Classifying multiple categories through label and feature grouping

Girish Pandey
IIT Jodhpur
Jodhpur, India
m22cs056@iitj.ac.in

Kwanit Gupta
IIT Jodhpur
Jodhpur, India
gupta.45@iitj.ac.in

ABSTRACT

Managing cases with several labels at once is the goal of multi-label classification (MLC), a difficult task in machine learning. Effective feature selection and dimensionality reduction strategies are necessary to enhance the performance and efficiency of MLC models when dealing with the high-dimensional and noisy data often associated with MLC jobs. To make use of the label-specific features and the label correlations in multi-label datasets, we offer a unique MLC framework that combines feature selection with label clustering. First, we use feature-label mutual information and feature interaction to select features; second, we use label co-occurrence and label similarity to cluster labels; and third, we use these features and labels to train and infer an MLC model. Several real-world multi-label datasets from various domains, including AmazonTitles, Bibtex, Delicious, etc., are used to test the efficacy of our framework. Both clustering-based and non-clustering-based MLC approaches are compared to our system. As can be seen from the experimental findings, our system provides competitive outcomes across a range of assessment criteria, including Precision@k, DCG@k, and nDCG@k. Additionally, our framework exhibits clear benefits in terms of computational efficacy and scalability. We conclude that by capitalising on the feature-label relationships and the label structure, our approach is well-suited for managing multi-label datasets of high dimensionality and label cardinality.

KEYWORDS

Big Data, MultiLabel Classification, Bag-Of-Word Features

ACM Reference Format:
Girish Pandey and Kwanit Gupta. 2018. Classifying multiple categories through label and feature grouping. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In machine learning, multi-label classification (MLC) refers to cases that are tagged with more than one label. It is possible to assign numerous labels to a picture, such as "cat," "dog," and "grass," or to divide a prose document into distinct sections, such as "sports," "politics," and "entertainment." Multiple-label classification (MLC) is useful in many fields, from biology to music classification.

In the context of big data, whereby data are characterised by high volume, velocity, diversity, validity, and value [5], MLC is a hard challenge. There are several difficulties that big data presents to MLC, such as:

The high dimensionality of big data necessitates the use of efficient feature selection and dimensionality reduction methods to boost the effectiveness of MLC models. In order to deal with the complexity of the label space and the label imbalance problem, big data often entail a large number of labels. Label correlations, label hierarchies, and label noise are only a few examples of the complicated label relationships that frequently arise in big data, necessitating advanced techniques for making use of the label information and label quality. In this article, we examine the current state of MLC in big data, discussing its many applications and analysing the pros and cons of the various approaches now in use. We zero in on binary relevance (BR), classifier chain (CC), label powerset (LP), and random k-labelsets (RAkEL) as our primary technique categories. The MLC issue is broken down into smaller, more manageable chunks that may then be handled by common, single-label classifiers in these approaches. The fundamental ideas behind these approaches are:

This paper will follow the following outline: In Section 2, we provide some context for multi-label classification (MLC) and big data in general; in Section 3, we review the four types of methods mentioned above and discuss their strengths and weaknesses; in Section 4, we present some experimental results on several real-world multi-label datasets from different domains; and in Section 5, we conclude this paper and make some suggestions for future work.

2 METHODOLOGY

2.1 Data Abstraction and Preprocessing

The information is provided as a LIBSVM-formatted file, which is a standard sparse data format. There are labels and features for each instance in the file, with an index and value for each feature. The characteristics are separated by spaces, while the labels are separated by commas. This is how an example might look:

1,2 3:0.5 5:0.7 9:0.2

This indicates that the instance contains three characteristics with indices 3, 5, and 9, with values of 0.5, 0.7, and 0.2, respectively, and that it belongs to labels 1 and 2. The information is split into a label column and a features column. The features are dictionaries with the keys as indices and the values as the values, while the

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

labels are lists. As an illustration, the situation stated above may be written as:

[1, 2], 3:0.5, 5:0.7, 9:0.2

So, we have segregated labels and index-value pair into 2 columns , the index-value pairs are stored in dictionary format, further we have generated 2 different sparse representation from those columns in form of 2 different 2-D arrays

2.2 Diverse Training

2.2.1 Multilabel Containers.

- In Binary Relevance , This technique trains a different classifier for each label, as if it were a unique binary classification issue. While this approach is easy to implement at scale, it risks producing subpar results due to its reliance on a large number of redundant labels.
- In Classifier Chain , This strategy builds a series of classifiers, each of which is in charge of a different label and incorporates the predictions of its predecessors as new characteristics. The label dependencies throughout the chain may be captured using this approach, however the method is sensitive to the order of the labels and may experience error propagation if the order is changed.
- In LabelPower Set, it considers each distinct label combination to be a separate class. Despite its ability to comprehensively capture label dependencies, this approach may struggle with data sparsity and is not suitable for use with very large label spaces.
- In RAkEL, the label space is randomly divided into many subgroups of k labels, and the LP approach is then applied to each subset. Although this technique can simplify and sparsify the LP approach, it also adds unnecessary noise and duplication.

