# Multi Label Classification of Discrete Data

Bhupesh Akhand
Department of Computer Science and Automation, Indian Institute of Science
Bangalore, India
Email: akhand123@csa.iisc.ernet.in

V.Susheela Devi
Department of Computer Science and Automation, Indian Institute of Science
Bangalore, India
Email: susheela@csa.iisc.ernet.in

***Abstract*—The paper describes an algorithm for multi-label classification. Since a pattern can belong to more than one class, the task of classifying a test pattern is a challenging one. We propose a new algorithm to carry out multi-label classification which works for discrete data. We have implemented the algorithm and presented the results for different multi-label data sets. The results have been compared with the algorithm multi-label KNN or ML-KNN and found to give good results.**

**Keywords: multi-label classification; fuzzy pattern recognition**

## I. Introduction

In Multi-label learning, instances have multiple labels simultaneously, which is used in many applications. Formally, let X = $R^d$ be the d-dimensional instance domain and Y = $\{l_1, l_2, ....., l_q\}$ be the set of labels or classes. Then, the main idea of multi-label learning is to learn a function h : X $\rightarrow$ $2^Y$ which maps each instance x $\in$ X into a set of labels h(x)$\subseteq$ Y. Multi-label classification also introduces information about ranking of labels [multi-label ranking]. In multi-label ranking corresponding to each instance we predict labels with their ranking.

The motivation for multi-label learning is increasing with a number of new applications like annotation of images and video, text classification, functional genomics(gene and protein function) and music categorization into emotions [Mining Multi-label Data].

There are many different approaches for the solving multi-label problem. According to Tsoumakasand Katakis [1] there are two methods for tackling multi-label problem: (a) problem transformation methods and (b) algorithm adaptation methods. Problem transformation methods transform the multi-label problem into a single-label problem e.g.Random k label-sets (RAkEL) [2], pruned sets method and these single-label problems work independently. Output of single-label problems is transformed back to multi-label representation. On the other hand algorithm adaptation methods adapt the algorithms to directly perform multi-label classification e.g. ML-KNN [3], Bayes optimal multi-label classification via probabilistic classifier chains [4], Improved multi-label classification with neural networks [5] etc.

In this study we compared our algorithm with ML-KNN [3]. We have used image and scene data sets with various performance metrics to compare the two methods.

## II. Related work

Several methods have been proposed for multi label classification although most of them are for continuous data. The literature on discrete data is very sparse as not much work has been done in this regard.

ML-KNN [3] is a multi-label lazy learning approach which is derived from the traditional k-nearest neighbour algorithm.In this method prior probability and the posterior probability of a label using k nearest algorithm [6] is calculated. Using these probabilities we find the labels of a given instance.

Label power-set(LP) multi-label learning approach [7], only deals with subset of L label-set. Here we consider a label-set as a single label. If the number of instances and labels will increase then this algorithm will take lot of computational cost and it will be difficult to understand. On the other hand, LP can only predict those label-sets which are available in training data.In order to deal with limitation of LP, new algorithm RAkEL(RAndom k labELsets) [2] is introduced. According to RAkEL label-sets are divided in random small label-sets and LP is applied to train the particular classifier.In the Random walk-KNN method [8], a link graph is built with the k nearest neighbours for each instance. For an unseen instance, a random walk is performed in the link graph. The final probability is computed according to the random walk results.

The Ensembles of Pruned Sets scheme (EPS) [9] first use pruned sets method in which multi label problem is first converted to single label problem. It considers the label set corresponding to each training example as a single class provided that the label set occurs more than a certain number of times called the threshold. If it does not, then it is broken into smaller subsets such that each occurs more than the threshold.

Multi-Label Learning with Label-Specific Features [10] algorithm constructs features specific vectors for each label by conducting clustering analysis on its positive and negative instances, and then performs training and testing by querying the clustering results. Section III describes the methodology of our approach. Section IV gives information about the data sets used by us. Section V defines the performance matrices used by us to evaluate the quality of the classification. The implementation and results are detailed in section VI. Finally, the conclusions are explained in section VII.

