International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988Volume 11 (2019) pp. 161-169
© MIR Labs,www.mirlabs.net/ijcisim/index.html

Received: 19 Nov, 2018; Accepted: 15 Jun, 2019; Published: 4 July, 2019

A Novel Algorithm for Multi-label Classification by Exploring Feature and Label Dissimilarities

Vaishali S. Tidake[1], Shirish S. Sane[2]

[1] Research Scholar, Matoshri College of Engineering and Research Center, Nashik, India
Dept. of Computer Engineering, MVPS's KBT College of Engineering, Nashik, India
Savitribai Phule Pune University
tidake.vaishali@kbtcoe.org

[2]Dept. of Computer Engineering, K. K. Wagh Institute of Engineering Education and Research, Nashik, India
Savitribai Phule Pune University
sssane@kkwagh.edu.in

***Abstract*: Selection of appropriate nearest neighbors greatly affects predictive accuracy of nearest neighbor classifier. Feature similarity is often used to decide the set of k nearest neighbors. Predictive accuracy of multi-label kNN could further be enhanced if in addition to the feature similarity, difference in feature values and dissimilarity of the instance labels are also taken into account to decide the set of k nearest neighbors. This paper deals with an algorithm called "ML-FLD" that not only takes into account features similarity of the instances, but also considers feature difference and label dissimilarity in order to decide the k nearest neighbors of a given unseen instance for the prediction of its labels. The algorithm when tested using well-known datasets and checked with the existing well known algorithms, provides better performance in terms of example-based metrics such as hamming loss, ranking loss, one error, coverage, average precision, accuracy, F measure as well as label-based metrics like macro-averaged and micro-averaged F measure.**

Keywords: classification, multi-label, algorithm adaptation, feature similarity, label dissimilarity, k nearest neighbors

I. Introduction

Classification is very commonly used tasks in data mining. It is referred to as *supervised learning.* Many applications like Text categorization (TC), direct marketing, bioinformatics, discovery of drugs, tag recommendation, prediction of gene function, etc. use supervised learning [1, 2, 3, 4, 11]. Text categorization classifies documents according to their contents. Many a times, a document to be categorized has multiple semantic meanings. Hence unlike traditional classification, such documents may belong to a set of predefined categories and therefore associated with more than one class labels.

An approach discussed by Zhang and Zhou in [25] follows an *algorithm adaptation* approach and is a modified version of well-known nearest neighbor algorithm. It is used by several researchers to perform multi-label classification. It utilizes feature similarity to determine nearest neighbors [8, 10, 11, 12, 13]. In case of multi-label classification, since the instances are associated with multiple labels, label dissimilarity may also help for better determination of set of nearest neighbors.

In this paper, a novel algorithm adaptation approach called ML-FLD is presented. ML-FLD not only takes into account features but also labels of instances to determine nearest neighbors while assigning weights to the neighbors. When the features of two instances are similar, chances of its selection as the nearest neighbor is high. However, if the labels of such instances are dissimilar, chances of its selection as nearest neighbor is low. Experimentation presented shows the importance of use of both features and labels to improve performance of the classifier.

Remaining paper is ordered as follows. Section II deals with the related work while the details of ML-FLD are in section III. Section IV presents the experimental work and the concluding comments are in section V.

II. Related Work

From the last two decades, many researchers are working in the area of multi-label classification as many applications are found to exhibit multiple semantic meanings [1, 2, 3, 4, 11]. As reported in [12, 13, 14, 15, 16, 17, 18, 25, 28], transformation and algorithm adaptation are the general approaches to build multi-label classifier.

BR (Binary Relevance) [5, 6, 7, 8, 22, 28] classifiers first transform the multi-label data to the single label data and then for each label, they build classifier using any existing conventional single label classifier separately.

Read J. et al. [21, 26] proposed CC (Classifier Chains) that also follows the same mechanism like BR, but in a particular sequence of labels. Both the algorithms, LP [12, 13, 14, 15, 16, 17, 18, 22, 28, 32] and RAkEL [5, 32] consider set of labels at a time.

LP treats each combination of labels appeared in underlying dataset as a separate class and then the classifier follows a multiclass classifier approach. However, its performance degrades due to large number of class labels with their presence in only a few instances. Thus considering only a subset of the labels has been suggested in RAkEL as proposed by Tsoumakas G. et al. [5].

Algorithm adaptation approach involves modification in existing algorithms to tackle with multi-label data directly without any transformation and generally provides better performance.

ML-kNN is an adaptation of conventional kNN to handle multi-label data as proposed by Zhang and Zhou [8]. It predicts labels of unseen instances using Maximum a posteriori and provides better performance when compared with other well-known algorithms. However, the algorithm does not consider relationship between labels.

E. Spyromitros et al. [10] proposed an algorithm BRkNN using the lazy approach. It uses kNN before applying transformation approach. Once neighbors are obtained, then BR classifier uses these neighbors independently for prediction of each label.

BPMLL is a modification of a conventional neural network with feed-forward for dealing with multi-label data where Zhang and Zhou [1] designed a global error function by considering the label correlation.

Jiang et al. [11] computed three terms namely the membership degree of each term in each category, that of each term in each document and that of each document in each category and combined them to get final membership degree. Use of clustering has been suggested to reduce the computational cost of kNN.

Y. Yu et al. [12] proposed two approaches MLRS and MLRS-LC that suggest global and local computation of neighbors respectively.

