldsymbol{y}}_t = \mathcal{H}_t(\boldsymbol{x}_t)$ is the predicted label vector, where $\boldsymbol{x}_t$ may be associated with both new labels and known labels.*

3.2 The Approach

Directly estimating $\mathcal{H}_t$ in the dynamic learning environment defined above is extremely hard, since $\boldsymbol{v}_t$ has an unknown variable, i.e., it is not known whether an observed instance $\boldsymbol{x}_t$ has a new label. In addition, we assume that the ground truth is not available throughout the entire data stream. Thus, the ability to accurately identify the new label when it emerges in $\boldsymbol{x}_t$ is critical in not only detecting its emergence, but also in maintaining a highly accurate classifier for the expanded set of known labels throughout the data stream.

We approach the problem by creating a detection function for any previously unseen labels, i.e., $\mathcal{D}_t(\boldsymbol{x}_t)$, which outputs 1 if $\boldsymbol{x}_t$ holds a new label; or -1 otherwise. For each instance $\boldsymbol{x}_t$ in the data stream, the current classifier is used to do the prediction, yielding $\hat{\boldsymbol{y}}_t = \mathcal{H}_t(\boldsymbol{x}_t)$, where $\mathcal{H}_t = [h_{t,1}, h_{t,2}, \cdots, h_{t,\ell}, \mathcal{D}_t]$.

Algorithm 1 summarizes the `MuENL` approach. It has three key components: (i) the multi-label classifier ($\mathcal{H}_t = [h_{t,1}, \cdots, h_{t,\ell}]$) for the known labels; (ii) the detector ($\mathcal{D}_t$) for the new label; (iii) Updating $\mathcal{H}_t$ and $\mathcal{D}_t$ to become $\mathcal{H}_{t+1}$ and $\mathcal{D}_{t+1}$.

After making prediction $\hat{\boldsymbol{y}}_t$ for $\boldsymbol{x}_t$, the classifier and detector update process begins when both of the following conditions are satisfied:

1) When $\mathcal{D}_t(\boldsymbol{x}_t) = 1$, the newly arrived instance associated with the new label is added to buffer B.
2) $|B|$ reaches the preset maximum buffer size.

The classifier and the detector are updated using $\mathcal{T}_t = (X_t, \mathcal{H}_t(X_t))$ as shown in line 11 of Algorithm 1. This is the time when the known label set is expanded by including the new label $\ell' = \ell + 1$: $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} \cup \{\ell'\}$. Note that, when $0 < |B| <$ MAX_BUFFER_SIZE, even though the new label has been detected, the data is not sufficient to train/update a good performing classifier. In this situation, the output of the detector is applied as the prediction for the new label.

Algorithm 1 MuENL

```
Input: $X_0$, $Y_0$, $\{\boldsymbol{x}_t, t \in \{1, 2, \cdots, T\}\}$
Output: $\hat{\boldsymbol{y}}_t$ for each $\boldsymbol{x}_t$
 1: Train $\mathcal{H}_0$ with $X_0$ and $Y_0$; build detector $\mathcal{D}_0$ on $X_0$; set $t = 1$;
 2: Initialize sampling weight vector $\boldsymbol{s}_0 = \mathbf{1}_{|X_0|}$;
 3: $\mathcal{H}_1 = [\mathcal{H}_0, \mathcal{D}_0]$; $\mathcal{D}_1 = \mathcal{D}_0$;
 4: repeat
 5:     Receive a new instance $\boldsymbol{x}_t$, $X_t = [X_{t-1}; \boldsymbol{x}_t^\top]$;
 6:     Update sampling weight vector $\boldsymbol{s}_t = [\boldsymbol{s}_{t-1}; 1]$;
 7:     $\hat{\boldsymbol{y}}_t = \mathcal{H}_t(\boldsymbol{x}_t)$, where $\mathcal{H}_t = [h_{t,1}, h_{t,2}, \cdots, h_{t,\ell}, \mathcal{D}_t]$;
 8:     if $\mathcal{D}_t(\boldsymbol{x}_t) = 1$
 9:         Add $\boldsymbol{x}_t$ to $B$;
10:         if $|B|$ = MAX_BUFFER_SIZE
11:             Create $\mathcal{D}_{t+1}$ and $\mathcal{H}_{t+1}$ by using $\mathcal{T}_t = (X_t, \mathcal{H}_t(X_t))$;
12:             Empty $B$;
13:             The new label is converted to the known label:
                $\ell \leftarrow \ell + 1$;  $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} \cup \{\ell\}$;
14:             Update sampling weight vector $\boldsymbol{s}_t \leftarrow 0.8\boldsymbol{s}_t$;
15:             $X_t \leftarrow$ Select a subset of $X_t$ based on $\boldsymbol{s}_t$;
16:             Update $\boldsymbol{s}_t$ based on the updated $X_t$;
17:         end if
18:     end if
19:     $t \leftarrow t + 1$;
20:     $\boldsymbol{v}_t = \boldsymbol{v}_{t-1}$;  $\mathcal{D}_t = \mathcal{D}_{t-1}$;  $\mathcal{H}_t = \mathcal{H}_{t-1}$;
21: until $t = T$.
```

In order to reduce the storage and computational complexity, not all the historical data are stored in X_t, i.e., after the classifier and the detector have been updated for the new class, only a subset of X_t is stored for future processing. Specifically, in order to give preference to recent instances, we maintain a sampling weight for each instance in the trunk, i.e., $\boldsymbol{s}_t$, which is initialized as 1.0 when $\boldsymbol{x}_t$ is first observed (line 6 in Algorithm 1). Then, $\boldsymbol{s}_t$ is multiplied by a decay factor 0.8 after the classifier and the detector has been updated for the new class (line 14 in Algorithm 1) [3]. Finally, a subset of X_t is selected as follows: an instance in X_t is selected only if a randomly generated number is smaller than the instance's weight in $\boldsymbol{s}_t$ (line 15 in Algorithm 1).

Although we have mentioned only one new label in the definition, the algorithm admits multiple new labels to emerge in the same period. In this scenario, the multiple new labels can be treated as a single meta new label for the purpose of prediction of $\boldsymbol{x}_t$ and model update.

In the following three sections, we detail the design of the multi-label classifier, the detector, and their updates.

3.2.1 The Multi-Label Classifier (PLR)

We use a linear classifier $(\boldsymbol{w}_i, b_i)$ for each label i, i.e., $h_i(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}_i^\top \boldsymbol{x} + b_i)$. In addition to the ordinary misclassification loss minimization, we also minimize the pairwise label ranking loss in order to exploit label correlations to obtain a performance better than using either one of them. The resultant classifier is named Pairwise Label Ranking classifier or `PLR` for short.

The optimization process is an iterative process, conducted for each $\boldsymbol{w}_i$ for label i while having other $\boldsymbol{w}_j$ fixed ($j \neq i$), until it converges. Specifically, the optimization problem for each $\boldsymbol{w}_i$ is formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{w}_i, b_i, \xi, \zeta} \; & \frac{1}{2}\|\boldsymbol{w}_i\|^2 + C_1 \sum_{k=1}^{n} \xi_k + C_2 \sum_{j=1}^{\ell} \sum_{k=1}^{n} \zeta_{j,k}, \\ \text{s.t.} \quad & y_{i,k} f_{i,k} \geq 1 - \xi_k, \\ & \Delta_{j,k}(f_{i,k} - f_{j,k}) \geq 1 - \zeta_{j,k}, \\ & \xi_k \geq 0, \; \zeta_{j,k} \geq 0, \\ & j \in \{1, 2, \cdots, \ell\}, \; k \in \{1, 2, \cdots, n\}, \end{aligned} \tag{1}$$

[2] Then, ℓ will be automatically changed to ℓ'.

[3] This factor is set by experience, it is more robust to different datasets, compared with other settings.

where $\Delta_{j,k} = y_{i,k} - y_{j,k}$, $f_{i,k} = \boldsymbol{w}_i^\top \boldsymbol{x}_k + b_i$. and C_1, C_2 are two parameters to trade off.

To simplify the optimization process, $(\boldsymbol{w}_i, b_i)$ is converted to $\boldsymbol{w}_i$ by adding an attribute with value 1 at the end of x_k, i.e., $f_{i,k} = \boldsymbol{w}_i^\top[\boldsymbol{x}_k; 1]$. Eqn. (1) can be rewritten as

$$\begin{aligned}\min_{\boldsymbol{w}_i} & \sum_{j=1}^{\ell}\sum_{k=1}^{n}[1 - (y_{i,k} - y_{j,k})(f_{i,k} - f_{j,k})]_+ \\ & + \lambda_1 \sum_{k=1}^{n}[1 - y_{i,k} f_{i,k}]_+ + \frac{\lambda_2}{2}\|\boldsymbol{w}_i\|^2,\end{aligned} \tag{2}$$

where λ_1 and λ_2 are two trade-off parameters. To solve Eqn. (2), we first calculate the subgradient of the objective function; then apply the gradient descent method.

3.2.2 New Label Detection

The appearance of a new label may be due to a previously unseen set of feature values or a previously unseen pattern of known labels or both. Therefore, we take both feature and label spaces into account: If a new instance has different characteristics from known instances in the feature space, it is likely to hold a new label. Also, in the label space, if a rare co-occurrence of label pattern appears, a new label is likely to occur.

Using this idea, we propose `MuENLForest` as the detector $\mathcal{D}$. Similar to the earlier work [11], [20], we encode the label information into the feature space. Once the data is encoded, a detector for new labels is built based on an effective anomaly detection technique using random trees called `iForest` [16]. The main differences between `MuENLForest` and `iForest` are listed as follows:

- `iForest` considers only the feature space, whereas we encode the label information into the feature space in `MuENLForest`, considering both the feature difference and label relations. Note that the labels of all instances, arrived after the initial training set, are not available. The predicted value vectors are used for all these instances in training a `MuENLForest`.
- In each node of a tree in `iForest`, the test attribute is randomly selected; so is the cut-off value. In contrast, each node of a tree in `MuENLForest` is based on a fixed number of randomly selected attributes. The split is an outcome of a clustering process based on the selected attributes.
- When evaluating a test instance, `iForest` employs the average path length, that the test instance traverses over all trees, as the anomaly score. A small value suggests that the test instance is located in a sparse region, which is more likely to be an outlier. This does not work in the multi-label setting since instances with new labels may share the same dense region of instances with some common known labels. `MuENLForest` captures the characteristics in both the feature space and the label patterns. In addition to building the trees, a ball is constructed in each leaf node based on the training instances which fall into the leaf node. A test instance is predicted to have a new label if it falls outside the ball; otherwise, it has similar data characteristics and label patterns as the training instances used to build `MuENLForest`.

