Online Semi-supervised Multi-label Classification with Label Compression and Local Smooth Regression

Peiyan Li[1] , **Honglian Wang**[1] , **Christian Böhm**[2] and **Junming Shao**[1,*]
[1]Data Mining Lab, University of Electronic Science and Technology of China
[2]Ludwig-Maximilians-Universität München
peiyanli.uestc@gmail.com, hlw@std.uestc.edu.cn, boehm@ifi.lmu.de, junmshao@uestc.edu.cn

Abstract

Online semi-supervised multi-label classification serves a practical yet challenging task since only a small number of labeled instances are available in real streaming environments. However, the mainstream of existing online classification techniques are focused on the single-label case, while only a few multi-label stream classification algorithms exist, and they are mainly trained on labeled instances. In this paper, we present a novel Online Semi-supervised Multi-Label learning algorithm (OnSeML) based on label compression and local smooth regression, which allows real-time multi-label prediction in a semi-supervised setting and is robust to evolving label distributions. Specifically, to capture the high-order label relationship and to build a compact target space for regression, OnSeML compresses the label set into a low-dimensional space by a fixed orthogonal label encoder. Then a locally defined regression function for each incoming instance is obtained with a closed-form solution. Targeting the evolving label distribution problem, we propose an adaptive decoding scheme to adequately integrate newly arriving labeled data. Extensive experiments provide empirical evidence for the effectiveness of our approach.

1 Introduction

In multi-label learning, each instance is associated with multiple labels simultaneously. This is a common learning paradigm in real-world applications. For example, people in a social network usually has multiple identities [Wang and Sukthankar, 2013], a water body is associated with many functions [Yang *et al.*, 2015], and a web query can be related to several topics [Tang *et al.*, 2009]. Similar applications can also be found in many online systems. For example, a micro-blogging system should suggest tags (labels) for new blogs, and users can choose tags for their blogs, which serves as a feedback to improve the performance of tag-suggestion. The main challenges of such applications are two-fold. (a) The real-time demand for accurate predictions when labeled data and unlabeled data are arriving randomly at a high-speed. (b) The system should possess the ability to deal with evolving label distributions. To be specific, the labels of arriving instances in data streams often come randomly and usually associated with only a few labels compared to the whole label set in a short time range (e.g., a small-size data chunk). Thus the underlying label distributions in different time slices usually vary significantly.

To date, we have witnessed the success of multi-label learning on static data (see two excellent surveys [Tsoumakas *et al.*, 2009; Zhang and Zhou, 2014]). One of its sub-fields, semi-supervised multi-label learning, has gained increasing attention due to the scarcity of labeled data in real-world applications. Following the classic taxonomy, semi-supervised multi-label learning can be either transductive or inductive. The former only focuses on making predictions on existing unlabeled data while the latter can generalize to unseen instances. For example, graph-based semi-supervised methods generally work in a transductive setting by first constructing a similarity graph among labeled and unlabeled data, and then predicting the labels of unlabeled instances via label propagation [Kong *et al.*, 2013; Wang *et al.*, 2013] or manifold regularization [Chen *et al.*, 2008; Jing *et al.*, 2015]. Since requiring all unlabeled data to be available during the training phase may not be realistic, inductive multi-label learning is more applicable in a real scenario. Compared to transductive multi-label learning techniques, methods in this line should learn multi-label classifiers to predict labels for unseen instances. For example, the co-training framework for multi-label classification [Zhan and Zhang, 2017; Xing *et al.*, 2018] constructs two or multiple views of the data, and then pairwise ranking predictions on unlabeled data are communicated between view-specific classifiers for model refinement. Despite their success, most of these algorithms can not be easily transformed into the online semi-supervised setting. Designing efficient and effective online semi-supervised multi-label classification algorithm is thus of significant importance. To the best of our knowledge, there is only one proposed algorithm [Boulbazine *et al.*, 2018] targeting the online semi-supervised multi-label classification problem, which constructs a dynamic graph with incoming data based on the growing neural gas model [Zaki and Yin, 2008].

[*]Corresponding Author

In this paper, we propose a new online semi-supervised multi-label classification algorithm, called OnSeML, based on local smooth regression and label compression. OnSeML allows making real-time multi-label predictions in a semi-supervised setting and is robust to evolving label distributions. To this end, OnSeML first uses a fixed orthogonal-initialized encoding matrix to encode the label set and then learns the local smooth regression model for each incoming instance based on the regularized moving least square (RMLS) [Chang and Yeung, 2007]. Basically, RMLS is designed for local metric learning, which learns local affine transformations for each data. We adapt the core idea of RMLS by approximating the regression function for each incoming instance with its neighbors in the already arrived data. Since we cannot store all historical data, two budgets are set to bound the model size. Considering the evolving label distributions, we update the decoding matrix periodically. Accordingly, two update strategies are proposed, including (a) the adaptive update strategy; and (b) the adjustment strategy. The first strategy enables real-time predictions for unlabeled incoming instances, while the latter strategy refines the label decoder when the prediction accuracy is low. Finally, extensive experiments demonstrate that OnSeML can even achieve comparable results compared to offline semi-supervised multi-label models and online supervised multi-label models. In summary, the main contributions of this paper are multi-fold:

- We propose OnSeML, a novel locally defined regression model for semi-supervised multi-label learning on data streams, and a closed-form solution is derived to guarantee its efficiency.
- We use a label encoding scheme to capture label relationships in a low-dimensional space, and an adaptive decoding matrix is learned for handling the potential issues of evolving label distributions.
- Our experiments demonstrate that OnSeML outperforms many state-of-the-art semi-supervised or online multi-label learning algorithms.

