A Survey on different Multi label Classification Using Deep/ML

Gaurav Kumar
Department of Computer Science & Engineering
Sikkim Manipal Institute of technology
gaurav_201800295@smit.smu.edu.in

Aditi Karmakar
Department of Computer Science & Engineering
Sikkim Manipal Institute of technology
aditi_201800107@smit.smu.edu.in

Guided by:
Mrs. Chitrapriya Ningthoujam
Assistant Professor
Department of Computer Science & Engineering
Sikkim Manipal Institute of technology
chitrapriya.n@smit.smu.edu.in

Abstract

This survey paper emphasizes the idea of multi-label classification, analyses several assessment measures, and compares the available techniques. This work also connects multi-label issues to related but distinct problems that are frequently converted to multi-label problems in order to get access to a diverse set of multi-label methods. Multi-labelled classification has a wide range of applications, including text categorization, semantic picture labelling, and gene functioning classification, among others, and the scope and interest in these applications is growing. Depending on the characteristics of the classification problem, neural network models can be built to handle multi-label classification and perform well.

1. Introduction

In machine learning, single label classification is a common classification problem where the goal is to learn from a set of instances, each associated with a unique class label from a set of disjoint class labels. Depending on the total number of disjoint classes in Label, the problem can be identified as binary classification (when |Label Set| = 2) or multi-class classification (when |Label Set| > 2) problem. Unlike these problems, multi-label classification allows the instances to be associated with more than one class.

Even though multi-label classification was primarily motivated by the emerging need for automatic text-categorization and medical diagnosis, recent realization of the omnipresence of multi-label prediction tasks in real world problems drawn more and more research attention to this domain [1]. Traditional binary and multi-class problems both can be posed as specific cases of multi-label problem. However, the generality of multi-label problems makes it more difficult than the others [2].

2. Problem Transformation

2.1. Label Powerset(LP)

Label Powerset is a straight forward method that considers each unique set of labels in a multi-label training data as one class in the new transformed data. Therefore, the new transformed problem is a single label classification task. For a new instance, LP outputs the most probable class which actually is a set of classes in the original data. It is also possible to produce a ranking of labels using LP, given the classifier can output a probability distribution over all newly formed classes [3]. Table 1 shows the dataset transformed using LP method. The computational complexity of LP is upper bounded by

min(n, 2^k), where n is the total number of data instances and k is the total number of classes in the training data (before transformation).When the number of classes increases the number of distinct label combinations can grow exponentially[3]. This easily leads to combinatorial explosion and thus computational infeasibility. In practice complexity would be much less than 2^k, still for large values of n and k this can be an issue. The second problem with this approach is that, a large number of classes (set of labels in the original data) would be associated with very few examples and that would also pose extreme class imbalance problem for learning. We have the data set like this, where X is the independent feature and Y's are the target variable.

X	Y_1	Y_2	Y_3	Y_4
X_1	0	1	1	0
X_2	1	0	0	0
X_3	0	1	0	0
X_4	0	1	1	0
X_5	1	1	1	1
X_6	0	1	0	0

Here x1 and x4 have the same labels, similarly, x3 and x6 have the same set of labels.
So, label powerset transforms this problem into a single multi-class problem as shown below.

X	Y_1
X_1	1
X_2	2
X_3	3
X_4	1
X_5	4
X_6	3

Table 1: Transformed data using Label Powerset method

Algorithm:-

```
# using Label Powerset
from skmultilearn.problem_transform import LabelPowerset

# initialize label powerset multi-label classifier
classifier = LabelPowerset(LogisticRegression())

# train
```

```
classifier.fit(x_train, y_train)

# predict
predictions = classifier.predict(x_test)
```

2.2. Binary Relevance

An ensemble of single-label binary classifiers is trained, one for each class. Each classifier predicts either the membership or the non-membership of one class. The union of all classes that were predicted is taken as the multi-label output[5]. Binary Relevance is one of the most popular approaches as a transformation method that actually creates k datasets (k = |Label set|, total number of classes), each for one class label and trains a classifier on each of these datasets. Each of these datasets contains the same number of instances as the original data, but each dataset $D\lambda_j$, $1 \leq j \leq k$ positively labels instances that belong to class λ_j and negative otherwise[6]. Table 2 show the example dataset transformed for Binary Relevance.