3.2.3 MuENLForest Construction

Recall that the training set at time $t > 0$ is $\mathcal{T}_t = (X_t, \mathcal{H}_t(X_t))$; and $\mathcal{T}_0 = (X_0, Y_0)$. `MuENLForest` consists of g `MuENLTrees`; and each `MuENLTree` is built using a random subset of $\mathcal{T}_t$ of size ψ using sampling weight $\boldsymbol{s}_t$. The definition of `MuENLTree` is given as follows.

Definition 2. *`MuENLTree` is a binary tree consists of internal nodes and leaf nodes. Let $\boldsymbol{a} = [\boldsymbol{x}, \mathcal{H}_t(\boldsymbol{x})]$ denote a training sample with predictive values. Each internal node has test: $\|\boldsymbol{a}^{\boldsymbol{q}} - \boldsymbol{p}_1\| \leq \|\boldsymbol{a}^{\boldsymbol{q}} - \boldsymbol{p}_2\|$ which splits into two son nodes, where $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are two cluster centers having $\boldsymbol{q}$ attributes and $\boldsymbol{a}^{\boldsymbol{q}}$ is the $\boldsymbol{q}$ projection of $\boldsymbol{a}$. Each leaf node defines a ball covering S (i.e., the set of all training instances falling into this leaf node) having radius $r = \max_{\boldsymbol{x}\in S}\|\boldsymbol{a} - \boldsymbol{m}\|$, where $\boldsymbol{m} = \mathrm{mean}(S)$.*

To grow a `MuENLTree` during the training process, the training set is recursively divided as internal nodes are constructed until any one of the following conditions ($\boldsymbol{C}$) is satisfied: (a) the tree reaches a height limit e_m; (b) $|S| = 1$; (c) all instances in S have the same $\boldsymbol{x}^{\boldsymbol{q}}$ value. Algorithm 1 summarizes the construction of `MuENLTree`.

Procedure 1 MuENLTree

Input: Training sample S, current tree height e, maximum tree height e_m, number of randomly selected attributes k

Output: MuENLTree

```
if any condition in $\boldsymbol{C}$ is satisfied:
    A ball[4] is built having radius $r = \max_{\boldsymbol{a}\in S}(\|\boldsymbol{x} - \boldsymbol{m}\|)$,
    where $\boldsymbol{m}$ = mean($S$);
    return $\boldsymbol{N}\{\boldsymbol{N}.S \leftarrow S, \boldsymbol{N}.\boldsymbol{m} \leftarrow \boldsymbol{m}, \boldsymbol{N}.r \leftarrow r\}$;
else
    Let $Q_1$ be the input attribute set in $S$;
    Let $Q_2$ be the predicted attribute set in $S$;
    Randomly select $k$ attributes $\boldsymbol{q}_1 \subset Q_1$;
    Randomly select $k$ attributes $\boldsymbol{q}_2 \subset Q_2$;
    $\boldsymbol{q} = [\boldsymbol{q}_1, \boldsymbol{q}_2]$;
    Cluster centers: $\{\boldsymbol{p}_1, \boldsymbol{p}_2\} \leftarrow$ Clustering($\boldsymbol{q}, S$);
    $S_l = \{\boldsymbol{a} \in S \mid \|\boldsymbol{a}^{\boldsymbol{q}} - \boldsymbol{p}_1\| \leq \|\boldsymbol{a}^{\boldsymbol{q}} - \boldsymbol{p}_2\|\}$;
    $S_r = \{\boldsymbol{a} \in S \mid \|\boldsymbol{a}^{\boldsymbol{q}} - \boldsymbol{p}_1\| > \|\boldsymbol{a}^{\boldsymbol{q}} - \boldsymbol{p}_2\|\}$;
    return $\boldsymbol{N}\{\boldsymbol{N}.S \leftarrow S, \boldsymbol{N}.\boldsymbol{m} \leftarrow \boldsymbol{m}, \boldsymbol{N}.r \leftarrow r,$
        $\boldsymbol{N}.\boldsymbol{q} \leftarrow \boldsymbol{q}, \boldsymbol{N}.\{\boldsymbol{p}_1, \boldsymbol{p}_2\} \leftarrow \{\boldsymbol{p}_1, \boldsymbol{p}_2\},$
        $\boldsymbol{N}.\boldsymbol{N}_{left} \leftarrow$ MuENLTree($S_l, e+1, e_m, k$)
        $\boldsymbol{N}.\boldsymbol{N}_{right} \leftarrow$ MuENLTree($S_r, e+1, e_m, k$)$\}$;
end if
```

For the construction of `MuENLForest`, we augment each instance with its predictive values, so that both the feature and label information can be considered. Projected on a set of randomly selected attributes, each internal node is split based on a cluster center on either branch. As a result, instances within the same leaf node must be similar on some attributes of either features or predictive values, or both.

3.2.4 MuENLForest Detection

Once `MuENLForest`, i.e., $\mathcal{D}_t(\cdot)$, is constructed, it is ready for prediction. In evaluating a test instance $\boldsymbol{x}_t$ in each `MuENLTree`, $\mathcal{D}_t(\boldsymbol{x}_t) = 1$, i.e., having a new label if $\boldsymbol{x}_t$ falls outside the ball. Otherwise, $\mathcal{D}_t(\boldsymbol{x}_t) = -1$, i.e., $\boldsymbol{x}_t$ has no new labels. The final output of `MuENLForest` is decided via majority voting.

The key idea is explained as follows. Recall that instances within the same leaf node must be similar on some attributes of either features or predictive values, or both. As a result,

[4] In the case that $|S| = 1$, the center and the radius of its parent node are used instead.

if a test instance falls into a certain leaf node but outside the ball. This suggests that this instance must be similar on some of the attributes to the other instances within that leaf node, but it is very different on the other attributes (features or predictive values). Thus, this instance may hold a new label with high probability.

3.2.5 Multi-Label Classifier Update

When buffer B is full, $\mathcal{H}_t$ is to be updated. This update includes the construction of a new classifier for the new label, and the update of the existing multi-label classifier. Because the detector is not perfect, it may miss some positive instances with the new label or mistake some negative ones to be positive. We aim to produce a robust classifier which can tolerate this kind of errors to some extent.

The basic idea is to introduce a latent variable which estimates the true label assignment of each instance in $\mathcal{X}_t$, where a predicted label by the detector is the initial value of the latent variable. Then the optimization learns this latent label assignment and the classifier simultaneously that best fits the data. In this way, the learned classifier is more tolerant to the errors of the detector.

Let X_B be the instances collected in the buffer and X_U be the set of instances with (predicted) known labels only, where $X_U = X_t \setminus X_B$. Let $\boldsymbol{d} = [d_1, d_2, \cdots, d_m]^\top$ be the potential assignment of the new label of $X_t = [X_B; X_U]$, where m is the number of instances in $[X_B; X_U]$; and $d_k = 1$ if $\boldsymbol{x}_k \in [X_B; X_U]$ holds a new label; $d_k = 0$ otherwise. Note that $\boldsymbol{d}$ is initialized with the labels predicted by the detector.

In order to obtain a robust classifier, we propose Multi-label learning with New Labels (MNL) to simultaneously learn the $\boldsymbol{d}$ and classifier $\boldsymbol{w}_a$ for the new label $\ell \leftarrow \ell+1$. This is done by modifying Eqn. (2) in Section 3.2.1 i.e., replacing $y_{i,k}$ with $2d_k - 1$, that optimizes both the pairwise label ranking loss (the first term, to encourage positive labels to be ranked before negative labels), the misclassification loss (the second term, to encourage that labels are correctly predicted), and the regularizers (the last two terms). The optimization problem of building classifier $\boldsymbol{w}_a$ and learning $\boldsymbol{d}$ for the new label ℓ is cast as follows:

$$\begin{aligned}\min_{\boldsymbol{w}_a,\boldsymbol{d}} &\sum_{j=1}^{\ell}\sum_{k=1}^{m}[1-(2d_k-1-y_{j,k})(f_{\ell,k}-f_{j,k})]_+ \\ &+\lambda_1\sum_{k=1}^{m}[1-(2d_k-1)f_{\ell,k}]_+ +\frac{\lambda_2}{2}\|\boldsymbol{w}_a\|^2+\frac{\lambda_3}{2}\|\boldsymbol{d}\|^2, \\ &\textbf{s.t. } d_k\in\{0,1\}, k\in\{1,2,\cdots,m\},\end{aligned} \tag{3}$$

where $y_{j,k} = h_{t,j}(\boldsymbol{x}_k)$ and $\lambda_1, \lambda_2, \lambda_3$ are three parameters.

The above optimization problem is a NP-Hard problem. To simplify the problem, we relax the constraint from $d_k \in \{0, 1\}$ to $d_k \in [0, 1]$. The process optimizes $\boldsymbol{d}$ and $\boldsymbol{w}_a$ alternatively.