2 Proposed Approach

In this section, we introduce our online semi-supervised multi-label classification algorithm, which mainly contains two parts: label compression and local smooth regression. In the following, we first give the problem statement.

2.1 Problem Statement

Formally, let $\mathbf{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} \in \mathbb{R}^{d \times N}$ be a multi-label data stream with labeled and unlabeled instances coming randomly, where d denotes the feature dimension, and $\boldsymbol{x}_N$ is the last seen instance in the data stream. The online classifier should predict the proper label vector $\boldsymbol{y} \in \mathbb{B}^l$ (l is the number of labels) for the unlabeled incoming data. Different from the classic online semi-supervised learning paradigm [Jia *et al.*, 2009], which divides the data stream into consecutive batches and assumes that the first part of data in each batch is labeled. Here, we have no assumption on the arriving order of labeled and unlabeled instances, which makes our problem setting more general.

2.2 Label Compression with Least Square

To capture the high-order label relationship, and to obtain a compact space for regression, we compress the label set into a low-dimensional space. To be specific, a fixed orthogonal encoding matrix $\mathbf{P} \in \mathbb{R}^{k \times l}, k < l$ is initialized by Gram-Schmidt orthogonalization [Björck, 1967], and then we encode the binary label vector of the i-th instance $\boldsymbol{y}_i$ via projection:

$$\boldsymbol{h}_i = \mathbf{P}\boldsymbol{y}_i \tag{1}$$

It is worth noting that we build a regression model for each instance (e.g., x_i) whether it is labeled or not, i.e., $f_i : \boldsymbol{x}_i \to \hat{\boldsymbol{h}}_i$. After obtaining $\hat{\boldsymbol{h}}_i$, we use the decoding matrix $\mathbf{Q}_t \in \mathbb{R}^{l \times k}$ to project $\hat{\boldsymbol{h}}_i$ back to the original label space, i.e.,

$$\hat{\boldsymbol{y}}_i = \mathbf{Q}_t \hat{\boldsymbol{h}}_i \tag{2}$$

A similar label compression strategy can be found in [Ahmadi and Kramer, 2018; Zhou *et al.*, 2017]. However, we do not use an additional binarization step like $sign(\boldsymbol{h}_i)$, since we employ regression to perform classification. Furthermore, the binarization step in previous methods tends to yield a significant reconstruction error due to the non-linearity.

For a given multi-label data stream, when receiving a set of labeled instances, i.e., $[\mathbf{X}_t, \mathbf{Y}_t]$ with $\mathbf{X}_t \in \mathbb{R}^{d \times N_t}, \mathbf{Y}_t \in \mathbb{R}^{l \times N_t}$, the decoding matrix $\mathbf{Q}_t \in \mathbb{R}^{l \times k}$ can be obtained by minimizing the least square error as follows.

$$\min \ ||\mathbf{Y}_t - \mathbf{Q}_t \mathbf{P} \mathbf{Y}_t||_F^2, \tag{3}$$

where $|| \cdot ||_F$ is the Frobenius norm. We can derive a closed-form solution for the decoding matrix $\mathbf{Q}_t$ as follows.

$$\mathbf{Q}_t = \mathbf{Y}_t \mathbf{H}_t^T (\mathbf{H}_t {\mathbf{H_t}}^T)^{-1}, \tag{4}$$

where $\mathbf{H}_t = \mathbf{P}\mathbf{Y}_t$ is the encoded label set.

As aforementioned, since the arriving data at different time slices usually have different label distributions, in section 2.4, we will illustrate how to learn an adaptive decoding matrix to handle this issue.

2.3 Local Smooth Regression

After projecting the label set to a low-dimensional space, the next step is to build the regression model for each incoming data to support classification. As a direct application of well-known linear regression to the classification task, we formulate our local smooth regression model as follows.

$$f_i(\boldsymbol{x}_i) = \mathbf{W}_i \boldsymbol{x}_i + \mathbf{b}_i, \tag{5}$$

where $\mathbf{W}_i \in \mathbb{R}^{k \times d}$ and $\mathbf{b}_i \in \mathbb{R}^k$ are the two parameters to be learned for the i-th instance, whether it is labeled or not. In principle, to learn the local smooth regression model for each instance, we need at least two labeled instances, which is acceptable in the real scenario. Although this method has to determine a large number of parameters, in the following, we will illustrate its efficiency and introduce the strategy for reducing the model size.

In this paper, we extend the regularized moving least square model (RMLS) [Chang and Yeung, 2007] to the online semi-supervised setting. Here, we first review the idea of RMLS by its optimization function, which mainly contains two parts: approximation and regularization. Firstly, assuming $\boldsymbol{x}_i$ is similar to $\boldsymbol{x}_j$ in the feature space, and the ground truth (i.e., the regression target) of $\boldsymbol{x}_j$ is given and denoted as $\boldsymbol{h}_j$, then $\{\boldsymbol{x}_j, \boldsymbol{h}_j\}$ can be used to approximate the i-th instance's regression function as follows.

$$\text{If } \boldsymbol{x}_j \in k\text{NN}(\boldsymbol{x}_i), \text{ then } f_i(\boldsymbol{x}_j) \rightarrow \boldsymbol{h}_j$$

Next, assume that similar instances usually have similar targets, which further formulates the regularization as follows.

