IE 506: Machine Learning - Principles and Techniques **Jan-May 2023**

Programming Challenge : AdaBoost to handle multi-label classification

Name: Jayshankar Yadav *Rollno: 22n0460*

Abstract

This report explores the use of AdaBoost, a popular ensemble learning algorithm, for multilabel classification problems. Specifically, the focus is on two problem transformation methods: Binary Relevance and Classifier Chains. Both methods are commonly used to tackle multilabel classification problems by transforming the original problem into multiple binary classification sub-problems. The report discusses the details of each method and their implementation using AdaBoost. An experimental analysis is conducted on several benchmark datasets to evaluate the performance of both methods in terms of accuracy, F1 score, and execution time.

1 Introduction

Multilabel classification is a challenging task in machine learning that involves predicting multiple output labels for each input instance. It has numerous practical applications, such as text classification, image annotation, and biological data analysis. Several machine learning algorithms have been proposed to solve multilabel classification problems, including problem transformation methods like Binary Relevance and Classifier Chains.

In this report, we investigate the use of AdaBoost, an ensemble learning algorithm, with Binary Relevance and Classifier Chains for multilabel classification. AdaBoost is a powerful algorithm that combines weak learners to build a strong classifier. Binary Relevance and Classifier Chains are two popular problem transformation methods that transform a multilabel classification problem into multiple binary classification sub-problems, which can be easily solved by AdaBoost.

The main objective of this report is to provide a comprehensive understanding of the use of AdaBoost with Binary Relevance and Classifier Chains for multilabel classification. We first introduce the problem of multilabel classification and discuss its importance and challenges. Next, we describe the AdaBoost algorithm, Binary Relevance, and Classifier Chains in detail. We then present an experimental analysis of the methods using several benchmark datasets and evaluate their performance in terms of accuracy, F1 score, and execution time.

2 Methods and Approaches

To train AdaBoost for multi-label classification, I have followed the steps of transforming the problem into binary relevance and classifier chains. After that, I have implemented AdaBoost for single-label classification.

1 **Binary Relevance**
This is the simplest technique, which basically treats each label as a separate single class classification problem.

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

For example, let us consider a case as shown below. We have the data set like this, where X is the independent feature and Y's are the target variable.

2 **Classifier Chains** 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.

Let's try to this understand this by an example. In the dataset given below, we have X as the input space and Y's as the labels.

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	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.

This is quite similar to binary relevance, the only difference being it forms chains in order to preserve label correlation.

Here are some details on the training procedure, hyperparameter tuning, and cross-validation used in training:

X	y1
x1	0
x2	1
x3	0

Classifier 1

X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2

X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

Classifier 3

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Classifier 4

1. Training Procedure:

a. **Model Training:** For training the AdaBoost model, i have used the binary relevance and classifier chains transformation to convert the multi-label problem into a set of single-label classification problems. Then, I have trained an AdaBoost model for each of these single-label problems.

b. **Prediction:** After training the model, I have made predictions using the testing data. To make predictions, i have used the ensemble of AdaBoost models for each single-label classification problem.

2. Hyperparameter Tuning:

To optimize the performance of the AdaBoost model, I have performed hyperparameter tuning. The hyperparameters that i have tuned include:

a. Number of estimators: The number of estimators is the number of weak classifiers used in the AdaBoost model. You have used cross-validation to find the optimal number of estimators.

b. Learning rate: The learning rate is the rate at which the algorithm learns from the mistakes of the previous iterations..

In my experiment i have fixed learning rate which is 0.5.

3 Experiments and Results

1 **Binary Relevance**

For Training data

	Model	Accourcy(n =50)	F1_score(n =50)	Accourcy(n =100)	F1_score(n =100)	Accourcy(n =150)	F1_score(n =150)
0	logistic	0.522701	0.316405	0.522426	0.316610	0.516365	0.313917
1	decision_tree	0.445100	0.281993	0.463158	0.292109	0.470004	0.296475
2	naive_bayes	0.458457	0.291577	0.453663	0.288918	0.453975	0.288232
3	svm	0.690067	0.389428	0.694634	0.391832	0.695702	0.392057

For Test data

2 **Classifier Chains**

over Traing data

	Model	Accourcy(n =50)	F1_score(n =50)	Accourcy(n =100)	F1_score(n =100)	Accourcy(n =150)	F1_score(n =150)
0	logistic	0.460024	0.286669	0.441087	0.279262	0.438738	0.278304
1	decision_tree	0.421397	0.270143	0.411802	0.270927	0.373952	0.254543
2	naive_bayes	0.352619	0.243385	0.328127	0.227052	0.362706	0.245716
3	svm	0.426294	0.272709	0.476444	0.295263	0.440833	0.280494

	Model	Accourcy(n =50)	F1_score(n =50)	Accourcy(n =100)	F1_score(n =100)	Accourcy(n =150)	F1_score(n =150)
0	logistic	0.687717	0.396135	0.691561	0.397238	0.686940	0.395259
1	decision_tree	0.624395	0.373197	0.613559	0.368477	0.611967	0.368503
2	naive_bayes	0.540399	0.334840	0.549750	0.338163	0.557406	0.342579
3	svm	0.730638	0.411921	0.732261	0.413108	0.732869	0.413119

over test data

	Model	Accourcy(n =50)	F1_score(n =50)	Accourcy(n =100)	F1_score(n =100)	Accourcy(n =150)	F1_score(n =150)
0	logistic	0.614968	0.367587	0.641230	0.374263	0.639730	0.376433
1	decision_tree	0.607040	0.365897	0.589754	0.360341	0.580111	0.350756
2	naive_bayes	0.457810	0.298508	0.471992	0.303464	0.499373	0.316016
3	svm	0.675833	0.388579	0.662516	0.386012	0.666484	0.385044

4 Guide for using code

In the code i have implemented several classification method as a base classifer because of them it take time to run hole code. for checking validation of code you can run the last cell of the code which for experiment purpose. While running last cell you need to input a test data after than you need to choose base classifier and also need to choose which method you want to implement binary relevance or classifier chains. In all above code cell i have implemented adaboost by choosing different different base classifier and also check for different number of classifer and reported the accuracy and f1score for all implimented methods.

5 Conclusion

Using above method for multi-label classifiction is giving not very good result but not very bad also. and Adaboost finds that it imporve the result. To imporve the result we need to use multi-label Adaboost that my increse the performance of the model.

6 reference

1.https://arxiv.org/pdf/1312.6086v1.pdf
2.https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/
3.https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/
4.chatgpt, google colab, jupyter notebook