IE 506: Machine Learning - Principles and Techniques Jan-May 2023 End-term Project Report

: Multi-Instance Multi-Label Learning

Team Name: Data Poltergeist *Team Members: 22m0423*

Abstract

This project report contains details of my project which deals with the Multi-Instance Multi-Label datasets (MIML) which are generally tackled in many real-world tasks where one object contains multiple instances and multiple labels associated with it with complicated semantics, such as Image or text. For handling such datasets, many techniques have been studied and implemented but with some limitations associated with them. This eventually creates a barrier to what extent we could handle such datasets. This project comes up with a new approach MIMLFast which works efficiently on such datasets without any kind of limitations associated with it. MIMLFast efficiently handles large datasets and achieves excellent performance and also discovering sub-concepts for complicated labels. The experimental results show that MIMLFast is more effective than previous techniques and has the potential to enable more accurate and efficient machine learning models for MIML challenges.

1 Introduction

In general, four distinct types of learning frameworks have been identified. Working with a single instance associated with a single label is typical of traditional supervised learning. A multi instance learning involves the association of numerous objects with a single label, where each item has several instances. MIML learning is also known as multiple labels for multiple instances. In image classification tasks, for example, a picture may be associated with many semantic labels and separated into segments, each of which is represented by an instance.[1]

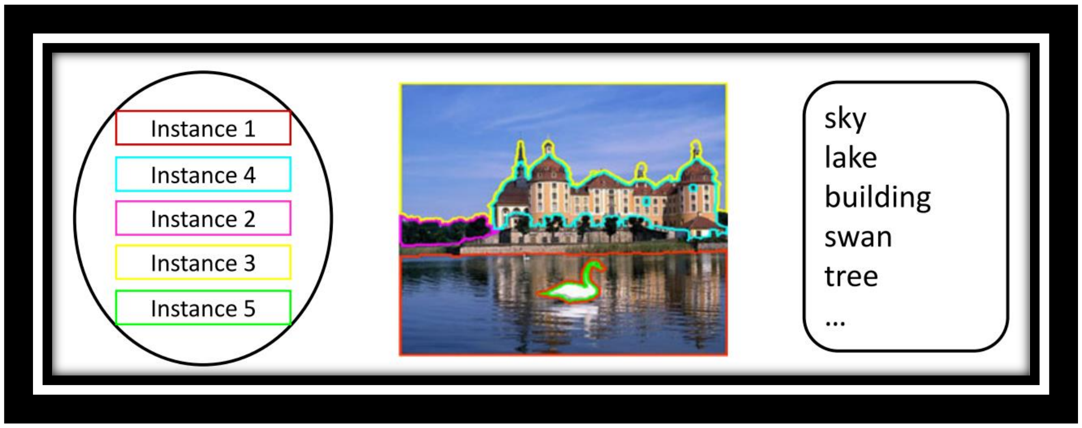


Figure 1: A MIML Example [1]

Many MIML algorithms have been proposed throughout the years. Some of them work in a degenerative manner. MIMLBoost [11] , for example, degenerates the MIML problem into a set of multi-instance single-label learning tasks, one for each label; while MIMLSVM [11] degenerates the MIML work into multi-label learning tasks by transforming a bag into a single instance. In contrast, the DMIMLSVM technique directly optimises the loss function for the original MIML problem. These approaches produced acceptable results and validated the superiority of the MIML framework in a variety of applications. However, as expressive power increases, MIML's hypothesis space expands considerably, resulting in the high complexity and low efficiency of existing techniques. These approaches are typically time-consuming and unable of handling massive amounts of data, severely limiting the application of multi-instance multi-label learning.

To address this issue, a novel approach called MIMLfast [1] is proposed in this research to learn on multi-instance multilabel data quickly. MIMLfast gives an effective approximation of the original MIML problem despite the use of simple linear models for efficiency.