$$\text{If } \boldsymbol{x}_j \in k\text{NN}(\boldsymbol{x}_i), \text{ then } f_i(\boldsymbol{x}_i) \approx f_j(\boldsymbol{x}_j)$$

Based on the two parts, the optimization problem of RMLS is given as:

$$\begin{aligned} J(\mathbf{W}_i, \mathbf{b}_i) = &\sum_j \theta_{ij} ||f_i(\boldsymbol{x}_j) - \boldsymbol{h}_j||^2 + \\ &\lambda \sum_j \omega_{ij} ||f_i(\boldsymbol{x}_i) - f_j(\boldsymbol{x}_j)||^2, \end{aligned} \tag{6}$$

where $\theta_{ij} = 1/||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$ is the moving weight, indicating that to which degree $\boldsymbol{x}_j$ can use $f_i(\cdot)$ to approximate h_j. Moreover, $\omega_{ij} = exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/\sigma^2)$ with $\sigma > 0$ controlling the spread, is introduced as a regularization term. It preserves local neighborhood relationships and is widely used in graph-based semi-supervised learning approaches (e.g., [Zhu and Goldberg, 2009]). Further, λ is a user-defined parameter to balance the two terms.

Since an instance's regression function can be approximated by its neighbors, we can naturally extend RMLS to the online semi-supervised setting. For example, the first part of Eq. (6) can be reformed to the empirical loss of all arrived labeled instances, while the second part serves as the regularization term by treating the learned regression $f_j(\boldsymbol{x}_j)$ as pseudo target. Moreover, considering the intrinsic property of online classification, two other objectives should be achieved, including (1) bounding the model size; and (2) preventing error-propagation.

Note that if we do not set a candidate set for $\boldsymbol{x}_j$, the model size will grow linearly with the number of all arriving instances, which is infeasible for streaming data. To bound the model size, we set two budgets B_L and B_A for labeled instances and arrived instances, respectively. Accordingly, we set two pre-defined sizes for B_L and B_A as s_L and s_A, respectively. If $|B_L| > s_L$ or $|B_A| > s_A$, the budget maintenance strategy should be triggered. Here, we remove the oldest instance for both cases. To be specific, once the number of instances in B_A or B_L overflows its pre-defined size, the oldest instance in B_A or B_L will be removed. This strategy focuses more on recently arrived instances, making it more robust to concept drift. When the number of instances in the two budgets reaches the pre-defined size, the computational complexity for calculating the newly arrived instance's regression function will remain the same. Also, we only need to store the corresponding parameters of the instances in the two budgets. Therefore, the model size is bounded. We summarize the budget maintenance procedure for local smooth regression in Algorithm 1.

On the other hand, with more incoming instances, an emerging problem is how to prevent error-propagation. In the second term of Eq. (6), if some of the past learned regression functions have a large bias, the regularizer would propagate errors to $f_i(\cdot)$. This error-propagation problem comes naturally with semi-supervised data stream mining since it is learning-order-sensitive. To ease this problem, we adopt the active sampling strategy, by further restricting $\boldsymbol{x}_j$ in the second term to be selected in the nearest neighbors of $\boldsymbol{x}_i$ in B_A. With more informative instances for $\boldsymbol{x}_i$ to be selected, fewer errors can be propagated. Finally, we have the following local smooth regression function for multi-label data streams.

Algorithm 1 Budget Maintenance for B_L and B_A

```
1:  Receive a new instance $x$
2:  if $x$ is labeled then
3:      Add $x$ to $B_L$ and $B_A$
4:  else
5:      Add $x$ to $B_A$
6:  end if
7:  for $B = \{B_L, B_A\}$ do
8:      if $B$ overflowed then
9:          Remove the oldest instance in $B$
10:     end if
11: end for
```

$$\begin{aligned} J(\mathbf{W}_i, \mathbf{b}_i) = &\sum_{j \in B_L} \theta_{ij} ||f_i(\boldsymbol{x}_j) - \boldsymbol{h}_j||^2 + \\ &\lambda \sum_{j \in B_A \cap nei(i)} \omega_{ij} ||f_i(\boldsymbol{x}_i) - f_j(\boldsymbol{x}_j)||^2 \end{aligned} \tag{7}$$

We remark that Eq. (7) has a closed-form solution. After substituting Eq. (5) into Eq. (7) and setting $\hat{B}_i = B_A \cap nei(i)$ for short, we obtain the following objective function.

$$\begin{aligned} J(\mathbf{W}_i, \mathbf{b}_i) = &\sum_{j \in B_L} \theta_{ij} ||\mathbf{W}_i \boldsymbol{x}_j + \mathbf{b}_i - \boldsymbol{h}_j||^2 + \\ &\lambda \sum_{j \in \hat{B}_i} \omega_{ij} ||\mathbf{W}_i \boldsymbol{x}_i + \mathbf{b}_i - \hat{\boldsymbol{h}_j}||^2, \end{aligned} \tag{8}$$

where $\hat{\boldsymbol{h}_j} = \mathbf{W}_j \boldsymbol{x}_j + \mathbf{b}_j$. Eq. (8) has two parameters $\mathbf{W}_i$ and $\mathbf{b}_i$. By setting the derivative of each parameter to 0, we obtain the following results.

$$\begin{aligned} \mathbf{W}_i &= (\mathbf{D}_i - \bar{\boldsymbol{h}}_i \bar{\boldsymbol{x}}_i^T)(\mathbf{C}_i - \bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^T)^+ \\ \mathbf{b}_i &= \bar{\boldsymbol{h}}_i - \mathbf{W}_i \bar{\boldsymbol{x}}_i \end{aligned} \tag{9}$$

