Multi-Label Learning with Emerging New Labels

Yue Zhu[1], Kai-Ming Ting[2], Zhi-Hua Zhou[1]
[1]*National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China*
[2]*School of Engineering and Information Technology, Federation University, Australia*
Email: [1]*{zhuy, zhouzh}@lamda.nju.edu.cn* [2]*kaiming.ting@federation.edu.au*

***Abstract*—Multi-label learning is widely applied in many tasks, where an object possesses multiple concepts with each 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 open and new concepts may emerge with previously unseen instances. In order to maintain good predictive performance in this environment, a multi-label learning method must have the ability to detect and classify those instances with emerging new labels. To this end, we propose a new approach called Multi-label learning with Emerging New Labels (MuENL). It builds models with three functions: classify instances on currently known labels, detect the emergence of a new label in new instances, and construct a new classifier for each new label that works collaboratively with the classifier for known labels. Our empirical evaluation shows the effectiveness of MuENL.**

***Keywords*-multi-label learning; incremental learning; emerging new labels; learnware**

I. Introduction

In traditional supervised learning, one instance is associated with a single label; whereas in many applications, one instance may possess multiple labels. Multi-label learning is the learning paradigm to handle this kind of data, and has attracted much attention in recent years [13]. Previous studies of 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 open. Previously unseen instances may emerge with new labels together with currently known labels.

A learning system in an open environment should be reusable and adaptive to the changing environment [14]. For a dynamic multi-label learning system, a pre-trained multi-label model shall be revised as new instances emerge; and new models are established for emerging new labels only. These demands are non-trivial, and no existing research in the literature can deal with them, as far as we know.

To meet these additional demands, 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 multi-label classifier for known labels; (2) A detector of emerging new labels; and (3) Update the models by enlarging the set of known labels to include a new label when a sufficient number of instances of it has been detected.

The most challenging part in MuENL 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 impossible to separate instances with new labels only from those with the known labels only.

Moreover, because the detection might not be perfect, the error will accumulate as more and more new labels emerge in a data stream. Thus, the environment demands robust models for new labels in order to maintain high performance in detection and prediction continuously in the data stream.

To meet all the above challenges, the MuENL approach (1) optimizes the pairwise label ranking together with the classification loss on the known labels, (2) builds a specially designed detector based on both the input features and predicted value attributes, and (3) constructs a robust classifier for the new label.

The contribution of this work is summarized as follows:

- Formalizing the dynamic multi-label learning problem—a multi-label training set with known labels is available initially before the data stream begins. Each newly arrived instance in the data stream may be associated with multiple known labels and possibly new label(s).
- Proposing the MuENL approach to address the dynamic multi-label learning problem, incorporating a novel detector and a robust classifier for new labels.

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.

II. Related Work

Multi-label learning can be divided into three main categories based on the order of label correlations [13]. For the first-order strategy [1], none of the label correlations are considered. For the second-order strategy [4], pairwise label relations are taken into account. For the high-order strategy [8], a label is assumed to be influenced by all other labels. 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 frequent data update is involved or when it is desirable not to re-train the model from scratch. According to [15] , there are roughly three major incremental learning settings, i.e., example incremental learning (E-IL) [9], where new instances arrive after the learning system has been trained; attribute incremental learning (A-IL) [12], where new attributes may appear; and class incremental learning (C-IL) [2], [6], where the class label set may be enlarged.

The dynamic multi-label learning setting with emerging new 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 [11]. 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 [10] learns a boundary for instances with known labels, and decides instances outside the boundary as outliers; iForest [7] 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 [3] has proposed a transductive multi-label zero-shot learning. This work extends the idea of [2] of identifying new class by the exploitation of unlabeled data. It is under a transductive setting, i.e., all test instances are observed during the training and all the names of 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.

III. The MuENL Approach

Our dynamic multi-label learning problem faces the following challenges: (A) detecting instances with both new and known labels; (B) building a robust model for the new label. In this section, we propose MuENL to handle the dynamic multi-label learning problem with emerging new labels, addressing the two challenges. Specifically, MuENL consists of three components: (i) A multi-label classifier for the known labels; (ii) a detector of emerging new labels; (iii) update of current models, including construction of a robust model for the new label recently detected, and update of the current multi-label classifier and the detector.