An approach called, RF-ML, to search for the k neighbors and use of dissimilarity of instances to find the importance of features has been suggested by Newton Spolaˆor et al. [15].

Based on our survey and to the best of our knowledge none of the reported work takes into account label dissimilarity while determining nearest neighbors. We present here a novel algorithm ML-FLD, that determines the nearest neighbors based on feature similarity and differences as well as label dissimilarity.

III. Proposed Algorithm: ML-FLD

Proposed algorithm ML-FLD aims to improve performance of the multi-label classifier through proper selection of neighbors. It uses labels along with the features while searching for the neighbors. This information is utilized for the estimation of the likelihood probabilities of each label. These probabilities along with computed prior probabilities of particular label are further utilized to predict that label for an unseen instance.

Let C be a set of *c* disjoint labels. Let Q be a dataset having q instances $\{X_1 \dots X_q\}$. Let each instance X_j be represented by a pair of vectors, (x_j, y_l), where x_j, ($j = 1, 2 \dots f$) be the set of features and vector y_l ($l = 1, 2 \dots c$) be a set of labels. The aim is to construct a function *g(x)* that maps a given feature vector x_j to a vector y_l.

Figure 1 shows the algorithm for Multi-Label classification using Feature and Label Dissimilarity (ML-FLD) as shown in lines (7) and (24). Algorithm ML-FLD takes four parameters as an input: a training dataset Q with *q*

```
Algorithm ML-FLD
Input:
- A training dataset Q with q instances
- An unseen instance t
- Number of nearest neighbors k
- A user-defined threshold Th
Output: Prediction of labels for unseen instance t

begin
1)  for each label c in each instance q
2)      Compute Prior_c: P (H_c=1) and P (H_c=0) using
        Eq. (1) and Eq. (2) respectively
3)  end for

4)  for each instance X_i (1 ≤ i ≤ q)
5)      N_i = Ø // neighbors of X_i
6)      for each instance X_j (1 ≤ j ≤ q, i ≠ j)
7)          W_j = fs (x_i, x_j) x diff (x_i, x_j) x ld (y_i, y_j)
8)          if |N_i| ≤ k
9)              N_i = N_i U { X_j }
10)         else
11)             find instance X_m in N_i having max weight W_m
12)             if W_m > W_j  // Replace X_m by X_j
13)                 N_i = N_i - { X_m }
14)                 N_i = N_i U { X_j }
15)             end if
16)         end if
17)     end for
18) end for

19) for each label c in j neighbors (0 ≤ j ≤ k)
20)     Estimate Likelihood_c: P (E=j | H_c=1) and
        P (E=j | H_c=0) using Eq. (3) and Eq. (4) respectively
21) end for

22) N_t = Ø // neighbors of instance t
23) for each instance X_i (1 ≤ i ≤ q) and unseen instance t
24)     W_i = fs (x_i, x_t) x diff (x_i, x_t)
25)     if |N_t | ≤ k
26)         N_t = N_t U { X_i }
27)     else
28)         Find instance X_m in N_t with max weight W_m
29)         if W_m > W_i  // Replace X_m by X_i
30)             N_t = N_t - { X_m }
31)             N_t = N_t U { X_i }
32)         end if
33)     end if
34) end for

35) for each label c
36)     Predict t_c for instance t using Prior_c and Likelihood_c
        as given in Eq. (5) and Eq. (6) respectively
37) end for
end
```

Figure 1 Algorithm ML-FLD

instances, an unseen instance t, number of nearest neighbors k and a user-defined threshold Th.

Initially, prior probabilities of each label are computed by counting the number of instances from train set Q that belong to each label c (Lines 1-3). This count $cnt^{(c)}$ along with the smoothing parameter p and size of the training dataset q, are used to compute the prior probabilities of every label in the set C of labels using Eq. (1) and (2) given below. Two kinds of prior probabilities are estimated for each label c: the probability P ($H_c = 1$) of the event that "an instance belongs to a label c" and the probability P ($H_c = 0$) of the event that "an instance does not belong to a label c".

$$P(H_c = 1) = (p + cnt^{(c)}) \;/\; (2p + q) \tag{1}$$

$$P(H_c = 0) = 1 - P(H_c = 1) \tag{2}$$

Once the prior probabilities are obtained, then likelihood probabilities are estimated. It requires the knowledge obtained from k nearest neighbors (kNN). These neighbors are computed for each instance X in set Q (Lines 4-18). If the traditional kNN algorithm is applied, it selects k neighbors with minimum distance and then votes are obtained from the k nearest neighbors to decide the class of particular unseen instance. However, since the instances in case of multi-label data may have one or more class labels, ML-FLD not only considers features but it also takes into account class labels while deciding nearest neighbors. ML-FLD uses Euclidean distance [9] for checking the similarity of features between the instances (function fs(.) in Line 7) and uses Hamming distance [15] to find the label similarity (function ld(.)). Hamming distance between two strings is the number of positions where the characters of two strings are not same. ML-FLD uses a similar method to obtain the statistics from the total number of distinct labels of two instances collectively and the total number of common labels between two instances. The difference between these two values divided by total number of labels is used to update the weight of a neighbor [30, 31]. Along with fs (.) and ld (.), function diff (.) is also used to compute weights of neighbors. The diff (.) function finds the difference between values of corresponding features between the two instances [14, 15]. Summation of absolute values of differences in features is returned by diff (.) function. Thus neighbors are weighed using the information obtained from features as well as labels together (Line 7). Initial k computed weights for instance X_i are directly considered as its k neighbors denoted by set N_i (Lines 8-10). Thereafter, maximum of previously computed k weights in set N_i is searched and it is replaced by the new weight if new weight is smaller (Lines 11-16).