Here $(\cdot)^+$ means the pseudo-inverse function since the inverse of $(\mathbf{C}_i - \bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^T)$ may not exist. $\bar{\boldsymbol{x}}_i$, $\bar{\boldsymbol{h}}_i$, $\mathbf{C}_i$ and $\mathbf{D}_i$ are denoted as follows.

$$\begin{aligned} \bar{\boldsymbol{x}}_i &= \frac{\sum_{j \in B_L} \theta_{ij} \boldsymbol{x}_j + \lambda \sum_{j \in \hat{B}_i} \omega_{ij} \boldsymbol{x}_i}{\sum_{j \in B_L} \theta_{ij} + \lambda \sum_{j \in \hat{B}_i} \omega_{ij}} \\ \bar{\boldsymbol{h}}_i &= \frac{\sum_{j \in B_L} \theta_{ij} \boldsymbol{h}_j + \lambda \sum_{j \in \hat{B}_i} \omega_{ij} \hat{\boldsymbol{h}}_j}{\sum_{j \in B_L} \theta_{ij} + \lambda \sum_{j \in \hat{B}_i} \omega_{ij}} \\ \mathbf{C}_i &= \frac{\sum_{j \in B_L} \theta_{ij} \boldsymbol{x}_j \boldsymbol{x}_j^T + \lambda \sum_{j \in \hat{B}_i} \omega_{ij} \boldsymbol{x}_i \boldsymbol{x}_i^T}{\sum_{j \in B_L} \theta_{ij} + \lambda \sum_{j \in \hat{B}_i} \omega_{ij}} \\ \mathbf{D}_i &= \frac{\sum_{j \in B_L} \theta_{ij} \boldsymbol{h}_j \boldsymbol{x}_j^T + \lambda \sum_{j \in \hat{B}_i} \omega_{ij} \hat{\boldsymbol{h}}_j \boldsymbol{x}_i^T}{\sum_{j \in B_L} \theta_{ij} + \lambda \sum_{j \in \hat{B}_i} \omega_{ij}} \end{aligned}$$

2.4 Adaptive Update

In this section, we introduce how to learn an adaptive model to handle the problem of evolving label distributions. Basically, we achieve this goal by learning an adaptive decoding matrix. Eq. (4) gives the solution of label decoder, when receiving a set of labeled data, which contains an inverse operator with $O(k^3)$ computational complexity in general. Thus it will cause high computation cost if we update the decoding matrix each time, when a labeled instance arrives. Thereby, we update $\mathbf{Q}$ periodically. To be specific, we initialize the decoding matrix $\mathbf{Q}$ by $\mathbf{Q} = \mathbf{P}^T$ since $\mathbf{P}$ is orthogonal. After that, we update $\mathbf{Q}$ when the size of newly arrived labeled data $([\mathbf{X}_{t+1}, \mathbf{Y}_{t+1}])$ reaches a pre-defined number s_Q from time t to time $t+1$.

$$\begin{aligned}\mathbf{Q}_{t+1} &= \begin{bmatrix}\mathbf{Y}_t\\ \mathbf{Y}_{t+1}\end{bmatrix}\begin{bmatrix}\mathbf{H}_t\\ \mathbf{H}_{t+1}\end{bmatrix}^T\left(\begin{bmatrix}\mathbf{H}_t\\ \mathbf{H}_{t+1}\end{bmatrix}\begin{bmatrix}\mathbf{H}_t\\ \mathbf{H}_{t+1}\end{bmatrix}^T\right)^{-1}\\ &= (\mathbf{Y}_t\mathbf{H}_t^T + \mathbf{Y}_{t+1}\mathbf{H}_{t+1}^T)(\mathbf{H}_t\mathbf{H}_t^T + \mathbf{H}_{t+1}\mathbf{H}_{t+1}^T)^{-1}\end{aligned} \quad (10)$$

Similar to the adaptive update method introduced in [Venkatesan *et al.*, 2016; Ahmadi and Kramer, 2018], the above function can be solved incrementally. For simplicity, we substitute $\mathbf{Y}_t\mathbf{H}_t^T = \mathbf{Q}_t\mathbf{K}_t^{-1}$ (the deformation of Eq.(4)) where $\mathbf{K}_t = (\mathbf{H}_t\mathbf{H}_t^T)^{-1}$ into Eq. (10), then the above expression becomes:

$$\begin{aligned}\mathbf{Q}_{t+1} &= (\mathbf{Q}_t\mathbf{K}_t^{-1} + \mathbf{Y}_{t+1}\mathbf{H}_{t+1}^T)(\mathbf{K}_t^{-1} + \mathbf{H}_{t+1}\mathbf{H}_{t+1}^T)^{-1}\\ &= \mathbf{Q}_t + (\mathbf{Y}_{t+1} - \mathbf{Q}_t\mathbf{H}_{t+1})\mathbf{H}_{t+1}^T\mathbf{K}_{t+1}\end{aligned} \quad (11)$$

where:

$$\mathbf{K}_{t+1} = (\mathbf{K}_t^{-1} + \mathbf{H}_{t+1}\mathbf{H}_{t+1}^T)^{-1} \quad (12)$$

We remark that the adaptive update procedure for $\mathbf{Q}$ can be accelerated in the specific case when $s_Q < k$, and k is the dimension of encoded label space. By using the Sherman-Morrison-Woodbury formula, we rewrite Eq. (12) as follows.

$$\mathbf{K}_{t+1} = \mathbf{K}_t - \mathbf{K}_t\mathbf{H}_{t+1}(\mathbf{I}_{s_Q} + \mathbf{H}_{t+1}^T\mathbf{K}_t\mathbf{H}_{t+1})^{-1}\mathbf{H}_{t+1}^T\mathbf{K}_t \quad (13)$$

Therefore, when $s_Q < k$, by keeping $\mathbf{K}_t$ for updating $\mathbf{Q}_{t+1}$, the time complexity becomes $O(s_Q^3)$, which is smaller than directly calculating $\mathbf{Q}_{t+1}$ by Eq. (10). For a better illustration, we give the pseudo-code in Algorithm 2.

