Expede Herculem: Learning Multi Labels from Single Label

Dejun Mu , Junhong Duan, Xiaoyu Li, Hang Dai, Xiaoyan Cai and Lantian Guo
Shenzhen Research Institute, Northwestern Polytechnical University, Shenzhen 518057, China

Corresponding author: Dejun Mu (e-mail: madejun@nwpu.edu.cn).

This work was supported by the Natural Science Foundation of China under Grant 61672433, in part by the Fundamental Research Fund for Shenzhen Science and Technology Innovation Committee under Grant 201703063000511 and 201703063000517, in part by the National Power Grid Gansu Electric Power Company Science and Technology Project 522722180007, and in part by the Shaanxi Provincial Key R&D Program 2018KW-005.

ABSTRACT

Although there has been a lot of research in multi-label learning task, little attention has been paid on the weak label problem, in which only a subset of labels has been assigned to each instance in the training set. The extreme form of weak label learning is to predict all the label from just one label set in the training phase. In this paper, we focus on dealing with this kind of weak label learning task which is commonly met in old legacy information system, it is also called “Hercules Learning”. We propose a label-group-optimization based Hercules learning algorithm LGO, which divides the entire label set into multiple groups according to the classifier’s capability to distinguish them, so for each group we can train a classifier which can predict instance’s label within the group with high accuracy. The experimental results show that our algorithm is obviously superior to the existing weak label learning algorithm.

INDEX TERMS Multi-label classification, weak-supervised learning, genetic algorithm.

I. INTRODUCTION

One of the most studied tasks in machine learning is the single-label classification problem, which learns the samples associated with a single category, predicts the new instance’s category. However, even the most advanced algorithms failed to make desired performance on some real-world datasets. The common reason for that is many instances cannot be accurately labeled with a single category: either because these samples actually belong to multiple classes or because those classes are conceptually overlap.

Multi-label classification problem [1][2][3][4] has generated significant interest in recent years. The task of multi-label classification is to learn a function \( f: \mathcal{X} \rightarrow \mathbb{L} \) between the instance space \( \mathcal{X} \) and the label space \( \mathbb{L} \) from the training set \( \{(x_i, l_i)|1 \leq i \leq m\} \) [5]. Most of multi-label classification algorithms assume that all the proper labels of each training instance have been given. In many applications, however, only a subset of full labels is available, getting all the proper labels is usually too difficult. The problem of label prediction with a subset of full labels is called the “weak label” problem [6][7-10]. The extreme form of weak-label learning task is that the training sample contains only a single label, and the goal is to infer all the categories that the sample actually belongs to. We call it “Hercules Learning” Problem (coming from the Latin quotation “Expede Herculeus”, which means “from his foot we can measure Hercules”). This kind of scenario is often encountered when dealing with the data of old legacy information systems, such as book categories, news sections and social organization classifications.

In this paper, we propose a label-group-optimization based Hercules learning algorithm LGO (Label Group Optimization), which divides the entire label set into multiple groups according to the classifier’s capability to distinguish them. So, for each group, a classifier can be trained to predict instance’s label within the group with high accuracy. The final label set for each sample is the combination of the labels given by all these classifiers. We also apply several tricks to improve the efficiency and effectiveness of proposed algorithm. For example, fusion matrix is used instead of sample sets to estimate grouping effects, genetic algorithm is used to optimize grouping strategies.

The rest of this paper is organized as follows. Section II reviews related work. Section III illustrates the proposed
Hercules Learning algorithm. Section IV evaluates the experimental results and Section V presents the conclusion.

II. Related Work

Existing approaches of multi-label classification can be categorized into two different groups[1]: (i) problem transformation methods, and (ii) algorithm adaptation methods. The former one transforms the multi-label classification problem either into one or more single-label classification or regression problem. Various classification or regression algorithms can be applied after the transformation. The latter one extends specific classification algorithms in order to handle multi-label data directly.

