EECE-7313

Pattern Recognition

Spring 2013

Final Project

Solving the Multi-Label Problems Based On Adaboost Algorithm

Submitted Date: ___/ ___/2013

Name (Last, First): _____

IN-CLASS (BOSTON CAMPUS) STUDENT

Introduction

After finished the homework 3b which is about the algorithm of clustering, I am kind of interested the problem of classification or clustering. However, in the homework we just simply focus on the data which belong to single label. Since that, I was wondering, what if we face some multi-label problem, what kinds of method should we use to solve the multi-label problems. In reality, multi-label problem is much more common and complicated than the single label ones. For example, the movie called Silver Linings Playbook can not only be classified to the category of Comedy but also the category of Love story. Similarly, when we search Abraham Lincoln in the Google, we both can see the picture of Lincoln and the latest movie which also call Lincoln. Since the applications of the multi-label classification are become more and more important, I would like to find some useful algorithm or method, especially in the area of SVM and AdaBoost algorithm, to solve multi-label classification during my project.

People in the past concluded the methods for multi-label classification into two main categories: a) problem transformation methods, and b) algorithm adaptation methods. (Grigorios Tsoumakas, Ioannis Katakis, Multi-Label Classification: An Overview). One sample of the first method is that people transforms each different set of labels that exist in the multi-label data into a single label, which means that they combine the all different labels which belong to the same data into a new single label. However, the problem of this method is that it may lead to data sets with a large number of classes and few examples per class (Boutell et al., Learning muti-label scene classification).

EX.	SPORT	XSPORT	EX.	POLITICS	XPOLITICS
1	X		**1**	X	
2		X	**2**	X	
3	X		**3**		X
4		X	**4**		X
EX.	**RELIGION**	**XRELIGION**	**EX.**	**SCIENCE**	**XSCIENCE**
1	X		**1**		X
2	X		**2**	X	
3		X	**3**		X
4		X	**4**	X	

(One special example of problem transforms method)

In the algorithm adaptation methods, one of them is using the Adaboost algorithm to solve the multi-label problems. And ML-KNN (Min-Ling Zhang, Zhi-Hua Zhou, A k-Nearest Neighbor Based Algorithm for Multi-label Classification) is one of the special extension of this algorithm.

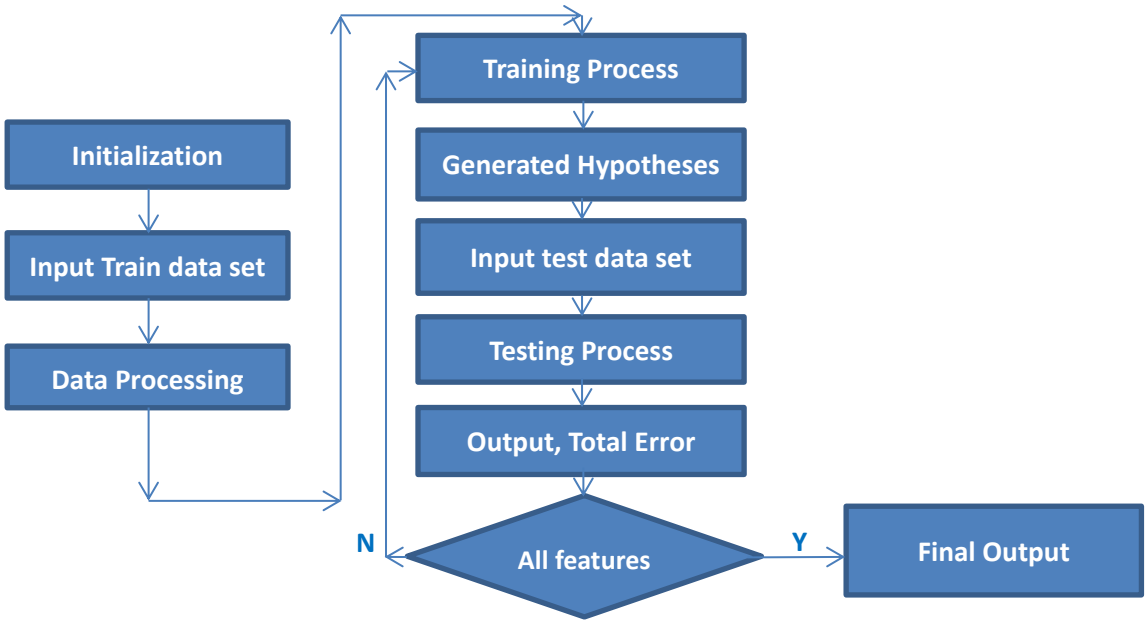
After reading these papers, I would like to implement the Adaboost algorithm and try to use them to solve the multi-label problem. The Adaboost is so call the adaptive boosting algorithm. In Adaboost each training pattern receives a weight that determines its probability of being selected for training set for an individual component classifier. If a training pattern is accurately classified, then its chance of being used again in a subsequent component classifier is reduced; conversely, if the pattern is not accurately classified, then its chance of being used again is raised. In this way, Adaboost "focuses in" on the informative or "difficult" patterns.