The performance of the detector is essential in maintaining an accurate predictive model in the dynamic multi-label learning environment. The new label detection problem is not only an unsupervised learning problem, but the new label can be masked by known labels because both the new and known labels are associated with an instance. No research has been conducted in the current literature, as far as we known. We design a new detector which can detect new labels with high accuracy.

Even with an accurate detector, the accumulated errors over time in the data stream can seriously compromise the performance of the predictive model, if it cannot tolerate the accumulated errors over time. Thus, it is imperative to design a robust model for the new label.

A. Problem Formulation

Let $\mathcal{X}$ denote the feature space and define $X_0 = [\boldsymbol{x}_{-n+1}, \cdots, \boldsymbol{x}_{-1}, \boldsymbol{x}_0]^\top \in \mathcal{X}$ as the initially observed n labeled instances. New instances successively arrive, and let $X_t = [X_{t-1}; \boldsymbol{x}_t^\top], t \in \{1, 2, \cdots, T\}$, denote all the instances observed thus far at time t, where $\boldsymbol{x}_t$ denotes the unlabeled new instance emerging at time t.

Let $\mathcal{Y}_0 = \{c_1, c_2, \cdots c_l\}$ be the known class label set of length l initially at $t = 0$. When the new label is to be converted to known label, $\mathcal{Y}_t = \mathcal{Y}_{t-1} \cup \{c_a\}$; otherwise $\mathcal{Y}_t = \mathcal{Y}_{t-1}$, where c_a is the new label at time t with $a = |\mathcal{Y}_{t-1}|+1$. Let $Y_0 = [\boldsymbol{y}_{-n+1}, \cdots, \boldsymbol{y}_{-1}, \boldsymbol{y}_0] \in \{-1, 1\}^{l \times n}$ be the known label matrix of X_0, and $\boldsymbol{y}_t \in \{-1, 1\}^{|\mathcal{Y}_t|}$ be the label vector of the newly arrived instance $\boldsymbol{x}_t$. Notice that none of the elements in $\boldsymbol{y}_t$ are observed, since $\boldsymbol{x}_t$ is unlabeled.

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 $\mathcal{H}_t = [h_{t,1}, h_{t,2}, \cdots, h_{t,|\mathcal{Y}_t|}]$, where $h_{t,i} : \mathcal{X} \to \{-1, 1\}^{|\mathcal{Y}_t|}$ represents the model for class label c_i at time $t \in \{1, 2, \cdots, T\}$. For each $\boldsymbol{x}_t$, the models produce $\hat{\boldsymbol{y}}_t = \mathcal{H}_t(\boldsymbol{x}_t)$ which is the predicted label vector, where $\boldsymbol{x}_t$ may be associated with new labels to co-occur with known labels.*

B. The Approach

Directly estimating $\mathcal{H}_t$ in the dynamic learning environment defined above is extremely hard, since $\mathcal{Y}_t$ has an unknown variable, i.e., it is not known whether any new label is associated with a newly arrived instance $\boldsymbol{x}_t$. 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 predicting their existence, but also in updating models to maintain the high predictive accuracy of classifiers 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., $\mathcal{F}_t(\boldsymbol{x}_t)$, 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 model is used to do the prediction, yielding $\hat{\boldsymbol{y}}_t = \mathcal{H}_t(\boldsymbol{x}_t)$, where $\mathcal{H}_t = [h_{t,1}, h_{t,2}, \cdots, h_{t,|\mathcal{Y}_t|}, \mathcal{F}_t]$. After making prediction $\hat{\boldsymbol{y}}_t$ for $\boldsymbol{x}_t$, the model update process begins when both of the following conditions are satisfied:

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

The models are updated using $\mathcal{T}_t = (X_t, \mathcal{H}_t(X_t))$ as shown in line 9 of Algorithm 1. This is the time when the known label set is expanded by including the new label: $\mathcal{Y}_t = \mathcal{Y}_{t-1} \cup \{c_a\}$. 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 model.