There also exist three kinds of strategies based on the order of label correlations[5]: first-order strategy, second-order strategy and high-order strategy. Generally speaking, first-order strategy is tackled in a label-by-label style, having the side effect to its effectiveness while obtaining the efficiency. Second-order strategy is tackled by considering pairwise relations between instance labels, achieving a balance between effectiveness and efficiency. High-order strategy takes high-order relations among labels into account, getting stronger capabilities, while on the other hand is computationally more intensive and less scalable.

For the case where the annotation of the sample cannot be given full consideration, Sun[6] put forward the concept of “weak label” problem and proposed the WELL algorithm. More and more researchers focus on weak-label learning, which is an important branch in multi-label learning task, such as WSABIE[11], MPU[7], LEML[12] and MLML[9]. For WSABIE algorithm, the loss function is defined based on the ranking error[15], which indicates the count of labels which ranks higher than its given label. But when most of its “true” labels are missing, a higher rank loss does not equal to a worse prediction. So, we need to study this kind of weak label learning problem, especially its extreme form - "Hercules Learning” task.

III. Hercules Learning Algorithm

A. Basic Ideas

For each instance in a dataset, various labels can be defined from different conceptual aspects. For example, “male” or “female” can be labeled based on a person’s gender, “student”, “teacher” or “engineer” can be labeled based on the person’s occupation. The labels derived from one aspect are usually conceptually disjoint, but those from different aspects are often overlap.

The Label-Group-Optimization (LGO) algorithm attempts to automatically group all the labels into multiple different aspects so that within each group, there are sufficient discriminations between the instances with these labels. In this situation, for each label group, we can train a good enough classifier which can predict every instance’s label almost perfectly, and the instance’s final label set is simply the union of all the labels predicted by these classifiers. As shown in Figure 1, the LGO algorithm tries to find a good grouping solution, so that the samples of different class within the group do not overlap each other as much as possible in feature space.

![Figure 1 Label Grouping](image)
B. Distinguishability Estimation with Confusion Matrix

The first problem is how to quickly assess the distinguishability of a set of labels for a certain classification algorithm. We could actually train classifiers for each candidate group and evaluate the performance metrics, but it is too slow.

In the field of classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another). It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

Our approach is to use the submatrix of the confusion matrix for the initial assessment. We use the entire training set to train a classifier and get the confusion matrix of the classifier on the training set. For any candidate label group, we use the corresponding submatrix (the red part and blue part in Figure 2) to evaluate its performance on these classes.

![Confusion Matrix for Label Groups](image)

Not all traditional classification algorithm metrics are suitable in weak label problem. Assuming that there are missing labels on the dataset, then the label for the instance only means that the sample should belong to this category, but the conclusion that the sample does not belong to other categories cannot be obtained. In this case, for the classification results of the classifier, we can only confirm a subset of the True-Positive cases (see ⭐ in Table 1) and Type II errors (see ✦ in Table 1). The gray part of the table cannot be calculated.

![Confusion Matrix](image)

Table 1: Confusion matrix for multi-label dataset with label missing

<table>
<thead>
<tr>
<th></th>
<th>True Condition</th>
<th>Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Condition Positive</td>
<td>Condition Positive Unlabeled</td>
</tr>
<tr>
<td>Total Population</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predicted Condition Positive</td>
<td>True Positive</td>
<td>False Positive Type I error</td>
</tr>
<tr>
<td>Predicted Condition Negative</td>
<td>False Negative</td>
<td>True Negative</td>
</tr>
<tr>
<td></td>
<td>True Positive Labeled</td>
<td>False Negative Unlabeled Type II error</td>
</tr>
<tr>
<td></td>
<td>False Negative Labeled</td>
<td>Type II error</td>
</tr>
</tbody>
</table>
Therefore, in the strict sense, the conventional measurement of classifier, including Recall, Accuracy, and F1 Score, cannot be obtained from datasets with incomplete label set. Similar to the definition of Recall, we define
\[
\text{Recall} = \frac{\text{True Positive}}{\text{Condition Positive Labeled}}. 
\]
Although the calculation process of Recall and Recal is exactly the same in the classifier training process, we specifically emphasize the differences between traditional full-label problem and weak-label problem.

