*ECAI 2016*
*G.A. Kaminka et al. (Eds.)*


*doi:10.3233/978-1-61499-672-9-261*

# A Scalable Clustering-Based Local Multi-Label Classification Method

**Lu Sun**[1] and **Mineichi Kudo**[1] and **Keigo Kimura**[1]

**Abstract.** Multi-label classification aims to assign multiple labels to a single test instance. Recently, more and more multi-label classification applications arise as large-scale problems, where the numbers of instances, features and labels are either or all large. To tackle such problems, in this paper we develop a clustering-based local multi-label classification method, attempting to reduce the problem size in instances, features and labels. Our method consists of low-dimensional data clustering and local model learning. Specifically, the original dataset is firstly decomposed into several regular-scale parts by applying clustering analysis on the feature subspace, which is induced by a supervised multi-label dimension reduction technique; then, an efficient local multi-label model, meta-label classifier chains, is trained on each data cluster. Given a test instance, only the local model belonging to the nearest cluster to it is activated to make the prediction. Extensive experiments performed on eighteen benchmark datasets demonstrated the efficiency of the proposed method compared with the state-of-the-art algorithms.

## 1 INTRODUCTION

Originated from traditional single-label classification, multi-label classification (MLC) enables to associate an instance with multiple labels. MLC has been used to tackle a number of real-world applications like text categorization [11], semantic image classification [3], video annotation [16] and music emotions detection [24], etc. Various MLC decomposition methods, such as Binary Relevance [3], Classifier Chains [18, 6], Calibrated Label Ranking [8] and Label Powerset [26], have been proposed by decomposing a multi-label problem into one or a set of single-label classification problems.

As the rapid increasing of web-related applications, more and more recent multi-label datasets emerge in large-scale, whose numbers of instances, features and labels are far from the regular-size. For example, there are millions of videos in the video-sharing website Youtube, while each one can be tagged by some of millions of candidate categories. Such large-scale problems challenge the existing MLC methods. Several methods [38, 5] have been proposed to tackle such a situation by training a multi-label model on the feature or the label subspaces. The common assumption behind these methods is that noisy features exist in the original data and the training label matrix is low-rank. Although these methods achieved much success in a number of MLC applications, further improvement in terms of time complexity and prediction accuracy is recently required.

In this study, we put on two assumptions about the locality in MLC setting: (a) meta-labels, i.e. reasonable and strong label combinations, exist implicitly in the label space; (b) only a fraction of features and instances are relevant to a meta-label. These assumptions are supported by several observations. For example, in Enron dataset, 53 labels are categorized into only four meta-labels, and in image annotation, an object typically relates to only a few regions in the high-dimensional feature space.

Hence, we presume that MLC can be tackled by decomposing the original large-scale data into several regular-scale datasets, each of which is relevant to only several meta-labels in a feature subspace with a fraction of training instances. Based on this assumption, a Clustering-based Local MLC (CLMLC) method is proposed in this paper. CLMLC consists of two stages, low-dimensional data clustering and local model learning. In the first stage, a supervised dimension reduction is firstly conducted to project the original high-dimensional data into a low-dimensional feature subspace, while preserving feature-label correlation. Then clustering analysis is applied to partition the low-dimensional data into several regular-scale datasets. In the second stage, within each data cluster, meta-labels are mined by saving both label similarity and instance locality, and then classifier chains over meta-labels are built as the local MLC model. Given a test instance, prediction is made on the basis of the local model corresponding to its nearest data cluster. To empirically evaluate the performance of CLMLC, extensive experiments on regular/large-scale datasets from various domains are carried out with the state-of-the-art MLC algorithms.

## 2 RELATED WORKS

To handle large-scale MLC problems, recently many research efforts have been paid to Feature Space Dimension Reduction (FS-DR) and Label Space Dimension Reduction (LS-DR). In FS-DR, traditional supervised dimension reduction approaches, such as Latent Semantic Indexing, Linear Discriminant Analysis, Canonical Correlation Analysis and Hypergraph Spectral Learning, are specifically extended to match the MLC setting [35, 29, 22, 21]. On the other hand, in order to improve the discriminative ability for each label, LIFT [37] and LLSF [9] are proposed to extract label-specific features. In LS-DR, based on the assumption of low-rank of label matrix, several embedding methods, such as Compressive Sensing [12], CPLST [5] and FaIE [13], encode the sparse label space by preserving label correlations and maximizing predictability of latent label space. By combining FS-DR and LS-DR, several methods have been proposed in recent years. WSABIE [31] learns a low-dimensional joint embedding space by approximately optimizing the precision on the top $k$ relevant labels. By modeling MLC as a general empirical risk minimization problem with a low-rank constraint, LEML [34] scales to very large datasets even with missing labels. To handle the extreme MLC problems with a large number of labels, a tree-based

[1] Hokkaido University, Sapporo 060-0814, Japan, email: {sunlu, mine, kkimura}@main.ist.hokudai.ac.jp

method, FastXML, is proposed in [15] by directly optimizing a specific ranking loss function, nDCG, and by efficiently executing its formulation in light of an alternating minimization algorithm.

The above methods can be categorized as global MLC methods, since they assume that feature-label relationship can be modeled on the whole training data. The global methods probably contradict real-world problems, harming classification accuracy and bringing in high time complexity, especially in large-scale datasets. To relax the assumption, local MLC methods are proposed, aiming to solve a complex problem by dividing it into multiple simpler ones. The local strategy has two advantages. First, simpler problems can be solved by simpler techniques, like transforming a global nonlinear problem into a local linear problem. Second, the training and testing can be more efficient, making the algorithm tractable for large-scale datasets.

As local MLC methods, Hierarchical Multi-Label Classification (HMC) [19, 1, 28] builds a hierarchy of single-label classifiers. Under the hierarchy constraint, the training data for each classifier is restricted so that it contains only the instances associated with parent labels. However, HMC's applications are limited on particular problems in text categorization and genomics analysis. Applying the same strategy of HMC, HOMER [25] breaks the constraint on the predefined label hierarchy. It builds the label hierarchy by recursively conducting balanced $k$-means on the label space, transforming the original task into a tree-shaped hierarchy of simpler tasks, each one relevant to a subset of labels. The local strategy is also applied by directly finding data clusters. CBMLC [14] partitions the original multi-label datasets into multiple small-scale datasets, on which multi-label classifiers are built individually. Given a test instance, it is feed only to the classifier corresponding to the nearest cluster. To speed up the $k$NN classification, SLEEC [2] partitions the original training data into several clusters, learning a local nonlinear embedding per cluster and conducting $k$NN only within the test sample's nearest cluster. On the other hand, in the regression setting, several regression tree based methods, RETIS [10], M5 system [17] and HTL [23], also employ the local strategy. Similar with the classical regression tree algorithm like CART [4], such methods divide the input space into mutually exclusive regions described by propositional assertions on the input features. The difference is that RETIS, M5 and HTL build several alternative regression models in the leaves of a tree to improve predictive accuracy. In [36], Regression Clustering partitions the original dataset into several subsets. Each regression is conducted on its own subset with a simpler distribution, leading to a better generalization ability.

Based on the above survey, we notice that seldom research works focus on local MLC methods. Motivated by the work of CBMLC [14], in this paper, we propose the Clustering-based Local MLC (CLMLC) method. In CLMLC, we assume a large-scale problem can be divided into a number of small or medium-scaled problems without loss of discriminative information. Different with CBMLC, CLMLC is built on a feature subspace and employs different local models for different data clusters.

# 3 THE CLMLC METHOD

In the scenario of MLC, an instance is typically represented by a pair $(\mathbf{x}, \mathbf{y})$, which contains a feature vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ and the corresponding label vector $\mathbf{y} \in \mathcal{Y} \subseteq \{0,1\}^L$, where $y_\ell = 1$ if and only if $\ell$-th label is associated with instance $\mathbf{x}$, and $y_\ell = 0$ otherwise, $\ell \in \{1, ..., L\}$. Assume that we are given a dataset of $N$ instances $\mathcal{S} = [\mathbf{X}_S, \mathbf{Y}_S]$, where $\mathbf{X}_S = [\mathbf{x}_1, ..., \mathbf{x}_N]^\intercal$ and $\mathbf{Y}_S = [\mathbf{y}_1, ..., \mathbf{y}_N]^\intercal$ denote the feature and label matrix, respectively. Given a testing dataset $\mathcal{T} = [\mathbf{X}_T, \mathbf{Y}_T]$, the task of MLC is to find an optimal classifier $h : \mathcal{X} \to \mathcal{Y}$ which assigns a label matrix $\hat{\mathbf{Y}}_T$ to test data $\mathbf{X}_T$ such that $h$ minimizes a loss function on $\hat{\mathbf{Y}}_T$ and $\mathbf{Y}_T$.

