LibMultiLabel: a Library for Multi-label Classification

LibMultiLabel Project Authors*

January 16, 2024

Abstract

LibMultiLabel is an open source software for binary, multi-class and multi-label classification, supporting various neural network architectures and linear classifiers. LibMultiLabel can be found at https://www.csie.ntu.edu.tw/~cjlin/libmultilabel/. This paper provides the mathematical formulations and implementation details of LibMultiLabel.

Contents

1 Metrics ........................................... 2
  1.1 Precision and Recall at K ................. 2
  1.2 R-Precision at K ............................ 2
  1.3 Normalized Discounted Cumulative Gains at K ... 3
  1.4 F-measure ..................................... 4
  1.5 Choosing the Suitable Metrics ............. 5

2 Handling zero-shot labels .................. 6
  2.1 The Default Behavior in LibMultiLabel .......... 6
  2.2 When to Set the Option to be True? .......... 6
  2.3 Remarks on Labels in Training/Validation Sets 6

*See contributors at https://github.com/ASUS-AICS/LibMultiLabel/graphs/contributors
1 Metrics

Metrics are functions that represent the performance of models during evaluation. When predicting, we use a model to calculate the scores for an instance associated with labels. For example, let \( \mathbf{w} \) be the weight of a linear model for label \( l \). Then for a given instance \( \mathbf{x} \), the score of \( \mathbf{x} \) for label \( l \) is calculated by \( \mathbf{w}^T \mathbf{x} \). This score will be used to decide whether this instance is associated with label \( l \). For this reason, we called this score decision value. If a given instance \( \mathbf{x} \) has the label \( l \), we say that the label \( l \) is relevant to \( \mathbf{x} \).

For a given data instance, let \( L \) be the number of labels and 
\[
\mathbf{p} = \begin{bmatrix} p_1 & p_2 & \cdots & p_L \end{bmatrix} \in \mathbb{R}^L, \\
\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_L \end{bmatrix} \in \{0, 1\}^L, \\
\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_L \end{bmatrix} \in \{0, 1\}^L
\]
be the decision values, the predictions, and the ground truths associated with the instance respectively. The value of 1 indicates a relevant label and 0 indicates an irrelevant label. Define \( I_p = \{i_1, i_2, ..., i_L\} \) to be the sorted index of \( \mathbf{p} \) by decision values.

1.1 Precision and Recall at \( K \)

Precision@\( K \) aims to check that among the top-\( K \) predictions for a given instance, how many labels are relevant to the instance. So, precision@\( K \) for the instance is defined as follows
\[
P@K = \frac{\text{#relevant labels in the top-}K \text{ predictions}}{K} = \frac{\sum_{s=1}^{K} y_{i_s}}{K}. 
\]
(1)

On the other hand, recall@\( K \) shows that among labels associated with the given instance, how many are in the top-\( K \) predictions. Recall@\( K \) for the instance can be defined as follows
\[
R@K = \frac{\text{#relevant labels in the top-}K \text{ predictions}}{\text{#relevant labels}} = \frac{\sum_{s=1}^{K} y_{i_s}}{\sum_{s=1}^{L} y_s}. 
\]
(2)

Note that, we set \( R@K = 0 \) if the number of relevant labels associated with the instance is zero.

The values of \( P@K \) and \( R@K \) over the entire dataset is the average of \( n \) instances, calculated as follows:
\[
P@K = \frac{1}{n} \sum_{j=1}^{n} P@K \text{ for the } j\text{th instance}, \\
R@K = \frac{1}{n} \sum_{j=1}^{n} R@K \text{ for the } j\text{th instance.}
\]

1.2 R-Precision at \( K \)

For instances where the number of relevant labels is less than \( K \), even with a perfect prediction, \( P@K \) will be smaller than 1. This is because
\[
P@K = \frac{\text{#relevant labels in the top-}K \text{ predictions}}{K} < \frac{K}{K} = 1.
\]
On the other hand, when $K$ is smaller than the number of relevant labels, then even with a perfect prediction, $R@K$ will be smaller than 1. The reason is

$$R@K = \frac{\text{#relevant labels in the top-}K\text{ predictions}}{\text{#relevant labels}} < \frac{\text{#relevant labels}}{\text{#relevant labels}} = 1.$$ 

For example, if the instance associates with two labels, then the value of $P@5$ for a perfect prediction is 0.4 and the value of $R@1$ for a perfect prediction is 0.5. For this reason, we cannot ensure that the values of $P@K$ and $R@K$ over different datasets for perfect predictions are always 1. To ensure the maximum value of the metric is 1, R-Precision at $K$ (RP@K) may be used.