Next, the algorithm decides how many instances have the total number of 0, 1…k neighbors respectively such that each neighbor is associated with the label c. This knowledge is stored in arrays $F_1^{(c)}[0 \ldots k]$ and $F_0^{(c)}[0 \ldots k]$ respectively, depending on whether particular instance whose neighbors are observed, is associated or not associated with the label c. Using this information, the likelihood probabilities of each label are estimated (Lines 19-21). Two kinds of likelihood probabilities are estimated: First the probability that an instance x has j neighbors associated with label c when "an instance x belongs to the label c" as computed in Eq. (3) and second, the probability that an instance x has j neighbors associated with label c when "an instance x does not belong to the label c" as computed in Eq. (4).

$$P(\mathrm{E} = \mathrm{j} | H_c = 1) = \frac{\mathrm{p} + F_1^{(c)}[j]}{p(1+\mathrm{k}) + \sum_{r=0}^{k} F_1^{(c)}[r]}, 0 \leq j \leq k \tag{3}$$

$$P(\mathrm{E} = \mathrm{j} | H_c = 0) = \frac{\mathrm{p} + F_0^{(c)}[j]}{p(1+\mathrm{k}) + \sum_{r=0}^{k} F_0^{(c)}[r]}, 0 \leq j \leq k \tag{4}$$

Next, computation of the Euclidean distance and the difference of features of an unseen instance t with each instance in the dataset Q is done using fs (.) and diff (.) functions respectively (Line 24). It is followed by the selection of k nearest neighbors denoted by set N_t for the unseen instance t based on the minimum weights (Lines 25-33).

Finally prior and likelihood probabilities of each label c, and count of neighbors of the unseen instance t belonging to each label c (Eq. (5)) is used for the prediction of label c as per the Bayes theorem (Lines 35-37) [8]. Eq. (5) measures the number of neighbors of an unseen instance t from the set N_t that are associated with each label c. This count j along with the prior and the likelihood probabilities are used in Eq. (6) to find the ratio to decide whether the unseen instance t is associated with the label c or not. The '*Th*' in Eq. (6) represents threshold and is one of the input to the algorithm. It can be user-defined or calibrated. In our experimentation, *Th* was set to 0.5.

$$\mathrm{j} = \sum_{m=1}^{k} N_m^{(c)} \tag{5}$$

$$t_c = 1 \text{ if} \left(\frac{\mathrm{P}(H_c{=}1) \times \mathrm{P}(\mathrm{E} = \mathrm{j} | H_c = 1)}{\mathrm{P}(H_c{=}1) \times \mathrm{P}(\mathrm{E} = \mathrm{j} | H_c = 1) + \mathrm{P}(H_c{=}0) \times \mathrm{P}(\mathrm{E} = \mathrm{j} | H_c = 0)} \right) \geq Th \quad \ldots (6)$$

Average time required for both the algorithms, ML-kNN proposed by Zhang and Zhou [8] and ML-FLD is comparable. ML-kNN requires $O(q^2.f + c.q.k)$ for computing prior and likelihood probabilities and $O(q.f + c.k)$ for computation related to unseen instances [25]. Whereas the time complexity of ML-FLD is $O(q^2. (f + c) + c.q.k)$ and $O(q.(f + c) + c.k)$ respectively. Here k, f, c, and q represent number of nearest neighbors, features, labels, and instances in dataset Q respectively. Thus average time complexity of ML-FLD is slightly higher than that of ML-kNN. However, ML-FLD provides better performance in terms of various performance parameters as discussed in section IV.

IV. Experimentation and Performance Evaluation

A. *Multi-label Datasets*

Table I shows details of benchmark multi-label datasets used in the experimentation [5, 8, 10, 16, 17, 23, 25, 28, 32]. Datasets are ordered according to number of examples including train and test. Columns in Table I labeled LC (label cardinality) and LD (label density) denote average number of classes per example and the ratio of LC to the number of distinct classes respectively. All the datasets consist of numeric features only.

B. Experimental work and Discussions

Experiments were conducted to compare the performance of algorithm ML-FLD with multi-label and state-of-the-art classifiers that include BR, LP, CC, RAkEL, BRkNN, BPMLL and ML-kNN that are available in Mulan [17]. In case of BR, LP and CC classifiers, J48 decision tree algorithm was considered as base classifier, while LP using J48 was considered as base classifier in case of RAkEL. BPMLL is run with default parameter values in the Mulan.

Parameters such as k and Laplace smoothing factor were set to 10 and 1 respectively as used in most of the reported work [1, 12, 17, 24, 26, 29]. Also threshold value was set to 0.5 as in the reported work.

All experiments were carried on Intel(R) Core(TM) i5-6200U CPU @2.30 GHz with 8GB RAM. Java code libraries available in Mulan, as well as MEKA and WEKA [16, 17, 18, 19, 20] were used.