Specifically, when fixing $\boldsymbol{w}_a$ and updating $\boldsymbol{d}$, it solves the following subproblem:

$$\begin{aligned}\min_{\boldsymbol{d}} &\sum_{j=1}^{\ell}\sum_{k=1}^{m}[1-(2d_k-1-y_{j,k})(f_{\ell,k}-f_{j,k})]_+ \\ &+\lambda_1\sum_{k=1}^{m}[1-(2d_k-1)f_{\ell,k}]_+ +\frac{\lambda_3}{2}\|\boldsymbol{d}\|^2, \\ &\textbf{s.t. } d_k\in[0,1], k\in\{1,2,\cdots,m\}.\end{aligned} \tag{4}$$

Procedure 2 MNL

```
Input: $X_B$, $X_U$, $\boldsymbol{w}_i, i \in \{1, 2, \cdots, \ell\}$, $\boldsymbol{d}_{init}$, $\boldsymbol{w}_{a,init}$
Output: $\boldsymbol{w}_a$
Initialize $\boldsymbol{d}_0 \leftarrow \boldsymbol{d}_{init}$; $\boldsymbol{w}_{a,0} \leftarrow \boldsymbol{w}_{a,init}$;
$t = 1$;
repeat:
    $\boldsymbol{d}_t \leftarrow$ solve Eqn. (4) with a warm start using $\boldsymbol{d}_{t-1}$;
    $\boldsymbol{w}_{a,t} \leftarrow$ solve Eqn. (5) with a warm start using $\boldsymbol{w}_{a,t-1}$;
    $t \leftarrow t + 1$;
until converge or the maximum number of iterations is reached;
$\boldsymbol{w}_a = \boldsymbol{w}_{a,t}$.
```

Then, when updating $\boldsymbol{w}_a$ with $\boldsymbol{d}$ fixed, it solves the following subproblem:

$$\begin{aligned}\min_{\boldsymbol{w}_a} &\sum_{j=1}^{\ell}\sum_{k=1}^{m}[1-(2d_k-1-y_{j,k})(f_{\ell,k}-f_{j,k})]_+ \\ &+\lambda_1\sum_{k=1}^{m}[1-(2d_k-1)f_{\ell,k}]_+ +\frac{\lambda_2}{2}\|\boldsymbol{w}_a\|^2,\end{aligned} \tag{5}$$

To solve Eqn. (4) and (5), we calculate the subgradient of the objective function and perform the gradient descent. After each update, we project $\boldsymbol{d}$ to $[0, 1]$: $\boldsymbol{d} \leftarrow \min(\boldsymbol{1}, [\boldsymbol{d}]_+)$, to satisfy the box constraint.

In order to achieve a faster convergence and a good result, we adopt a warm start strategy. For the first iteration, we employ X_B as positive instances and $[X_0; X_U]$ as negative instances to train a linear multi-label classifier. For each subsequent iteration, the optimization begins with the optimization result obtained in the last iteration.

Procedure 2 summarizes the procedure of building a classifier for MNL.

When an instance with a new label appears, the influence of the new label shall be taken into consideration in updating the existing classifier. To achieve this goal, we slightly adjust each existing classifier $\boldsymbol{w}_i, i \in \{1, \cdots, \ell-1\}$ via minimizing the ranking loss, and penalizing a large change of classifier $\boldsymbol{w}_i$ as follows:

$$\min_{\boldsymbol{w}'_i}\sum_{j=1}^{\ell}\sum_{k=1}^{n}[1-(y_{i,k}-y_{j,k})(f'_{i,k}-f_{j,k})]_+ + \|\boldsymbol{w}'_i-\boldsymbol{w}_i\|_2^2.$$

When $j = l_{mx}$, we use $\boldsymbol{d}$ as the new label vector. Note that this optimization does not affect the order of predicted values on all labels for each instance, since the relative ranking between any two known labels is not changed. Besides, a large change in the classifier is prevented via minimizing $\|\boldsymbol{w}'_i - \boldsymbol{w}_i\|_2^2$, since we wish to slightly adjust the existing classifier and avoid a sudden change. In a nutshell, the formulation considers the influence of new class model and it is also robust to errors made by the new label classifier to update known label classifiers.

3.2.6 Detector Update

To update the detector, we simply rebuild the detector based on X_t as follows: Each of g MuENLTrees is built using ψ instances selected from X_t based on sampling weight $\boldsymbol{s_t}$. This produces $\mathcal{D}_{t+1}$ to replace $\mathcal{D}_t$.

4 The MuENLHD Approach

In some real applications, streaming data are high-dimensional and in sparse representations. Micro-blogs for example [9], if bag-of-words features are applied, there will be over $10,000$ features and each instance has sparse representation because a few out of the over $10,000$ features are related to a specific topic.

When we need to detect the appearance of a new topic from a sparse high-dimensional data stream, the previous tree-based detector is not suitable. Recall that in MuENLTree, we randomly select some attributes (including features and predicted values) to split the node into two son nodes. However, when the data is high dimensional and sparse, attributes with 0 value are selected with high probability. As a result, even instances belonging to different topics may be deemed similar because they both hold 0 on the selected attributes. Thus the detection accuracy is poor. Neither the use of non-zero attributes only nor building a tree which includes all possible attribute combinations is a feasible solution. Globally, most attributes have non-zero values in at least some instances in the training set, even though individual instances have few non-zero attributes. As the number of attribute combinations is exponentially large w.r.t. the number of attributes, both the time complexity and tree size are prohibitive in practice.

A practical way to handle the novel label detection problem in sparse high-dimensional data is to do a dimension reduction first. Then, the MuENL approach can be applied to the data with reduced dimensions.

Streaming kernel principal component analysis approach or SKPCA [8] is a state-of-the-art non-linear dimension reduction approach on data streams, with high efficiency and strong theoretical guarantee. Specifically, it first generates random Fourier features which are equivalent to non-linear kernel mappings. Then, a frequent direction approach is applied in order to keep the most informative direction on the spectral space via a truncated singular value decomposition (SVD), i.e., only half of the largest eigenvalues are kept.

Note that SKPCA is for dimension reduction only. In this work, we incorporate SKPCA into MuENL to handle high dimensional data stream with emerging new labels. SKPCA consists of two components:

- The random feature map (Procedure 3) is defined as: $\boldsymbol{\phi} = [\phi_1, \cdots, \phi_m]$, where $\phi_i(\boldsymbol{a}) = \cos(r_i \boldsymbol{a} + \gamma_i), i \in \{1, 2, \cdots, m\}$, $r_i \sim \mathcal{N}(0, 2sI)$, $\gamma_i \sim \mathcal{U}(0, 2\pi)$, s is a parameter[5] and I is the identity matrix. With this random feature mapping, sparse high dimensional sparse data is transformed into dense m dimensional representations. However, the size m of $\boldsymbol{\phi}$ is usually chosen to be hundreds or thousands in order to achieve a good performance. $m = 2000$ and $s = 1$ are fixed in our experiments.
- The Frequent Direction (FD) update rule performs further dimension reduction in a data stream (Procedure 4), where it converts the m dimensional space to a d' dimensional space. The lower dimension d' is usually set much smaller than m. We use $d' = 100$ in the experiments.

Incorporating these two components with MuENL, we have MuENLHD summarized in Algorithm 5. Specifically, the output of RandomFeatureMap is a set of mapping functions, each is a random Frontier transformation. This is produced before the beginning of the stream. It needs to be done once only; and the mapping functions are used for the rest of the stream.

Procedure 3 RandomFeatureMap

```
Input: $s$, $m$
Output: mapping function $\boldsymbol{\phi} = [\phi_1, \cdots, \phi_m]$
for each $i \in \{1, \cdots, m\}$
    sample $r_i \sim \mathcal{N}(0, 2sI)$, where $\mathcal{N}$ represents normal distribution ;
    sample $\gamma_i \sim \mathcal{U}(0, 2\pi)$, where $\mathcal{U}$ represents uniform distribution;
    $\phi_i = \cos(r_i \boldsymbol{a} + \gamma_i)$, where $\boldsymbol{a}$ is the input variable for $\boldsymbol{\phi}$;
end
```

Procedure 4 FDUpdate

```
Input: Matrix $P$, low dimensionality $d'$
Output: $P$ and $W$
$[\,\cdot, \Sigma, W] = \mathrm{SVD}(P)$, where $\Sigma = \mathrm{diag}(\{\sigma_i|, i \leq i \leq d'\})$;[6]
$\mathrm{mid} = \mathrm{Round}(d'/2)$;
$\tau = \Sigma_{\mathrm{mid},\mathrm{mid}}$;
$P \leftarrow \sqrt{\max(0, \Sigma^2 - \tau^2)} W$.
```

The output of FDUpdate is an updated matrix P and a linear mapping W. According to SKPCA [8], P is a good spectral approximation of the data stream with a bounded error, and W is the corresponding mapping. In other words, P gradually captures the main information of the data stream in the spectral space.

The detector and the classifier in MuENLHD are built with the new mapped space, from the output of FDUpdate. In order to reduce the complexity, we only update the mapping when buffer B is full. The classifier and the detector are updated using B.

5 Time complexity

In this section, the time complexity of each component of MuENL and MuENLHD is presented. The common components of MuENL and MuENLHD are the multi-label classifier (PLR), the detector MuENLForest, and update the classifier (MNL) and the detector (which is equivalent to building a detector). For MuENLHD, there are two additional components RandomFeatureMap and FDUpdate.

For the multi-label classifier (PLR) and update (MNL), a pairwise label ranking optimization is involved which has time complexity $O(|\boldsymbol{v}|^2 nd)$, where $|\boldsymbol{v}|$ is the size of the class label set, n is the number of instances, and d is the number of dimensions. For the detector construction, each node involves k-means clustering with 2 clusters which have time complexity $O(|\boldsymbol{q}|n)$. The MuENLTree construction involves ψ instances having tree height limit e_m. Thus, the time complexity is $O(|\boldsymbol{q}|e_m\psi)$. As a result, the time complexity of MuENLForest with g MuENLTrees is $O(|\boldsymbol{q}|ge_m\psi)$. The detector update simply rebuilds the detector, which will be $O(|\boldsymbol{q}|e_m\psi)$. RandomFeatureMap and FDUpdate have $O(mdn)$ and $O(d'mn)$ respectively, where m is the number of random features, and d' is the number of features at the output of FDUpdate. The time complexity of each component is summarized in Table 1.