Now we present the proposed CLMLC method, which can scale to MLC problems in large $N$, $D$ and $L$. CLMLC comprises low-dimensional data clustering and local model learning.

## 3.1 Low-dimensional data clustering

We assume that a large-scale dataset could be decomposed into several smaller local sets. To this end, clustering analysis is introduced to find the local clusters. However, directly applying cluster analysis would probably produce unstable outputs and suffer from high computational cost, especially when the dimensionality of the original feature space is relatively high. In this sense, a dimensionality reduction approach is necessary as a pre-processing technique before applying clustering analysis.

Let $\mathbf{X}$ and $\mathbf{Y}$ be already centered so as to $\mathbf{X}^\intercal\mathbf{1} = \mathbf{0}$ and $\mathbf{Y}^\intercal\mathbf{1} = \mathbf{0}$. The Partial Least Squares (PLS) [30] finds the directions of maximum covariance between $\mathbf{X}$ and $\mathbf{Y}$ by Singular Value Decomposition (SVD) as follows:

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{X}^\intercal\mathbf{Y} - \mathbf{U}\mathbf{\Lambda}_d\mathbf{V}^\intercal\|_F^2, \tag{1}$$

where $\mathbf{\Lambda}_d$ is a diagonal matrix $(\lambda_1, \lambda_2, ..., \lambda_d)$ with the largest $d$ singular values of $\mathbf{X}^\intercal\mathbf{Y}$, and $\|\cdot\|_F$ denotes the Frobenius norm. This is also the solution of the maximization problem:

$$\begin{aligned} \max_{\mathbf{U},\mathbf{V}} \quad & \mathrm{Tr}(\mathbf{U}^\intercal\mathbf{X}^\intercal\mathbf{Y}\mathbf{V}) \\ \text{s.t.} \quad & \mathbf{U}^\intercal\mathbf{U} = \mathbf{V}^\intercal\mathbf{V} = \mathbf{I}_d. \end{aligned} \tag{2}$$

One of limitations of PLS is the lack of invariance to arbitrary linear transformations on $\mathbf{X}$ [32].

To overcome this limitation, Orthonormalized PLS (OPLS) [32] is proposed by orthonormalizing $\mathbf{X}$ to $\mathbf{X}(\mathbf{X}^\intercal\mathbf{X})^{-\frac{1}{2}}$ in (1), and we have

$$\min_{\mathbf{U},\mathbf{V}} \|(\mathbf{X}^\intercal\mathbf{X})^{-\frac{1}{2}}\mathbf{X}^\intercal\mathbf{Y} - \mathbf{U}\mathbf{\Lambda}_d\mathbf{V}^\intercal\|_F^2. \tag{3}$$

Similar with (2), (3) can be also rewritten to a maximization problem:

$$\begin{aligned} \max_{\mathbf{U}} \quad & \mathrm{Tr}(\mathbf{U}^\intercal\mathbf{X}^\intercal\mathbf{Y}\mathbf{Y}^\intercal\mathbf{X}\mathbf{U}) \\ \text{s.t.} \quad & \mathbf{U}^\intercal\mathbf{X}^\intercal\mathbf{X}\mathbf{U} = \mathbf{I}. \end{aligned} \tag{4}$$

The solution $\mathbf{U}$ consists of eigenvectors $\mathbf{u}$ corresponding to the largest $d$ eigenvalues of a generalized eigenvalue problem

$$(\mathbf{X}^\intercal\mathbf{Y}\mathbf{Y}^\intercal\mathbf{X})\mathbf{u} = \lambda(\mathbf{X}^\intercal\mathbf{X})\mathbf{u}. \tag{5}$$

To avoid the singularity of $\mathbf{X}^\intercal\mathbf{X}$ and reduce the model complexity, in practice a regularization term $\gamma\mathbf{I}$ with $\gamma > 0$ is commonly introduced to (5), leading to

$$(\mathbf{X}^\intercal\mathbf{Y}\mathbf{Y}^\intercal\mathbf{X})\mathbf{u} = \lambda(\mathbf{X}^\intercal\mathbf{X} + \gamma\mathbf{I})\mathbf{u}. \tag{6}$$

In general, directly solving the generalized eigenvalue problem (6) suffers from an expensive cost and thus might not scale to large-scale problems. In this study, we use an efficient two-stage approach [20] to address the problem. In the first stage, a penalized least squares problem is solved by regressing the centered feature matrix $\mathbf{X}$ to the centered label matrix $\mathbf{Y}$; after projecting $\mathbf{X}$ into the subspace by the regression, in the second stage, the resulting generalized eigenvalue problem is solved by SVD.

**Algorithm 1** Low-dimensional data clustering

**Input:** $\mathbf{X}$: centered data matrix, $\mathbf{Y}$: centered label matrix, $d$: size of feature subspace, $K$: number of data clusters
**Output:** $\mathbf{U}$: projection matrix, $\mathbf{R}, \mathbf{C}$: clustering output

1: Solve the least squares problem:

$$\min_{\mathbf{U}_1} \|\mathbf{X}\mathbf{U}_1 - \mathbf{Y}\|_F^2 + \|\mathbf{U}_1\|_F^2;$$

2: $\mathbf{H} = \mathbf{U}_1^\intercal \mathbf{X}^\intercal \mathbf{Y}$;
3: Decompose $\mathbf{H} = \mathbf{U}_H \Lambda_d \mathbf{U}_H^\intercal$ by SVD;
4: $\mathbf{U} = \mathbf{U}_1 \mathbf{U}_2$, where $\mathbf{U}_2 = \mathbf{U}_H \Lambda_d^{-\frac{1}{2}}$;
5: $[\mathbf{R}, \mathbf{C}] \leftarrow k$-means$(\mathbf{Z}, K)$ by (7), where $\mathbf{Z} = \mathbf{X}\mathbf{U}$.

Through (6), we find an orthonormal basis $[\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_d]$ to form $\mathbf{U}$. Therefore we can have a low-dimensional expression $\mathbf{z} \in \mathbb{R}^d$ by projection $\mathbf{z} = \mathbf{U}^\intercal \mathbf{x}$, $\mathbf{Z} = \mathbf{X}\mathbf{U}$ as well. Then we conduct clustering on $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N$ in the light of elimination of most of noisy features. In this paper, $k$-means is employed, aiming to approximately solve the following optimization problem:

$$\begin{aligned} \min_{\mathbf{R},\mathbf{C}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{K} r_{ij}\|\mathbf{z}_i - \mathbf{c}_j\|_2^2 \\ \text{s.t.} \quad & \forall i,\ \|\mathbf{r}_i\|_0 = 1,\ \|\mathbf{r}_i\|_1 = 1, \end{aligned} \tag{7}$$

where $\mathbf{R}$ represents the $N \times K$ indicator matrix, indicating the assignment from data points to centroids, while the centroid matrix $\mathbf{C} = [\mathbf{c}_1, ..., \mathbf{c}_K]^\intercal$, whose $\mathbf{c}_j = \sum_i r_{ij}\mathbf{x}_i / \sum_i r_{ij}$. $\|\cdot\|_0$, $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the $\ell_0$, $\ell_1$ and $\ell_2$ norm, respectively. In general, $k$-means is realized as an iterative algorithm. The pseudo code of low-dimensional data clustering is given in Algorithm 1.

## 3.2 Local model learning

In the second stage, we perform local model learning in each cluster. By expecting the existence of meta-labels, We use Laplacian eigenmap to learn meta-labels within each cluster, and then build classifier chains over meta-labels for local model learning. For each data cluster, we construct a graph $\mathbf{G} =< \mathbf{V}, \mathbf{E} >$ in the label space, where $\mathbf{V}$ is the vertex/label set, and $\mathbf{E}$ is the edge set containing edges between each label pair. Given an appropriate affinity matrix $\mathbf{A}$ on $\mathbf{E}$, meta-label learning can be considered as a graph cut problem: cutting the graph $\mathbf{G}$ into a set of sub-graphs.

For constructing affinity matrix $\mathbf{A}$, we use two different sources: the label space and the instance space. In this study, we utilize Jaccard index and heat kernel affinity to represent the label similarity and instance locality, respectively.