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 $\mathcal{F}_0$ on $X_0$; set $t = 1$;
 2: $\mathcal{H}_1 = [\mathcal{H}_0, \mathcal{F}_0]$; $\mathcal{F}_1 = \mathcal{F}_0$
 3: repeat
 4:     Receive a new instance $\boldsymbol{x}_t$, $X_t = [X_{t-1}; \boldsymbol{x}_t^\top]$;
 5:     $\hat{\boldsymbol{y}}_t = \mathcal{H}_t(\boldsymbol{x}_t)$, where $\mathcal{H}_t = [h_{t,1}, h_{t,2}, \cdots, h_{t,|\mathcal{Y}_t|}, \mathcal{F}_t]$;
 6:     if $\mathcal{F}_t(\boldsymbol{x}_t) = 1$
 7:         Add $\boldsymbol{x}_t$ to $B$;
 8:         if $|B| =$ MAX_BUFFER_SIZE
 9:             Update $\mathcal{F}_{t+1}$ and $\mathcal{H}_{t+1}$ using $\mathcal{T}_t = (X_t, \mathcal{H}_t(X_t))$;
10:             Empty $B$;
11:             The new label is converted to the known label:
                $\mathcal{Y}_t = \mathcal{Y}_{t-1} \cup \{c_a\}$;
12:         end if
13:     end if
14:     $t \leftarrow t + 1$;
15:     $\mathcal{Y}_t = \mathcal{Y}_{t-1}$;  $\mathcal{F}_t = \mathcal{F}_{t-1}$;  $\mathcal{H}_t = \mathcal{H}_{t-1}$;
16: until $t = T$.
```

Algorithm 1 summarizes the MuENL approach. It has three key components: (i) the multi-label model ($\mathcal{H}_0 = [h_{0,1}, \cdots, h_{0,|\mathcal{Y}_0|}]$) for the known labels; (ii) the detection model ($\mathcal{F}_t$) for the new label; (iii) the update of $h_{t,k},\ k \in \{1, 2, \cdots, |\mathcal{Y}_t|\}$ and the update of $\mathcal{F}_t$.

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

IV. The Multi-label Model

We use a linear model $(\boldsymbol{w}_i, b_i)$ 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 better performance. To simplify the optimization process, model $(\boldsymbol{w}_i, b_i)$ is converted to $(\boldsymbol{w}_i)$ by adding an attribute with value 1 at the end of x_k, i.e., $f_{i,k} = \boldsymbol{w}_i^\top [\boldsymbol{x}_k; 1]$. The optimization problem is formulated as follows:

$$\min_{\boldsymbol{w}_i} \sum_{j=1}^{|\mathcal{Y}_t|} \sum_{k=1}^{n} [1 - (y_{i,k} - y_{j,k})(f_{i,k} - f_{j,k})]_+ + \lambda_1 \sum_{k=1}^{n} [1 - y_{i,k} f_{i,k}]_+ + \frac{\lambda_2}{2} \|\boldsymbol{w}_i\|^2, \qquad (1)$$

where λ_1 and λ_2 are two trade-off parameters. To solve Eqn. (1), we first calculate the subgradient of the objective function; then apply the gradient descend method.

V. New Label Detection

The appearance of a new label may owe to the difference in feature space or known label patterns 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 like to occur.

Using this idea, we propose MuENLForest as our detection model $\mathcal{F}$. Similar to the earlier work [5], [8], 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 [7]. 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 as 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, 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 growing 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.

A. 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

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

Algorithm 2 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[1] is built having radius $r = \max_{\boldsymbol{x} \in S}(\|\boldsymbol{x} - \boldsymbol{m}\|)$,
    where $\boldsymbol{m} = \text{mean}(S)$;
    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 feature 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$(\boldsymbol{q}, S)$;
    $S_l = \{\boldsymbol{x} \in S \mid \|\boldsymbol{x}^{\boldsymbol{q}} - \boldsymbol{p}_1\| \leq \|\boldsymbol{x}^{\boldsymbol{q}} - \boldsymbol{p}_2\|\}$;
    $S_r = \{\boldsymbol{x} \in S \mid \|\boldsymbol{x}^{\boldsymbol{q}} - \boldsymbol{p}_1\| > \|\boldsymbol{x}^{\boldsymbol{q}} - \boldsymbol{p}_2\|\}$;
    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
```

of g MuENLTrees; and each MuENLTree is built using a random subset of $\mathcal{T}_t$ of size ψ.

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

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 2 summarizes the construction of MuENLTree.