So, we set the optimization goal of label grouping is to maximum the average Recall for all the labels in each group. Let the confusion matrix \( C_{nxn} = [c_{ij}]_{nxn} \), the element \( c_{ij} \) in the matrix \( C \) is the number of training samples that are labeled as the \( i \)th class but are predicted to be the \( j \)th class. For any given group number \( K \), the optimization goal is:
\[
\text{Recall}_{Label} = \arg \max_{L=1}^{K} \sum_{i=1}^{n} \frac{\sum_{j=i}^{n} c_{ij}}{\sum_{j=1}^{n} \sum_{i=1}^{n} c_{ij}} \text{ for all groups} \quad (1)
\]

In the formula, \( L_i \) represents the label group, the conditions \( L = \bigcup_{i=1}^{K} L_i \) and \( L_i \cap L_j = \emptyset \) mean the entire label set is dividing into \( K \) non-overlap groups. The numerator is the sum of count of true-positive samples within each group. So, it is the sum of the diagonal elements of the confusion matrix. The denominator is the sum of samples in all label groups. So, the first part of the denominator is sum of count of true-positive samples as the numerator, the second part of is the sum of other elements in the \( K \) submatrix of the confusion matrix.

C. Label Group Optimization with Genetic Algorithm

After clarifying the optimization goal, we are looking for how to calculate this optimization process. First of all, what needs to be estimated is the size of the solution space. Under the premise of determining the number of groups \( K \), the elements \( N \) are divided into \( K \) groups, each group includes at least one element, and the number of combinations is known as the Second Stirling Number \( S(N, K) \). The recurrence calculation formula is:
\[
\begin{align*}
S(N, K) &= S(N - 1, K - 1) + KS(N - 1, K) \\
S(N, 1) &= 1 \\
S(N, N) &= 1
\end{align*}
\]

Obviously, it is usually an extremely large number so the solution cannot be found in a brute-force way. Therefore, we need to use certain optimization algorithms.

In the optimization method, the genetic algorithm is a kind of optimization method that has evolved from some phenomena in evolutionary biology and is a kind of evolutionary algorithm. In genetic optimization, the solution to an optimization problem is called an individual, expressed as a sequence of variables, commonly referred to as the chromosome. Chromosomes are generally expressed as simple strings or strings of numbers. This process is called coding. The algorithm randomly generates a certain number of individuals at the beginning. In each generation, each individual is evaluated and a fitness value is obtained by calculating the fitness function.

Subsequently, according to a certain selection strategy (in general, the higher the fitness is, the higher the probability of being selected), a certain number of individuals are selected to enter the breeding stage. The breeding stage generally includes two operators, Crossover and Mutation. The crossover refers to that under certain crossover probability (general range is 0.6–1), the chromosomes of two selected individuals are interchanged at the mating point, and two new chromosomes are generated instead of the original individuals. The mutation refers to changing the original chromosome at a random position according to a certain mutation probability (usually less than 0.1). Through multiple iterations, the newly-generated individual generation develops in the direction of increasing the degree of fitness until a solution satisfying the conditions is obtained.

In this optimization problem, we set \( \text{Recall}_{Label} \) as fitness. And we use integer coding as the coding strategy. The length of the chromosome is the number of categories \( N \), and the value of each gene on the chromosome is in the range of the group number \( \{1, 2, \ldots, K\} \). The category with the same value \( K \) on the chromosome represents the points in the same group. Such a chromosome represents a feasible solution, and all feasible solutions can be represented by chromosomes. After any chromosome is processed by the crossover and mutation operator, the result is still a feasible solution.

D. Novelty Detection with one-class SVM

Another consideration is what if an instance dose not belongs to any class within a label group. For we have divided all the class to \( K \) label groups, so we will always get \( K \) predicted label for any given instance. Maybe it is not true in some situation.