- Label similarity $\mathbf{A}^{(L)} = \{A_{\ell m}^{(L)}\}_{\ell,m=1}^{L}$,

$$A_{\ell m}^{(L)} \coloneqq \frac{\sum_{i=1}^{N} y_{i\ell} \cdot y_{im}}{\sum_{i=1}^{N}(y_{i\ell} + y_{im} - y_{i\ell} \cdot y_{im})}. \tag{8}$$

- Instance locality $\mathbf{A}^{(I)} = \{A_{\ell m}^{(I)}\}_{\ell,m=1}^{L}$,

$$A_{\ell m}^{(I)} \coloneqq \mathrm{e}^{-\|\boldsymbol{\mu}_\ell - \boldsymbol{\mu}_m\|_2^2},\ \text{where } \boldsymbol{\mu}_\ell = \frac{\sum_{i=1}^{N} \mathbf{z}_i \cdot y_{i\ell}}{\sum_{i=1}^{N} y_{i\ell}}. \tag{9}$$

By combining these two matrices, we obtain the following affinity matrix $\mathbf{A} = \{A_{\ell m}\}_{\ell,m=1}^{L}$,

$$A_{\ell m} \coloneqq \frac{1}{2}\left(A_{\ell m}^{(L)} + A_{\ell m}^{(I)}\right). \tag{10}$$

**Algorithm 2** Local model learning

**Input:** $\mathbf{Z}^{\mathbf{c}}$: local data matrix, $\mathbf{Y}^{\mathbf{c}}$: local label matrix, $n$: number of meta-labels, $\mathfrak{L}$: meta-label classifier
**Output:** $\mathbf{h}^{\mathbf{c}}$: local classifier

1: Compute $\mathbf{A}$ according to (10);
2: $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \mathrm{diag}(\sum_\ell A_{\ell m})$;
3: Solve $\mathbf{L}\mathbf{w} = \lambda \mathbf{D}\mathbf{w}$ by $n$ smallest eigenvalues;
4: $\mathbf{R}^{\mathbf{c}} \leftarrow k$-means$(\mathbf{W}, n)$, where $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_n]$;
5: **for** $k \in \{1, ..., n\}$ **do**
6: $\quad id = \mathrm{find}(\mathbf{R}^{\mathbf{c}} == k)$
7: $\quad h_k^{\mathbf{c}} \leftarrow \mathfrak{L}(\mathbf{Z}^{\mathbf{c}}, \mathbf{Y}^{\mathbf{c}}(:, id))$;
8: $\quad \mathbf{Z}^{\mathbf{c}} = \mathbf{Z}^{\mathbf{c}} \cup \mathbf{Y}^{\mathbf{c}}(:, id)$;
9: $\mathbf{h}^{\mathbf{c}} \leftarrow \{h_k^{\mathbf{c}}\}_{k=1}^{n}$.

To cut the graph $\mathbf{G}$ into $n$ sub-graphs ($n$ meta-labels) is equivalent to perform $k$-means on the $n$ smallest eigenvectors $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_n]$ of the generalized eigenvalue problem:

$$\mathbf{L}\mathbf{w} = \lambda \mathbf{D}\mathbf{w}, \tag{11}$$

where $\mathbf{D} = (D_{\ell\ell}) = (\sum_m A_{\ell m})$, and $\mathbf{L}$ is the Laplacian matrix, $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Thus, the label assignment to $n$ meta-labels is obtained by applying $k$-means on the rows of $\mathbf{W}$.

After finding meta-labels, a sophisticated multi-label classifier $\mathfrak{L}$ could be applied to capture the strong label correlations within each meta-label. On the other hand, to model relatively weak meta-label correlations, a simple MLC method is also necessary in the meta-label space. In this way, label correlations can be well captured without much time cost. To this end, we introduce an efficient classifier chains method [18] over the meta-label space. In general, for each meta-label within a meta-label chain, we expand its training data by taking previous meta-labels as extra features before feeding the data into $\mathfrak{L}$. The outline of local model learning is given in Algorithm 2.

## 3.3 Prediction

Given a test instance $\mathbf{x} \in \mathbf{X}_T$, the prediction can be made by two steps. Firstly, $\mathbf{x}$ is encoded into the feature subspace by $\mathbf{z} = \mathbf{U}^\intercal \mathbf{x}$. Secondly, the local classifier $\mathbf{h}^{\mathbf{c}}$ corresponding to $\mathbf{z}$'s nearest cluster $\mathbf{c}$ such as,

$$\mathbf{c} = \arg\min_{\mathbf{c} \in \mathbf{C}} \|\mathbf{z} - \mathbf{c}\|_2^2, \tag{12}$$

is activated to predict the label assignment by $\hat{\mathbf{y}} = \mathbf{h}^{\mathbf{c}}(\mathbf{z})$. Note that $\mathbf{C}$ in (12) is the centroid matrix obtained according to (7).

## 3.4 Remarks

The complete procedure of CLMLC, including training (Steps 1 to 5) and testing (Steps 6 to 8), is outlined in Algorithm 3. It is worth noting that CLMLC is able to serve as a *meta-strategy* for large-scale MLC problems. For example, other dimension reduction or clustering analysis techniques could be used to replace the OPLS or $k$-means in Algorithm 1, in order to handle specific problem settings or data patterns. Similarly, any MLC method can be directly applied for local model learning in Algorithm 2. It shows the high flexibility of CLMLC to address various MLC problems.

# 4 EXPERIMENTS

## 4.1 Datasets and evaluation metrics

In order to evaluate the performance of the proposed CLMLC method and other MLC methods, we conducted experiments on eigh-

**Algorithm 3** CLMLC

**Input:** $\mathbf{X}$: centered data matrix, $\mathbf{Y}$: centered label matrix, $\mathbf{x}$: test instance, $d$: size of feature subspace, $K$: number of data clusters, $n$: number of meta-labels, $\mathfrak{L}$: meta-label classifier

**Output:** $\hat{\mathbf{y}}$: predicted label set

**Training**:

$[\mathbf{U}, \mathbf{R}, \mathbf{C}] \leftarrow$ <Algorithm 1> $(\mathbf{X}, \mathbf{Y}, d, K)$;

$\mathbf{Z} = \mathbf{X}\mathbf{U}$;

**for** $\mathbf{c} \in \mathbf{C}$ **do**

  Find local dataset $[\mathbf{Z^c}, \mathbf{Y^c}]$ by $\mathbf{R}$;

  $\mathbf{h}^c \leftarrow$ <Algorithm 2> $(\mathbf{Z^c}, \mathbf{Y^c}, n, \mathfrak{L})$;

**Testing**:

$\mathbf{z} = \mathbf{U}^\intercal \mathbf{x}$;

Find $\mathbf{z}$'s nearest cluster $\mathbf{c}$ by (12);

$\hat{\mathbf{y}} \leftarrow \mathbf{h^c}(\mathbf{z})$;

teen benchmark datasets in Mulan [27], where nine datasets come from two data sources, Rcv1 and Corel16k. The statistics of the datasets are summarized in Table 1. For the convenience of parameter setting, we treat the first six sets as regular-scale datasets, and last twelve sets as large-scale datasets, respectively.

**Table 1**: The statistics of experimental multi-label datasets. "Card.", "Den." and "Dist." denote the label cardinality, label density and the number of distinct label combinations, respectively.

| Dataset | $N$ | $D$ | $L$ | Card. | Den. | Dist. | Domain |
|---|---|---|---|---|---|---|---|
| Birds | 645 | 260 | 19 | 1.014 | 0.053 | 133 | audio |
| Genbase | 662 | 1186 | 27 | 1.252 | 0.046 | 32 | biology |
| Medical | 978 | 1449 | 45 | 1.245 | 0.028 | 94 | text |
| Enron | 1702 | 1001 | 53 | 3.378 | 0.064 | 753 | text |
| Scene | 2407 | 294 | 6 | 1.074 | 0.179 | 15 | image |
| Yeast | 2417 | 103 | 14 | 4.237 | 0.303 | 198 | biology |
| Corel5k | 5000 | 499 | 374 | 3.522 | 0.009 | 1453 | image |
| Rcv1s1 | 6000 | 944 | 101 | 2.880 | 0.029 | 837 | text |
| Rcv1s2 | 6000 | 944 | 101 | 2.634 | 0.026 | 800 | text |
| Rcv1s3 | 6000 | 944 | 101 | 2.614 | 0.026 | 783 | text |
| Rcv1s4 | 6000 | 944 | 101 | 2.667 | 0.022 | 629 | text |
| Bibtex | 7395 | 1836 | 159 | 2.402 | 0.015 | 1654 | text |
| Corel16k1 | 13766 | 500 | 153 | 2.859 | 0.019 | 1791 | image |
| Corel16k2 | 13761 | 500 | 164 | 2.882 | 0.018 | 1782 | image |
| Corel16k3 | 13760 | 500 | 154 | 2.829 | 0.018 | 1718 | image |
| Corel16k4 | 13837 | 500 | 162 | 2.842 | 0.018 | 1760 | image |
| Corel16k5 | 13847 | 500 | 160 | 2.858 | 0.018 | 1784 | image |
| Delicious | 16105 | 500 | 983 | 19.020 | 0.019 | 3937 | text |

Given a test dataset $\mathcal{T} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_T}$, we use four evaluation metrics for the experimental results. Here $\mathbb{1}$ denotes the indicator function.