Note that, even though there are two additional components for MuENLHD, MuENLHD is faster than MuENL when handling high dimensional data in practice. The reasons are: (a) In MuENLHD, much lower dimensional data are involved in PLR, MuENLForest and MNL, which will contribute to

[5] Note that s is equivalent to the parameter in a rbf kernel because $\boldsymbol{\phi}(\boldsymbol{a}_i)^\top \boldsymbol{\phi}(\boldsymbol{a}_j)$ is a good approximate to Gaussian kernel $K(\boldsymbol{a}_i, \boldsymbol{a}_j) = \exp(-s\|\boldsymbol{a}_i - \boldsymbol{a}_j\|^2)$.

[6] The number of rows of P should be larger or equal to d'. We assume that singular value decomposition SVD returns singular values in a descending order (σ_i denotes the i-th largest singular value).

Table 1
Time complexity of each component of MuENL and MuENLHD

	PLR	MNL	MuENLForest	RandomFeatureMap	FDUpdate
MuENL	$O(\|\boldsymbol{v}\|^2nd)$	$O(\|\boldsymbol{v}\|^2nd)$	$O(\|\boldsymbol{q}\|ge_m\psi)$	–	–
MuENLHD	$O(\|\boldsymbol{v}\|^2nd')$	$O(\|\boldsymbol{v}\|^2nd')$	$O(\|\boldsymbol{q}\|ge_m\psi)$	$O(mdn)$	$O(d'mn)$

Algorithm 5 MuENLHD

```
Input: $X_0$, $Y_0$, $\{\boldsymbol{x}_t, t \in \{1, 2, \cdots, T\}\}$, low dimension $d'$
Output: $\hat{\boldsymbol{y}}_t$ for each $\boldsymbol{x}_t$
Generate a random feature map
  $\boldsymbol{\phi} = \mathrm{RandomFeatureMap}(s, m)$;
Initialize $P, W : [P, W] = \mathrm{FDUpdate}(\sqrt{\frac{2}{m}}\boldsymbol{\phi}(X_0), d')$;
Initialize sampling weight vector $\boldsymbol{s}_0 = \mathbf{1}_{|X_0|}$;
Initialize $B = \emptyset$;
Train $\mathcal{H}_0$ with $Z_0 = \sqrt{\frac{2}{m}}\boldsymbol{\phi}(X_0)W$ and $Y_0$;
  build detector $\mathcal{D}_0$ on $Z_0$;
$\mathcal{H}_1 = [\mathcal{H}_0, \mathcal{D}_0]$; $\mathcal{D}_1 = \mathcal{D}_0$; $W_p = W$; $t = 1$;
repeat
    Receive a new instance $\boldsymbol{x}_t$, obtain its low dimensional
      representation $\boldsymbol{z}_t = \sqrt{\frac{2}{m}}\boldsymbol{\phi}(\boldsymbol{x}_t)W$; $X_t = [X_{t-1}; \boldsymbol{x}_t^\top]$;
    Update sampling weight vector $\boldsymbol{s}_t = [\boldsymbol{s}_{t-1}; 1]$;
    Insert $\boldsymbol{z}_t$ into zero valued rows in $P$;
    if $P$ has no rows having zero value.
        $[P, W_p] = \mathrm{FDUpdate}(P, d')$;
    end if
    $\hat{\boldsymbol{y}}_t = \mathcal{H}_t(\boldsymbol{z}_t)$, where $\mathcal{H}_t = [h_{t,1}, h_{t,2}, \cdots, h_{t,|\boldsymbol{v}_t|}, \mathcal{D}_t]$;
    if $\mathcal{D}_t(\boldsymbol{z}_t) = 1$
        Add $\boldsymbol{z}_t$ to $B$;
        if $|B|$ = MAX_BUFFER_SIZE
            $W \leftarrow W_p$; $Z_t = \sqrt{\frac{2}{m}}\boldsymbol{\phi}(X_t)W$;
            Update $\mathcal{D}_{t+1}$ and $\mathcal{H}_{t+1}$ using $\mathcal{T}_t = (Z_t, \mathcal{H}_t(Z_t))$;
            Empty $B$;
            The new label is converted to the known label:
              $\ell \leftarrow \ell + 1$;  $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} \cup \{\ell\}$;
            Update sampling weight vector $\boldsymbol{s}_t \leftarrow 0.8\boldsymbol{s}_t$;
            Update $X_t$ and $\boldsymbol{s}_t$ via keep or discard instances
              according to $\boldsymbol{s}_t$;
        end if
    end if
    $t \leftarrow t + 1$;
    $\boldsymbol{v}_t = \boldsymbol{v}_{t-1}$;  $\mathcal{D}_t = \mathcal{D}_{t-1}$;  $\mathcal{H}_t = \mathcal{H}_{t-1}$;
until $t = T$.
```

much shorter running time than MuENL; (b) the time complexities of RandomFeature and FD are much smaller than those of PLR and MNL; (c) with a much smaller number of dimensions, the best $|\boldsymbol{q}|$ selected for MuENLForest in MuENLHD is much smaller than that for MuENL.

6 Experiments on Low dimensional data

6.1 Configuration

To evaluate the predictive performance of the MuENL approach, we use 5 multi-label benchmark datasets (*"Birds", "CAL500", "Emotions", "Enron" and "Yeast"*, detailed information is shown in Table. 2)[7]. A data stream is simulated by generating the initial set of labeled data (i.e., (X_0, Y_0)) and individual unlabeled instances $\boldsymbol{x}_t$ arrive progressively in time step $t \in \{1, 2, \ldots, T\}$.

Figure 1 shows an example simulation of a data stream using the Yeast dataset, where 5 new labels (A to E) are observed in different time periods. At t_0, only the initial training set with known labels ($\boldsymbol{v}_0$) is observed; and a multi-label classifier is trained using this training set. Then, instances with possibly new label A begins to appear in the $t_0 - t_2$ period. At t_1, which denotes that the buffer (of instances having detected with the new label) is full, MuENL updates the multi-label classifier ($[h_{t,1}, \cdots, h_{t,\ell}]$) for the expanded known label set $\{\boldsymbol{v}_0, A\}$; and also updates the detector ($\mathcal{D}_t$) so that it can detect the next new label.

Table 2
Characteristics of datasets used. # inst is the number of instance, # dim is the dimension, #label is the total number of class labels, and #cardinarity is the average number of labels for each instance

Dataset	# inst.	# dim.	#labels	# cardinality
Birds	645	260	19	1.014
CAL500	502	68	174	26.044
Emotions	593	72	6	3.378
Enron	1702	1001	53	3.378
Yeast	2417	103	14	4.237
Weibo	6000	15461	10	1.593

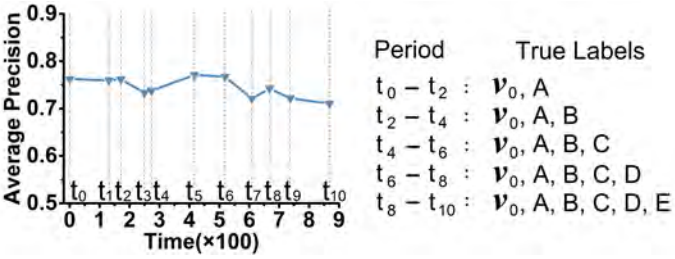


Figure 1. Performance on Yeast dataset. 5 new labels are involved.

The updated classifier and detector are then used for prediction at $t_1 + 1$, which is the time that A becomes part of the known labels. At t_2, new label B begins to appear, and the same process in the $t_0 - t_2$ period repeats in the $t_2 - t_4$ period; and the subsequent periods up to $t_8 - t_{10}$ for label E. The classifier and detector used for prediction in each period are summarized in Table 3. Note that for detection, we always use MuENLForest; for prediction, when the classifier has not been built for the new label, we use the output of the detector MuENLForest as the prediction result; after MNL has been updated for the new label, we use MNL for prediction.

It is interesting to note that the overall predictive performance does not degenerate much with the successive appearance of new labels, as shown in Figure 1, after about 900 time steps with 5 consecutive new labels.

Table 3
Models used for predicting new labels in the simulated time periods.

Time	Model for predicting new labels	True labels
$t_0 - t_1$	MuENLForest	$\boldsymbol{v}_0$, A
$t_1 - t_2$	MNL	$\boldsymbol{v}_0$, A
$t_2 - t_3$	MuENLForest	$\boldsymbol{v}_0$, A, B
$t_3 - t_4$	MNL	$\boldsymbol{v}_0$, A, B
$t_4 - t_5$	MuENLForest	$\boldsymbol{v}_0$, A, B, C
$t_5 - t_6$	MNL	$\boldsymbol{v}_0$, A, B, C
...	...	...

For a given dataset, the above simulation is conducted as follows. The label set $\boldsymbol{v}$ in a given dataset is split into two subsets, i.e., $\boldsymbol{v}_N = \{\boldsymbol{v}_{a1}, \ldots, \boldsymbol{v}_{a5}\}$ the candidate new label set of 5 new labels, and $\boldsymbol{v}_K$ the known label set. Let P_K be the data subset without $\boldsymbol{v}_N$; and P_N be the data subset

[7] http://mulan.sourceforge.net/datasets.html

with $\boldsymbol{v}_N$. Note that we need to do some pre-processing of the given dataset: (1) if two labels are highly correlated, i.e., often co-occur, they are combined by taking a union of the two labels, since it is impossible to distinguish these labels without any prior knowledge. (2) If a label is independent of other labels, i.e., seldom co-occur with other labels, it is not as a new label, since this kind of label will degenerate to a new class detection in class incremental learning, which is an easier task; (3) those moderately correlated labels (around median in the ranked list) are potential new labels. These labels sometimes co-occur with the same label and sometimes with different labels [8].

We sample 90% of P_K to form the initial training set $\mathcal{T}_0 = (X_0, Y_0)$. For $t > 0$, $\boldsymbol{x}_t$ is randomly selected uniformly (without replacement) from $P_K \setminus \mathcal{T}_0$ and a subset of P_N having label $\boldsymbol{v}_{ai}$ for time period $t_{2i-2} - t_{2i}$ where $i = 1, \ldots, 5$.