There several reasons for me to decide to choose using the Adaboost algorithm.

1. When we receive a bunch of data, we may sometime not sure about the type of the data set. Either the data set can be a Gaussian distribution, or a Uniform distribution. So if we using the Adaboost algorithm which combines lots of weak classifiers into a strong one. It can help us to fit the data set much better.
2. The Adaboost algorithm is not hard to apply. Since it just have to use some weak learners which even better than the random guess. After compute the error and the entropy of each classification, and then update the weight for the next iteration.

Methods

The whole flow chat of the total algorithm can be showed as below:



(The flow chats of the completed algorithm)

In the beginning, I have to do some initializations, which included parameters

define and the data set generation. After that, I should do some transformation of my data set which will show as follow. I input the transformed train data set to the Adaboost algorithm training process. After that I can generate a bunch of hypotheses and the entropies (or weights) of each hypothesis. As soon as I finished the training process, I input the test sample in the criterion we get from the training process. And finally obtain the outputs. Since In the data processing part, I have computed the total number of the label, and I do the classification separately, hence I design the times of running the loop is equal to the total number of label. And After running the total algorithm, I can get a ideal result.

Details in the Adaboost algorithm

When I decided the way of solving the multi-label problem, the first thing I have to do is to build the Adaboost algorithm. The Adaboost procedure is as follow:

1. Begin initialize D={x^1 , y_1, … , x_n, y_n }, k_{max} , $W_1(i)=1/n$, i=1, … ,n
2. k←0
3. Do k←k+1
4. Train weak learner C_k using D sampled according to $W_k(i)$
5. E_k← training error of C_k measured on D using $W_k(i)$
6. $a_k \leftarrow \frac{1}{2}\ln[(1-Ek)/\mathrm{Ek}]$
7. $$W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & \text{if } h_k(\mathbf{x}^i) = y_i \text{ (correctly classified)} \\ e^{\alpha_k} & \text{if } h_k(\mathbf{x}^i) \neq y_i \text{ (incorrectly classified)} \end{cases}$$
8. Until k=k_{max}
9. Return C_k and a_k for k=1 to k_{max} (ensemble of classifiers with weights)
10. End

Since the Adaboost focus on the informative or the “difficult” patterns, we can easily find that each time I generate a hypothesis, it will come out with entropy. I can define this entropy as the weight of the hypothesis. Finally, When we finished the training process, the final classification decision of test point x is based on a discriminant function that is merely the weighted sums of the outputs given by the component classifiers:

$$g(x) = \left[\sum_{k=1}^{k_{\max}} a_k h_k(x) \right]$$

Data processing

Since we get the original data set, the each sample can belong to more than 1 label in the same time, I have to transform the data set in a form that each column belongs to only one label. In each column I also transformed the label