Novelty detection can be defined as the task of recognizing that test data differ in some respect from the data that are available during training[16]. One of the most commonly used novelty detection algorithm is one-class SVM(OCSVM)[17]. One-class SVM was proposed for estimating the support of a high-dimensional distribution. Given training vectors \( x_i \in \mathbb{R}^n, i = 1, \ldots, l \), the dual form of its prime optimization object is:
\[
\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha 
\]
subject to
\[
0 \leq \alpha_i \leq 1, \quad i = 1, \ldots, l
\]
\[
e^T \alpha = vl
\]
where \( Q_{i,j} = K(x_i, x_j) \) and \( K(\cdot) \) is the kernel function. The parameter \( v \) approximates the fraction of training errors and support vectors.

The decision function is
\[
\text{sgn} \left(\sum_{i=1}^{l} \alpha_i K(x_i, x) - \rho\right)
\]
We chose the Gaussian Radial Base Function (RBF) as the kernel function:
\[
K(x, x') = \exp \left(-\frac{\|x-x\|^2}{2\sigma^2}\right)
\]
E. LGO Algorithm
Considering all above points, we propose the LGO algorithm as follows:

In training phase, as shown in Figure 3, we first (1) train a classifier on whole training set by using a single label classification algorithm and produce the confusion matrix of this classifier; (2) get the optimized label group solution by genetic algorithm, and split the whole training set to multi subset according to the group such that each subset contains only samples whose labels belong to the group; (3) for each subset, train a classifier; (4) for each subset, train a novelty detector by using one-class SVM algorithm.

In the evaluation phase, as shown in Figure 4, for every test instance, and for each group, (1) first test if the instance belongs to this group by pre-trained novelty detector; (2) get the most probably label by pre-trained classifier; (3) combine all the labels get from all the group.

IV. Experimental Setup

A. Datasets
Table 2 presents the three datasets used for the experiment. These datasets are often used as a performance measure in multi-label classification. The table lists the total number of instances, the number of features per instance, and the total number of unique labels contained in the dataset. We chose yeast[18], [19] as a small dataset, mediammill[19], [20] as a medium-sized dataset, and nus-wide[19], [21] as a large-scale real-world dataset to test our algorithm.

In addition, we divide each dataset into a training set and a validation set by random sampling. The table also lists the total number of instances of training and validation sets.

Since this paper focuses on learning multiple label through a single label, we also deal with the labels of the training set instances. In data processing, we performed shuffle processing on the training set's label set and took the first as the training sample's label. Although the labels of the validation instances are likely to be incomplete, we consider these labels as the full set that the sample should actually correspond to for evaluation purpose.

<table>
<thead>
<tr>
<th>Name</th>
<th>#attribute</th>
<th>#examples</th>
<th>#labels</th>
<th>#train</th>
<th>#validate</th>
</tr>
</thead>
<tbody>
<tr>
<td>yeast</td>
<td>103</td>
<td>2417</td>
<td>14</td>
<td>1209</td>
<td>1208</td>
</tr>
<tr>
<td>mediamill</td>
<td>120</td>
<td>43907</td>
<td>101</td>
<td>21089</td>
<td>21088</td>
</tr>
<tr>
<td>nus-wide</td>
<td>128</td>
<td>209347</td>
<td>81</td>
<td>125449</td>
<td>83898</td>
</tr>
</tbody>
</table>
B. Comparison with Other Approaches

The experiment used random forest (RF) algorithm for single label classification as the evaluation baseline.