For evaluation, we consider a commonly applied metric for multi-label learning, i.e., Average Precision. It is the average fraction of positive labels ranked higher than a particular positive label. Let p denote the number of test instances; $\boldsymbol{C}_i^+$ and $\boldsymbol{C}_i^-$ be the sets of positive and negative labels, respectively, associated with instance $\boldsymbol{x}_i$. The set of positive predicted labels which are ranked lower than label c for $\boldsymbol{x}_i$ is defined as: $\hat{\boldsymbol{Q}}_{i,c} = \{j \mid \text{rank}(\boldsymbol{x}_i, j) \leq \text{rank}(\boldsymbol{x}_i, c), j \in \boldsymbol{C}_i^+\}$, where $\text{rank}(\boldsymbol{x}_i, c)$ is the rank of label c in the predicted label ranking (sorted in descending order). Then, Average Precision $= \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|\boldsymbol{C}_i^+|}\sum_{c\in\boldsymbol{C}_i^+}\frac{|\hat{\boldsymbol{Q}}_{i,c}|}{\text{rank}(\boldsymbol{x}_i,c)}$. The larger the Average Precision, the better.

The experiment is conducted based on the simulation described above. The predictive performance is measured at $t_1, \ldots, t_6$; and two kinds of evaluations are conducted: (i) $\boldsymbol{v}_t$-evaluation: it measures the performance with the emergence of new labels in the simulation; (ii) $\boldsymbol{v}_0$-evaluation: it assesses how well MuENL performs on the initial label set $\boldsymbol{v}_0$ only throughout the entire period, i.e., the performance on the emerging new labels are not assessed.

For each dataset, the average result and standard deviation of 10 independent runs of simulations are reported.

The performances of two algorithms are said to have a significant difference if the difference is more than two standard errors.

6.2 Baselines

We employ two sets of baselines to compare with MuENL: (i) the state-of-the-art multi-label approaches; (ii) variants of MuENL having different MuENL components.

(a) State-of-the-art multi-label approaches

We compare MuENL with BR [3], CLR [7], ECC [20], PLR, LIMO [28], and GenEML [12], which are multi-label learning approaches considering the initially known labels only. BR, CLR, and ECC are first-order, second-order and high-order multi-label learning approaches, respectively. PLR is the multi-label approach proposed in Section 3.2.1. It is a degenerated version of MuENL without a detector and model update. LIMO and GenEML are the most recently proposed approaches to handle multi-label learning: LIMO considers both the instance-wise and label-wise margin and GenEML is a generative model. Further details are listed as follows:

- BR trains a linear classifier for each label independently.
- CLR transforms the multi-label learning problem into the label ranking problem and incorporates a virtual label to separate the relevant and irrelevant labels.
- ECC is the ensemble of classifier chains. In each chain, ground truth labels are encoded into the feature space gradually; thus high order label relations are exploited.
- PLR takes advantage of pairwise label ranking.
- LIMO maximizes both the label-wise margin and instance-wise margin.
- GenEML is a flexible and scalable generative model based on a latent factor model for the label matrix.

(b) Variants of MuENL

To further validate the effectiveness of each component, we compare MuENL with its variants.

- MuENL_{IF}: Use iForest [16] (instead of MuENLForest) as the detector.
- MuENL_{OC}: Use OC-SVM [23] (instead of MuENLForest) as the detector.
- $\text{MuENL}_{\text{SVM}}$: Use MuENLForest as the detector, but use a different classifier: train a linear classifier for the new label via SVM by using the same training set as used in MNL.
- MuENL_{OR}: Use MuENLForest as the detector and assume that an oracle is accessible to provide the ground truth for model updates. Its performance will be the upper bound.

These variants are summarized in Table 4.

Table 4
MuENL variants

Approach	Classifier	Detector	Classifier Update
$\text{MuENL}_{\text{SVM}}$	PLR	MuENLForest	SVM
MuENL_{IF}	PLR	iForest	MNL
MuENL_{OC}	PLR	OC-SVM	MNL
MuENL_{OR}	PLR	MuENLForest	Oracle+PLR
MuENL	PLR	MuENLForest	MNL

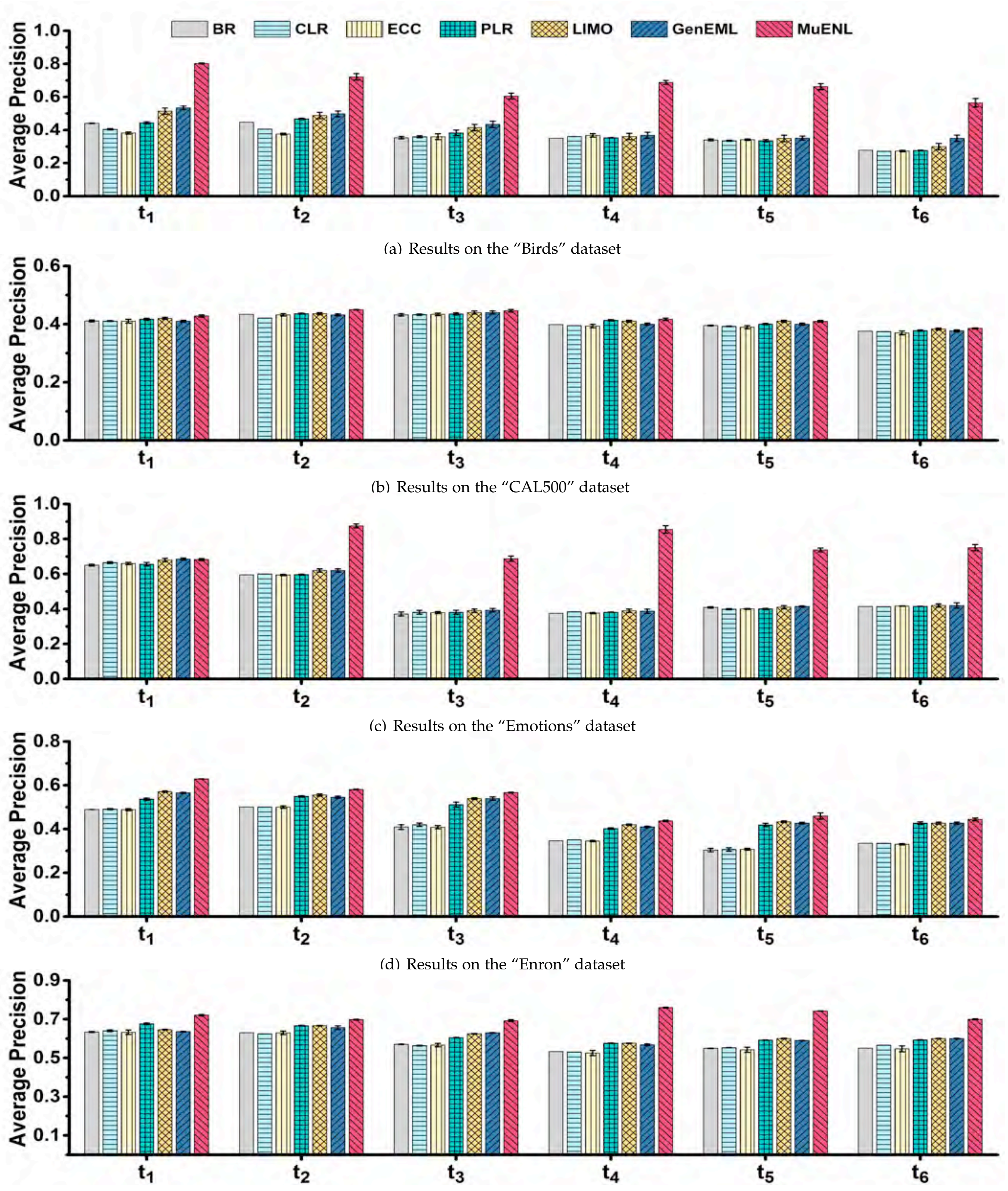
The following codes are used to implement MuENL: LIBLINEAR toolbox [5] is applied as the linear SVM; MANOPT toolbox [2] is utilized to implement the steepest descent with a line search procedure to update the multi-label classifier. kmeans with $k = 2$ is applied to implement the splitting criterion in MuENLTree which is a binary tree.

For each method under comparison, its parameters are tuned using the initial training set at t_0 via 5-fold cross-validation. Then, these settings are employed for the rest of the data stream.

To select the appropriate settings for the detector, a label randomly selected from Y_0 is used for validation, and the rest of the labels are used to train a detector. This process is repeated 5 times for a different randomly selected label in Y_0 in order to choose the appropriate parameter settings for the detector. The ranges of values used are: buffer size $|B| \in \{10, 15, 30, 60\}$, maximum tree height $e_m \in \{-2, -1, 0, 1\} + \text{ceil}(log_2\psi)$ (where $\text{ceil}(log_2\psi)$ is the average height of the binary tree with sample size ψ), number of selected attributes $|\boldsymbol{q}| \in \{1, 3, 5\}$. The other parameters are fixed by default: tree number $g = 100$, sample size $\psi = 256$ and $\lambda_3 = 1$.

Detail ranges (or values) of the parameters of MuENL and the baselines are summarized in Table 5.

[8] The correlation between two labels is measured based on cosine similarity. In order to rank label correlation among labels, we use the sum of cosine similarities of pair-wise labels. A, B, C three label vectors for example, the score for A is $\cos(A, B) + \cos(A, C)$.



(a) Results on the "Birds" dataset

(b) Results on the "CAL500" dataset

(c) Results on the "Emotions" dataset

(d) Results on the "Enron" dataset

(e) Results on the "Yeast" dataset

Figure 2. Compare `BR`, `CLR`, `ECC`, `PLR` and `MuENL`: Results using $\boldsymbol{v}_t$-evaluation. $t_1 - t_6$ on the horizontal axis are the 6 time points.