RP@K is very similar to $P@K$ and $R@K$ as the only difference is in the denominator. The denominators of $P@K$ and $R@K$ are $K$ and the number relevant labels respectively. Instead, the denominator of RP@K is

$$\min(K, \text{#relevant labels of the instance}).$$

With this change, the maximum value of RP@K is always 1. The definition of RP@K for the instance is as follows

$$\text{RP@K} = \frac{\text{#relevant labels in the top-}K\text{ predictions}}{\min(K, \text{#relevant labels of the instance})} = \frac{\sum_{s=1}^{K} y_{is}}{\min(K, \text{#relevant labels of the instance})}.$$ 

Similarly, the value of RP@K over the entire dataset is the average of $n$ instances, calculated as follows:

$$\text{RP@K} = \frac{1}{n} \sum_{j=1}^{n} \text{RP@K for the } j\text{th instance}.$$ 

### 1.3 Normalized Discounted Cumulative Gains at $K$

When the number of relevant labels in the top-$K$ predictions are the same for two predictions, then by (1) and (2), these two predictions will have the same value of $P@K$ and $R@K$. In this case, these metrics cannot discriminate between the two predictions. For example, consider the ground truth and two predictions for an instance as follows

ground truth = [0, 1, 1, 0, 0],

decision values of prediction 1 = [0.1, 0.3, 1.0, -0.3, -0.7],

decision values of prediction 2 = [0.8, 0.2, 0.7, -0.1, -0.5].

In this case, $P@5$ for these two predictions are both 0.4, but have different orders of labels.

To understand why this matters, consider a search engine. If these are the search results, we hope that positive labels appear first. That is, positive labels have higher ranks. From this perspective, prediction 1 is better than prediction 2 in the example above.

To solve this problem, we use another metric called normalized discounted cumulative gains at $K$ (NDCG@K). Before introducing how to compute NDCG@K, we need to understand what DCG@K and IDCG@K are.

Discounted cumulative gains at $K$ (DCG@K) measures the top-$K$ predictions by taking discounts for different ranks. With this metric, the above two predictions will have different values of DCG@K and we can use these values to compare which is better. DCG@K for the instance is defined as follows

$$\text{DCG@K} = \sum_{s=1}^{K} \frac{y_{is}}{\log_2(s + 1)}.$$
A problem with DCG@K is that it is not comparable across instances with a different number of relevant labels. For example, consider these two instances

\begin{align*}
growth\text{ truth } 1 &= [0, 1, 0, 0, 0], \\
\text{decision values of prediction } 1 &= [0.1, 1.2, -0.9, -0.7, -0.5], \\
growth\text{ truth } 2 &= [1, 0, 1, 0, 1], \\
\text{decision values of prediction } 2 &= [0.3, 1.0, 0.4, -0.9, 0.1].
\end{align*}

Then DCG@5 for these two instances will be 1 and 1.52 respectively. Despite the first instance having the best possible prediction, it has a lower DCG@5 than the second instance. To solve this problem, one way is to consider the ratio of DCG@K for a prediction and DCG@K for the best prediction. DCG@K for the best prediction is called ideal DCG@K (IDCG@K) and this ratio called normalized DCG@K (NDCG@K).

IDCG@K is the maximum value of DCG@K. The maximum value of DCG@K occurs when all of the relevant labels are ranked higher than irrelevant labels. In other words, let \( I = \min(K, \|y\|_0) \). Note that \( \|y\|_0 \) is the 0-norm of \( y \), which is the number of non-zero elements of \( y \). Then the maximum value of DCG@K occurs when the top-\( I \) predictions for the given instance are all relevant. Thus, the expression of IDCG@K for the instance is defined as

\[
\text{IDCG@K} = \sum_{i=1}^{\min(k, \|y\|_0)} \frac{1}{\log_2(i + 1)}.
\]

NDCG@K shows how close the prediction is to the best possible prediction, calculated as follows:

\[
\text{NDCG@K} = \frac{\text{DCG@K for the instance}}{\text{IDCG@K for the instance}}.
\]

The value of NDCG@K over the entire dataset is the average of each instance, calculated as follows:

\[
\text{NDCG@K} = \frac{1}{n} \sum_{j=1}^{n} \text{NDCG@K for the } j\text{th instance}.
\]

### 1.4 F-measure

In the above, we introduced some ranking measures. They only check top-K predictions and \( K \) is usually a small number. When we need to consider the whole predictions, a ranking metric may not be a good choice. Instead, we may choose some classification measures like the F-measure. F-measure is one of the most used performance measures for information retrieval systems. It is the harmonic means of precision (\( P \)) and recall (\( R \)).

Precision shows that among predictions for all instances, how many positive predictions are correct. Recall shows that among positive instances, how many are predicted. Precision and recall for label \( l \) are expressed as follows:

\[
P_l = \frac{\#\text{TP for label } l}{\#(\text{TP + FP}) \text{ for label } l} \quad \text{and} \quad R_l = \frac{\#\text{TP for label } l}{\#(\text{TP + FN}) \text{ for label } l},
\]

where TP, FP, and FN are defined in Table \([1] \).
Table 1: Definition of TP, FP, FN, and TN.