X	Y_1	Y_2	Y_3	Y_4
X_1	0	1	1	0
X_2	1	0	0	0
X_3	0	1	0	0
X_4	1	0	0	1
X_5	0	0	0	1

In binary relevance, this problem is broken into 4 different single class classification problems as shown in the figure below.

X	**Y_1**
X_1	0
X_2	1
X_3	0
X_4	1
X_5	0

X	**Y_2**
X_1	1
X_2	0
X_3	1
X_4	0
X_5	0

X	**Y_3**
X_1	1
X_2	0
X_3	0
X_4	0
X_5	0

X	**Y_4**
X_1	0
X_2	0
X_3	0
X_4	1
X_5	1

Table 2: Transformed data sets produced by Binary Relevance method

Once these datasets are ready, it is easy to train on binary classifier for each. For any new instance, Binary Relevance outputs the union of the labels λj that are positively predicted by the k classifiers. While Binary Relevance has been used in many practical

applications, it has been widely criticized for its implicit assumption of label independence.

Algoritm:-

```
# using binary relevance
from skmultilearn.problem_transform import BinaryRelevance
from sklearn.naive_bayes import GaussianNB

# initialize binary relevance multi-label classifier
# with a gaussian naive bayes base classifier
classifier = BinaryRelevance(GaussianNB())

# train
classifier.fit(X_train, y_train)

# predict
predictions = classifier.predict(X_test)
```

2.3. Classifier Chains

A chain of binary classifiers C0, C1, . . . , Cn is constructed, where a classifier Ci uses the predictions of all the classifier Cj , where j < i. This way the method, also called classifier chains, can take into account label correlations[7]. In this, the first classifier is trained just on the input data and then each next classifier is trained on the input space and all the previous classifiers in the chain. The total number of classifiers needed for this approach is equal to the number of classes, but the training of the classifiers is more involved. In the dataset given below, we have X as the input space and Y's as the labels.

X	Y_1	Y_2	Y_3	Y_4
X_1	0	1	1	0
X_2	1	0	0	0
X_3	0	1	0	0

In classifier chains, this problem would be transformed into 4 different single label problems, just like shown below. Here yellow colored is the input space and the white part represent the target variable.

X	Y_1
X_1	0
X_2	1
X_3	0

Classifier 1

X	Y_1	Y_2
X_1	0	1
X_2	1	0
X_3	0	1

Classifier 2

X	Y_1	Y_2	Y_3
X_1	0	1	1
X_2	1	0	0
X_3	0	1	0

Classifier 3

X	Y_1	Y_2	Y_3	Y_4
X_1	0	1	1	0
X_2	1	0	0	0
X_3	0	1	0	0

Classifier 4

Algorithm:-

```
# using classifier chains
from skmultilearn.problem_transform import ClassifierChain
from sklearn.naive_bayes import GaussianNB

# initialize classifier chains multi-label classifier
# with a gaussian naive bayes base classifier
classifier = ClassifierChain(GaussianNB())

# train
classifier.fit(X_train, y_train)

# predict
predictions = classifier.predict(X_test)
```

2. Adapted Algorithm

Algorithm adaptation methods for multi-label classification concentrate on adapting single-label classification algorithms to the multi-label case usually by changes in cost/decision functions. Adapted algorithm to directly perform multi-label classification, rather than transforming the problem into different subsets of problems. Here we use a multi-label lazy learning approach named ML-KNN which is derived from the traditional K-nearest neighbor (KNN) algorithm[8].

The skmultilearn.adapt module implements algorithm adaptation approaches to multi-label classification, including but not limited to ML-KNN.