- **Exact-Match** $:= \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1}_{\hat{\mathbf{y}}_i = \mathbf{y}_i}$,
- **Hamming-Score** $:= \frac{1}{N_T} \sum_{i=1}^{N_T} \frac{1}{L} \sum_{\ell=1}^{L} \mathbb{1}_{\hat{y}_{i\ell} = y_{i\ell}}$,
- **Macro-F1** $:= \frac{1}{L} \sum_{\ell=1}^{L} \frac{2 \sum_{i=1}^{N_T} \hat{y}_{i\ell} \cdot y_{i\ell}}{\sum_{i=1}^{N_T} \hat{y}_{i\ell} + \sum_{i=1}^{N_T} y_{i\ell}}$,
- **Micro-F1** $:= \frac{2 \sum_{\ell=1}^{L} \sum_{i=1}^{N_T} \hat{y}_{i\ell} \cdot y_{i\ell}}{\sum_{\ell=1}^{L} \sum_{i=1}^{N_T} \hat{y}_{i\ell} + \sum_{\ell=1}^{L} \sum_{i=1}^{N_T} y_{i\ell}}$.

The above metrics can be cast into two categories, instance-based metrics (Exact-Match and Hamming-Score) and label-based metrics (Macro-F1 and Micro-F1 [33]). Exact-Match is the most stringent measure, since it does not evaluate partial match of labels. In spite of that, it is a good metric to measure how well label correlations are modeled. Hamming-Score emphasizes on the prediction accuracy on label-instance pairs, and is able to evaluate the performance on each single label. However, since Hamming-Score treats equally false positives ad false negatives, it is weak in imbalanced MLC problems. The label-based metrics overcome the limitations of the two instance-based metrics. Macro-F1 computes F1-Score locally over each label, which is more sensitive to the performance on the labels in minority. In contrast, Micro-F1 computes F1-Score globally over all labels, thus it tends to be influenced more by the labels in majority.

## 4.2 Configuration

The proposed **CLMLC** method was compared with four state-of-the-art MLC methods:

- **ECC** [18]: an ensemble of classifier chains, where chain orders are generated randomly. Each classifier of a single CC is trained by taking previously assigned labels as extra attributes.
- **MLHSL** [21]: an FS-DR MLC method. A dataset is encoded by mapping features into a subspace, and then an MLC method is built on the basis of the encoded dataset.
- **CPLST** [5]: an LS-DR MLC method. The label space is encoded by a feature-aware principal label space transformation, and the round-based decoding [5] is used to predict the label set.
- **CBMLC** [14]: a first attempt on applying clustering analysis on the dataset before feeding the data to a multi-label classifier.

ECC is adopted due to its superior performance compared with other MLC decomposition methods, such as BR [3] and CC [18], as shown in [18]. As global MLC methods, MLHSL is chosen as a representative of FS-DR methods, while CPLST is chosen by its performance advantage, especially in Hamming-Score, over several LS-DR methods, such as Compressive Sensing, PLST and orthogonally constraint CCA, as shown in [5]. As a local MLC method, CBMLC is selected for comparison in cluster analysis. Note that SLEEC [2] is excluded from the comparing methods, although it employs the similar local strategy with CLMLC. This is because SLEEC focuses on extreme MLC [15], where standard multi-label evaluation metrics like our four metrics are not appropriate.

In the experiments, *5-fold cross validation* was performed to evaluate the classification performance. For fair comparison, CC with ridge regression[1] was used as the baseline classifier for CBMLC, MLHSL, CPLST and CLMLC. In parameter setting, for CLMLC, we set the size of feature subspace $d$ by $\min\{L, 30\}$, and the number of clusters $K$ by 20/100 for regular/large-scale datasets, respectively. For a cluster $\mathbf{c}$, the number of meta-labels $n$ was set to $\lceil L^{\mathbf{c}}/5 \rceil$. CLMLC employed an ensemble of 2 CCs as the meta-label classifier $\mathfrak{L}$. ECC used an ensemble of 10 CCs. In addition, in order to scale up ECC, random sampling was applied to randomly select 75% of instances and 50% of features for building each CC in ECC, as recommended in [18]. CBMLC and MLHSL shared the same value of $K$ and $d$ with CLMLC, respectively. For CPLST, we set the ratio for LS-DR by 0.8/0.6 for regular/large-scale datasets, respectively. Note that the parameters were chosen for the comparing methods in order to balance the classification accuracy and execution time, according to the experimental results on conducting grid search in the parameter spaces (detailed discussion will be made in Section 4.4). We obtained the MATLAB codes of CPLST[1] and MLHSL[2] given by the authors, and implemented the MATLAB codes of ECC[3], CBMLC[3] and CLMLC[3] by ourselves. Experiments were performed in a computer configured with an Intel Quad-Core i7-4770 CPU at 3.4GHz with 4GB RAM.