To make use of the relationships between several labels, we first learn a shared space from the original features, and then train label-specific linear models from the shared space. To determine the key instance to represent a bag for a certain label, we train the classification model on an instance-by-instance basis and then choose the instance with the highest prediction as the key instance. We use stochastic gradient descent (SGD) to optimise an approximated ranking loss to make learning more efficient. MIMLfast randomly picks a triplet consisting of a bag, a relevant label of the bag, and an irrelevant label at each step of SGD, then optimises the model to rank the relevant label before the irrelevant one if this order is violated. In comparison to state-of-the-art MIML approaches, the suggested MIMLfast approach delivers highly competitive performance and improves efficiency by more than 100 times on big data sets. Based on the instance-level predictions, this technique may naturally pick the most representative case for each label. Furthermore, for sophisticated labels, this strategy attempts to utilise sub-concepts. The suggested method can recognise many sub-concepts encoded in a difficult label by adaptively learning numerous models for each label.

The rest of the paper is organised as follows. Section 2 discusses previous work, Section 3 explains the MIMLfast technique, and Section 4 discusses the type of dataset used and its details. Section 5 contains details on the dataset's experiments, while Section 6 has the outcomes. Section 7 discusses future work, followed by Section 8's conclusion.

2 Literature Survey

First, consider two learning frameworks that are linked to MIML: multi-label learning and multi-instance learning.

Each object in multi-label learning is represented by a single instance while being associated with many labels. Multilabel learning has been extensively researched over the last several decades, and numerous methods have been presented. Existing approaches can be divided into two categories: problem transformation methods and algorithm adaptation methods [3]. The simplest strategy in the first group is to divide a multilabel job into a sequence of binary classification problems [4], one for each label, and then learn each label independently [5]. Other methods attempt to convert the multi-label task into multi-class problems, with each class representing a possible label subset [6]. There are also several methods that attempt to convert the multi-label job into label ranking issues, with the goal of ranking the labels, with the goal of ranking relevant labels ahead of irrelevant ones in each instance [7].

Methods of algorithm adaptation attempt to adapt popular learning techniques to a multi-label situation. Boosting style method AdaBoost [8].MH, slow learning algorithm ML-kNN [9], decision tree based algorithm ML-DT [10], and neural networks based algorithm are examples of representative approaches.

Furthermore, either active querying or incomplete annotations have been used to study weakly supervised multi-label learning. Exploiting the link between multiple labels has recently gained popularity. Efforts in this direction include using prior knowledge of label structures and automatically exploiting relationships between labels.

There are also several algorithms that attempt to convert the multi-label job into label ranking issues, with the goal of ranking relevant labels ahead of irrelevant ones in each instance. Learning the connection between many labels is another essential feature of multilabel learning that can assist improve the model's performance.

MIML techniques with excellent efficiency are required under this framework to manage large size data.

3 The MIMLFAST Approach

A MIML dataset consists of n examples, where each bag X_i has Z_i instances.

$\{ (X_1, Y_1) , (X_2, Y_2) , (X_3, Y_3) , \ldots\ldots (X_n, Y_n) \}$

$X_i = \{ x_1 , x_2 , x_3 , \ldots ., x_{i,z_i} \}$

Y_i contains the labels associated with each X_i

$Y_i = \{ y_1 , y_2 , y_3 \ \ldots . y_L \}$ set of all possible labels associated to X_i

In the next section, I will first discuss the suggested model for MIML prediction, followed by the goal of the proposed learning task. Following that, a quick algorithm for optimising the objective function will be proposed. Finally, I provide an improved version of the algorithm that makes better use of instance label information.

3.1 Work done before mid-term project review

3.1.1 The Prediction Model

Let's start by talking about how to develop a classification model at the instance level, and then try to derive bag labels using instance predictions. The simplest way to solve an issue with multiple labels is to degenerate it into a series of single label problems by training one model for each label independently. However, because it handles labels independently and overlooks their relationships, such a degenerating strategy may lose information.

In this approach, the model is first developed as a mixture of two components. The first component learns a linear mapping from the original feature space to a low-dimensional space that all labels share. The second component then learns label-specific models from the shared area. The two components are optimised interactively to fit training examples from all labels.