Now, we give the adapt update strategy for the decoding matrix, which enables real-time prediction. Since we update $\mathbf{Q}_t$ to $\mathbf{Q}_{t+1}$ after receiving a pre-defined number of labeled instances, the prediction is performed by the decoding matrix $\mathbf{Q}_t$ rather than $\mathbf{Q}_{t+1}$. Considering that the labeled data are not fully utilized, the idea of error-sensitive adjustment is proposed to refine the adapt update strategy. From time t to time $t+1$, we train a set of regression models for newly arrived instances. Since some instances are labeled, an average prediction error δ is computed. Defining an error threshold ϵ, if $\delta \le \epsilon$, which indicates $\mathbf{Q}_t$ can fit the data received from time t to time $t+1$, otherwise, an adjustment strategy will be triggered. To be specific, a number of recently arrived labeled instances including $\mathbf{Y}_{t+1}$ will be selected to train a new label decoder $\mathbf{Q}_{t+1}$, and then it will be used to make the offline predictions to get the final results. Note that if we directly update $\mathbf{Q}_{t+1}$ by $\mathbf{Y}_{t+1}$, it will be lack of generality since $\mathbf{Y}_{t+1}$ may only contain a part of the label set. We summarize this strategy in Algorithm 3.

Algorithm 2 Adaptive Update Strategy for $\mathbf{Q}$

```
Input: $\mathbf{Q}_t$, $\mathbf{K}_t$, $\mathbf{P}$
       newly arrived label set $\mathbf{Y}_{t+1}$
Output: $\mathbf{Q}_{t+1}$
1: Set $\mathbf{H}_{t+1} = \mathbf{P}\mathbf{Y}_{t+1}$
2: if $s_Q < k$ then
3:     Get $\mathbf{K}_{t+1}$ via Eq.(13)
4: else
5:     Get $\mathbf{K}_{t+1}$ via Eq.(12)
6: end if
7: Get $\mathbf{Q}_{t+1}$ via Eq.(11)
8: return $\mathbf{Q}_{t+1}$
```

Algorithm 3 Adjustment Strategy for $\mathbf{Q}$

```
Input: $\mathbf{Q}_t$, $\mathbf{Y}_{t+1}$, error threshold $\epsilon$
       the label set received in the past $\hat{\mathbf{Y}}_t$
Output: $\mathbf{Q}_{t+1}$
1: Record the prediction results for data at $[t, t+1]$
2: Compute the average prediction error $\delta_t$
3: if $\delta_t \le \epsilon$ then
4:     Get $\mathbf{Q}_{t+1}$ via Algorithm 2
5: else
6:     Select label subset $\mathbf{Y}_t^*$ in $\hat{\mathbf{Y}}_t$
7:     Training $\mathbf{Q}_{t+1}$ via $[\mathbf{Y}_t^*; \mathbf{Y}_{t+1}]$ and Eq. (4)
8: end if
9: return $\mathbf{Q}_{t+1}$
```

Algorithm 4 OnSeML

```
Input: A multi-label data stream: $\mathbf{X}$;
       Regularization parameter $\lambda$;
       Budget size $s_L$ and $s_A$;
       The dimension of the encoded label set: $k$;
       The number of labeled instances to update $\mathbf{Q}$: $s_Q$;
Output: The predicted labels for unlabeled data $\mathbf{Y}_p$.
1: // Gram-Schmidt orthogonalization
2: Generate an orthogonal matrix $\mathbf{P} \in R^{k \times l}$;
3: Initialize $\mathbf{Q} = \mathbf{P}^T$;
4: Initialize $B_L = \varnothing$, $B_A = \varnothing$;
5: // Training a set of regression models
6: for $i = 1 \to N$ do
7:     Receive an incoming instance $\boldsymbol{x}_i$
8:     Obtain $\mathbf{W}_i, \mathbf{b}_i$ via Eq. (9)
9:     $\hat{\boldsymbol{h}}_i = \mathbf{W}_i\boldsymbol{x}_i + \mathbf{b}_i$
10:    $\hat{\boldsymbol{y}}_i = \mathbf{Q}\hat{h}_i$
11:    Update $B_L$ and $B_A$ with Algorithm 1
12:    for each time receive $s_Q$ labeled data do
13:        Update $\mathbf{Q}$ via Algorithm 2 or 3.
14:        if using Algorithm 3 then
15:            Update the classification results
16:        end if
17:    end for
18: end for
```

2.5 Online Semi-supervised Multi-Label Learning

Building upon local smooth regression and the label encoding-decoding scheme, we summarize the pseudo-code of OnSeML in Algorithm 4. Firstly, the encoding matrix $\mathbf{P}$ is initialized to be an orthogonal matrix with k independent components. Then the decoding matrix $\mathbf{Q}$ is computed by $\mathbf{P}$. For each incoming data, we train a regression model online, and then we project its regression result to the original label space. Meantime, we update two budgets B_L and B_A to bound the model size. If we receive sufficient labeled data or the prediction error is higher than a given threshold, an adaptive update strategy or adjustment strategy will be triggered to update the label decoder $\mathbf{Q}$, respectively. For label decision, we get the predicted labels with top K values.