B. MuENLForest Detection

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

VI. Model Update

A. Multi-Label Model Update

When buffer B is full, $\mathcal{H}_t$ is to be updated. This update includes the construction of a new model for the new label, and the update of the existing multi-label model (without re-training the model).

Because the detector is not perfect, it may miss some positive instances with the new label or mistaken some negative ones to be the positive ones. Thus, we need to design a robust model for both the new label and the known labels.

When an instance with a new label appears, the influence of the new label shall be taken into consideration in the training of a new model and the updating of the existing model. To achieve this goal, we apply an iterative optimization strategy, i.e., the multi-label model is fixed while learning a robust model $h_{t,a}$ for the new label; and the learned $h_{t,a}$ is fixed while updating the existing multi-label model.

Let X_B be the instances collected in the buffer and X_U be the set of instances with (predicted) known labels only since the last model update. Let $\boldsymbol{d} = [d_1, d_2, \cdots, d_m]^\top$ be the indicator vector of $[X_B; X_U]$, 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 to be learned simultaneously with the classifier, which corresponds to the potential assignment of the new label.

In order to obtain a robust model, we propose MuENL$_{\text{MNL}}$ to simultaneously learn the indicator $\boldsymbol{d}$ and model $\boldsymbol{w}_a$ for the new label c_a. Similar to the strategy described in Section IV that optimizes both the misclassification loss and the pairwise label ranking loss, the optimization problem of building model $\boldsymbol{w}_a$ for new label c_a is cast as follows:

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

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, thus 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, we calculate the subgradient of the objective function and perform the gradient descend. After each update, we project $\boldsymbol{d}$ to $[0, 1]$: $\boldsymbol{d} \leftarrow \min(\boldsymbol{1}, [\boldsymbol{d}]_+)$, to satisfy the box constrain.

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 model. For the following iterations, the optimization begins with the optimization result obtained in the last iteration.

Updating the existing multi-label model considers both the detected new label and corresponding learned model. It updates the model in a way similar to Eqn. (1) in Section IV.

B. Detection Model Update

To give preference to recent data, we apply a weighted sampling strategy, where recent data has a higher weight than older data. Specifically, a weight is associated with each instance, i.e., $\boldsymbol{v}$. Each newly arrived instance has an initial weight of 1.0. Each time the buffer is full, update $\boldsymbol{v} \leftarrow 0.8\boldsymbol{v}$, thus the weight of older instances will diminish as time progress.

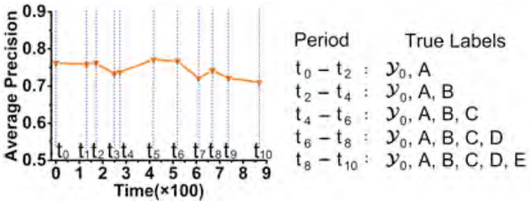


Figure 1. Example: MuENL's performance when there are 5 new labels.

VII. Experimental Setup

A. Configuration

To evaluate the predictive performance of MuENL, we use 5 multi-label benchmark datasets (*"Birds", "CAL500", "Emotions", "Enron" and "Yeast"*)[2]. 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.

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 ($\mathcal{Y}_0$) are observed; and a multi-label model 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 model ($[h_{t,1}, \cdots, h_{t,|\mathcal{Y}_t|}]$) for the expanded known label set $\{\mathcal{Y}_0, A\}$; and also updates the detection model ($\mathcal{F}_t$) for the next new label (B).

A commonly used multi-label performance metric "average precision" is applied in the evaluation, which is the larger the better. 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 900 time steps with 5 consecutive new labels.

The experiment is conducted based on the simulation described above, except only three new labels are used instead of five new labels because of space limitation. The predictive performance is measured at $t_1, \ldots, t_6$; and two evaluations are made: (i) $\mathcal{Y}_t$-evaluation, where it measures the performance in the dynamic learning environment as in the simulation; (ii) $\mathcal{Y}_0$-evaluation, where it assesses how well MuENL works on the initial label set $\mathcal{Y}_0$ only throughout the entire period.

B. 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.

State-of-the-art multi-label approaches: We compare MuENL with BR [1], CLR [4], ECC [8] and PLR, 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 IV. 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 ensembled 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.