For a given instance x, we define classification model on label l as follows:

$$F_l(x) \ = \ W_l^T . W_o \ . x$$

W_o is a mxd matrix which maps the original feature vectors to the shared space. W_l is a m dimensional weight vector for label l. d and m are the dimensionalities of the feature space and the shared space, respectively.

A model like this has two big advantages. First, because we usually have m<<d, the number of variables to memorise is lowered from d x L to (d+L) x m. This will result in significant savings in both memory and computational costs. Second, the relationship between labels can be used. Examples from each label will help to optimise the shared area, and labels with strong relationships are expected to assist each other.

The two-level model is depicted in the figure below.

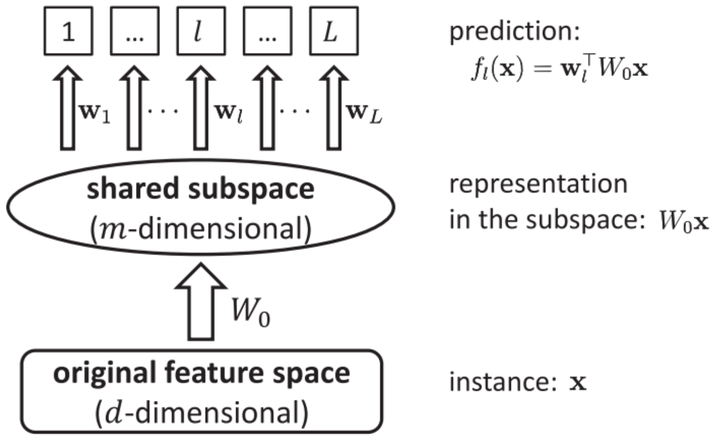


Figure 2 :The two-level model: Wo maps the original feature vectors to a shared subspace, and Wl is the weight vector for label l [1].

The model is then improved to handle complex labels with sub-concepts. Objects in multi-instance multi label learning tasks typically have sophisticated semantics; hence, samples with various contents may be assigned the same label. It is challenging to train a single model to categorise photos/images with such disparate contents. Instead, it is proposed to train multiple models for a label, one for each subconcept, and then automatically determine which subconcept each case belongs to. Each sub-concept's model is more simpler, and thus more easily trained to suit the data. It is assumed that each label has K sub-concepts. The sub-concept to which a particular example with label l belongs is automatically determined by first assessing the prediction values of the K models and then selecting the sub-concept with the highest prediction value.

Therefore, the prediction of instance x on label l is redefined as:

$$F_l(x) \;=\; max\;(\,W_{l,k}^T\,.\,W_o\,.\,x\,) \qquad \text{k=1,2,3}\ldots\text{K}$$

where $W_{l,k}^T$ represents the kth sub-concept of label l. Although it is assumed that each label has K sub-concepts, empty sub-concepts are permitted, i.e., examples of a simple label may be dispersed in only a few or even one sub-concept.

Then we'll look at how to get bag predictions from instance level models. A bag is assumed to be positive if and only if it includes at least one positive instance. Under this premise, the prediction of a bag X on label l can be defined as the sum of all predictions from this bag:

$$F_l(X) \;=\; max\;F_l(\,x\,) \qquad x \in X$$

Whichever instance has maximum prediction for that label, we call that instance as 'Key instance' of X for that label.

3.1.2 The Objective

The MIML classification problem is converted into a label ranking problem. And the goal is to prioritise useful labels over irrelevant labels for all samples.

For any bag 'X' with one of its relevant label 'l', we define R(X,l) as:

$$\text{R(X , l)} = \sum_{j\epsilon\bar{Y}} I\;[\,F_j(X)\;>\;F_l(X)\,]$$

$\bar{Y}$ denotes the set of irrelevant labels of X and I[ ] is the indicator function which returns 1 if the argument is 'true' and 0 otherwise. Essentially, R(X , l) counts how many irrelevant labels ae ranked before label l on the bag X.