Table II to Table X show comparative performance of algorithm ML-FLD with that of seven other well-known multi-label classifiers with respect to seven example-based and two label-based performance measures [3, 7, 8, 11, 25]. The symbol (↓) shown in titles of Table II to Table V indicates that smaller value indicates better performance and the symbol (↑)in the title of Table VI to Table X indicates that a larger value for the metric provides better performance.

Discussions:

Experimentation was performed on normalized datasets and values were obtained for seven example-based performance measures, namely hamming loss, ranking loss, one error, coverage, average-precision, accuracy and F-measure and two label-based performance measures, namely macro-averaged and micro-averaged F-measure [3, 7, 8, 11, 25]. Bold values in Table II to Table X represent best values for the particular performance metric in the respective rows.

Following observations are made after experimentation.

1) Hamming loss of ML-FLD was improved than that of ML-kNN in case of Scene and Image datasets shown in Table II. It is comparable in case of the remaining datasets. However, average hamming loss of ML-FLD was lesser by approx. 2%.

2) Ranking loss and one error of ML-FLD were improved for smaller datasets shown in Table III and IV. For larger datasets we could not obtain value for ranking loss. One error was found to be comparable for larger datasets. Average one error was improved by more than 6%.

3) ML-FLD provided better values for coverage for all the datasets except Mediamill where it was slightly higher shown in Table V. Average value of coverage for ML-FLD was the least among comparing algorithms.

4) For smaller datasets, average precision and accuracy of ML-FLD were the best among all the algorithms shown in Table VI and VII. But, we could not get its values for larger datasets. Average values of both metrics were improved by approximately 8% and 18% respectively as compared to other algorithms.

5) For all the datasets ML-FLD showed improvement in the example-based as well as label-based F measure except Mediamill in Table IX. This improvement was by 13% in case of example-based F measure as shown in Table VIII and approx. 6% and 3% in case of macro-averaged and micro-averaged F measure shown in Tables IX and X respectively.

After experimentation, it was observed that, for some test instances, no label was predicted. This definitely affects performance of the classifier. Handling such scenario may improve performance as well as may help to generate values for performance measures where no value was generated specially for larger datasets.

Table I Characteristics of Multi-label Datasets used

Dataset	Domain	#Features	#Labels	Train / Test	#Examples	LD	LC	Type of data
Emotions	Multimedia	72	6	Train	391	0.302	1.813	Nu
				Test	202	0.329	1.975	Nu
Scene	Multimedia	294	6	Train	1211	0.177	1.062	Nu
				Test	1196	0.181	1.086	Nu
Image	Multimedia	294	5	Train	2000	0.247	1.236	Nu
				Test	600	0.228	1.14	Nu
Cbmi09-bow	Multimedia	100	101	Train	22000	0.042	4.291	Nu
				Test	21907	0.044	4.461	Nu
Mediamill	Multimedia	120	101	Train	30993	0.043	4.363	Nu
				Test	12914	0.044	4.406	Nu

Table II Experimental Results for *Hamming Loss* (↓)

Dataset	BR	LP	CC	RAkEL	BRkNN	BPMLL	ML-kNN	ML-FLD
Emotions	0.3144	0.3226	0.3317	0.3053	0.2170	0.2690	**0.2162**	0.2186
Scene	0.1364	0.1469	0.1377	0.1307	0.1080	0.2465	0.0962	**0.0851**
Image	0.1390	0.2323	0.1683	0.1840	0.1153	0.3833	0.1147	**0.1127**
Cbmi09-bow	0.0472	0.0544	0.0467	0.0434	0.0331	0.0599	**0.0331**	0.0341
Mediamill	0.0412	0.0494	0.0424	0.0409	0.0318	0.0645	**0.0316**	0.0317
Average	0.1356	0.1611	0.1454	0.1409	0.1010	0.2046	0.0984	**0.0964**

Table III Experimental Results for *Ranking Loss* (↓)

Dataset	BR	LP	CC	RAkEL	BRkNN	BPMLL	ML-kNN	ML-FLD
Emotions	0.3650	0.4050	0.4086	0.2951	0.1694	0.2064	0.1781	**0.1570**
Scene	0.2315	0.2171	0.2350	0.1591	0.1173	0.1643	0.0930	**0.0830**
Image	0.1382	0.2240	0.1999	0.1769	0.0924	0.1853	0.1154	**0.0888**
Cbmi09-bow	0.2939	0.3865	0.2684	0.3487	0.0967	0.0631	**0.0604**	NaN
Mediamill	0.2142	0.3668	0.2116	0.3465	0.0853	**0.0516**	0.0533	NaN
Average	0.2486	0.3199	0.2647	0.2653	0.1122	0.1341	**0.1000**	0.1096

Table IV Experimental Results for *One Error* (↓)

Dataset	BR	LP	CC	RAkEL	BRkNN	BPMLL	ML-kNN	ML-FLD
Emotions	0.4356	0.5396	0.4901	0.4059	0.3069	0.3812	0.3218	**0.2970**
Scene	0.4189	0.4047	0.3687	0.3470	0.3010	0.5217	0.2425	**0.2191**
Image	0.2417	0.4100	0.3383	0.3350	0.2267	0.2883	0.2517	**0.2183**
Cbmi09-bow	0.5836	0.7697	0.3734	0.4300	**0.2020**	0.2375	0.2043	0.2032
Mediamill	0.5256	0.6951	0.3495	0.4236	0.1810	0.2205	**0.1804**	0.1809
Average	0.4411	0.5638	0.3840	0.3883	0.2435	0.3298	0.2401	**0.2237**