## III. Methodology

Our approach works only on discrete or categorical data which has some finite number of values for each attribute.

TABLE I. DATA SETS

| DATA SET | DOMAIN | INSTANCES | ATTRIBUTES | LABELS | CARDINALITY | DENSITY |
|---|---|---|---|---|---|---|
| image | images | 2000 | 294 | 5 | 1.236 | 0.247 |
| scene | images | 2407 | 294 | 6 | 1.074 | 0.179 |

URL 1 http://mulan.sourceforge.net/datasets.html
URL 2 http://cse.seu.edu.cn/people/zhangml/Resources.data

However it can be used for continuous data by first converting it into discrete data. We have first converted the data by using discretization technique and then applied the algorithm on the discrete data.

### A. Discretization of continuous data

An attribute is called continuous if it has very large(infinite) number of possible values while discrete attributes only have small(finite) number of possible values. In discretization we partition attributes of continuous data into intervals for converting it to discrete data. Discretization methods are categorized as unsupervised or supervised. In unsupervised methods no information of target attribute is used while in the supervised methods information of target attribute is used.

For preprocessing step of our algorithm we used a discretization method which is based on k- means algorithm [11]. We choose the initial centers of the clusters for clustering real valued data points. We assume that the number of cluster is already given to us.

The main idea of the algorithm is to choose initial centers of clusters in such a way that they are in increasing order so that recomputed centers will also be in increasing order. Therefore for determining the closest cluster for each valued attribute it does less comparisons. For an attribute, the closest cluster will remain the same or it will be one of two neighbouring clusters. The number of comparisons will be less for reallocation of cluster but in k means algorithm it is k. Also we do not have to order all values like in equal-frequency interval discretization.

The boundary points are defined as minimum and maximum of the attributes and the midpoints of the centers of clusters. In this algorithm we used K=8 for discretization.

### B. Proposed Algorithm

Consider a continuous data set having n attributes $\{a_1, .....a_n\}$. After preprocessing step(discretization) each attribute $a_i$ has p discrete possible values $\{a_{i1}, .....a_{ip}\}$. The dataset has q classes $\{c_1, ....c_q\}$. Now consider a matrix M of size q$\times$np where n is the no. of attributes. Each row of M corresponds to one class label. Each column gives the fuzzy membership value of an attribute value to the various classes. The total number of columns is therefore n$\times$p.

The matrix M is updated for every training data. Once we have gone through the entire training data, the matrix M reflects the membership value of every attribute to the various class labels.

### C. Algorithm for formation of membership matrix M:

Step 1: For every training instance, the class labels to which this instance belongs is noted and all those rows are updated to reflect the instance. In each of these rows, the particular elements which represent the values of the attribute in this instance are incremented by one.

Step 2: Each row is divided by number of instances of the class in the training data. The matrix M now gives the membership value of every feature value to every class. The values in the i$^{th}$ row correspond to the membership values of i$^{th}$ class.

*Consider an example with 3 training data & 3 classes for formation of membership matrix M:*

- We have training set $A = \begin{bmatrix} x & y \\ z & w \\ p & q \end{bmatrix}$ where each row corresponds to one pattern and after discretization it will be converted to
$$A = \begin{bmatrix} x_1 & x_2 & x_3 & y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 & w_1 & w_2 & w_3 \\ p_1 & p_2 & p_3 & q_1 & q_2 & q_3 \end{bmatrix}$$
- Let us suppose after discretization of training matrix we got matrix
$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$ and we have label matrix
$X = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$
where rows shows number of classes and column shows number of instances. For making matrix M we will check label matrix X and matrix A. According to algorithm we will update the matrix M.
- Consider the first training pattern. Since this has class labels 2 & 3, M gets updated as given below
$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$
- After updating M for second pattern we get
$$M = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$
- Updating M for third training pattern
$$M = \begin{bmatrix} 0 & 0 & 2 & 1 & 0 & 1 \\ 0 & 1 & 2 & 2 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

which is the final M after finishing step 1.