Based on the R(X , l), we further define 'ranking eror' with respect to bag X on label l.

$$\boldsymbol{\varepsilon}\text{(X , l)} = \sum_{i=1}^{R(X\,,l)} \frac{1}{i}$$

It is obvious that ranking error would be larger for lower l being ranked. Finally we have the ranking error on the while dataset as :

$$\text{Rank Error} = \sum_{j=1}^{n} \sum_{l\epsilon\,\text{Yj}} \varepsilon\text{(Xj , l)}$$

It is rather difficult to optimize the above equation so we instead explore the following loss,

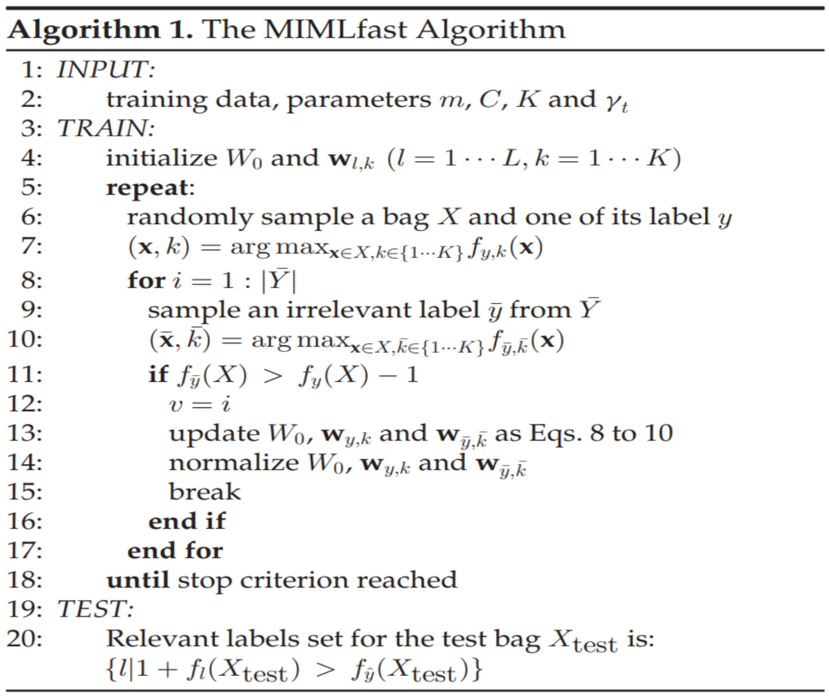
$$\Psi(X,l) = \sum_{j \in \bar{Y}} \epsilon(X,l) \frac{|1 + f_j(X) - f_l(X)|_+}{R(X,l)}$$

Therefore, to rank the relevant labels before the irrelevant labels, we only need to minimize this rank loss on training data.

$$\min \sum_{i=1}^{n} \sum_{l \in Y_i} \Psi(X_i, l).$$

3.2 Work done after mid-term project review

The pseudo code of MIMLfast is presented in Algorithm 1. First, each column of W_o and W_y^k for all labels y and all sub-concepts k are initialized at random with mean 0 and standard deviation 1=sqrt(d). Then at each iteration of SGD, a triplet (X,y,$\bar{y}$) is randomly sampled, and their corresponding key instance and sub-concepts are identified. After that, gradient descent is performed to update the three parameters: W_o, $W_y^k$and $W_{\bar{y}}^{\bar{k}}$. At last, the updated parameters are normalized such that their norms will be upper bounded by C. This procedure is repeated until some stop criteria reached.



Algorithm 1. The MIMLfast Algorithm

```
1:  INPUT:
2:      training data, parameters $m$, $C$, $K$ and $\gamma_t$
3:  TRAIN:
4:      initialize $W_0$ and $\mathbf{w}_{l,k}$ $(l = 1 \cdots L, k = 1 \cdots K)$
5:      repeat:
6:        randomly sample a bag $X$ and one of its label $y$
7:        $(\mathbf{x}, k) = \arg\max_{\mathbf{x} \in X, k \in \{1 \cdots K\}} f_{y,k}(\mathbf{x})$
8:        for $i = 1 : |\bar{Y}|$
9:          sample an irrelevant label $\bar{y}$ from $\bar{Y}$
10:         $(\bar{\mathbf{x}}, \bar{k}) = \arg\max_{\mathbf{x} \in X, \bar{k} \in \{1 \cdots K\}} f_{\bar{y},\bar{k}}(\mathbf{x})$
11:         if $f_{\bar{y}}(X) > f_y(X) - 1$
12:           $v = i$
13:           update $W_0$, $\mathbf{w}_{y,k}$ and $\mathbf{w}_{\bar{y},\bar{k}}$ as Eqs. 8 to 10
14:           normalize $W_0$, $\mathbf{w}_{y,k}$ and $\mathbf{w}_{\bar{y},\bar{k}}$
15:           break
16:         end if
17:       end for
18:     until stop criterion reached
19: TEST:
20:     Relevant labels set for the test bag $X_{\text{test}}$ is:
        $\{l | 1 + f_l(X_{\text{test}}) > f_{\hat{y}}(X_{\text{test}})\}$
```