Table V Experimental Results for *Coverage* (↓)

Dataset	BR	LP	CC	RAkEL	BRkNN	BPMLL	ML-kNN	ML-FLD
Emotions	3.0050	3.1634	3.2030	2.6089	1.9158	2.0545	1.9356	**1.8119**
Scene	1.2834	1.1982	1.3035	0.9013	0.6873	0.9239	0.5661	**0.5184**
Image	0.7333	1.0250	0.9550	0.8583	0.5100	0.9217	0.6083	**0.5000**
Cbmi09-bow	69.7428	66.8468	61.7521	67.2632	31.4247	21.1593	20.1887	**20.1426**
Mediamill	58.7190	64.4715	53.4878	67.5235	28.8667	**18.2272**	18.8066	18.8340
Average	26.6967	27.3410	24.1403	27.8310	12.6809	8.6573	8.4211	**8.3614**

Table VI Experimental Results for *Average Precision* (↑)

Dataset	BR	LP	CC	RAkEL	BRkNN	BPMLL	ML-kNN	ML-FLD
Emotions	0.6540	0.6082	0.6270	0.6946	0.7916	0.7484	0.7810	**0.8024**
Scene	0.7143	0.7247	0.7312	0.7751	0.8154	0.6970	0.8511	**0.8653**
Image	0.8377	0.7342	0.7712	0.7862	0.8692	0.8073	0.8456	**0.8718**
Cbmi09-bow	0.4116	0.2169	0.4549	0.2725	0.6547	0.6366	**0.6738**	NaN
Mediamill	0.4880	0.2670	0.5156	0.2833	0.6850	0.6782	**0.7005**	NaN
Average	0.6211	0.5102	0.6200	0.5623	0.7632	0.7135	0.7704	**0.8465**

Table VII Experimental Results for *Accuracy* (↑)

Dataset	**BR**	**LP**	**CC**	**RAkEL**	**BRkNN**	**BPMLL**	**ML-kNN**	**ML-FLD**
Emotions	0.3173	0.3774	0.3859	0.3672	0.4612	0.4905	0.4818	**0.5227**
Scene	0.5173	0.5787	0.5975	0.5596	0.5439	0.3441	0.6597	**0.6958**
Image	0.6308	0.5794	0.6165	0.5444	0.6294	0.1290	0.6492	**0.7008**
Cbmi09-bow	0.2981	0.2690	0.3032	0.1866	0.3899	0.3276	**0.4009**	NaN
Mediamill	0.3477	0.3140	0.3545	0.2078	0.4176	0.3374	**0.4200**	NaN
Average	0.4222	0.4237	0.4515	0.3731	0.4884	0.3257	0.5223	**0.6398**

Table VIII Experimental Results for *F measure* (↑)

Dataset	**BR**	**LP**	**CC**	**RAkEL**	**BRkNN**	**BPMLL**	**ML-kNN**	**ML-FLD**
Emotions	0.3936	0.4589	0.4749	0.4630	0.5416	0.5836	0.5662	**0.6074**
Scene	0.5551	0.5917	0.6177	0.5893	0.5530	0.4586	0.6793	**0.7124**
Image	0.6713	0.6055	0.6378	0.5683	0.6428	0.2004	0.6667	**0.7233**
Cbmi09-bow	0.4193	0.3729	0.4079	0.2857	0.4933	0.4673	**0.5083**	NaN
Mediamill	0.4706	0.4231	0.4648	0.3139	0.5263	0.4766	**0.5317**	NaN
Average	0.5020	0.4904	0.5206	0.4440	0.5514	0.4373	0.5904	**0.6810**

Table IX Experimental Results for *Macro-averaged F measure* (↑)

Dataset	**BR**	**LP**	**CC**	**RAkEL**	**BRkNN**	**BPMLL**	**ML-kNN**	**ML-FLD**
Emotions	0.4294	0.4563	0.4680	0.5063	0.5909	0.6195	0.5880	**0.6339**
Scene	0.6209	0.5938	0.6280	0.6388	0.6285	0.5514	0.7156	**0.7518**
Image	0.4930	0.4078	0.4721	0.4665	0.5666	0.2009	0.5904	**0.6104**
Cbmi09-bow	0.0742	0.0696	0.0705	0.0242	0.0618	0.0678	0.0939	**0.1234**
Mediamill	**0.1349**	0.1102	0.1192	0.0361	0.1056	0.0933	0.1063	0.1128
Average	0.3505	0.3275	0.3516	0.3344	0.3907	0.3066	0.4188	**0.4464**

Table X Experimental Results for *Micro-averaged F measure* (↑)

Dataset	**BR**	**LP**	**CC**	**RAkEL**	**BRkNN**	**BPMLL**	**ML-kNN**	**ML-FLD**
Emotions	0.4356	0.4835	0.4911	0.5119	0.6104	0.6192	0.6278	**0.6545**
Scene	0.6132	0.5870	0.6185	0.6290	0.6380	0.5215	0.7156	**0.7474**
Image	0.6941	0.5459	0.6273	0.5856	0.7048	0.3314	0.7166	**0.7412**
Cbmi09-bow	0.4343	0.3848	0.4247	0.2960	0.5033	0.4836	0.5184	**0.5307**
Mediamill	0.4882	0.4352	0.4824	0.3259	0.5415	0.4935	0.5442	0.5433
Average	0.5331	0.4873	0.5288	0.4697	0.5996	0.4898	0.6245	**0.6434**

*NaN denotes that ML-FLD program has not generated value for that performance metric.