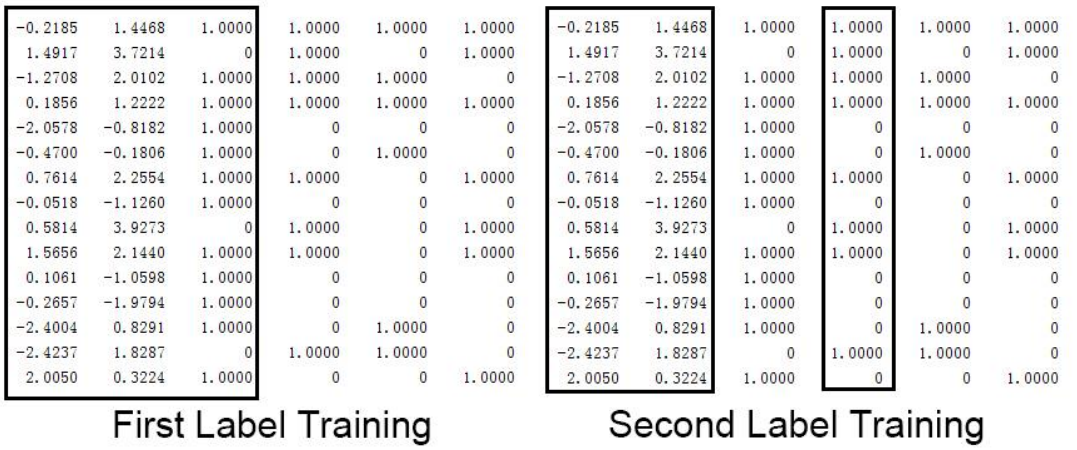
into a binary type which means that the label will be marked as 1 if the data sample belongs to this label. In contrary, the sample will be marked as 0 if the data sample doesn't belong to the label. And the process of data transforming can be showed as below:

Feature1	Feature2	labels
1	3	1 2 3
2	2	2 4
3	1	1 3 4

(The original data set)

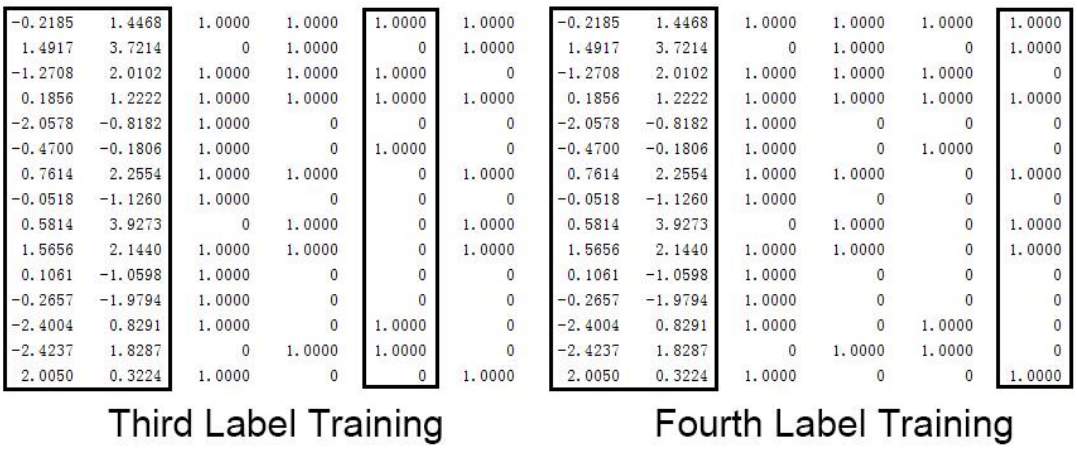
Feature1	Feature2	Label1	Label2	Label3	Label4
1	3	1	1	1	0
2	2	0	1	0	1
3	1	1	0	1	1