[1] https://github.com/hsuantien/mlc_lsdr

[2] http://www.public.asu.edu/~jye02/Software/MLDR/

[3] https://github.com/futuresun912/CLMLC.git

**Table 2**: Experimental results (mean (rank)) on eighteen multi-label datasets in four evaluation metrics.

| Method | Exact-Match | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Birds | Genbase | Medical | Enron | Scene | Yeast | Corel5k | Rcv1s1 | Rcv1s2 | Rcv1s3 | Rcv1s4 | Bibtex | Corel16k1 | Corel16k2 | Corel16k3 | Corel16k4 | Corel16k5 | Delicious |
| ECC | 0.515 (3) | 0.974 (4) | 0.640 (3) | 0.121 (2) | 0.607 (2) | 0.197 (2) | 0.006 (3) | 0.098 (4) | 0.214 (4) | 0.216 (4) | 0.329 (2) | 0.157 (3) | 0.009 (3.5) | 0.007 (4) | 0.009 (3) | 0.008 (3) | 0.008 (3.5) | 0.001 (4.5) |
| MLHSL | **0.524 (1.5)** | **0.982 (1)** | 0.676 (2) | 0.120 (3) | 0.596 (3) | 0.196 (3) | 0.004 (5) | 0.114 (3) | 0.220 (3) | 0.219 (3) | 0.320 (4) | 0.119 (5) | 0.009 (3.5) | 0.008 (3) | 0.007 (4) | 0.007 (4) | 0.008 (3.5) | 0.002 (3) |
| CPLST | 0.502 (4) | 0.980 (2) | 0.583 (4) | 0.092 (5) | 0.479 (5) | 0.149 (5) | 0.005 (4) | 0.063 (5) | 0.176 (5) | 0.173 (5) | 0.290 (5) | 0.148 (4) | 0.007 (5) | 0.006 (5) | 0.006 (5) | 0.006 (5) | 0.007 (5) | 0.001 (4.5) |
| CBMLC | 0.375 (5) | 0.973 (5) | 0.578 (5) | 0.101 (4) | 0.553 (4) | 0.163 (4) | 0.012 (2) | 0.163 (2) | 0.255 (2) | 0.248 (2) | 0.326 (3) | 0.164 (2) | 0.016 (2) | 0.014 (2) | 0.017 (2) | 0.017 (2) | 0.015 (2) | **0.012 (1)** |
| CLMLC | **0.524 (1.5)** | 0.979 (3) | **0.688 (1)** | **0.147 (1)** | **0.627 (1)** | **0.205 (1)** | **0.029 (1)** | **0.224 (1)** | **0.316 (1)** | **0.319 (1)** | **0.400 (1)** | **0.172 (1)** | **0.030 (1)** | **0.029 (1)** | **0.030 (1)** | **0.028 (1)** | **0.030 (1)** | 0.004 (2) |
| avg. rank | **CLMLC (1.194)** ≻ CBMLC (2.833) ≻ MLHSL (3.194), ECC (3.194) ≻ CPLST (4.583) | | | | | | | | | | | | | | | | | |
| Method | Hamming-Score | | | | | | | | | | | | | | | | | |
| | Birds | Genbase | Medical | Enron | Scene | Yeast | Corel5k | Rcv1s1 | Rcv1s2 | Rcv1s3 | Rcv1s4 | Bibtex | Corel16k1 | Corel16k2 | Corel16k3 | Corel16k4 | Corel16k5 | Delicious |
| ECC | 0.951 (3) | **0.999 (3)** | 0.989 (2) | 0.933 (3) | **0.896 (1)** | 0.793 (2) | 0.990 (2.5) | **0.973 (2)** | **0.977 (2)** | **0.977 (2)** | **0.982 (1.5)** | **0.988 (1.5)** | **0.981 (2)** | 0.982 (2.5) | **0.982 (2)** | 0.982 (2,5) | **0.982 (2)** | 0.981 (2.5) |
| MLHSL | **0.954 (1.5)** | **0.999 (3)** | **0.990 (1)** | 0.936 (2) | 0.875 (4) | 0.786 (3) | 0.990 (2.5) | **0.973 (2)** | **0.977 (2)** | **0.977 (2)** | 0.981 (3) | 0.986 (4) | **0.981 (2)** | 0.982 (2.5) | **0.982 (2)** | 0.982 (2.5) | **0.982 (2)** | 0.981 (2.5) |
| CPLST | 0.950 (4) | **0.999 (3)** | 0.986 (5) | 0.911 (5) | 0.887 (2) | **0.797 (1)** | **0.991 (1)** | **0.973 (2)** | **0.977 (2)** | **0.977 (2)** | **0.982 (1.5)** | **0.988 (1.5)** | **0.981 (2)** | **0.983 (1)** | **0.982 (2)** | **0.983 (1)** | **0.982 (2)** | **0.982 (1)** |
| CBMLC | 0.887 (5) | **0.999 (3)** | 0.987 (4) | 0.930 (4) | 0.869 (5) | 0.750 (5) | 0.988 (4) | 0.966 (5) | 0.971 (5) | 0.969 (5) | 0.976 (5) | 0.986 (4) | 0.972 (5) | 0.976 (5) | 0.975 (5) | 0.975 (5) | 0.975 (5) | 0.976 (5) |
| CLMLC | **0.954 (1.5)** | **0.999 (3)** | 0.988 (3) | **0.940 (1)** | 0.885 (3) | 0.779 (4) | 0.986 (5) | 0.969 (4) | 0.973 (4) | 0.973 (4) | 0.979 (4) | 0.986 (4) | 0.977 (4) | 0.979 (4) | 0.978 (4) | 0.979 (4) | 0.979 (4) | 0.979 (4) |
| avg. rank | **CPLST (2.167)**, **ECC (2.167)** ≻ MLHSL (2.417) ≻ CLMLC (3.583) ≻ CBMLC (4.667) | | | | | | | | | | | | | | | | | |
| Method | Macro-F1 | | | | | | | | | | | | | | | | | |
| | Birds | Genbase | Medical | Enron | Scene | Yeast | Corel5k | Rcv1s1 | Rcv1s2 | Rcv1s3 | Rcv1s4 | Bibtex | Corel16k1 | Corel16k2 | Corel16k3 | Corel16k4 | Corel16k5 | Delicious |
| ECC | 0.290 (3) | 0.725 (5) | 0.340 (4) | **0.196 (1)** | **0.703 (1)** | 0.354 (4) | 0.014 (4) | 0.118 (4) | 0.131 (3) | 0.108 (3.5) | 0.109 (3) | 0.193 (3) | 0.014 (4.5) | 0.016 (4) | 0.017 (4) | 0.010 (4.5) | 0.013 (4) | 0.034 (4) |
| MLHSL | 0.302 (2) | **0.767 (1)** | 0.354 (3) | 0.160 (4) | 0.648 (4) | 0.354 (3) | 0.010 (5) | 0.104 (5) | 0.096 (5) | 0.091 (5) | 0.089 (5) | 0.095 (5) | 0.014 (4.5) | 0.014 (5) | 0.014 (5) | 0.010 (4.5) | 0.012 (5) | 0.025 (5) |
| CPLST | 0.287 (4) | 0.761 (2.5) | **0.374 (1)** | 0.167 (3) | 0.639 (5) | 0.351 (5) | 0.016 (3) | 0.125 (3) | 0.110 (4) | 0.108 (3.5) | 0.108 (4) | 0.186 (4) | 0.015 (3) | 0.018 (3) | 0.021 (3) | 0.013 (3) | 0.015 (3) | 0.048 (3) |
| CBMLC | 0.188 (5) | 0.738 (4) | 0.312 (5) | 0.195 (2) | 0.655 (3) | **0.418 (1)** | 0.032 (2) | 0.204 (2) | 0.195 (2) | 0.185 (2) | 0.176 (2) | **0.257 (1)** | **0.068 (1)** | **0.063 (1)** | **0.058 (1)** | **0.070 (1)** | **0.060 (1)** | **0.143 (1)** |
| CLMLC | **0.369 (1)** | 0.761 (2.5) | 0.358 (2) | 0.153 (5) | 0.689 (2) | 0.400 (2) | **0.038 (1)** | **0.215 (1)** | **0.210 (1)** | **0.198 (1)** | **0.189 (1)** | 0.210 (2) | 0.056 (2) | 0.057 (2) | 0.053 (2) | 0.051 (2) | 0.047 (2) | 0.067 (2) |
| avg. rank | **CLMLC (1.861)** ≻ CBMLC (2.056) ≻ CPLST (3.333) ≻ ECC (3.528) ≻ MLHSL (4.222) | | | | | | | | | | | | | | | | | |
| Method | Micro-F1 | | | | | | | | | | | | | | | | | |
| | Birds | Genbase | Medical | Enron | Scene | Yeast | Corel5k | Rcv1s1 | Rcv1s2 | Rcv1s3 | Rcv1s4 | Bibtex | Corel16k1 | Corel16k2 | Corel16k3 | Corel16k4 | Corel16k5 | Delicious |
| ECC | 0.440 (4) | 0.990 (3) | 0.800 (2) | **0.499 (1)** | **0.694 (1)** | **0.642 (1)** | 0.126 (4) | 0.325 (4) | 0.356 (4) | 0.350 (4) | 0.430 (3) | 0.381 (4) | 0.092 (3) | 0.079 (4) | 0.076 (4) | 0.077 (4) | 0.073 (4.5) | 0.096 (4) |
| MLHSL | 0.452 (2) | **0.992 (1.5)** | **0.812 (1)** | 0.483 (2) | 0.640 (4) | 0.627 (4) | 0.140 (3) | 0.310 (5) | 0.330 (5) | 0.317 (5) | 0.392 (5) | 0.279 (5) | 0.089 (4) | 0.084 (3) | 0.065 (5) | 0.085 (3) | 0.090 (3) | 0.063 (5) |
| CPLST | 0.450 (3) | **0.992 (1.5)** | 0.756 (4) | 0.414 (5) | 0.635 (5) | 0.631 (3) | 0.106 (5) | 0.349 (3) | 0.371 (3) | 0.365 (3) | 0.440 (2) | 0.382 (3) | 0.070 (5) | 0.078 (5) | 0.079 (3) | 0.070 (5) | 0.073 (4.5) | 0.194 (3) |
| CBMLC | 0.265 (5) | 0.988 (4) | 0.740 (5) | 0.463 (4) | 0.641 (3) | 0.581 (5) | 0.151 (2) | 0.371 (2) | 0.387 (2) | 0.376 (2) | 0.426 (4) | 0.393 (2) | 0.163 (2) | 0.157 (2) | 0.154 (2) | **0.161 (1)** | **0.156 (1)** | **0.268 (1)** |
| CLMLC | **0.474 (1)** | 0.987 (5) | 0.782 (3) | 0.480 (3) | 0.676 (2) | 0.632 (2) | **0.173 (1)** | **0.401 (1)** | **0.423 (1)** | **0.422 (1)** | **0.472 (1)** | **0.396 (1)** | **0.164 (1)** | **0.164 (1)** | **0.160 (1)** | 0.157 (2) | 0.148 (2) | 0.214 (2) |
| avg. rank | **CLMLC (1.722)** ≻ CBMLC (2.722) ≻ ECC (3.250) ≻ MLHSL (3.639) ≻ CPLST (3.667) | | | | | | | | | | | | | | | | | |

## 4.3 Experimental results

Experimental results of five comparing MLC methods on benchmark datasets are reported in Table 2, where the averaged rank of each method over all datasets is shown in the last row of each metric. For each evaluation metric, the larger the value, the better the performance. Among the five comparing methods, the best performance is highlighted in boldface.