Values of the 'label-wise F measure' in case of smaller datasets, namely, 'Emotions', 'Scene' and 'Image' were also observed for the ML-FLD and the other competing algorithms. The performance is represented graphically in Figure 2. From Figures 2 (a), 2 (b) and 2 (c), it can be observed that 'F measure' values of ML-FLD are either improved or similar to that of other competing algorithms for majority of the labels.

Both the larger datasets, namely, 'Mediamill' and 'Cbmi09-bow' have 101 class labels and therefore graphical or tabular representation of performance of all the algorithms is difficult. Also, as the ML-kNN provided better performance for almost all datasets in Table II to Table X, we made comparison of performance of ML-FLD and ML-kNN only and was in terms of 'label-based F measure' for these datasets. Out of 101 class labels in 'Mediamill' dataset, both algorithms provided value of zero for 41 class labels and for the remaining 60 class labels, ML-FLD provided better performance for 35 class labels and almost similar performance for the remaining 24 class labels. For the dataset

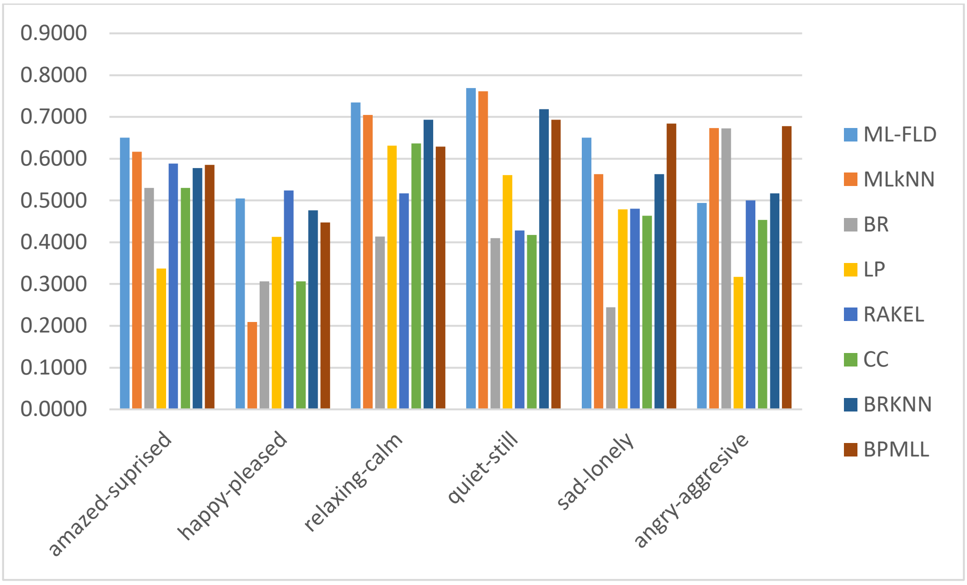

Figure 2 (a) Label-wise F measure for Emotions dataset

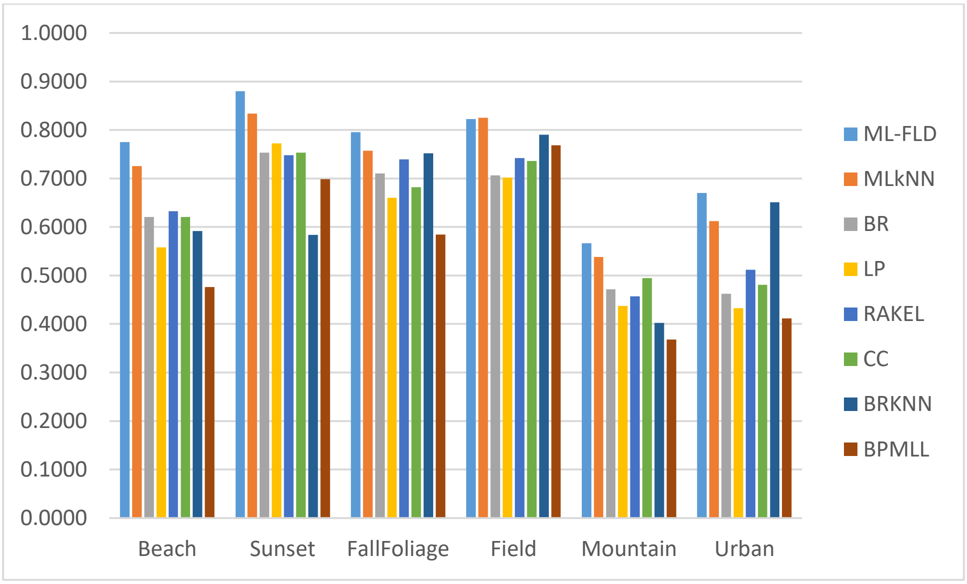

Figure 2 (b) Label-wise F measure for Scene dataset

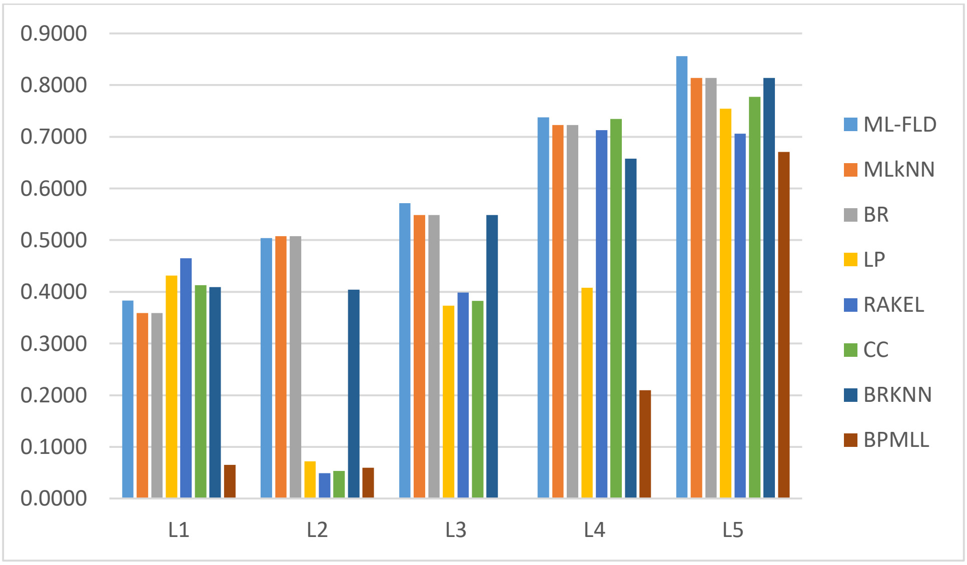

Figure 2 (c) Label-wise F measure for Image dataset

'Cbmi09-bow', both ML-FLD and ML-kNN provided value of zero in case of 39 class labels out of 101. For the remaining 62 class labels, ML-FLD showed improved performance for 52 class labels. This indicates that ML-FLD has better ability to assign correct class labels when compared with all the other competing algorithms.

V. Conclusions

Multi-label learning has received significant attention of the researchers. Multi-label k-nearest neighbors (ML-kNN) is widely adapted by various researchers for classification of multi-label data [8, 10, 11, 12, 13]. Selection of appropriate neighbors for a given instance is necessary to improve accuracy of multi-label classifiers. All existing algorithms determine nearest neighbors using feature similarity only. This paper presents a novel algorithm called ML-FLD that suggests use of label dissimilarity in addition to the feature similarity and difference to determine the neighbors. ML-FLD takes into account the effect of similarity and differences in feature as well as label dissimilarity. It performs better with respect to average values of several example-based as well as label-based measures when compared with the state-of-the-art multi-label classifiers.