- After step 2 Matrix M will be

$$M = \begin{bmatrix} 0 & 0 & 1 & .5 & 0 & .5 \\ 0 & .33 & .66 & .66 & 0 & .33 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

*D. Algorithm for formation of decision vector P:*

Step 1: For every test value do

Step 2: For a vector N of size np×1. According to the values of the attribute in the test instance, update the corresponding entries in N to 1, keeping the other values to zero.

Step 3: Multiply M and N to get P of size q×1.
i.e. M×N = P

Step 4: Let Max be the maximum value in P. Divide every value in P by Max. The $i^{th}$ value in P is the membership value of the test pattern to $class_i$

*E. Selecting the threshold:*

Selecting the threshold is crucial to the working of this algorithm. There are two ways of doing this.

1. For each test pattern put values in P in increasing order and find difference between successive values. Put threshold as mid value between 2 values having maximum difference.

2. To select the threshold t, a validation data set is needed. After forming the matrix M, calculate the error on the validation set for different values of the threshold and choose that value of the threshold for which the error is minimum.

**Algorithm 1** Algorithm for forming M

Input: Training data with c patterns & d features and each feature takes p values
begin

for i:=1 to c
for j:=1 to d×p
{ $M_{ij}$=0; }
for i:=1 to c do
{
For each training data corresponding every class label update $i^{th}$ row of M. Increment the appropriate columns by 1.
}
for i:=1 to c do
for j:=1 to d×p do
{
count = Number of patterns having the class label i
$M_{ij} = \frac{M_{ij}}{count}$
}
end

## IV. Data sets

We experiment with two datasets (semantic image and scene analysis), in which a picture can be categorized into one or more classes. In the scene data, for example, pictures can have the following classes: sunset, beach, foliage, mountain, field and urban.Each image is represented by a feature vector using the same method as in (Boutell etal. 2004), so that image is transformed into a 294-dimensional feature vector. Table 1 shows certain standard characteristics of these datasets, such as the number of numeric, the number of examples and the number of labels, along with multi-label data statistics, such as the label density [1] and the label cardinality. Label cardinality is the average number of labels per instance, while label density is label cardinality divided by the the number of labels.

**Algorithm 2** Algorithm for classifying test patterns

Input: matrix M of size c×dp and m test patterns
begin

for i:=1 to m do
{
1. Form a matrix N of size dp×1 and initialize to 0.

2. Depending on values of test patterns i make appropriate entries in N as 1.

3. Multiply M and N to get P.

4. Find maximum value in P. Divide every value in P by it.

Max = max(P);

5. Classify test instance

for j:=1 to c do

$p_j = \frac{P_j}{Max}$

for k:=1 to c do
{
if $P_k >$ threshold
then pattern k $\in$ to $class_k$
}
}
end

## V. Evaluation metrics

In Multi-label learning an instance is associated with multiple labels simultaneously. In this case finding accuracy is more difficult than single-label learning. The evaluation matrices used for single label classification can not be used for multi-label classification since we need to consider the multiple labels.

In this paper we use five criteria often used for multi-label learning [12]. In a given multi-label data set S = { $(X_1,Y_1)$, $(X_2,Y_2)$,.....,$(X_p,Y_p)$ }, these criteria are defined as below. Here, $h(X_i)$ returns a set of proper labels of $X_i$ ; $h(X_i,y)$ returns a real-value indicating the confidence for y to be a proper label of $X_i$ ; rank $h(X_i,y)$ returns the rank of y derived from $h(X_i,y)$.

Hamming loss:

$$\frac{1}{p}\sum_{i=1}^{p}\frac{1}{|y|}|h(X_i)\triangle Y_i| \tag{1}$$

Here $\triangle$ denotes the symmetric difference between two sets. The hamming loss evaluates how many times an example-label pair is misclassified,i.e., a proper label is missed or a wrong label is predicted.

One error:

$$\frac{1}{p}\sum_{i=1}^{p}\|[argmax_{y\in Y}h(X_i,y)\notin Y_i]\| \tag{2}$$

The one-error evaluates how many times the top-ranked label is not a proper label of the object.