We choose two typical multi-label classification algorithms (WSABIE, LEML) with weak-label learning capabilities as comparison. For novelty detection, we chose OCSVM algorithm in combination with LGO. To determine whether novelty detection is necessary, we prepared two version of LGO, one with novelty detection (LGO-ND) and the other without (LGO). The algorithm implementations, parameters and hyperparameters are shown in Table 3.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Parameters &amp; Hyperparameters</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>max_depth=5, ntrees=100, nfold=5</td>
<td>H2O Deep Learning</td>
</tr>
<tr>
<td>LEML</td>
<td>hidden_space_dimension=1, l2_regularization=0.01, number_of_iter=10</td>
<td><a href="https://github.com/algorithmdog/Representation-based-Multilabel-Learning">https://github.com/algorithmdog/Representation-based-Multilabel-Learning</a></td>
</tr>
<tr>
<td>WSABIE</td>
<td>hidden_space_dimension=1, l2_regularization=0.01, number_of_iter=10, sparse_threshold=0.01, learning_rate=0.001</td>
<td><a href="https://github.com/algorithmdog/Representation-based-Multilabel-Learning">https://github.com/algorithmdog/Representation-based-Multilabel-Learning</a></td>
</tr>
<tr>
<td>LGO</td>
<td>group=3, (GA) population = 100, (GA) crossover_prob = 0.35, (GA) mutation_prob = 1/12, (GA) iter = 100</td>
<td>Java + jgap</td>
</tr>
<tr>
<td>LGO-OCSVM</td>
<td>group=3, (GA) population = 100, (GA) crossover_prob = 0.35, (GA) mutation_prob = 1/12, (GA) iter = 100, (SVM) nu=0.5, (SVM) degree=10</td>
<td>Java + jgap + svmlib</td>
</tr>
</tbody>
</table>

((GA) for Genetic Algorithm, (SVM) for One-Class SVM Algorithm)

C. Evaluation Measures

We use multi-label classification criteria Hamming_Loss, Accuracy, Macro_F1 and Instance_F1 to measure the performance.

Hamming_Loss evaluate the fraction of the wrong labels to the total number of labels. The Δ operator returns the symmetric difference between $Y_i$, the real label set of the $i$th instance, and $Z_i$, the predicted one. The $|r|$ operator counts the number of 1s in this difference, in other words the number of miss predictions. The total number of mistakes in the $n$ instances is aggregated and then normalized taking into account the number of labels and number of instances.[22]

$$\text{Hamming Loss} = \frac{1}{n} \sum_{i=1}^{n} |Y_i \Delta Z_i|$$  

In the multilabel field, Accuracy is defined as the
proportion between the number of correctly predicted labels and the total number of active labels, in the both real label set and the predicted one. The measure is computed by each instance and then averaged, as all example-based metrics.

\[
\text{Accuracy} = \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{|Y_i \cap Z_i|}{|Y_i|} \right]
\] (7)

Macro \( F_1 \) score is a familiar extension of the \( F_1 \) score for binary classification. To compute this metric, we average the \( F_1 \) score of all label columns in the data. Instance \( F_1 \) score is the F-measure averaging on each instance.

\[
\text{Macro} \_ F_1 = \frac{1}{k} \sum_{j=1}^{k} \frac{2 \sum_{i=1}^{n} y_{ij} \bar{y}_{ij}}{\sum_{i=1}^{n} y_{ij} + \sum_{i=1}^{n} \bar{y}_{ij}}
\] (8)

\[
\text{Instance} \_ F_1 = \frac{1}{n} \sum_{i=1}^{n} \frac{2 \sum_{j=1}^{k} y_{ij} \bar{y}_{ij}}{\sum_{j=1}^{k} y_{ij} + \sum_{j=1}^{k} \bar{y}_{ij}}
\] (9)

D. Experimental Results and Discussion

The experimental results are as Table 4. The LGO algorithm has achieved a significant lead in all the metrics except the Hamming Loss on all three datasets.

An interesting fact is that in these three datasets, adding novelty detection ability to the LGO algorithm does not seem to significantly improve the performance of the algorithm. From the experimental results, novelty detection only slightly reduced the Hamming Loss, but at the same time decreased the Accuracy, Macro \( F_1 \) and Instance \( F_1 \) slightly. So, novelty detection may be not a must-have in some situation.