Figure 3 : MIMLFast Algorithm1 [1]

4 Data set Details

For the training and testing of this MIMLFast algorithm, two datasets were used. Both datasets were multi-instance multi-label in nature. In both of these datasets, 80 percent of the data was chosen for training and 20% for testing purposes. Dataset 1 is named "example_data" and consists of numerous bags (in total 1000), each of which has 243 examples with a varied number of features, and each bag is associated with seven labels out of which some are relevant and some are not.

Dataset 2 is titled "kaggle_dataset" and was obtained from the "kaggle" website [12]. The picture data set contains 2,000 natural scene photographs. The picture data collection comprises of 2,000 natural scene photographs, each with a unique set of labels attached to it. Desert, mountains, sea, sunset, and trees are all acceptable class labels. The number of photos belonging to more than one class (for example, sea+sunset) accounts for more than 22% of the data set; however, many combined classifications (for example, mountains+sunset +trees) are exceedingly rare. Each image is assigned 1.24 class labels on average.

5 Experiments

First of all, I created 3 functions which were basically MIMLFast, MIML_train and MIML_test. MIMLFast is the function which basically executes the algorithm for training and test and hence predicting the test labels values. MIML_train basically is the function which first convert the original space to a shared subspace and hence then trains different classification models each for a label in that subspace. MIML_test is the function which basically takes the weight outputs given by MIML_train function after training and hence based on these outputs the MIML_test gives the predictions on the test data where the labels which accuracy scores are higher than that of dummy label are considered to be relevent ones otherwise irrelevant.

The two previously described datasets were taken and separated into training and test datasets, with 80 percent of the data used for training. After splitting, the train data, train labels, and test data were passed to the MIMLFast algorithm.

5.1 MIMLFast Function

This function begins by declaring the arguments and assigning values to them. The parameters consisted of D, which is the dimension of the shared subspace, which was set to 100. Then the upper bound for the norm of each vector was set to 10, the maximum number of iterations was set to 10 (maxiter), the initial step size for stochastic gradient descent was set to 0.005, the number of subconcepts was set to 5, and finally a dictionary 'opts' was created that contained average_begin and average size keys with their values.

After defining the parameters, the following step was to build a dummy label and add it to the label set's array. Following that, the Initialization stage was carried out, in which the W and V matrices were generated at random with a mean of 0 and a standard deviation of 1/sqrt(m), where m is the number of instances contained in each bag. Following that, W and V were normalised. After the initialization stage is complete, we assign all of these parameters to the MIMLtrain and MIMLtest functions.

5.2 MIML_train Function

First, a vector named 'tmpnums' was generated, which contained the total number of labels relevant to each bag. Then training pairs were formed, with each pair having one bag and one label. For example, if bag 1 is associated with two labels, the train pairs are as follows: bag 1 with label 1 as train pair 1 and bag 1 with label 2 as train pair 2. Once the train pairs for all of the bags and labels have been formed, we build a random number array in the range of train pair lengths and store it in 'random_idx'.

Then, depending on these randomly generated random indexes, we call each bag randomly and store its instances in 'xbag', as well as labels associated with that bag in 'idx_class'. Similarly, we store irrelevant labels in 'idx_irr' so that if 'idx_class' returns a dummy label class, we store just irrelevant labels in 'idx_irr' but if 'idx_class' doesn't returns dummy label class, we store both irrelevant labels and dummy labels in 'idx_irr'.