Table 5
Parameters used in **MuENL** and its baselines

Parameter	Model	Approach
$\lvert B\rvert \in \{10, 15, 30, 60\}$	—	**MuENL** MuENL variants
$\lambda_1, \lambda_2 \in \{0.001, 0.01, 0.1, 1\}$	PLR	**MuENL** MuENL$_{\text{IF}}$ MuENL$_{\text{OC}}$ MuENL$_{\text{OR}}$ PLR & MNL
$\lvert\boldsymbol{q}\rvert \in \{1, 3, 5\}$ $g = 100$ $\psi = 256$ $e_m \in \{6, 7, 8, 9\}$	MuENLForest	**MuENL** MuENL$_{\text{SVM}}$
$\lambda_3 = 1$	MNL	**MuENL** MuENL$_{\text{OC}}$ MuENL$_{\text{IF}}$
$C \in \{0.1, 1, 10, 100\}$	SVM	MuENL$_{\text{SVM}}$ BR, CLR, ECC
$g = 100$ $\psi = 256$ $e_m \in \{6, 7, 8, 9\}$	iForest	MuENL$_{\text{IF}}$
$nu \in \{0.3, 0.5, 0.7\}$	OC-SVM	MuENL$_{\text{OC}}$

6.3 Experimental Results

We conduct two sets of experiments which differ in the number of new labels which may appear in each time period. The first set has exactly one new label; whereas the second set has more than one new label. The second set assesses how multiple new labels in each time period may affect the predictive performance.

6.3.1 Results on one new label per time period

(a) Compare with existing Multi-label approaches

Figure 1 is an example plot of $\boldsymbol{v}_t$-evaluation on the Yeast dataset having one new label in each time period over 5 periods[9]. In general, MuENL maintains a good predictive performance spanning the entire duration from t_0 to t_{10}. This is a direct result of a good detector and a robust classifier in MuENL.

Figure 2 summarizes the comparison with BR, ECC, CLR, PLR, LIMO, and GenEML in terms of $\boldsymbol{v}_t$-evaluation in five datasets. It is interesting to note that MuENL almost always performs better than all baselines. Many of the differences are significant.

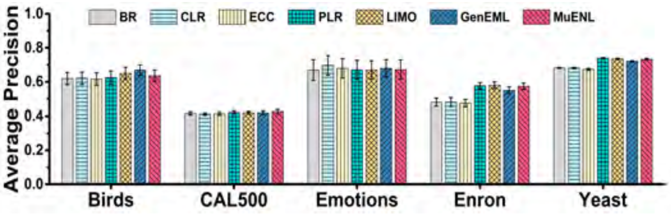


Figure 3. Compare BR, CLR, ECC, PLR, LIMO, GenEML, and MuENL: Results using $\boldsymbol{v}_0$-evaluation.

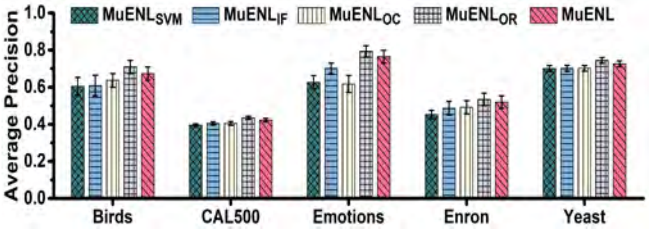


Figure 4. Compare MuENL variants: Results using $\boldsymbol{v}_t$-evaluation.

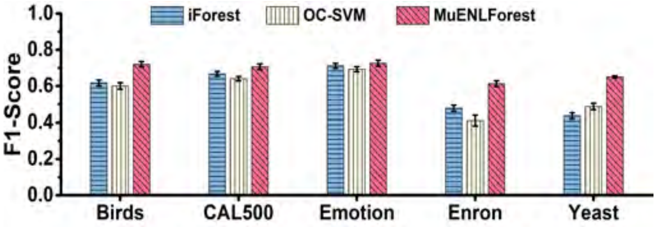


Figure 5. Detection performance of MuENLForest, OC-SVM and iForest (i.e., detectors in three MuENL variants) at t_1.

The summarised result on five datasets for $\boldsymbol{v}_0$-evaluation is provided in Figure 3 in terms of average precision. MuENL achieves better or comparable performance in comparison with all baselines. This shows that MuENL, which incorporates detection and prediction of new labels, not only does no harm to the performance on known labels, but can also gain better performance because pairwise label ranking is considered.

(b) Compare with MuENL variants

Figure 4 presents the result based on average precision.

Among the four MuENL variants, MuENL$_{\text{OR}}$ has the best performance since it uses the oracle which provides the ground truth that is not available to the other variants. Note that this oracle is available in practice and it is used to show the upper bound.

MuENL obtains a performance comparable with MuENL$_{\text{OR}}$ in all cases, where there is no significant difference in performance. This is a direct result of a good detector and a robust classifier for both known and new labels.

MuENL achieves a significantly better performance than MuENL$_{\text{SVM}}$ on all datasets. This shows that the robust classifier update procedure in MuENL works better than that in SVM; and it is essential in maintaining a good performance in a dynamic learning environment.

MuENL$_{\text{IF}}$ and MuENL$_{\text{OC}}$ replace the detector in MuENL with iForest and OC-SVM, respectively. Even though a robust update procedure is applied in both of them, MuENL still performs better than them (on 4 of 5 datasets, MuENL is significantly better). This validates the effectiveness of the detector in MuENL. Because new labels often co-occur with known labels, this kind of occurrences confuses existing detectors OC-SVM or iForest—leading to low detection accuracy. By taking both the feature and label information into account, MuENLForest is able to differentiate instances with a combination of features and label patterns due to new labels from that due to existing labels. This yields a better detection outcome as a result.

A more direct analysis of the above comparison is provided in Figure 5 which shows detection performance of MuENLForest, OC-SVM and iForest in terms of the F1-score evaluated at t_1. As expected, MuENLForest outperforms the other competitors.

For the classifier of new labels, we make a more direct comparison of the proposed MNL with SVM and PLR which treat the instances in the buffer as positive ones and other

[9] Notice that the time when the buffer gets full may be different in different time periods, and for different detectors in the same time period. Thus, the duration of one period may vary from one period to another and from one detector to another.

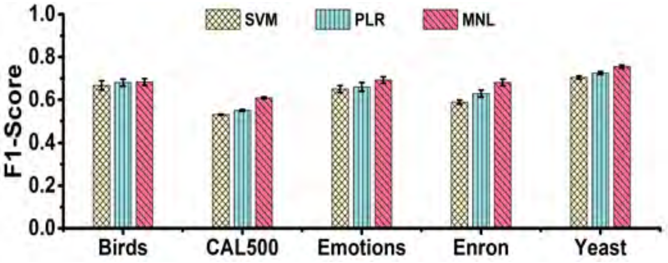


Figure 6. Classifier performance of `MNL`, `SVM` and `PLR` (i.e., classifiers in three `MuENL` variants) at t_2.

seen instances as negative ones in order to train a classifier. Figure 6 shows the performance evaluated at t_2 (In this experiment, the same detector `MuENLForest` is applied). Figure 6 shows that `MNL` is better than `SVM` and `PLR` because it employs a robust update, which works well even under the condition that a detector is imperfect.

To examine the significance of the relative performance among these algorithms, we perform a post-hoc Nemenyi test [10]. The result shown in Figure 7 reveals that the proposed `MuENL` is significantly better than all the traditional multi-label approaches (which do not consider any new labels) and other `MuENL` variants, except `MuENL`$_{\text{OR}}$ in which the oracle is available.

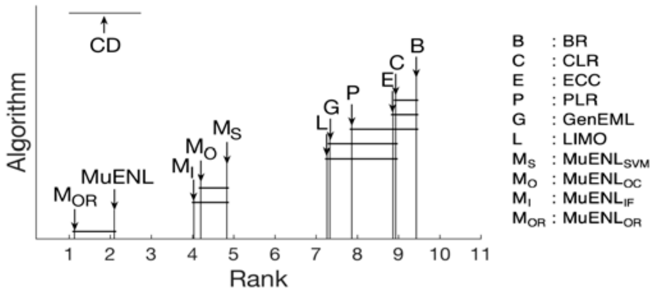


Figure 7. Critical difference (CD) diagram of the post-hoc Nemenyi test ($\alpha = 0.05$) for the comparison results with both the traditional multi-label learning approaches and `MuENL` variants. The difference between two algorithms is significant if the gap between their ranks is larger than the CD. There is a line between two algorithms if the rank gap between them is smaller than the CD.

(c) Compare with `LP-SENCForest`

`SENCForest` [19] is proposed recently to handle class incremental learning on streaming data, and has achieved a success. It provides a unified tree-based model for new class detection and known class prediction. `SENCForest` focuses on a multi-class setting—each instance belongs to a single class only. To apply `SENCForest` in the multi-label learning setting, we first apply `Label Powerset (LP)` [25] to encode different label combinations as different classes to transform multi-label learning into multi-class setting, before `SENCForest` is applied. We name it `LP-SENCForest`, which is the counterpart of `SENCForest` in our setting.

Figure 8 summarizes the comparison results of averaged performance on the whole data stream in five datasets. As can be observed, `MuENL` is much better than `LP-SENCForest`. This result is not surprising because: (1) `LP` transformation leads to too many classes, some of them hold a small number of instances. `SENCForest` does poorly because it requires a sufficient number of instances for each class in order to do well. (2) After `LP` transformation, the detected new class by `SENCForest` can be a new combination of existing labels, but does not contain any new labels.

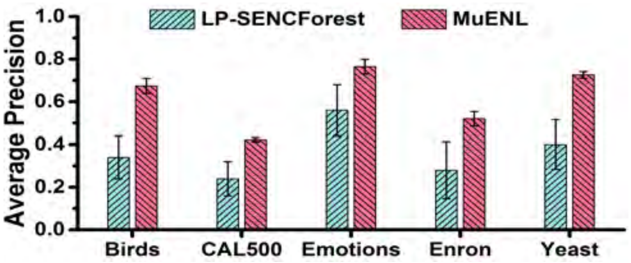


Figure 8. `MuENL` versus `LP-SENCForest`

(d) Results with different label sequences

In order to further validate the effectiveness of our approach, the evaluation is conducted multiple trials on a dataset, where each trial has a different random order of known and new labels on the simulated stream. The comparison is done with two best performing methods in each of two approaches, i.e., the multi-label approach: `PLR` and `LIMO` [10], and two`MuENL` variants: `MuENL`$_{OC}$ and `MuENL`$_{IF}$.