Experimentation for the performance evaluation of ML-FLD was carried out using numerical datasets only. In the future, it will be interesting to investigate the performance of ML-FLD for datasets with categorical features. Further investigation is necessary to observe effect of distance metrics on the performance of ML-FLD.

Acknowledgment

Thanks to Prof. Min-Ling Zhang for his valuable guidance and providing resources. Also thanks to Janez Demsar for providing guidance regarding comparison of multiple classifiers.

References

[1] Zhang ML, Zhou ZH (2006) Multi-label neural networks with applications to functional genomics and text categorization, IEEE Transactions on Knowledge and Data Engineering 18(10) 1338-1351

[2] Papanikolaou, Y., Tsoumakas, G., Laliotis, M., Markantonatos, N., and Vlahavas, I. (2017). Large-scale online semantic indexing of biomedical articles via an ensemble of multi-label classification models. *Journal of biomedical semantics*, *8*(1), 43. doi:10.1186/s13326-017-0150-0

[3] Rafal R et al (2005) Multi-label Associative Classification of Medical Documents from MEDLINE, Proceedings of the Fourth International Conference on Machine Learning and Applications (ICMLA'05) 0-7695-2495-8/05

[4] Yu, Qinghua & Wang, Jinjun & Zhang, Shizhou & Gong, Yihong & Zhao, Jizhong. (2016). Combining Local and Global Hypotheses in Deep Neural Network for Multi-label Image Classification. Neurocomputing. 235. 10.1016/j.neucom.2016.12.051

[5] Tsoumakas G., Katakis I. and Vlahavas I. (2011). Random k-Labelsets for Multi-Label Classification. IEEE Transactions on Knowledge Data Engineering, 23. 1079-1089. 10.1109/TKDE.2010.164

[6] S. S. Sane, Prajakta Chaudhari, V. S. Tidake, (2018) An Effective Multilabel classification using Feature Selection, Springer Nature 2018, Intelligent Computing and Info. and Comm., Adv. in Intelligent Systems and Computing 673, pp. 129-142

[7] Zhang, Min-Ling & Li, Yu-Kun & Liu, Xu-Ying. (2015), Towards Class-Imbalance Aware Multi-Label Learning, IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence Proceeding, Pages 4041-4047

[8] M.-L. Zhang and Z.-H. Zhou, (2007) ML-kNN: A lazy learning approach to multi-label learning, Pattern Recognition, vol. 40, no. 7, pp. 2038–2048

[9] J. Han, M. Kamber (2012), Data Mining: Concepts and Techniques, The Morgan Kaufmann Series in Data Management Systems