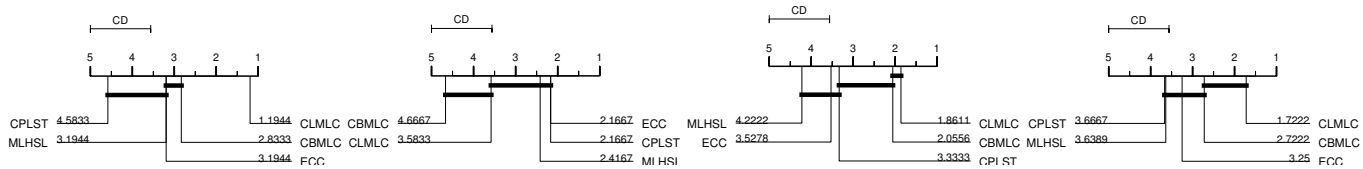
For all the 72 configurations (18 datasets $\times$ 4 evaluation metrics), CLMLC ranked 1st among five comparing MLC methods at $37.8\%$ cases, ranked 2nd at $18.9\%$ cases, and ranked 5th at only $3.3\%$ cases, which was remarkably better than the other methods. Specifically, CLMLC outperformed the other methods in Exact-Match (ranked 1st at $88.9\%$ cases) and Micro-F1 (ranked 1st at $55.6\%$ cases), and was competitive in terms of Macro-F1 (ranked 1st/2nd at $33.3\%/61.1\%$ cases). It demonstrates the effectiveness of the clustering-based local strategy adopted in CLMLC. The similar instances with similar label sets can be grouped together by CLMLC, leading to its strong capability on modeling label correlations and thus superior performance in Exact-Match. However, such grouped local data sometimes weaken the influence of minority labels, resulting in the worse performance of CLMLC in Hamming-Score (ranked 4nd at $66.7\%$ cases). CPLST and ECC performed better than the other methods in Hamming-Score (ranked 1st at $38.9\%$ and $16.7\%$ cases, respectively), since it is designed to be optimized in Hamming-Score, according to the theoretical analysis in [5]. In Hamming-Score, MLHSL ranked in 1st/2nd place at $11.1\%/61.1\%$ cases, but performed worse in other metrics, especially on large-scale datasets. It is probably because large-scale datasets typically need a sufficient number of instances for training, while FS-DR tends to remove too many features. CBMLC outperformed other methods except CLMLC in Exact-Match (ranked 1st/2nd at $5.6\%/55.5\%$ cases), Macro-F1 (ranked 1st/2nd at $44.4\%/33.3\%$ cases) and Micro-F1 (ranked 1st/2nd at $16.7\%/44.4\%$ cases), but worked worst in Hamming-Score (ranked 5th at $72.2\%$ cases). In addition, CBMLC worked worse than CLMLC on the average in all the four metrics, indicating that cluster analysis should be applied after appropriate feature dimension reduction. Note that the two local MLC methods, CLMLC and CBMLC, worked remarkably better than ECC, MLHSL and CPLST in terms of Exact-Match, Macro-F1 and Micro-F1 on the twelve large-scale datasets, demonstrating the superiority of local MLC strategy on real-world problems.

The execution time on seven large-scale datasets, including both training and prediction time, is reported in Table 3. The least time cost is highlighted in boldface. Among all the methods, MLHSL needed the least execution time on the average due to the low-dimensional feature subspace induced by FS-DR. CLMLC consumed the second least time on the average. Note that, CLMLC paid only slightly higher time cost than MLHSL on the Corel16k datasets. On datasets with large number of labels (large values in $L$), like delicious, CLMLC consumed more execution time than MLHSL and CPLST. Benefiting from LS-DR, CPLST cost the third least execution time, which was significantly less than ECC and CBMLC. But such superiority of CPLST decreased as the number of features increased (large values in $D$), like Bibtex. Due to its clustering analysis directly applied on high-dimensional datasets, CBMLC consumed the second largest time on all the datasets. ECC consumed the largest time on all the seven datasets, resulting from the ensemble strategy. In summary, the proposed CLMLC is one of the best choices for MLC in the balance of performance and execution time, especially when Exact-Match or Macro/Micro-F1 is the principal goal and the practical processing speed is required in a large-scale problem.

**Table 3**: Execution time ($10^3$sec) over seven large-scale datasets.

| | Corel5k | Rcv1s1 | Rcv1s2 | Bibtex | Corel16k1 | Corel16k2 | Delicious |
|---|---|---|---|---|---|---|---|
| ECC | 0.353 | 0.190 | 0.187 | 1.285 | 0.229 | 0.252 | 6.042 |
| MLHSL | 0.018 | **0.004** | **0.004** | 0.015 | **0.008** | **0.009** | **0.528** |
| CPLST | 0.042 | 0.045 | 0.036 | 0.223 | 0.042 | 0.042 | 0.558 |
| CBMLC | 0.097 | 0.112 | 0.127 | 1.002 | 0.131 | 0.151 | 1.916 |
| CLMLC | **0.005** | **0.004** | **0.004** | **0.014** | 0.010 | 0.010 | 0.567 |

To derive a more objective insistence on the experimental results, we conducted *Friedman test* [7] with significance level 0.05 (5 methods, 18 datasets). The results are shown in Table 4. Since the values of the Friedman Statistic $F_F$ in terms of all metrics were higher than the Critical Value, the null hypothesis of equal performance was rejected. Then, we proceeded to a *Nemenyi testing* to confirm the difference between any two methods. According to [7], the performance of two methods is regarded as significantly different if their average ranks differ by at least the Critical Difference (CD). Figure 1 shows the CD diagrams for four evaluation metrics at 0.05 significance level. In each subfigure, the value of CD is given as a rule above the axis, where the averaged rank is marked. In Figure 1, the algorithms which are not significantly different are connected by a




(a) Exact-Match (b) Hamming-Score (c) Macro-F1 (d) Micro-F1

**Figure 1**: CD diagrams (0.05 significance level) of five comparing methods.

thick line. In summary, among 90 comparisons (5 methods $\times$ 18 datasets), CLMLC achieved statistically superior performance than all the other methods in terms of Exact-Match. In Macro/Micro-F1, CLMLC achieved statistically comparable performances with CBMLC, and statistically superior performances than ECC, MLHSL and CPLST. Such observation demonstrates the competing performance of the proposed CLMLC in Exact-Match and Macro/Micro-F1, compared with the state-of-the-art MLC methods.

**Table 4**: Results of the Friedman Statistics $F_F$ (5 methods, 18 datasets) and the Critical Value (0.05 significance level). The null hypothesis as the equal performance is rejected, if the values of $F_F$ in terms of all metrics are higher than the Critical Value.

| Friedman Test | Exact Match | Hamming Score | Macro F1 | Micro F1 |
|---|---|---|---|---|
| $F_F$ | 24.166 | 15.992 | 11.680 | 6.051 |
| Critical Value | | 2.507 | | |

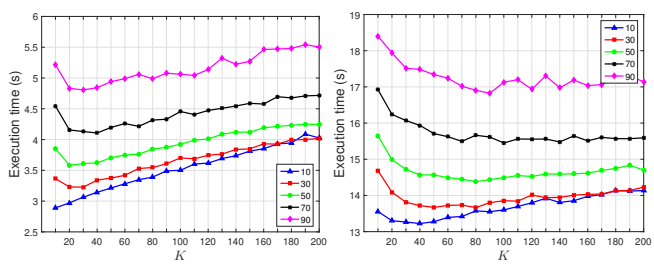
Table 5 reports the reduced sizes of training datasets in CLMLC, which are averaged by 5-fold cross validation. Here "std." shows the standard deviation of the values from $K$ clusters. As shown in Table 5, consistently with our previous assumptions, there is strong locality in datasets, especially on datasets in text domain, like Medical, Rcv1 and Bibtex, where $\overline{L}^{\mathbf{c}} \ll L$ in each data cluster $\mathbf{c}$. Indeed the problem sizes in terms of $N$, $D$ and $L$ have been significantly reduced. For example, in Bibtex, the average problem size ($\overline{N}^{\mathbf{c}} \times d \times \overline{L}^{\mathbf{c}}$) in each cluster $\mathbf{c}$ has been reduced to nearly $1/30000$ by CLMLC compared with the original set, bringing the fastest execution time on Bibtex (Table 3).