Coverage:

$$\frac{1}{p}\sum_{i=1}^{p}max_{y\in Y_i}rank^h(X_i,y)-1 \tag{3}$$

The coverage evaluates how far it is needed, on the average, to go down the list of labels in order to cover all the proper labels of the object.

Ranking Loss:

$$\frac{1}{p}\sum_{n=1}^{p}\frac{1}{|Y_i||\bar{Y_i}|}\{(y_1,y_2)|h(X_i,y_1)\leq h(X_i,y_2),(y_1,y_2)\in Y_i\times Y_i\}| \tag{4}$$

where $\bar{Y_i}$ denotes the complementary set of $Y_i$ in Y. The ranking loss evaluates the average fraction of label pairs that are misordered for the object.

Average precision:

$$\frac{1}{p}\sum_{n=1}^{p}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{|\{y'|rank^h(X_i,y')\leq rank^h(X_i,y),y'\in Y_i\}|}{rank^h(X_i,y)} \tag{5}$$

The average precision evaluates the average fraction of proper labels ranked above a particular label $y\in Y_i$.

The performance of algorithm will be perfect when hloss$_S$(h)=0 and the smaller the value of hloss$_S$(h) means the better the performance of h. Like Hamming loss performance is perfect when one-error$_S$(h)=0 and the smaller the value of one-error$_S$(h) means the better the performance of h. For Coverage, the smaller the value of coverage$_S$(h), the better the performance of h and performance is perfect when coverage$_S$(h)=0 and rloss$_S$(h)=0; the smaller the value of rloss$_S$(h), the better the performance of h. If avgprec$_S$(h)=1 performance will be perfect and the larger the value of avgprec$_S$(h), the better the performance of h.

## VI. Implementation and Results

We implemented our algorithm and fixed the threshold using validation set. We used multi-label data sets image and scene. These data sets have continuous attributes. These data sets are discretized by us before applying the algorithm.

TABLE II. Experimental results of the compared algorithms on image data set. For each evaluation criterion, ↓ indicates the smaller the better while ↑ indicates the bigger the better.

| Evaluation criteria | Algorithm | Results |
|---|---|---|
| Hamming Loss ↓ | ML-KNN | 0.2960 |
| | Proposed Algo | 0.2820 |
| One-error ↓ | ML-KNN | 0.7700 |
| | Proposed Algo | 0.1317 |
| Coverage ↓ | ML-KNN | 2.4417 |
| | Proposed Algo | 0.9850 |
| Ranking Loss ↓ | ML-KNN | 0.5717 |
| | Proposed Algo | 0.9565 |
| Average Precision ↑ | ML-KNN | 0.4401 |
| | Proposed Algo | 0.7525 |

TABLE III. Experimental results of the compared algorithms on scene data set. For each evaluation criterion, ↓ indicates the smaller the better while ↑ indicates the bigger the better.

| Evaluation criteria | Algorithm | Results |
|---|---|---|
| Hamming Loss ↓ | ML-KNN | 0.9331 |
| | Proposed Algo | 0.2094 |
| One-error ↓ | ML-KNN | 0.4391 |
| | Proposed Algo | 0.1302 |
| Coverage ↓ | ML-KNN | 0.9834 |
| | Proposed Algo | 1.3033 |
| Ranking Loss ↓ | ML-KNN | 0.1772 |
| | Proposed Algo | 0.7304 |
| Average Precision ↑ | ML-KNN | 0.7262 |
| | Proposed Algo | 0.6063 |

We used eight discrete values for every continuous value to cover entire range. The K- means algo has been used for discretization which is a supervised method of discretization. We compared our algorithm with multi-label KNN or ML-KNN with discrete image and scene data sets. The best threshold value for image data set is 0.92 and for scene data set is 0.85. Value of K is 10 for ML-KNN.

Tables 2 and 3 report the detailed experimental results on 1 & 2 data sets. It can be seen that for both the data sets, hamming loss, one-error, coverage and average precision is better using our algorithm as compared to ML-KNN.