The weights from W are then denoted as Wy and Wyn for each label in idx_class and idx_irr. As a result, we must first convert our original space into a shared subspace using V x xbag. Then, in that shared space, we train label specific models as Wy x V x xbag and save the score in 'fs'. Then we discover the 'fy', which is the maximum prediction made by that particular subspace, and hence the key instance for that particular label. Same thing we do for irrelevant labels as well.

Now, once this is done then we check if score of irrelevant label is greater than relevant ones. If 'true' then we go for updating step using 'gradient descent'. The parameter Wy, Wyn and V are then updated using the following equations:

$$W_0^{t+1} = W_0^t - \gamma_t S_{\bar{Y},v}(\mathbf{w}_{\bar{y},\bar{k}}^t \bar{\mathbf{x}}^\top - \mathbf{w}_{y,k}^t \mathbf{x}^\top) \qquad \mathbf{w}_{y,k}^{t+1} = \mathbf{w}_{y,k}^t + \gamma_t S_{\bar{Y},v} W_0^t \mathbf{x} \qquad \mathbf{w}_{\bar{y},\bar{k}}^{t+1} = \mathbf{w}_{\bar{y},\bar{k}}^t - \gamma_t S_{\bar{Y},v} W_0^t \bar{\mathbf{x}}$$

Here first equation updates the V matrix, the second equation updates the Wy matrix and the third equation updates the Wyn matrix.

5.3 MIML_test Function:

To begin, an empty array named 'pres' is formed to contain the scores of test dataset bags. Then, for each bag, identical procedures are taken, such as first utilising updated V to transform our xbag from its original space to a shared subspace, and then generating unique classification models for each label. As a result, we extract the scores for each bag and identify the subconcept with the highest score as well as the key instance.

Now we store the scores in the 'pres' array, sort it in descending order, and store it in the 'ord' array, and selecting the first five indices from the ord array. Then we make another array called 'labels' with every element set to -1.

Now, for a test bag, relevant labels are labels that have greater predictions than dummy labels, and the first five indices from the ord array are also relevant labels, so we assign the indices of such labels to one in the 'labels' array. This is ultimately what our model predicts.

6 Results

To facilitate a comparative analysis with other algorithms, the evaluation metric of Hamming loss was used as the baseline. Hamming loss is a commonly used metric for measuring the accuracy of multi-label classification models. This metric quantifies the proportion of incorrectly predicted labels in comparison to the total number of labels.

DATASET	NO OF BAGS	HAMMING LOSS	TIME COMPLEXITY
1	1000	0.27	5 seconds
2	2000	0.2	7 seconds

Table 1: Result Obtained for both datasets

The experimental results indicate that for Dataset 1, comprising of approximately 1000 bags, the Hamming loss was approximately 0.27. Conversely, for Dataset 2, consisting of approximately 2000 bags, the Hamming loss was around 0.2.

Furthermore, the computational efficiency of MIMLFast was evaluated in terms of time complexity. The results demonstrated that MIMLFast is a speedy algorithm, with a training and evaluation time of approximately 5 to 7 seconds for both the datasets.

Similarly when looking at the comparisons done by author for more larger datasets, similar kind of results were obtained as can be seen in this figure below.

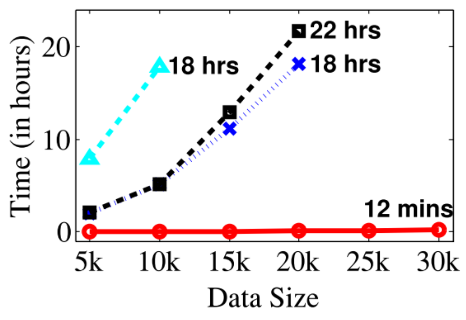


Figure 4: Time complexity Comparison for different models with MIMLFast[1]