2.6 Time Complexity

The time complexity of OnSeML mainly has two parts. For learning the regression function of each incoming instance, the major computation workload is the pseudo-inverse of a d-by-d matrix in Eq. (9), with time complexity of $O(d^3)$. As for adaptive update of the decoding matrix, in the case of $s_Q < k$, the time complexity can be reduced from $O(k^3)$ to $O(s_Q^3)$ by using the Sherman-Morrison-Woodbury formula. Assuming the multi-label data stream contains n data points, the adaptive decoder should be updated $\lceil \frac{n}{s_Q} \rceil$ times, and the time complexity of OnSeML is $O(nd^3 + \lceil \frac{n}{s_Q} \rceil \times \min(k^3, s_Q^3))$.

3 Experiments

3.1 Experimental Setup

Datasets. We evaluate OnSeML on four real-world datasets: Enron, Corel5k, Mirflickr and Mediamill. All data sets employed in our experiments are publicly available on Mulan[1] [Tsoumakas *et al.*, 2009] and Lambda[2] [Zhang and Zhou, 2010]. Table 1 lists the statistics of these data sets.

Comparison Methods. Since the area of online semi-supervised multi-label learning has rarely been investigated, for fair comparison, we use the same partition of labeled and unlabeled instances in the compared approaches[3]. Here four multi-label algorithms are used for comparison, including two transductive multi-label learning algorithms (e.g., TRAM [Kong *et al.*, 2011] and BSSML [Gönen, 2014]) and two online supervised classification algorithms. One is Online-BR (OBR) implemented by incremental support vector machine [Laskov *et al.*, 2006], and another one is OSML-ELM (OELM) [Venkatesan *et al.*, 2016], which is an online supervised neural network classifier for multi-label classification.

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

[2]http://lamda.nju.edu.cn/Data.ashx

[3]Only our approach allows labeled and unlabeled instances to come randomly, and the label set in different time slices can be different.

Data	Domain	N	d	l	C
Enron	text	1702	1001	53	3.38
Corel5k	image	5000	499	374	3.52
Mirflickr	image	25000	150	24	4.72
Mediamill	video	43097	120	101	4.38

Table 1: Statistic information of the data sets. Here N is the number of instances, d indicates the feature dimension, l is the number of labels, and C is the label cardinality.

Parameter Setting. For OnSeML, the budget size s_L and s_A are set to bound the model size. For larger data sets, e.g. Mirflickr and Mediamill, we simply set $s_L = 0.05N$ and $s_A = 0.1N$; while for small data sets like Enron and Corel5k, we leave the model size unbounded. For the second term in Eq. (7), the number of nearest neighbors is set to 10. The regularization parameter λ is tuned in $\{0.0001, 0.001, 0.1\}$. As for k and s_Q, they are largely decided by the size of the data set. Generally, we set $k = \lceil 0.1d \rceil$ and tuning s_Q in the range of $\{5, 10, 20, 40\}$. For TRAM, BSSML and OBR, we use their default parameter settings. For OELM, we tune the number of hidden layer neurons between $\{100, 200, 300, 400, 500\}$ and the batch size from 10 to 100 with the step size 10.

Evaluation Metrics. To evaluate the performance of OnSeML and the comparison methods, we use three widely-adopted metrics, including Precision at top-K (P@K, e.g., P@1, P@3, and P@5), Average Precision (AvgPre) and Micro-F1 (MicF1). The first two metrics are ranking based performance measure. P@K calculates the precision for top-K predicted labels, while AvgPre evaluates the average fraction of labels ranked higher than a particular true label. The two metrics are proper measures for multi-label algorithms when they make label decision based on a ranking list, e.g., BSSML, OBR, and OnSeML. To further evaluate the performance of classification, we report MicF1, which calculates an overall F-1 score among labels. We set 10% or 20% of the instances to be labeled, and the rest to be unlabeled. We run each experiment 10 times by randomly split the data as their original settings. The standard deviation is omitted due to space limitation, and the mean value of each evaluation metric is reported.

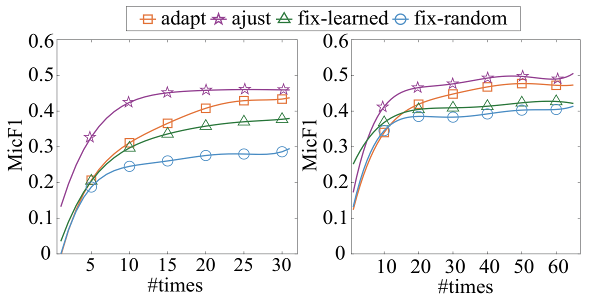
3.2 Performance Evaluation

In Table 2, we summarize the multi-label classification results on different data sets in terms of P@K, AvePre, and MicF1. Compared to others, OnSeML with the adaptive-update strategy and the adjustment strategy generally ranks the first or the second in prediction performance. Interestingly, by using the adjustment strategy, OnSeML gains better prediction performance on Enron and Corel5k. While on Mirflickr and Mediamill, the adaptive update strategy yields better performance. Considering that the label set of Mirflickr and Mediamill is small and instances coming in different time slices could have similar label distributions, the adjustment strategy may overfit the recently arrived data. Furthermore, we observe that TRAM and OELM achieve the worst results in most cases, while BSSML and OBR are sometimes inferior. To sum up, OnSeML generally achieves the best or the second-best results on the four benchmark datasets. We attribute this superiority to the well-approximated local smooth regression and the dynamic encoding-decoding scheme.