(The transformed data set)



First Label Training

Second Label Training



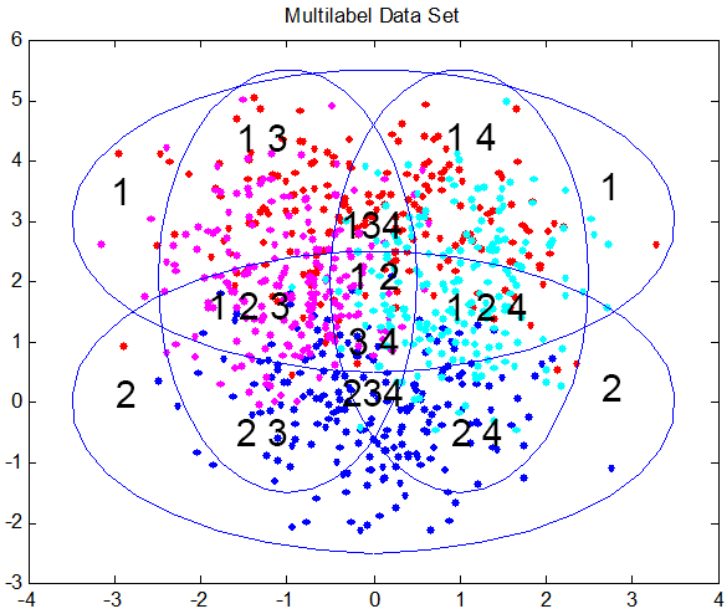
Third Label Training

Fourth Label Training

(Transformed data set in MATLAB)

About the original data set

Since I cannot find any ideal data set from the Internet, hence I generated the data set myself. It contains 4 different Gaussian distributions, and I generated 200 points randomly of each Gaussian distributions. The scatter of the data set can be showed as below:

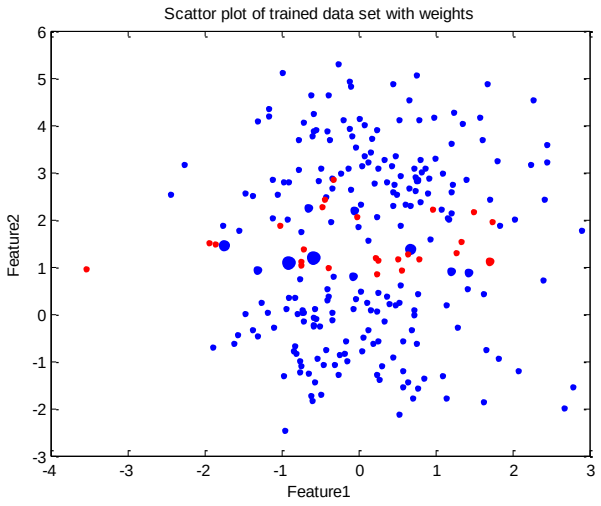


(Multi-Label data set in MATLAB)

The overlap shows that it belongs to more than 1 label in the same time, and I finally used this data set as my data source in this project. Besides, since I totally generated 800 data points, I use 4/5 of them as my training data set, and the left will become the testing data set.

Results

First, let's concentrate the Training Process. As we said that using weak learners in the Adaboost algorithm can change the weights of the data points, I Simulate this result by using MATLAB



The graph shows that after k times iteration, the weights of each data point will change associated with iteration. After updating the weights, in the next iteration classification, the classifiers will just have to focus on the point with heave weights. And finally we get a bunch of Hypotheses.

```
hypothesis =

   -0.4166   -1.0000    1.0000
    0.3687   -1.0000    1.0000
   -0.4673   -1.0000    1.0000
    1.6178   -1.0000    2.0000
    0.0023   -1.0000    1.0000
   -0.3640    1.0000    1.0000
   -0.4673   -1.0000    1.0000
   -1.3130    1.0000    1.0000
   -0.4673   -1.0000    1.0000
   -1.3130    1.0000    1.0000
   -0.4673   -1.0000    1.0000
    0.8539    1.0000    1.0000
   -1.3130    1.0000    1.0000
   -0.4673   -1.0000    1.0000
    1.6178   -1.0000    2.0000
    0.1270   -1.0000    1.0000
    0.8539    1.0000    1.0000
```