**Table 5**: Problem sizes of training datasets in CLMLC. The values were averaged by 5-fold cross validation. Here "std." denotes the standard deviation.

| Dataset | Original size | | | Reduced size | | | |
|---|---|---|---|---|---|---|---|
| | $N$ | $D$ | $L$ | $\overline{N}^{\mathbf{c}} \pm$ std. | $d$ | $\overline{L}^{\mathbf{c}} \pm$ std. | $K$ |
| Birds | 516 | 260 | 19 | 25.80±45.36 | 19 | 7.24±2.93 | 20 |
| Genbase | 530 | 1186 | 27 | 26.48±45.28 | 27 | 2.78±2.72 | 20 |
| Medical | 782 | 1449 | 45 | 39.12±39.47 | 30 | 4.68±2.97 | 20 |
| Enron | 1362 | 1001 | 53 | 68.08±66.81 | 30 | 23.85±6.54 | 20 |
| Scene | 1926 | 294 | 6 | 96.28±30.61 | 6 | 4.75±1.33 | 20 |
| Yeast | 1934 | 103 | 14 | 96.68±18.49 | 14 | 13.31±0.69 | 20 |
| Corel5k | 4000 | 499 | 374 | 40.00±18.18 | 30 | 54.08±25.19 | 100 |
| Rcv1s1 | 4800 | 944 | 101 | 48.00±28.92 | 30 | 19.54±12.62 | 100 |
| Rcv1s2 | 4800 | 944 | 101 | 48.00±31.34 | 30 | 18.51±11.78 | 100 |
| Rcv1s3 | 4800 | 944 | 101 | 48.00±30.69 | 30 | 18.13±12.00 | 100 |
| Rcv1s4 | 4800 | 944 | 101 | 48.00±33.80 | 30 | 14.36±9.94 | 100 |
| Bibtex | 5916 | 1836 | 159 | 59.16±39.44 | 30 | 29.95±20.41 | 100 |
| Corel16k1 | 11013 | 500 | 164 | 110.13±42.59 | 30 | 71.31±22.18 | 100 |
| Corel16k2 | 11009 | 500 | 164 | 110.09±48.86 | 30 | 71.32±24.37 | 100 |
| Corel16k3 | 11008 | 500 | 154 | 110.08±44.93 | 30 | 69.11±22.39 | 100 |
| Corel16k4 | 11070 | 500 | 162 | 110.70±48.46 | 30 | 70.73±22.91 | 100 |
| Corel16k5 | 11078 | 500 | 160 | 110.78±46.55 | 30 | 72.82±23.64 | 100 |
| Delicious | 12884 | 500 | 983 | 128.84±155.55 | 30 | 333.48±200.60 | 100 |

## 4.4 Parameter sensitivity analysis

To evaluate the potentiality of CLMLC, a parameter sensitivity analysis was conducted. First, the parameters $d$ and $K$ were dealt with the Rcv1s1 and Bibtex datasets, where $d$ controls the dimensionality of the feature subspace, and $K$ is the number of data clusters. In this experiment, we kept the value of $n$ by $\lceil L^{\mathbf{c}}/5 \rceil$, and increased $d$ from 5 to 100 by step 5, and $K$ from 10 to 200 by step 10. Figure 2 shows the experimental results in terms of four evaluation metrics, whose values are averaged by 5-fold cross validation. In Figure 2, the warmer the color, the better the performance. We observe that as the values of $d$ and $K$ increased, its performance in Exact-Match and Macro/Micro-F1 upgraded, and then became stable once $d$ and $K$ reached 30 and 100, respectively. In contrast, as the values of $d$ and $K$ increased, its performance in Hamming-Score degraded, although the change was very slight (within $0.5\%$). Figure 3 shows the execution time for parameter sensitivity analysis, where $d \in \{10, 30, 50, 70, 90\}$. On both two datasets, the execution time increased as the value of $d$ increased. As the value of $K$ increased, the execution time first decreased, and then increased on Rcv1s11 but became stable on Bibtex. Thus, by considering trade-off between classification accuracy and execution time, we set values of $d$ and $K$ to those as stated in Section 4.2.




(a) Rcv1s1 (b) Bibtex

**Figure 3**: The execution time (sec) over different values of the dimensionality $d$ of feature subspace and the number $K$ of clusters on the Rcv1s1 and Bibtex datasets.

Next, keeping the values of $d$ and $K$ to 30 and 100, we conducted a sensitivity analysis over $n$, where $n$ is the number of meta-labels for each cluster. Instead of directly varying the value of $n$, we increased $x$ from 2 to 20 by step 1 as $n = \lceil L^c/x \rceil$. Figure 4 shows the experimental results in four metrics averaged by 5-fold cross validation. For convenience, the values of each metric were normalized by its maximum. As the value of $x$ increased, the performance increased in Macro-F1, but decreased in Exact-Match. Note that performance in Hamming-Score seemed irrelevant to the change of $x$'s value. Thus, it was suggested to set smaller/larger value of $x$ if the objective is to optimize Exact-Match/Macro-F1.

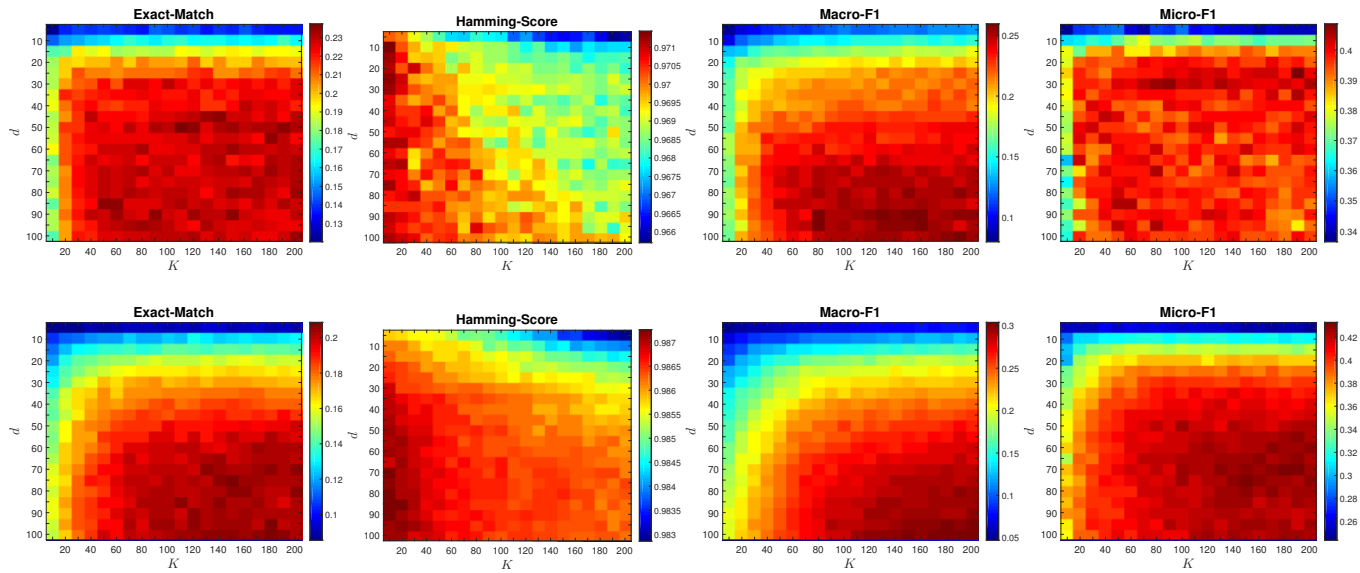
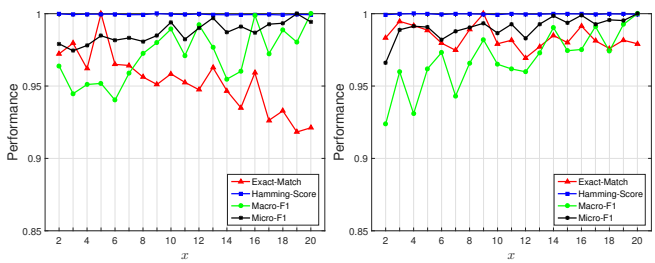



**Figure 2**: Parameter sensitivity analysis over the dimensionality $d$ of feature subspace and the number $K$ of clusters on the Rcv1s1 (the top row) and Bibtex (the bottom row) datasets ($n = \lceil L^{\mathbf{c}}/5 \rceil$). The size of $d/K$ was increased from 5/10 to 100/200 by step 5/10.