We report the averaged difference from the baseline `BR` in MicroF1, which is a label based measure: $\text{MicroF1} = \frac{2\sum_{i,j} y_{i,j} h_{i,j}}{\sum_{i,j} y_{i,j} + \sum_{i,j} h_{i,j}}, i \in \{1, \cdots, T\}, j \in \{1, \cdots, |\boldsymbol{v}_t|\}$, on the entire data stream as in Figure 9. The result that `MuENL` outperforms these four methods, even under random orders of new labels.

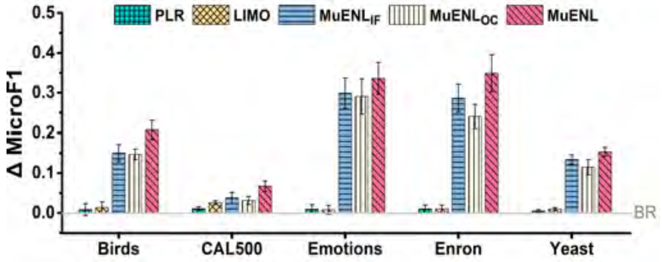


Figure 9. Compare `BR`, `PLR`, `LIMO`, `MuENL`$_{\text{OC}}$, `MuENL`$_{\text{IF}}$ with different label sequences as new labels: Results using the difference in MicroF1 from `BR`.

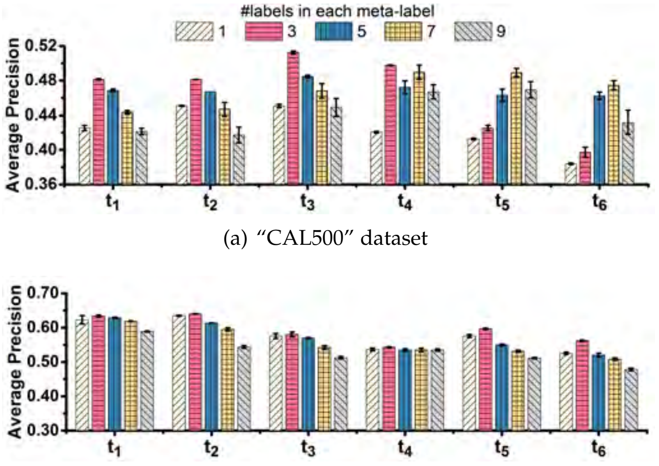
6.3.2 Results on multiple new labels per time period

When more than one new label occurs in a time period, `MuENL` can also be used by treating these new labels as a single meta label. Specifically, if any instance has a subset of labels, encapsulated by a meta-label, then the instance has this meta-label. The goal in this setting is to detect and classify a new meta label, if it exists, in each time period.

We conduct experiments, where the number of new labels varies from 1 to 9 in each time period, to examine the effect of multiple labels in comparison with the baseline which has only one new label per time period. The evaluation is based on the meta label we specified at the beginning of the data stream.

Figure 10 shows that results in two datasets. Both results show that the `MuENL` approach can handle multiple new labels; and in many cases, it performs better in the multiple-new-labels scenarios than in the single-new-label scenarios. This is because the learning task for a meta label of multiple

[10] `GenEML` is also comparable, not listed here due to the space issue.



(a) "CAL500" dataset

(b) "Enron" dataset

Figure 10. MuENL's performance in scenarios having different numbers of new labels in each time period

labels can be easier than that for a single label, depending on the data distribution. An example is shown Figure 10(a).

7 Experiments on high dimensional data

7.1 Dataset

We have collected a total of 6,000 Weibo instances in 10 topics: traffic, car, president, Nobel prize, TV show, music, video, advertisement, sports, finance (indexed as 1 - 10). We extracted the bag-of-words features which resulted in 15,461 features. As a consequence, the data is very sparse, i.e., only 0.2% of the total number of features have non-zero values; and each instance belongs to 1.61 labels on average.

When it begins at t_0, there are only 5 known labels, and the rest of them will successively emerge as new labels. Five new labels emerge successively at different times, denoted as $t_1, t_2, \cdots, t_5$. On average, the interval between the subsequent emergence of two new labels has 1,000 instances. For experiments, we permute the order of class labels, and conduct 5 runs to avoid the influence of new label order on the results.

7.2 Baselines

We compare MuENLHD with two baselines:

- MuENL: Directly apply MuENL on the high dimensional sparse data stream.
- MuENL+PCA: First, we perform PCA on the observed data at time t_0 to obtain the mapping from the original space to a low dimensional subspace (where $d' = 100$). Then, we transform each arriving instance to the low dimensional representation via this mapping. We apply MuENL on the transformed data.
- MuENL+MultiPCA: Unlike MuENL+PCA, which performs PCA only at t_0, MuENL+MultiPCA applies PCA multiple times, i.e., a new PCA is conducted on a data chunk collected during the period in-between the buffer is empty and full.

The parameter setting of MuENL is the same as that used in Section 6. The settings for MuENLHD are given as follows: the number of the random features $m = 2000$, and the number of attributes in the transformed space is set as $d' = 100$.

Table 6
Total time used to handle the data stream with 6,000 instances.

	MuENL	MuENL+PCA	MuENL+MultiPCA	MuENLHD
Time (s)	120,839	507	1417	1,442

7.3 Experimental results

We evaluate the performance in terms of Average Precision on 100 instances at every time point of t_0, t_1, $\cdots$, t_5. The performances of MuENLHD, MuENL+PCA, MuENL+MultiPCA are MuENL are exhibited in Figure 11. As expected, all algorithms have comparable performance at time t_0. However, MuENLHD achieves much better performance than the other contenders at t_1 to t_5. This is mainly due to fact that streaming kernel PCA captures the main information of the data stream in different time periods; and those transformed low dimensional representations work well collaboratively with the predictive information from the classifiers.

Although MuENL+PCA reduces the dimensionality to the same as that of MuENLHD, it performs worse because MuENL+PCA only trains PCA at t_0, and keeps the linear mapping throughout the stream. This strategy leads to a worse performance in the following detection, since emerging new instances are not considered. In contrast. MuENLHD adapts the dimension reduction throughout the stream.

For MuENL+MultiPCA, PCA is applied each time the MuENL's models are updated for an emerging new label. Even though PCA has been applied multiple times to adapt to the data stream, only the latest data chunk is considered. In contrast, MuENLHD adapts along the stream. As a result, MuENLHD outperforms MuENL+MultiPCA.

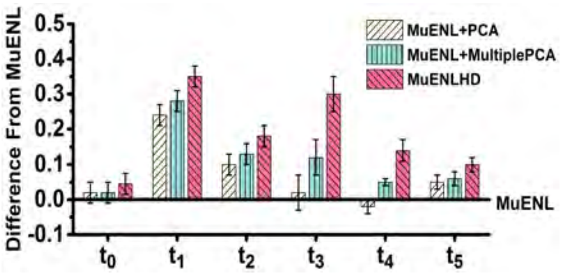
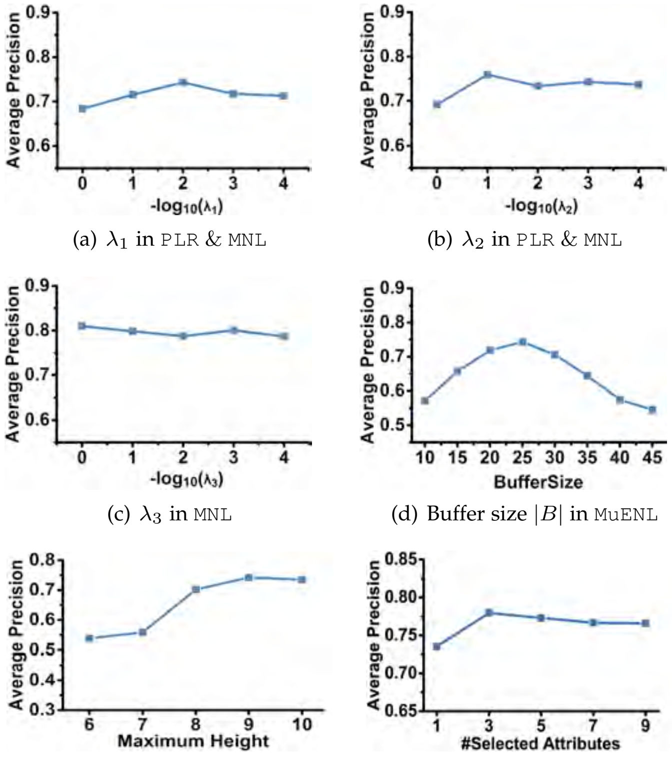


Figure 11. Compare MuENL+PCA, MuENL+MultiPCA and MuENLHD: Result shown is the difference from MuENL in terms of Average Precision at each time point.

We also make a comparison on the time spent to handle the whole stream, shown in Table 6. As can be observed, MuENL takes nearly 100 times longer than MuENLHD to handle the high dimensional data stream. This is consistent with the factor of reduction from 15000 to 100 dimensions. Though MuENL+PCA is 3 times faster than MuENLHD, they are in the same order. MuENLHD needs the extra time in order to update the nonlinear mapping throughout the stream. MuENLHD has about the same run time as MuENL+MultiPCA, but it achieves much better performance.

8 The effect of parameters

We study the effect of parameters in this section. The experiment is conducted in one run, where settings of the parameters are determined via 5-fold cross-validation before



(a) λ_1 in PLR & MNL

(b) λ_2 in PLR & MNL

(c) λ_3 in MNL

(d) Buffer size $|B|$ in MuENL

(e) Max height e_m in MuENLForest

(f) #Sampled attributes $|\boldsymbol{q}|$

Figure 12. Example plots of the effects of parameters in MuENL on "Emotions"

the start of the run. Then, a parameter is varied to study its effect while the other parameters are fixed.

Figure 12 shows the results of the study on varying the setting of parameters (buffer size $|B|$ in Algorithm 1; λ_1, λ_2 and λ_3 in Procedure 2; e_m and $|\boldsymbol{q}|$ in Procedure 1). The results show that the MuENL approach is not very sensitive to the settings of λ's and $|\boldsymbol{q}|$; but it is sensitive to $|B|$ and e_m. Its sensitivity to the buffer size is because (i) a good performing classifier cannot be trained for the new label if the buffer size is too small; and (ii) when the buffer size is too large, it may contain a mixture of different new labels, which will degenerate the performance[11].