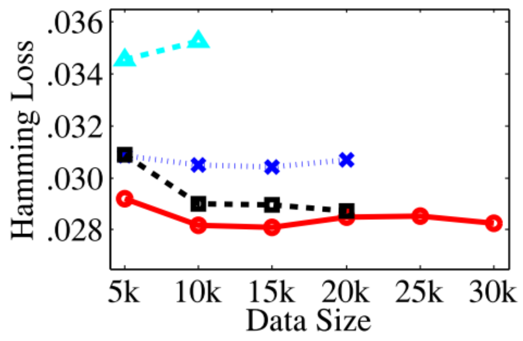
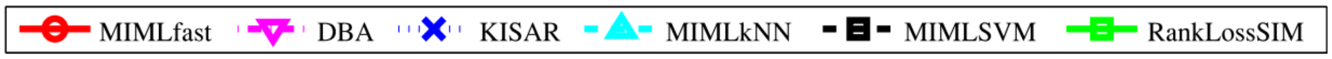


Figure 5: Hamming loss Comparison for different models with MIMLFast.[1]



7 Future Work

The further work could be done to improve the performance of implemented algorithm by exploring various techniques such as feature engineering, hyperparameter tuning, and ensemble methods. To experiment with different loss functions and regularization techniques to further reduce the Hamming loss more effectively.

8 Conclusion

The MIML framework was created to aid in the learning of complex objects and has proven to be a valuable tool in a variety of applications. Nonetheless, when dealing with large-scale problems, current MIML approaches have proven to be time-consuming.

To address this issue, a new technique called MIMLfast was presented as an efficient way to learn with MIML examples implemented in the Python programming language. My primary goal in developing this new model was to improve both efficiency and efficacy when compared to existing established methods. The suggested method achieves efficiency by optimising the approximated ranking loss using SGD using a two-level linear model. Furthermore, we increase effectiveness by utilising label relations in a shared space and discovering sub-concepts for complex labels.

Furthermore, MIMLfast can detect the key instances for each label automatically, providing an opportunity to investigate the relationship between input patterns and output label semantics.

References

1. Fast Multi-Instance Multi-Label Learning, Sheng-Jun Huang , Wei Gao, and Zhi-Hua Zhou, VOL. 41, NO. 11, NOVEMBER 2019.
2. Multi Instance Multi Label Learning by Zhi-Hua Zhou , Min-Ling Zhang, Sheng-Jun Huang, Yu-Feng Li, National Key Laboratory for Novel Software Technology.
3. M.-L. Zhang and Z.-H. Zhou, “A review on multi-label learning algo rithms,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 8, pp. 1819–1837, Aug. 2014.
4. M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng, “Binary relevance for multi-label learning: An overview,” Frontiers Comput. Sci., vol. 12, no. 2, pp. 191–202, 2018.
5. M. Boutell, J. Luo, X. Shen, and C. Brown, “Learning multi-label scene classification,” Pattern Recognit., vol. 37, no. 9, pp. 1757–1771, 2004
6. G. Tsoumakas, I. Katakis, and L. Vlahavas, “Random k-labelsets for multilabel classification,” IEEE Trans. Knowl. Data Eng., vol. 23, no. 7, pp. 1079–1089, Jul. 2011.
7. J. Furnkranz, E. H € ullermeier, E. Loza Menc € ıa, and K. Brinker, “Multilabel classification via calibrated label ranking,” Mach. Learn., vol. 73, no. 2, pp. 133–153, 2008.
8. R. E. Schapire and Y. Singer, “BoosTexter: A boosting-based system for text categorization,” Mach. Learn., vol. 39, no 2/3, pp. 135–168, 2000.
9. M.-L. Zhang and Z.-H. Zhou, “ML-kNN: A lazy learning approach to multi-label learning,” Pattern Recognit., vol. 40, no. 7, pp. 2038–2048, 2007
10. A. Clare and R. D. King, “Knowledge discovery in multi-label phenotype data,” in Proc. 5th Eur. Conf. Principles Practice Knowl. Discovery Databases, 2001, pp. 42–53
11. Z.-H. Zhou and M.-L. Zhang, “Multi-instance multi-label learning with application to scene classification,” in Proc. 19th Int. Conf. Neural Inf. Process. Syst., pp. 2007, 1609–1616.
12. Kaggle datasets