(a) Rcv1s1 (b) Bibtex

**Figure 4**: Parameter sensitivity analysis over $x$ ($n = \lceil L^{\mathbf{c}}/x \rceil$) on the Rcv1s1 and Bibtex datasets ($d = 30$, $K = 100$). The values of each metric were normalized by its maximum.

To optimize the parameters of MLHSL, CPLST and CBMLC, another set of parameter sensitivity analysis has been performed individually. Specifically, for MLHSL, $d$ shared the similar tendency with CLMLC. For CPLST, the ratio of LS-DR remarkably influenced the experimental results. As the ratio increased, its performance upgraded. As the ratio approached 0.8/0.6 on regular/large-scale datasets, the performance became stable, while execution time increased dramatically. For CBMLC, as the number of cluster $K$ increased, the values of evaluation metrics, except Hamming-Score, increased and became stable as $K$ approached 100. Such observations validate the effectiveness of parameter configurations in Section 4.2.

## 5 CONCLUSION

In this paper, we have proposed a Clustering-based Local Multi-Label Classification (CLMLC) method, relying on the assumption that a multi-label dataset can be decomposed into several datasets of smaller sizes, where meta-labels exist and are relevant to only a fraction of features and training data. In CLMLC, by applying clustering analysis on the feature subspace, similar instances associated with similar labels are grouped together and then fed into local models. Extensive experiments conducted on real-world benchmark datasets verified the validity of our assumption and demonstrated the efficiency of CLMLC. For the future work, we will seek a more appropriate method for building local models, which is currently a bottleneck for the application of CLMLC on extreme multi-label datasets.

## ACKNOWLEDGEMENTS

This work was partially supported by JSPS KAKENHI Grant Numbers 15H02719 and China Scholarship Council.

## REFERENCES

[1] Zafer Barutcuoglu, Robert E. Schapire, and Olga G. Troyanskaya, 'Hierarchical multi-label prediction of gene function', *Bioinformatics*, **22**(7), 830–836, (2006).
[2] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, 'Sparse local embeddings for extreme multi-label classification', in *Advances in Neural Information Processing Systems 28*, pp. 730–738, (2015).
[3] M. Boutell, J. Luo, X. Shen, and C. Brown, 'Learning multi-label scene classification', *Pattern Recognition*, **37**(9), 1757–1771, (2004).
[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth and Brooks, Monterey, CA, 1984.
[5] Y. Chen and H. Lin, 'Feature-aware label space dimension reduction for multi-label classification', in *Advances in Neural Information Processing Systems 25*, 1529–1537, (2012).
[6] Krzysztof Dembczynski, Willem Waegeman, and Eyke Hullermeier, 'An analysis of chaining in multi-label classification', in *Proceedings of the 20th European Conference on Artificial Intelligence*, pp. 294–299, (2012).
[7] Janez Demšar, 'Statistical comparisons of classifiers over multiple data sets', *Journal of Machine Learning Research*, **7**, 1–30, (2006).
[8] J. Fürnkranz, E. Hüllermeier, E.L. Mencia, and K. Brinker, 'Multilabel classification via calibrated label ranking', *Machine Learning*, **73**(2), 133–153, (2008).
[9] Jun Huang, Guorong Li, Qingming Huang, and Xindong Wu, 'Learning label specific features for multi-label classification', in *2015 IEEE International Conference on Data Mining, 2015*, pp. 181–190, (2015).
[10] Aram Karalič, 'Employing linear regression in regression tree leaves', in *Proceedings of the 10th European Conference on Artificial Intelligence*, pp. 440–441, (1992).

[11] I. Katakis, G. Tsoumakas, and Vlahavas I., 'Multilabel text classification for automated tag suggestion', in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2008 Discovery Challenge*, (2008).
[12] J. Langford, T. Zhang, D. Hsu, and S. Kakade, 'Multi-label prediction via compressed sensing', in *Advances in Neural Information Processing Systems 22*, 772–780, (2009).
[13] Z. Lin, G. Ding, M. Hu, and J. Wang, 'Multi-label classification via feature-aware implicit label space encoding', in *Proceedings of the 31st International Conference on Machine Learning*, pp. 325–333, (2014).
[14] G. Nasierding, G. Tsoumakas, and A. Kouzani, 'Clustering based multi-label classification for image annotation and retrieval', in *IEEE International Conference on Systems, Man and Cybernetics*, pp. 4514–4519, (2009).
[15] Y. Prabhu and M. Varma, 'Fastxml: a fast, accurate and stable tree-classifier for extreme multi-label learning', in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 263–272, (2014).
[16] G. Qi, X. Hua, Y. Rui, J. Tang, T. Mei, and H. Zhang, 'Correlative multi-label video annotation', in *Proceedings of the 15th ACM International Conference on Multimedia*, pp. 17–26, (2007).
[17] J. R. Quinlan, 'Learning with continuous classes', in *Proceedings of the Australian Joint Conference on Artificial Intelligence*, pp. 343–348. World Scientific, (1992).
[18] J. Read, B. Pfahringer, G. Holmes, and E. Frank, 'Classifier chains for multi-label classification', *Machine Learning*, **85**(3), 333–359, (2011).
[19] Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor, 'Learning hierarchical multi-category text classification models', in *Proceedings of the 22nd International Conference on Machine Learning*, pp. 744–751, (2005).
[20] L. Sun, B. Ceran, and J. Ye, 'A scalable two-stage approach for a class of dimensionality reduction techniques', in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 313–322, (2010).
[21] L. Sun, S. Ji, and J. Ye, 'Hypergraph spectral learning for multi-label classification', in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 668–676, (2008).
[22] L. Sun, S. Ji, and J. Ye, 'Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **33**(1), 194–200, (2011).
[23] Luís Torgo, 'Functional models for regression tree leaves', in *Proceedings of the 14th International Conference on Machine Learning, Nashville, Tennessee, USA, July 8-12, 1997*, pp. 385–393, (1997).
[24] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, 'Multi-label classification of music into emotions', in *Proceedings of the 9th International Conference on Music Information Retrieval*, pp. 325–330, (2008).
[25] G. Tsoumakas, I. Katakis, and I. Vlahavas, 'Effective and efficient multilabel classification in domains with large number of labels', in *Proceedings of ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, (2008).
[26] 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).
[27] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, 'Mulan: A java library for multi-label learning', *Journal of Machine Learning Research*, **12**, 2411–2414, (2011).
[28] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel, 'Decision trees for hierarchical multi-label classification', *Machine Learning*, **73**(2), 185–214, (2008).
[29] H. Wang, C. Ding, and H. Huang, 'Multi-label linear discriminant analysis', in *Proceedings of the 11th European Conference on Computer Vision*, volume 6316, 126–139, (2010).
[30] D. Watkins, *Chemometrics, mathematics and statistics in chemistry*, Reidel Publishing Company, Dordrecht, N etherlands, 1984.
[31] J. Weston, S. Bengio, and N. Usunier, 'Wsabie: Scaling up to large vocabulary image annotation', in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 2764–2770, (2011).
[32] K. Worsley, J. Poline, K. Friston, and A. Evans, 'Characterizing the response of PET and fMRI data using multivariate linear models', *Neuroimage*, **6**(4), 305–319, (1997).
[33] Yiming Yang and Xin Liu, 'A re-examination of text categorization methods', in *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pp. 42–49, New York, NY, USA, (1999). ACM.
[34] H. Yu, P. Jain, P. Kar, and S. Dhillon, 'Large-scale multi-label learning with missing labels', in *Proceedings of the 31st International Conference on Machine Learning*, pp. 593–601, (2014).
[35] Kai Yu, Shipeng Yu, and Volker Tresp, 'Multi-label informed latent semantic indexing', in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pp. 258–265, (2005).
[36] B. Zhang, 'Regression clustering', in *Proceedings of the 3rd IEEE International Conference on Data Mining*, pp. 451–458, (2003).
[37] M. Zhang and L. Wu, 'Lift: multi-label learning with label-specific features', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**(1), 107–120, (2015).
[38] Y. Zhang and Z. Zhou, 'Multi-label dimensionality reduction via dependence maximization', in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pp. 1503–1505, (2008).