Selecting an appropriate setting is not an issue as long as a cross-validation can be employed at the beginning of a data stream.

One issue related to memory is the expanded set of known labels as time progresses. We may exploit the "forgetting mechanism", i.e., remove models from $\mathcal{H}_t$ which have not been used for prediction (or equivalently, labels have not been seen) for a certain period of time. This mechanism will set a limit to the memory required to store the models. A recurrence of any of the removed labels will be treated as new labels.

9 Conclusions

Multi-label learning with emerging new labels is a practical problem that demands attention. This paper extends our preliminary research [36], which formalizes this problem and proposes the novel MuENL approach which has a model consists of three components: (1) a multi-label classifier for the known labels, (2) a detector for new labels, and (3) the updating processes for the classifier and the detector for each new label. Because existing methods only consider a fixed label set, they do not have the last two components. As a result, they are significantly less effective than the proposed approach in the dynamic learning environment, as verified in the empirical evaluation. The idea of converting part of the problem into an outlier detection problem has enabled the whole problem to be solved satisfactorily. The outlier detector we have designed has high detection rate—the key to ensuring a robust classifier that can maintain a high classification accuracy in data streams. We also show that MuENL can be easily extended to handle sparse high dimensional data streams by simply reducing the original dimensionality using Streaming Kernel PCA, and then applying MuENL on the reduced dimensional space. The empirical evaluation validates its effectiveness.

Acknowledgments

This research was supported by NSFC (61751306), 111 Project (B14020), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] W. Bi and J. T. Kwok. Multilabel classification with label correlations and missing labels. In *Proceedings of 28th AAAI Conference on Artificial Intelligence*, pages 1680–1686, 2014.

[2] N. Boumal., B. Mishra, P.-A. Absil., and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014.

[3] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[4] Q. Da, Y. Yu, and Z.-H. Zhou. Learning with augmented class by exploiting unlabeled data. In *Proceedings of 28th AAAI Conference on Artificial Intelligence*, pages 1760–1766, 2014.

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[6] Y. Fu, Y. Yang, T. Hospedales, T. Xiang, and S. Gong. Transductive multi-label zero-shot learning. *arXiv preprint arXiv:1503.07790*, 2015.

[7] J. Fürnkranz, E. Hüllermeier, E. L. Mencía, and K. Brinker. Multi-label classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, 2008.

[8] M. Ghashami, D. J. Perry, and J. Phillips. Streaming kernel principal component analysis. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 1365–1374, 2016.

[9] Y. Gong, Q. Zhang, X. Han, and X. Huang. Phrase-based hashtag recommendation for microblog posts. *SCIENCE CHINA Information Sciences*, 60(1):12109, 2017.

[10] M. Hollander, D. A. Wolfe, and E. Chicken. *Nonparametric statistical methods*. John Wiley & Sons, 2013.

[11] S.-J. Huang and Z.-H. Zhou. Multi-label learning by exploiting label correlations locally. In *Proceedings of 26th AAAI Conference on Artificial Intelligence*, pages 949–955, 2012.

[12] V. Jain, N. Modhe, and P. Rai. Scalable generative models for multi-label learning with missing labels. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1636–1644, 2017.

[13] I. Kuzborskij, F. Orabona, and B. Caputo. From n to n+1: multiclass transfer incremental learning. In *Proceedings of 2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3358–3365, 2013.

[11] Note that the evaluation here is different from that for meta new labels shown in Section 6.3.2. For a meta new label, if any one of those labels is detected, it is a correct detection. In this setting, it is correct only if the exact label is detected.

[14] F. Li and H. Wechsler. Open set face recognition using transduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1686–1697, 2005.

[15] X. Li and Y. Guo. Max-margin zero-shot learning for multi-class classification. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 626–634, 2015.

[16] F. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Proceedings of 8th IEEE International Conference on Data Mining*, pages 413–422, 2008.

[17] A. McCallum. Multi-label text classification with a mixture model trained by em. In *Working Notes of the AAAI'99 Workshop on Text Learning*, pages 1–7, 1999.

[18] T. Mensink, E. Gavves, and C. G. Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014*, pages 2441–2448, 2014.

[19] X. Mu, K. M. Ting, and Z.-H. Zhou. Classification under streaming emerging new classes: A solution using completely random trees. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1605–1618, 2017.

[20] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.

[21] S. Rüping. Incremental learning with support vector machines. In *Proceedings of the 1st IEEE International Conference on Data Mining*, pages 641–642, 2001.

[22] W. J. Scheirer, R. A. de Rezende, A. Spkota, and T. E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

[23] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

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

[25] G. Tsoumakas and I. P. Vlahavas. Random k -labelsets: An ensemble method for multilabel classification. In *Proceedings of 18th European Conference on Machine Learning*, pages 406–417, 2007.

[26] D. Turnbull, L. Barrington, D. Torres, and C. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, 2008.

[27] V. Vapnik, A. Vashist, and N. Pavlovitch. Learning using hidden information (learning with teacher). In *Proceedings of International Joint Conference on Neural Networks*, pages 3188–3195, 2009.

[28] X.-Z. Wu and Z.-H. Zhou. A unified view of multi-label performance measures. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3780–3788, 2017.

[29] M. Xu, R. Jin, and Z.-H. Zhou. Speedup matrix completion with side information: Application to multi-label learning. In *Advances in Neural Information Processing Systems 26*, pages 2301–2309. 2013.

[30] X. Xu, W. Wang, and J. Wang. A three-way incremental-learning algorithm for radar emitter identification. *Frontiers of Computer Science*, 10(4):673–688, 2016.

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

[32] Z.-H. Zhou. Learnware: on the future of machine learning. *Frontiers of Computer Science*, 10(4):589–590, 2016.

[33] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.

[34] Z.-H. Zhou and Z.-Q. Chen. Hybrid decision tree. *Knowledge-Based Systems*, 15(8):515–528, 2002.

[35] Y. Zhu, J. Kwok, and Z.-H. Zhou. Multi-label learning with global and local correlation. *IEEE Transactions on Knowledge and Data Engineering*, in press.

[36] Y. Zhu, K. M. Ting, and Z.-H. Zhou. Multi-label learning with emerging new labels. In *IEEE 16th International Conference on Data Mining*, pages 1371–1376, 2016.

Yue Zhu received the BSc degree in Computer Science and Technology from Nanjing Normal University, Nanjing, China, in 2011. In the same year, he was admitted to further study in Nanjing University, Nanjing, China. Now, he is currently a PhD candidate and a member of LAMDA Group. His research interests mainly include multi-instance learning, multi-label learning, multi-view learning, incremental learning, and novelty detection.

Kai Ming Ting After receiving his PhD from the University of Sydney (Australia), Kai Ming Ting worked at the University of Waikato, Deakin University and Monash University. He joined Federation University in 2014. He had previously held visiting positions at Osaka University (Japan), Nanjing University (China) and Chinese University of Hong Kong. His current research interests are in the areas of mass estimation and mass-based approaches, ensemble approaches and data stream data mining. He is an associate editor for Journal of Data Mining and Knowledge Discovery. He co-chaired the Pacific-Asia Conference on Knowledge Discovery and Data Mining 2008. He has served as a member of program committees for a number of international conferences including ACM SIGKDD, IEEE ICDM and ICML. His research projects are supported by grants from Australian Research Council, US Air Force of Scientific Research (AFOSR/AOARD), and Australian Institute of Sport. Awards received include the Runner-up Best Paper Award in 2008 IEEE ICDM, and the Best Paper Award in 2006 PAKDD.

Zhi-Hua Zhou (S'00-M'01-SM'06-F'13) received the BSc, MSc and PhD degrees in computer science from Nanjing University, China, in 1996, 1998 and 2000, respectively, all with the highest honors. He joined the Department of Computer Science & Technology at Nanjing University as an Assistant Professor in 2001, and is currently Professor and Standing Deputy Director of the National Key Laboratory for Novel Software Technology; he is also the Founding Director of the LAMDA group. His research interests are mainly in artificial intelligence, machine learning and data mining. He has authored the books *Ensemble Methods: Foundations and Algorithms* and *Machine Learning* (in Chinese), and published more than 150 papers in top-tier international journals or conference proceedings. He has received various awards/honors including the National Natural Science Award of China, the PAKDD Distinguished Contribution Award, the IEEE ICDM Outstanding Service Award, the Microsoft Professorship Award, etc. He also holds 22 patents. He is an Executive Editor-in-Chief of the *Frontiers of Computer Science*, Associate Editor-in-Chief of the *Science China Information Sciences*, Action or Associate Editor of the *Machine Learning*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* , *ACM Transactions on Knowledge Discovery from Data* , etc. He served as Associate Editor-in-Chief for *Chinese Science Bulletin* (2008-2014), Associate Editor for *IEEE Transactions on Knowledge and Data Engineering* (2008-2012), *IEEE Transactions on Neural Networks and Learning Systems* (2014-2017), *ACM Transactions on Intelligent Systems and Technology* (2009-2017), Neural Networks (2014-2016), *Knowledge and Information Systems* (2003-2008), etc. He founded ACML (Asian Conference on Machine Learning), served as Advisory Committee member for IJCAI (2015-2016), Steering Committee member for ICDM, PAKDD and PRICAI, and Chair of various conferences such as General co-chair of PAKDD 2014 and ICDM 2016, Program co-chair of SDM 2013 and IJCAI 2015 Machine Learning Track, and Area chair of NIPS, ICML, AAAI, IJCAI, KDD, etc. He is/was the Chair of the IEEE CIS Data Mining Technical Committee (2015-2016), the Chair of the CCF-AI(2012-), and the Chair of the Machine Learning Technical Committee of CAAI (2006-2015). He is a foreign member of the Academy of Europe, and a Fellow of the ACM, AAAI, AAAS, IEEE, IAPR, IET/IEE, CCF, and CAAI.