Algorithm:-

```
from skmultilearn.adapt import MLkNN
from scipy.sparse import csr_matrix, lil_matrix

classifier_new = MLkNN(k=10)

# Note that this classifier can throw up errors when handling sparse matrices.

x_train = lil_matrix(x_train).toarray()
y_train = lil_matrix(y_train).toarray()
x_test = lil_matrix(x_test).toarray()

# train
classifier_new.fit(x_train, y_train)

# predict
predictions_new = classifier_new.predict(x_test)
```

5. Ensemble approaches

Ensemble always outperforms the individual. The Scikit-Multilearn package includes many ensembling categorization routines that we may use to improve your findings. For multi-label classification tasks, ensemble techniques have been proved to be a useful tool. The idea behind ensemble classification is to learn a group of classifiers, called an ensemble of classifiers, and then use some type of voting to aggregate their predictions for the categorization of unseen cases[9].

Algorithm:-

.

Construction of the ensemble classifier:

Input: k-labelset matrix – M
The training set - T

Output: An ensemble of LP classifiers hi
1. for i 1 to the number of rows in M do

2. ***Yi the labelset represented by the i-th row in M;***
3. ***train an LP classifier hi : T P(Yi) on D;***
4. ***end for***

6. Evaluation Metrics:

We must average out the classes in order to measure a multi-class classifier.
Micro-averaging and macro-averaging are two different methods for achieving this.
The average of all TPs, TNs, FPs, and FNs for each class is calculated using micro-averaging.

. Microaveraging Precision $Prc^{micro}(D) = \frac{\sum_{c_i \in \mathcal{C}} TPs(c_i)}{\sum_{c_i \in \mathcal{C}} TPs(c_i) + FPs(c_i)}$

. Microaveraging Recall $Rcl^{micro}(D) = \frac{\sum_{c_i \in \mathcal{C}} TPs(c_i)}{\sum_{c_i \in \mathcal{C}} TPs(c_i) + FNs(c_i)}$

We add up the individual true positives, false positives, and false negatives of the system for distinct sets and apply them in the micro-averaging approach. The harmonic mean of the preceding two equations will be the micro-average F1-Score.

Macro-averaging is straight forward. We just take the average of the precision and recall of the system on different sets.

. Macroaveraging Precision $Prc^{macro}(D) = \frac{\sum_{c_i \in \mathcal{C}} Prc(D, c_i)}{|\mathcal{C}|}$

. Macroaveraging Recall $Rec^{macro}(D) = \frac{\sum_{c_i \in \mathcal{C}} Rcl(D, c_i)}{|\mathcal{C}|}$

When we want to know how the system performs overall across several data sets, we may utilize the macro-averaging approach. With this average, we should not make any specific decisions. Micro-averaging, on the other hand, can be a beneficial metric when our dataset is of varying size.

7. Conclusion

Problem transformation techniques and algorithm adaptation methods are the two basic approaches to solving a multi-label classification problem. The multi-label problem is transformed into a collection of binary classification problems, which may then be solved by single-class classifiers.

Whereas algorithm adaptation methods adapt the algorithms to directly perform multi-label classification. In other words, rather than trying to convert the problem to a simpler problem, they try to address the problem in its full form. In an extensive comparison with other approaches, label-powerset method scores best, followed by the one-against-all method.

8. Further improvements

We could utilise decision trees for increased speed, and ensemble models for a suitable trade-off between speed and accuracy.

The same problem can be solved using LSTMs in deep learning.

9. References

[1] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining Multi-label Data. O. Maimon, L. Rokach (Ed.), Springer, 2nd edition, 2010.

[2] J. Read. A Pruned Problem Transformation Method for Multi-label classification. In Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008), pages 143–150, 2008.

[3] Xiatian Zhang, Quan Yuan, Shiwan Zhao, Wei Fan, Wentao Zheng, and Zhong Wang. Multi label classification without the multi-label cost. In SDM, pages 778–789, 2010.

[4] G.Tsoumakas and I. Vlahavas. Random k-Labelsets: An Ensemble Method for Multilabel Classification. In Proceedings of the 18th European Conference on Machine Learning (ECML 2007), pages 406–417, Warsaw, Poland, September 2007

[5] Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label Classification Using En sembles of Pruned Sets. In ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, volume 0, pages 995–1000, Washington, DC, USA, 2008. IEEE Computer Society

[6] https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/

[7] https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff

[8] https://www.researchgate.net/publication/247153035_Ensemble_Methods_for_Multi-label_Classification

[9] https://www.researchgate.net/publication/263813673_A_Review_On_Multi-Label_Learning_Algorithms