## VII. Conclusion

According to the literature, MLKNN can be considered as the best multi label classification algorithm. In this paper, we have given an alternative approach for multi-label classification, which is giving good results and easy to implement. The only drawback is that it can be used only for discrete data. This paper also shows that how we can solve multi-label problem by using discretization. It is to be noted that if we use more number of discrete values we can get a better result. Fig. 1 and Fig. 2 shows that comparative evaluation matrices for our proposed method and the ML-KNN. They show how in most cases the proposed algorithm is far better then ML-KNN.

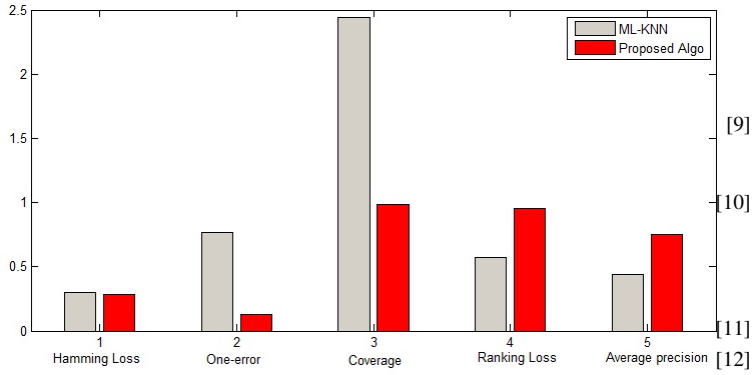


Fig. 1. Experimental results of the compared algorithms on image data set

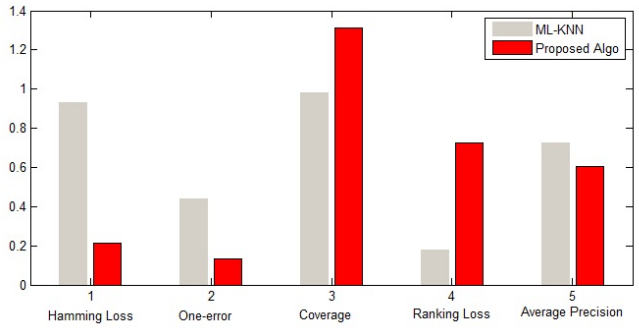


Fig. 2. Experimental results of the compared algorithms on scene data set

In future, we need to compare our method with data sets which are inherently discrete. We also plan to use a variety of different data sets to validate our method.

## REFERENCES


[1] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Int J Data Warehousing and Mining*, vol. 2007, pp. 1–13, 2007.

[2] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multilabel classification," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 7, pp. 1079 –1089, july 2011.

[3] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038 – 2048, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320307000027

[4] K. Dembczynski, W. Cheng, and E. Hüllermeier, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 279–286.

[5] R. Grodzicki, J. Mańdziuk, and L. Wang, "Improved multilabel classification with neural networks," *Parallel Problem Solving from Nature–PPSN X*, pp. 409–416, 2008.

[6] M.-L. Zhang and Z.-H. Zhou, "A k-nearest neighbor based algorithm for multi-label classification," in *Granular Computing, 2005 IEEE International Conference on*, vol. 2, july 2005, pp. 718 – 721 Vol. 2.

[7] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2004.03.009

[8] X. Xia, X. Yang, S. Li, C. Wu, and L. Zhou, "Rw.knn: a proposed random walk knn algorithm for multi-label classification," in *Proceedings of the 4th workshop on Workshop for Ph.D. students in information & knowledge management*, ser. PIKM '11. New York, NY, USA: ACM, 2011, pp. 87–90. [Online]. Available: http://doi.acm.org/10.1145/2065003.2065022

[9] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, dec. 2008, pp. 995 –1000.

[10] M.-L. Zhang, "Lift: multi-label learning with label-specific features," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two*, ser. IJCAI'11. AAAI Press, 2011, pp. 1609–1614. [Online]. Available: http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-270

[11] D. Joiţa, "Unsupervised static discretization methods in data mining."

[12] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li, "Multi-instance multi-label learning," *Artificial Intelligence*, vol. 176, no. 1, pp. 2291 – 2320, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370211001123