2.2.2 Base Estimators.

- To categorise cases based on their feature values, Gaussian Naive Bayes employs Bayes' theorem in a probabilistic method. All features are assumed to be uncorrelated with one another and to have a normal (Gaussian) distribution. Predicting the probability of each label given the input characteristics may be done using Gaussian Naive Bayes in multilabel models as a base estimator. It's great for high-dimensional datasets since it's computationally efficient and can process a lot of information.
- The likelihood of a binary result (such as the presence or absence of a label) may be modelled as a function of the input data using a linear approach called logistic regression. With the addition of a binary logistic regression model for each label in a multilabel model, Logistic Regression may be generalised to accommodate a large number of labels. It is a common choice for multilabel classification jobs due to being a simple, interpretable algorithm that performs well with sparse data.
- Classifying instances according to their feature values, the Decision Tree Classifier is a non-parametric method that learns a hierarchical set of rules. It builds a tree structure out of the data in a recursive fashion, splitting it at each node according to the characteristic that delivers the most information gain. Decision Tree Classifier may be used as a base estimator in multilabel models by fitting individual decision trees for each label. The algorithm's strength is in its ability to capture intricate interplays between features, while its weakness lies in its susceptibility to overfitting and its potential inability to generalise successfully to novel data.
- To boost overall performance and lessen overfitting, the Random Forest Classifier employs a collection of Decision Tree Classifiers to construct an ensemble algorithm. To get at a final prediction, it constructs numerous decision trees using random subsets of the data and characteristics. By fitting a distinct random forest for each label, the Random Forest Classifier may serve as a base estimator in multilabel models. It is a common choice for multilabel classification tasks due to the algorithm's robustness in the face of noisy data and nonlinear correlations between components.
- Another ensemble approach that uses random subsets of data to train several base estimators is the Bagging Classifier. In order to arrive at a conclusive conclusion, it takes the predictions and uses a majority vote (for classification) or an average (for regression). To utilise Bagging Classifier as a base estimator in multilabel models, just fit a separate bagging classifier for each label. However, other ensemble algorithms, such as Random Forest or Boosting, may be more efficient. This flexible algorithm can enhance the reliability and precision of base estimators.
- A boosting approach, AdaBoost Classifier gives more weight to misclassified examples with each iteration so as to combine numerous weak base estimators. It takes all the forecasts and uses a weighted majority to determine an overall prediction. AdaBoost Classifier may be used as a base estimator in multilabel models by training a different AdaBoost classifier for each label. It's a potent approach that can help base estimators perform better, especially when dealing with noisy data or unequal class representation.
- By projecting the input characteristics into a higher dimensional space, the RBF Support Vector Classifier is able to locate the hyperplane that most effectively divides the data into distinct classes. Specifically, it evaluates how similar two instances are using a kernel based on the radial basis function (RBF). By training a different support vector classifier for each label, the RBF Support Vector Classifier may serve as a baseline estimator in multilabel models. Although it is a powerful algorithm, its performance can be heavily impacted by factors such as the choice of kernel and hyperparameters, and it may be computationally expensive.

3 RESULTS

For seeing results on other datasets, kindly refer to the colab notebook [https://colab.research.google.com/drive/1-nWN21MTzhG93D_eX-xuOlF3eUmtyk9Z?usp=sharing].

Following are the time readings of different approaches:-

Approaches	Train Time (s)	Predict Time (s)
Binary Relevance	60 - 1683	1 - 412
Classifier Chains	49 - 1415	1 - 431
Label Power Set	1 - 564	0.1- 117
RakelD	16 - 1964	4 - 404

Table 1: Time Comparison with Approaches

Following are the precision@k and ndcg@k of different approaches:-

Approaches	P@1	P@3	P@5	ndcg@5
Binary Relevance	35.64	31.39	30.06	29.66
Classifier Chains	34.5	30.26	29.17	28.85
Label Power Set	28.73	28.94	28.15	27.81
RakelD	29.76	25.58	25.05	24.9

Table 2: Evaluation Comparison with Approaches

Following are the resident and virtual memory usages of different approaches:-

Approaches	Virtual Usage (MiB)	Residual Usage (MiB)
Binary Relevance	3325 - 5402	2119 - 4198
Classifier Chains	3271 - 3394	2092 - 2184
Label Power Set	3412 - 4641	2165 - 3376
RakelD	3604	2339

Table 3: Memory Comparison with Approaches

4 OBSERVATIONS AND LIMITATIONS

- To begin, the relationship between labels is not taken into account by Binary Relevance. When labels are interdependent, this can cause subpar efficiency. So, if we have two labels, "dog" and "animal," it's safe to assume that having the "dog" label also means having the "animal" label. Binary Relevance may miss this connection since it considers each label in isolation. Second, when the number of labels is huge, using Binary Relevance can lead to a large number of classifiers to train. If there are 10 labels, for instance, 10 individual binary classifiers will need to be trained. This may take a lot of RAM and is computationally intensive. In the end, Binary Relevance makes the potentially misleading assumption that all labels are of equal importance. Binary Relevance does not account for the possibility that some labels are more significant than others.
- Classifier Chains might be sensitive to the sequence in which labels are chained, which is a drawback. Finding the best possible order among many possible combinations is challenging. Classifier Chains, like Binary Relevance, can provide a huge number of classifiers that need to be trained if the number of labels is substantial. Classifier Chains also has the drawback of assuming that the order of the labels represents the interdependence between the labels, which is not always the case. However, this isn't always the case, and if things aren't in the right order, performance may suffer.
- The difficulty in training and producing predictions is hindered by the fact that the number of classes might expand exponentially with the number of labels. There's also the fact that in the LabelPower Set approach, it's assumed that the probabilities of any label combination are equal, which isn't necessarily the case. LabelPower Set does not account for the possibility that certain possible label combinations occur more frequently than others.
- RakelD's effectiveness is limited, for example, by how well the labels are clustered. If the clustering isn't done well, the resulting label subsets can not capture the important label relationships, resulting in subpar results. It can be computationally costly to train and forecast because, like with LabelPower Set, the number of classifiers grows exponentially with the number of labels.

5 CONTRIBUTIONS

We would like to definitely mention that we both have built together the codebases, wrote the report and made the poster.