Another point is the two algorithms (LEML and WSABIE) which we used for comparison do not show a significant performance improvement over the baseline algorithm. We speculate that it may be because each instance in the training set contains only one label that greatly weakens the prediction ability of these algorithms.

How to choose the group number K is also a problem which need to be study further. One idea is to explicitly specify how many labels an instance contains at most, then the number of groups can be determined based on the maximum number of labels. However, this method is subjective. The approach we take is to observe the difference between the fitness and the top K hits ratio of the classifier over the entire training set. When the difference reaches the maximum, we believe that the benefits brought by group optimization are also maximized.

We calculated the performance metrics of the LGO algorithm under different grouping numbers on all three datasets, and expressed all the performance metrics with Fitness and Top K Hits Ratio (denoted as Top_Hits) and their difference (denoted as delta) in the same coordinate axis (Figure 5). It can be seen that Hamming Loss and Macro \( F_1 \) generally show a trend of increasing gradually as the number of groups increases. It is worth noting that there is a peak on the delta values, and at this peak point (or nearby, as in nus-wide dataset), both Instance \( F_1 \) and Accuracy also have a distinct spike, and here Hamming Loss is also at a relatively better (smaller) value.

<table>
<thead>
<tr>
<th>dataset</th>
<th>Measure</th>
<th>Baseline</th>
<th>Comparisons</th>
<th>Our Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>RF</td>
<td>LEML</td>
<td>WSABIE</td>
</tr>
<tr>
<td>yeast</td>
<td>Hamming Loss↓</td>
<td>0.295234</td>
<td>0.28465</td>
<td>0.263955</td>
</tr>
<tr>
<td></td>
<td>Macro-F1†</td>
<td>0.109789</td>
<td>0.16123</td>
<td>0.123431</td>
</tr>
<tr>
<td></td>
<td>Instance-F1†</td>
<td>0.205262</td>
<td>0.208268</td>
<td>0.26726</td>
</tr>
<tr>
<td></td>
<td>Accuracy†</td>
<td>0.130158</td>
<td>0.142722</td>
<td>0.178025</td>
</tr>
<tr>
<td>mediamill</td>
<td>Hamming Loss↓</td>
<td>0.037886</td>
<td>0.036924</td>
<td>0.038081</td>
</tr>
<tr>
<td></td>
<td>Macro-F1†</td>
<td>0.044218</td>
<td>0.019244</td>
<td>0.012418</td>
</tr>
<tr>
<td></td>
<td>Instance-F1†</td>
<td>0.350514</td>
<td>0.345858</td>
<td>0.335039</td>
</tr>
<tr>
<td></td>
<td>Accuracy†</td>
<td>0.238076</td>
<td>0.260381</td>
<td>0.223628</td>
</tr>
<tr>
<td>nus-wide</td>
<td>Hamming Loss↓</td>
<td>0.030922</td>
<td>0.028921</td>
<td>0.029022</td>
</tr>
<tr>
<td></td>
<td>Macro-F1†</td>
<td>0.019914</td>
<td>0.031744</td>
<td>0.036301</td>
</tr>
<tr>
<td></td>
<td>Instance-F1†</td>
<td>0.300768</td>
<td>0.302434</td>
<td>0.304361</td>
</tr>
<tr>
<td></td>
<td>Accuracy†</td>
<td>0.255914</td>
<td>0.249902</td>
<td>0.238051</td>
</tr>
</tbody>
</table>

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2876014, IEEE Access
Figure 5 relationship between group number, GA fitness and performance metrics
V. Conclusion
This paper has presented a simple but powerful method for an extreme form of weak label learning problem: learning multi-labels from just one label per-instance dataset. We first train a traditional single-label classifier on the training set and use the training confusion matrix get the optimized label group. Then for each label group, we re-train a classifier and a novelty detector. Finally, we ensemble these classifiers and novelty detectors to predict instance’s label. Experiments show that our algorithm is obviously superior to the existing weak label learning algorithm.

REFERENCES