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

- MuENL_{IF}: Use iForest [7] (instead of MuENLForest) as the detection model.
- MuENL_{OC}: Use OC-SVM [10] (instead of MuENLForest) as the detection model.
- $\text{MuENL}_{\text{SVM}}$: Use MuENLForest as the detection model, but use a different classification model: build the model for the new label via SVM, which takes the instances in the buffer as positive ones, and the others as negative ones to train a linear model.
- MuENL_{OR}: Use MuENLForest as the detection model and assume that the oracle is accessible to provide the ground truth for model updates. Its performance will be the upper bound.

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.

VIII. Experimental Results

A. Compare with Multi-label approaches

Figure 1 is an example plot of $\mathcal{Y}_t$-evaluation on the Yeast dataset having one new label in each time period over 5 periods[3]. 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 and PLR in terms of $\mathcal{Y}_t$-evaluation in five datasets. It is interesting to note that MuENL almost always performs better than all baselines. Many of the differences are significant.

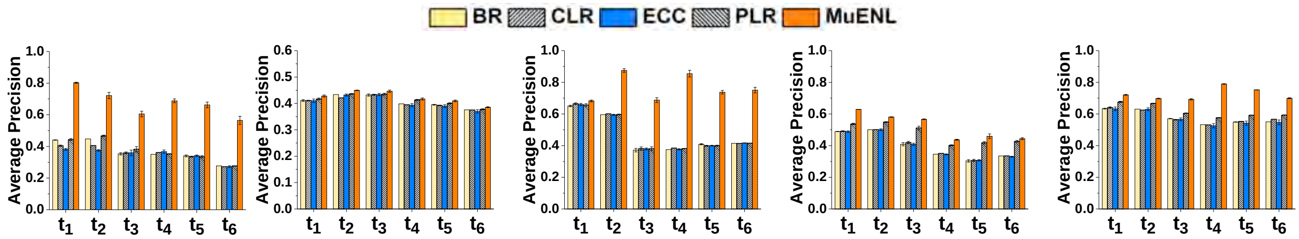
In terms of $\mathcal{Y}_0$-evaluation, MuENL achieves better or comparable performance in comparison with all baselines (results are not shown due to space limitation). MuENL can gain better performance on known labels because pairwise label ranking is considered.

B. Compare with MuENL variants

Table I presents the $\mathcal{Y}_t$-evaluation result.

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

[3] 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.



(a) Birds (b) Cal500 (c) Emotions (d) Enron (e) Yeast

Figure 2. Comparison results with multi-label approaches in $\mathcal{Y}_t$-evaluation. $t_1 - t_6$ on the horizontal axis are the 6 time points.

Table I
COMPARISON RESULTS WITH MUENL VARIANTS IN $\mathcal{Y}_t$-EVALUATION

	MuENL_{IF}	MuENL_{OC}	$\text{MuENL}_{\text{SVM}}$	MuENL_{OR}	MuENL
Birds	.60±.05	.63±.03	.60±.04	.69±.03	.67±.02
CAL500	.40±.01	.40±.01	.39±.01	.42±.01	.42±.01
Emotions	.70±.03	.62±.04	.63±.03	.78±.03	.76±.03
Enron	.48±.03	.49±.03	.45±.02	.52±.02	.52±.02
Yeast	.70±.01	.70±.01	.70±.01	.73±.01	.73±.01

Among the four MuENL variants, MuENL_{OR} has the best performance since it uses the oracle which provides the ground truth that is not available to other variants.

MuENL obtains a performance comparable with MuENL_{OR} in many cases. This is a direct result of a good detector and a robust classifier for both known and new labels.

MuENL achieves a better performance than $\text{MuENL}_{\text{SVM}}$ in most cases. This shows that the robust model update procedure in MuENL works and is essential in maintaining a good performance in a dynamic learning environment.

The comparisons with MuENL_{IF} and MuENL_{OC} show that MuENLForest is a better detector in the dynamic multi-label learning environment.

IX. Conclusion

Multi-label learning with emerging new labels is a practical problem that demands attention. In this work, we formalize this problem and propose the novel MuENL approach which consists of three components: (1) a multi-label classifier for the known labels, (2) a detector for new labels, and (3) a new classifier for each new label that works collaboratively with the classifiers for known labels. 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 designed has high detection rate—the key to ensuring a robust model to maintain high accuracy in data streams.

Acknowledgment

This research was supported by NSFC (61333014), 111 Project (B14020), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

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

[2] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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