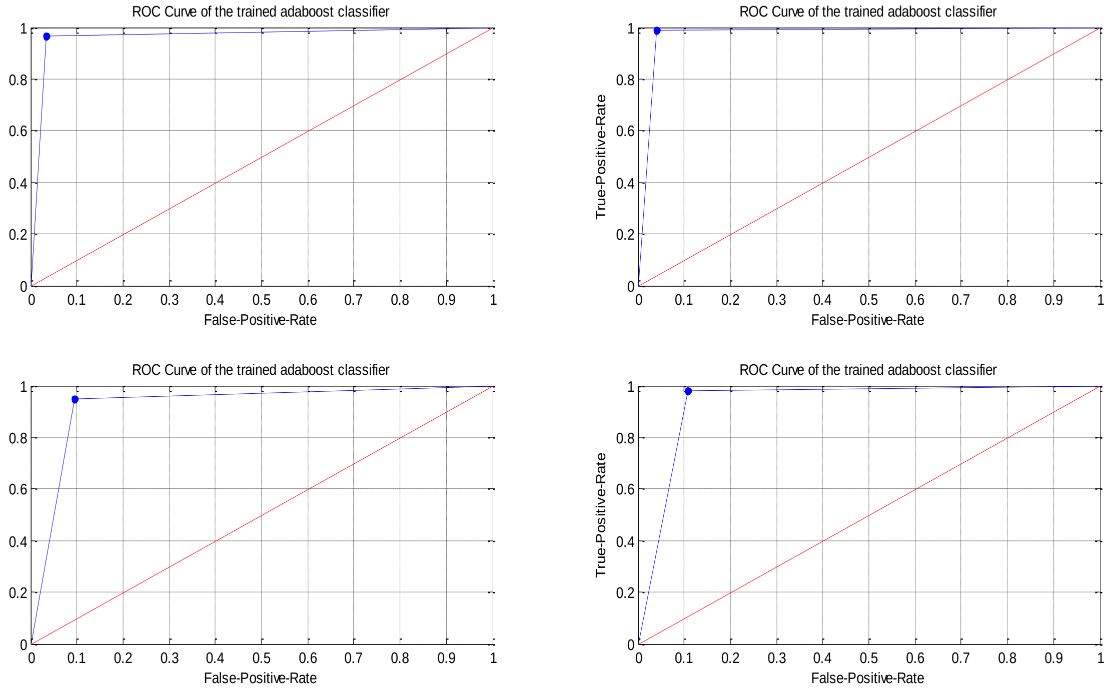
Hypotheses

After the training process, we can get a bunch of hypothesis; here is a scale of the whole hypotheses. As we can see that the hypotheses contain 3 columns. The first columns is the threshold of each classifier, the second one is the bias which is used to decide whether the result of the classification should flip over or not. The third one is so call feature which tells us the threshold belongs to which feature.
And the form of the Hypotheses can be showed as below:

Threshold	Bias	Features

ROC curve

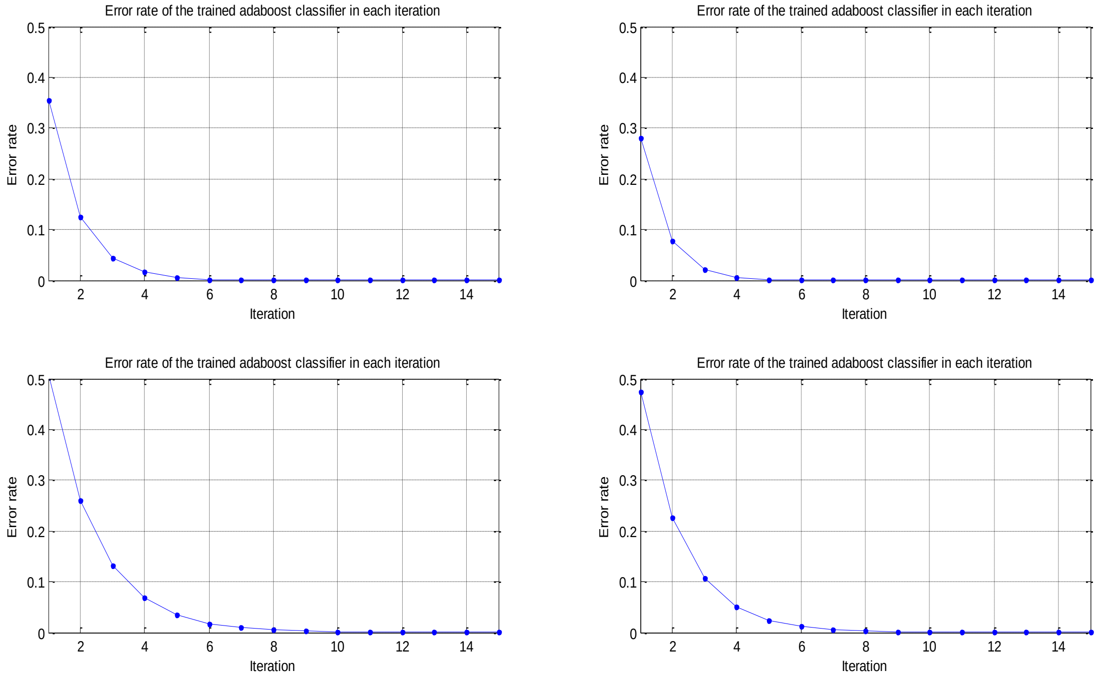
The ROC curve is so call receiver operating characteristic, and is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting the fraction of true positives out of the positives (TPR=true positive rate) vs. the fraction of false positives out of the negatives (FPR=false positive rate), at various threshold settings. And I plot the ROC curve to describe the quality of the final classifier which is combined by a bunch of weak learners. The ROC curve can be plot as below:



As we can see from the graph, if the ROC curve is much close to the red line, that means the quality of the classifier is bad and it just a little better than random guess. However, the reality shows after train the data set by Adaboost algorithm, the ROC curve is far away from the red line, which means that the quality of this classifier is good.

Total Error rate

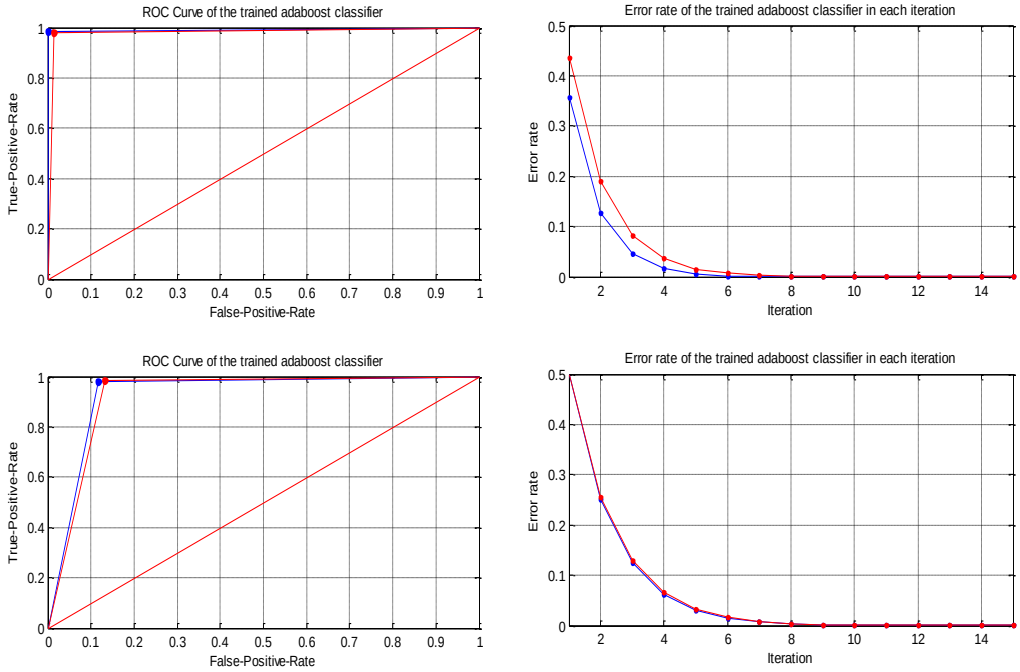
Besides plotting the ROC curve, I also plot the total error rate associated with iteration:



From the curve we can see, the total error rate of the final classifier is decreasing when the iteration is increasing. And that indicate that when we do the whole algorithm again and again, we can get a pretty well result.

Testing process

After the training process, we can generate a set of hypothesis. And this case I finally get 4 sets of hypotheses since my data set has 4 labels. However, how can we judge the different between the training result and the testing result? And can we just simply say that the results of them will be similar? Of cause I cannot. So I plot the comparison ROC curve and Total Error rate between these two data sets.



The red one indicates the result of the testing process and the blue one is the training one. From the plot of the comparison we can sure our assumption is right. The curves between the training data set and the testing data set. However, the result of the testing process still a little bit worse than the training process. That might because the number of the testing data set is less than the training one.
And the Accuracy off the whole classifier associated with the iteration can be given below:

Iteration	**10**	**30**	**50**	**70**	**100**
Accuracy	**0.8425**	**0.8450**	**0.8625**	**0.8650**	**0.8775**

Which the form shows that the Accuracy will be improve as the iteration increased.

Conclusion

After this final project about using the Adaboost algorithm to solve the multi-label problems, it provides an opportunity for me to explore the Boosting Algorithm especially the Adaboost algorithm.

With the study of the Adaboost algorithm, I find that there are several advantages of this algorithm. First, the Adaboost algorithm is not hard to apply, since it uses lots of weak learners and combines them into a strong one. Second, The Adaboost can fit the data set well even we cannot sure what type the data set is. And we still can get an ideal result from the strong classifier eventually after we run the algorithm many times.

However, it still exists some backward in this project. In this project I was just using the data set with 2 dimensional features with 4 labels. When The dimension of the data set and the number of the label grows up, it will lead to the algorithm become slow and complex. Moreover, since the weak learners I used this time which is decision stump, the boundary of this learner is not flexible than some other classifier, while the boundary of the decision stump is kind of stair shape. And I am still trying my best to solve this question and improve the accuracy of this method by using other algorithm. I hope I can get a much more ideal result in the future research.

References

[1] Multi-Label Classification: An Overview, Grigorios Tsoumakas, Ioannis Katakis, Dept. of Informatics, Aristotle University of Thessaloniki, 54124,Greece

[2] A k-Nearest Neighbor Based Algorithm for Multi-label Classification Min-Ling Zhang and Zhi-Hua Zhou, National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

[3] A kernel method for multi-labelled classification. In Thomas G. Dietterich, Susan Becker, and Zoubin Ghahramani, editors, Advances in Neural Information Processing Systems 14, 2002.

[4] A Comparison of Methods for Multiclass Support Vector Machines, Chih-Wei Hsu and Chih-Jen Lin, IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 13, NO. 2, MARCH 2002

[5] Learning multi-label scene classification, Boutell, et al, 2004; Pattern Recognition 37; pp. 1757-1771.