[10] E. Spyromitros-Xioufis, G. Tsoumakas, and I. Vlahavas (2008), An empirical study of lazy multilabel classification algorithms, in Proc. 5thHellenic Conf. Artificial Intelligence, Syros, Greece, pp. 401–406

[11] Jung-Yi Jiang, Shian-Chi Tsai, Shie-Jue Lee (2012) FSKNN: multi-label text categorization based on fuzzy similarity and k nearest neighbors, Expert Systems Applications, vol.39(3), pp. 2813-2821

[12] Ying Yu, Witold Pedrycz, Duoqian Miao (2014) Multi-label classification by exploiting label correlations, Expert Systems with Applications 41 (2014) 2989–3004

[13] Min-Ling Zhang, Lei Wu 2015. LIFT: Multi-label learning with label specific features. IEEE transactions on pattern analysis and machine intelligence, 37(1), pp.107-120

[14] Kenji Kira and Larry A. Rendell (1992) A practical approach to feature selection, Machine Learning Proceedings, Pages 249–256

[15] Newton Spolaˆor et al (2013) Relief for multi-label feature selection, Brazilian Conference on Intelligent Systems IEEE

[16] Read, Jesse, and Peter Reutemann (2012) MEKA: a multi-label extension to WEKA. URL http://meka.sourceforge.net

[17] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, (2011) MULAN: A Java library for multi-label learning, J. Machine Learning Res., vol. 12, pp. 2411-2414

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, Ian H. Witten (2009) The WEKA data mining software: An update, SIGKDD Explore, vol. 11, no. 1, pp. 10(18)

[19] C.-C. Chang and C.-J. Lin (2011) LIBSVM: A library for support vector machines, ACM Trans. Intelligent

Systems Technol., vol. 2, no. 3, Article 27. [Online]. Available: http://www.csie.ntu.edu.tw/cjlin/libsvm

[20] Tidake, Vaishali S., and Shirish S. Sane (2016) Multi-label Learning with MEKA, CSI Communications

[21] Liu B., Tsoumakas G., Making Classifier Chains Resilient to Class Imbalance, Submitted to ACML 2018 1-15

[22] Álvar Arnaiz-González, José-Francisco Díez-Pastor, JuanJ. Rodríguez, César García-Osorio, Study of data transformation techniques for adapting single-label prototype selection algorithms to multi-label learning, Expert Systems With Applications 109 (2018) 114–130

[23] Laurence A. F. Park, Jesse Read, A Blended Metric for Multi-label Optimisation and Evaluation, ECML PKDD 2018

[24] X. Wu and Z. Zhou. A unified view of multi-label performance measures. In ICML, volume 70, pages 3780-3788. PMLR, 2017

[25] Zhang, Min-Ling & Zhou, Zhi-Hua. (2014). A Review on Multi-Label Learning Algorithms. Knowledge and Data Engineering, IEEE Transactions on. 26. 1819-1837. 10.1109/TKDE.2013.39

[26] Read J, Pfahringer B, Holmes G, Frank E (2009) Classifier chains for multi-label classification. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II. ECML PKDD '09, Berlin, Heidelberg, Springer-Verlag pp. 254-269

[27] Fürnkranz J, Hüllermeier E, Mencía EL, Brinker K (2008) Multilabel classification via calibrated label ranking. Machine learning, 73(2), pp.133-153

[28] Tsoumakas G, Katakis I (2007) Multi-label classification: An overview, International Journal of Data Warehousing and Mining, vol. 3, no. 3, pp. 1-13

[29] Tsoumakas G, Zhang ML, Zhou ZH (2009) Tutorial on learning from multi-label data, in ECML PKDD, Bled, Slovenia [Online]. Available: http://www.ecmlpkdd2009.net/wpcontent/uploads/2009/08/learning-from-multi-label-data.pdf

[30] Carvalho A de, Freitas AA (2009) A tutorial on multi-label classification techniques, in Studies in Computational Intelligence 205, A. Abraham, A. E. Hassanien, and V. Snásel, Eds. Berlin, Germany: Springer, pp. 177–195

[31] Tsoumakas G et al (2010) Mining multilabel data, Data Mining and Knowledge Discovery Handbook, O. Maimon and L. Rokach, Eds. Berlin, Germany: Springer, pp. 667-686

[32] Madjarov G, Kocev D, Gjorgjevikj D, Džeroski S (2012) An extensive experimental comparison of methods for multi-label learning, Pattern Recognit., vol. 45, no. 9, pp. 3084-3104

Author Biographies

Vaishali S. Tidake is a research scholar of MCERC, Nashik. She completed her bachelor's degree in Computer Engineering from KKWCOE, Nashik in 1999 and M.E. in Computer Science and Engineering (IT) from VIT, Pune in 2008. She is working in Department of Computer Engineering at MVPS's KBTCOE, Nashik.

Prof. Shirish S. Sane obtained his bachelor's degree in Computer Engineering from PICT, Pune in 1987 and M. Tech. in Computer Science and Engineering from IIT, Mumbai in 1995. He is the first candidate being awarded the Ph.D. in Computer Engineering from University of Pune. He is working as Vice Principal and Head of Computer Engineering Department at KKWIEER, Nashik. He is the Chairman of Board of studies in Computer Applications and member of Board of studies in Computer Engineering, SPPU (Formerly, University of Pune). He has worked as Regional Vice President for CSI Region VI (Maharashtra and Goa).