Next, to experimentally justify the effectiveness of the

Data	**Metric**	10%						20%					
		TRAM	**BSSML**	**OBR**	**OELM**	**OnSeML**		**TRAM**	**BSSML**	**OBR**	**OELM**	**OnSeML**	
						Adapt	**Adjust**					**Adapt**	**Adjust**
Enron	P@1	0.455	0.544	0.536	0.518	0.555	**0.640**	0.539	0.540	0.545	0.549	0.587	**0.697**
	P@3	0.401	0.469	0.480	0.378	0.449	**0.505**	0.433	0.474	0.494	0.390	0.484	**0.522**
	P@5	0.244	0.396	**0.441**	0.249	0.369	0.401	0.264	0.391	**0.447**	0.269	0.386	0.410
	AvePre	0.362	0.368	**0.439**	0.372	0.344	0.417	0.395	0.369	**0.445**	0.384	0.367	0.426
	MicF1	0.387	0.460	**0.526**	0.430	0.441	0.494	0.428	0.462	**0.534**	0.443	0.461	0.502
Corel5k	P@1	0.147	0.221	0.223	0.182	**0.225**	0.217	0.219	0.228	0.235	0.200	0.242	**0.264**
	P@3	0.094	**0.205**	0.188	0.138	0.173	0.198	0.167	0.206	0.198	0.110	0.189	**0.216**
	P@5	0.064	0.171	0.155	0.101	0.144	**0.176**	0.111	0.173	0.163	0.069	0.157	**0.185**
	AvePre	0.059	0.121	0.116	0.088	0.102	**0.143**	0.093	0.122	0.121	0.092	0.116	**0.146**
	MicF1	0.090	0.201	0.182	0.142	0.169	**0.209**	0.155	0.202	0.191	0.144	0.185	**0.218**
Mirflickr	P@1	0.326	0.417	0.407	0.331	**0.442**	0.422	0.327	0.441	0.396	0.344	**0.445**	0.431
	P@3	0.386	0.395	0.358	0.166	**0.406**	0.396	0.388	0.396	0.367	0.178	**0.406**	0.402
	P@5	0.304	**0.369**	0.315	0.100	0.362	0.367	0.304	**0.370**	0.336	0.108	0.365	0.368
	AvePre	**0.451**	0.449	0.401	0.366	0.447	**0.451**	0.450	0.449	0.413	0.372	**0.455**	0.451
	MicF1	0.397	**0.420**	0.349	0.211	0.416	0.417	0.400	**0.421**	0.363	0.225	0.419	0.418
Mediamill	P@1	†	0.803	0.783	0.514	**0.823**	0.804	†	0.804	0.794	0.533	**0.811**	0.804
	P@3	†	0.612	0.632	0.469	**0.659**	0.640	†	0.613	0.641	0.487	**0.664**	0.652
	P@5	†	0.481	**0.540**	0.388	0.507	0.497	†	0.481	**0.542**	0.382	0.509	0.498
	AvePre	†	0.499	0.526	0.482	**0.532**	0.521	†	0.499	0.528	0.503	**0.532**	0.525
	MicF1	†	0.513	0.534	0.511	**0.538**	0.530	†	0.512	0.539	0.534	**0.540**	0.536

Table 2: The performance of different algorithms on 10% and 20% labeled data. “Adapt” and “Adjust” refers to the adaptive update strategy and the adjustment strategy, respectively. Best results are marked in bold, and the second-bests are underlined. Results marked with † indicate the failure due to runtime demands ($>$ 24hours).



(a) Enron + 10% labeled (b) Enron + 20% labeled

Figure 1: Prediction performance tested on arriving unlabeled instances. The x-axis corresponds to the time that the label decoder should be updated. The colored lines present the performance of different strategies used by OnSeML, including adapt, adjust, fixed label decoder, which is learned by 10/20% labeled instances, and fixed label encoder initialized by $\mathbf{P}$.

adapt update strategy and the adjustment strategy, we add two strategies by fixing the labeled decoder. Explicitly, in one strategy, the labeled decoder is initialized by a part of the labeled instances, while in another strategy, the labeled decoder is only decided by $\mathbf{P}$. We record the prediction performance (indicated by MicF1) for each time the labeled decoder is updated. As shown in Figure 1, OnSeML with the two proposed strategies achieves better results compared to the case when OnSeML uses a fixed label encoder. Moreover, after receiving enough labeled instances, the performance of OnSeML begins to stabilize, which demonstrates its robustness towards the online semi-supervised multi-label classification problem.

4 Related Work

In this section, we briefly review some related work on semi-supervised multi-label learning. Currently, most of the semi-supervised multi-label learning algorithms are under the transductive setting. For example, TRAM [Kong *et al.*, 2011] is a classical semi-supervised multi-label learning algorithm based on label propagation, while BSSML [Gönen, 2014] is a Bayesian algorithm which couples linear dimensionality reduction and linear binary classification in a semi-supervised fashion. As to inductive multi-label learning, iMLCU [Wu and Zhang, 2013] is a representative method by adapting S3VM [Zhu and Goldberg, 2009], and COINS [Zhan and Zhang, 2017] is proposed based on co-training. To model label on-the-fly, inductive methods usually assume that unseen labels are negative, then pairwise ranking is incorporated to learn the base classifiers. Thus inductive methods generally have a high time complexity and limited scalability.