<table>
<thead>
<tr>
<th></th>
<th>prediction</th>
<th>ground truth</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>TP (true positive)</td>
<td>FP (false positive)</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>FN (false negative)</td>
<td>TN (true negative)</td>
<td></td>
</tr>
</tbody>
</table>

Then the F-measure for label $l$ is

$$F_l = \frac{2 \cdot P_l \cdot R_l}{P_l + R_l} = \frac{2 \# \text{TP for label } l}{(2 \# \text{TP} + \# \text{FP} + \# \text{FN}) \text{ for label } l}$$

To extend the F-measure from single-label to multi-label, two approaches are developed in Tague (1981). The first is the macro-average F-measure, which is the unweighted mean of label F-measures,

$$\text{Macro-F1} = \frac{1}{L} \sum_{l=1}^{L} F_l = \frac{1}{L} \sum_{l=1}^{L} \frac{2 \# \text{TP for label } l}{(2 \# \text{TP} + \# \text{FP} + \# \text{FN}) \text{ for label } l}.$$  

Some use a different way, denoted as Macro*-F1, by calculating the average precision and recall over all labels first.

$$\bar{P} = \frac{1}{L} \sum_{l=1}^{L} P_l = \frac{1}{L} \sum_{l=1}^{L} \frac{\# \text{TP for label } l}{\#(\text{TP} + \text{FP}) \text{ for label } l},$$

$$\bar{R} = \frac{1}{L} \sum_{l=1}^{L} R_j = \frac{1}{L} \sum_{l=1}^{L} \frac{\# \text{TP for label } l}{\#(\text{TP} + \text{FN}) \text{ for label } l},$$

$$\text{Macro*-F1} = \frac{2 \cdot \bar{P} \cdot \bar{R}}{\bar{P} + \bar{R}}.$$

Opitz and Burst (2021) suggest that Macro*-F1 is less suitable to use.

The other multi-label measure is the micro-average F-measure, which calculates total TP, FP, and FN first.

$$\text{Micro-F1} = \frac{\sum_{l=1}^{L} \# \text{TP for label } l}{\sum_{l=1}^{L}(2 \# \text{TP} + \# \text{FP} + \# \text{FN}) \text{ for label } l}.$$

### 1.5 Choosing the Suitable Metrics

The choice of metrics should be motivated by the use case of the model. No metric fits every scenario equally well.

For example, if the model is used as a large-scale search engine, then the number of labels will be enormous. In this case, only the first few dozens of search results are important because no user will read every one of the results. For this reason, multi-label problems with a large amount of labels are often only concerned about the top few predictions. In this case, we might use P@K or NDCG@K with a choice of $K$ that reflects the use case well.

In contrast, multi-label problems with a small amount of labels are often concerned with predicting all the labels correctly. For example, illness prediction in medical data is usually concerned about every label. In such a case, we may choose to use Macro-F1 and Micro-F1.
2 Handling zero-shot labels

In some cases, there exist labels that only appear in the test data. These labels are called zero-shot labels. We provide an option include_test_labels in LibMultiLabel to handle these labels in evaluation. This option can be true or false to decide whether to include zero-shot labels for evaluation. In this section, we illustrate some details on how to choose a correct value of include_test_labels.

2.1 The Default Behavior in LibMultiLabel

The default value of include_test_labels is false because of the following reasons. Consider the case that models do not handle the zero-shot labels. If we include these labels for evaluation, the ranking measures are not affected. However, the classification measures such as Macro-F1 or Micro-F1 become different. In particular, because the F-measure of zero-shot labels is zero, the resulting Macro-F1, which is the unweighted mean of label F-measures, can be significantly different. In this situation, the zero-shot labels should not be included in the evaluation. Popular software such as scikit-learn does not include test labels for evaluation, and we hope to be consistent with them.

2.2 When to Set the Option to be True?

Sometimes, we may need to include zero-shot labels for evaluation. For example, if a paper experiments with approaches to handle zero-shot labels and report some classification measures, then to compare their results, the option include_test_labels should be true. For example, Chalkidis et al. (2019) propose the dataset EURLEX57K, which contains zero-shot labels. In their experimental results, they include zero-shot labels for evaluation and report Micro-F1. To compare with their results, we must set include test labels to be true.

2.3 Remarks on Labels in Training/Validation Sets

No matter which value of include_test_labels, we always consider the combined label set of training and validation sets. The reason is that training and validation instances are considered as all available data, so those labels that only appear in validation sets should not be regarded as zero-shot labels. Further, it is possible that we conduct training/validation splits several times. For example, we adopt the cross-validation strategy in our linear solvers. Therefore, it is better to consider the same label set across splits.

References