5 Conclusion

In this paper, we propose OnSeML, a novel online semi-supervised multi-label classification method based on local smooth regression and label encoding-decoding. This method enables real-time prediction and possesses the ability to deal with evolving label distributions. Specifically, OnSeML learns a locally defined regression function for each incoming instance, and a closed-form solution is derived to ensure its efficiency. To bound the model size, two budgets are set. Furthermore, to deal with the evolving label distribution problem, we propose an adaptive label decoder and two update strategies are introduced. Finally, experiments demonstrate the effectiveness and robustness of the proposed algorithm.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61976044), Fundamental Research Funds for the Central Universities (ZYGX2019Z014), Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China (161062), National key research and development program (2016YFB0502300).

References

[Ahmadi and Kramer, 2018] Zahra Ahmadi and Stefan Kramer. A label compression method for online multi-label classification. *Pattern Recognition Letters*, 111:64–71, 2018.

[Björck, 1967] Åke Björck. Solving linear least squares problems by gram-schmidt orthogonalization. *BIT Numerical Mathematics*, 7(1):1–21, 1967.

[Boulbazine *et al.*, 2018] Samira Boulbazine, Guénaël Cabanes, Basarab Matei, and Younés Bennani. Online semi-supervised growing neural gas for multi-label data classification. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[Chang and Yeung, 2007] Hong Chang and Dit-Yan Yeung. Locally smooth metric learning with application to image retrieval. In *11th International Conference on Computer Vision*, pages 1–7. IEEE, 2007.

[Chen *et al.*, 2008] Gang Chen, Yangqiu Song, Fei Wang, and Changshui Zhang. Semi-supervised multi-label learning by solving a sylvester equation. In *Proceedings of the International Conference on Data Mining*, pages 410–419. SIAM, 2008.

[Gönen, 2014] Mehmet Gönen. Coupled dimensionality reduction and classification for supervised and semi-supervised multilabel learning. *Pattern recognition letters*, 38:132–141, 2014.

[Jia *et al.*, 2009] Yangqing Jia, Shuicheng Yan, and Changshui Zhang. Semi-supervised classification on evolutionary data. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[Jing *et al.*, 2015] Liping Jing, Liu Yang, Jian Yu, and Michael K Ng. Semi-supervised low-rank mapping learning for multi-label classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1483–1491, 2015.

[Kong *et al.*, 2011] Xiangnan Kong, Michael K Ng, and Zhi-Hua Zhou. Transductive multilabel learning via label set propagation. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):704–719, 2011.

[Kong *et al.*, 2013] Xiangnan Kong, Michael K Ng, and Zhi-Hua Zhou. Transductive multilabel learning via label set propagation. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):704–719, 2013.

[Laskov *et al.*, 2006] Pavel Laskov, Christian Gehl, Stefan Krüger, and Klaus-Robert Müller. Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research*, 7(Sep):1909–1936, 2006.

[Tang *et al.*, 2009] Lei Tang, Suju Rajan, and Vijay K Narayanan. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th international conference on World wide web*, pages 211–220. ACM, 2009.

[Tsoumakas *et al.*, 2009] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[Venkatesan *et al.*, 2016] Rajasekar Venkatesan, Meng Joo Er, Shiqian Wu, and Mahardhika Pratama. A novel online real-time classifier for multi-label data streams. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1833–1840. IEEE, 2016.

[Wang and Sukthankar, 2013] Xi Wang and Gita Sukthankar. Multi-label relational neighbor classification using social context features. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 464–472. ACM, 2013.

[Wang *et al.*, 2013] Bo Wang, Zhuowen Tu, and John K Tsotsos. Dynamic label propagation for semi-supervised multi-class multi-label classification. In *Proceedings of the IEEE international conference on computer vision*, pages 425–432, 2013.

[Wu and Zhang, 2013] Le Wu and Min-Ling Zhang. Multi-label classification with unlabeled data: An inductive approach. In *Asian Conference on Machine Learning*, pages 197–212, 2013.

[Xing *et al.*, 2018] Yuying Xing, Guoxian Yu, Carlotta Domeniconi, Jun Wang, and Zili Zhang. Multi-label co-training. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2882–2888. AAAI Press, 2018.

[Yang *et al.*, 2015] Qinli Yang, Christian Boehm, Miklas Scholz, Claudia Plant, and Junming Shao. Predicting multiple functions of sustainable flood retention basins under uncertainty via multi-instance multi-label learning. *Water*, 7(4):1359–1377, 2015.

[Zaki and Yin, 2008] Shireen M Zaki and Hujun Yin. A semi-supervised learning algorithm for growing neural gas in face recognition. *Journal of Mathematical Modelling and Algorithms*, 7(4):425–435, 2008.

[Zhan and Zhang, 2017] Wang Zhan and Min-Ling Zhang. Inductive semi-supervised multi-label learning with co-training. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1305–1314. ACM, 2017.

[Zhang and Zhou, 2010] Yin Zhang and Zhi-Hua Zhou. Multilabel dimensionality reduction via dependence maximization. *ACM Transactions on Knowledge Discovery from Data*, 4(3):14, 2010.

[Zhang and Zhou, 2014] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2014.

[Zhou *et al.*, 2017] Wen-Ji Zhou, Yang Yu, and Min-Ling Zhang. Binary linear compression for multi-label classification. In *IJCAI*, pages 3546–3552, 2017.

[Zhu and Goldberg, 2009] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.