ASSESSMENT COVER SHEET

Assignment Details

Course: Introduction to Statistical Machine Learning

Semester/Academic Year: 2019 semester 2

Assignment title: Assignment 2

Assessment Criteria

Assessment Criteria are included in the Assignment Descriptions that are published on each course's web site.

Plagiarism and Collusion

Plagiarism: using another person's ideas, designs, words or works without appropriate acknowledgement.

Collusion: another person assisting in the production of an assessment submission without the express requirement, or consent or knowledge of the assessor.

Consequences of Plagiarism and Collusion

The penalties associated with plagiarism and collusion are designed to impose sanctions on offenders that reflect the seriousness of the University's commitment to academic integrity. Penalties may include: the requirement to revise and resubmit assessment work, receiving a result of zero for the assessment work, failing the course, expulsion and/or receiving a financial penalty.

DECLARATION

I declare that all material in this assessment is my own work except where there is clear acknowledgement and reference to the work of others. I have read the University Policy Statement on Plagiarism, Collusion and Related Forms of Cheating:

http://www.adelaide.edu.au/policies/?230

I give permission for my assessment work to be reproduced and submitted to academic staff for the purposes of assessment and to be copied, submitted and retained in a form suitable for electronic checking of plagiarism.

Kejie Fu
10/10/19

SIGNATURE AND DATE

AdaBoost

A1719087

Kejie Fu

1 Introduction

Boosting is a family of machine learning algorithms, that could boost weak learning algorithm into an accurate “strong” learning algorithm. Adaptive Boosting (Adaboost) is a well-known method of Boosting family introduced in this report.

2 Algorithm Description

$$D := weight\ (hardness) of\ each\ sample$$

$$\epsilon\ := weighted\ error\ of\ Hypothesis$$

$$\alpha := weight\ of\ Hypothesis$$

$$h := weak\ Hypothesis$$

$$H := final\ Hypothesis$$

Adaboost is a method that trains weak learner T times to get T weak hypothesizes and combines them with weights (based on hypothesizes’ accuracies) into a strong final hypothesis.

As a supervised based method, the inputs of Adaboost are sample features $x_1 \dots x_n$ and label $y_1 \dots y_n$, $y \in \{-1, 1\}$. Then we can get an initial distribution $D_1\ (i) = 1/n$, which is used to control weight of samples in each round of train. Then a weak learner should be chosen, at least slightly better than random.

2.1 Weighted Error ϵ and Choosing Hypothesis Weights α

For every round of training, same training dataset was trained by a weak learner with the distribution to find a weak hypothesis $h_t : X \rightarrow \{-1, +1\}$ that minimize the weighted error,

$$\epsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$$

In this assignment, I passed the distribution to find the best stump. Then a parameter α_t is chosen, this parameter is the weight of weak hypothesis when generating the final hypothesis. And α_t is corresponding the goodness of weak hypothesis, it gets larger as ϵ_t gets smaller, which the formula is $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.

2.2 Updating Sample Weights (Distribution) *D*

After the α_t is chosen, distribution should be updated. The correctly classified samples' weights should be decreased, the misclassified samples' weights should be increased. Which can improve the next hypothesis on predicting the misclassified samples correctly.

$$D_{t+1,i} = D_{t,i} e^{-\alpha_t y_i h(x_i)}$$

Then the new distribution should be renormalized, $D_{t+1,i} = \frac{D_{t+1,i}}{\sum D_{t+1}}$.

After T weak hypothesizes are gotten, final hypothesis can be combined as

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

3. Understanding

3.1 Rule of Thumb

Adaboost is a method combines weak hypothesizes into a final hypothesis, and each weak hypothesis should be better than randomly guessing for avoiding misclassifying. For choosing every hypothesis, it is similar to greedy algorithm, it will always choose the one with minimum weighted error (concentrate on misclassified samples). Adaboost will also give hypothesis a weight related to its weighted error, which makes it have more voice in the vote(combination) phase.

3.2 Overfitting

Adaboost is very sensitive to the noisy samples and outliners, because the exists of distribution makes later weak learner pay more attention on the misclassified samples (it can be noises or outliners). And because Adaboost is a linear combination of weak classifiers, if weak classifiers overfits (especially the sample is in high demission), AdaBoost can also be influenced.

However, if datasets are very sufficient (features are discriminative) and weak learner is not complex (not easy to overfit) such as C4.5, the bias of Adaboost exponentially converges and variance increases by geometrically diminishing while the rounds of boosting increases, which means it is less prone to overfit.

3.3 Weak Learners

Performance of Adaboost is depends on datasets and weak learner, choosing a good weak learner is important, if the weak learner is too complex, overfitting occurs rapidly, if the weak learner is too weak, Adaboost will also overfit. However, simple weak learners are used, such as decision stumps, then the algorithms are much less prone to overfitting.

4. Implementation

4.1 Setup

Firstly, the datasets were preprocessed, 30 features (X) and diagnoses (Y) were collected from csv file. There were 569 samples. First 300 samples were collected as training sets, and rest 269 samples were collected as testing sets. For diagnosis, malignant was decided to be labeled as 1, benign as -1. Every single weak hypothesis was recorded for output the final hypothesis.

4.2 Analysis of Implementation

My code was designed as the given pseudo code. The given decision stump was chosen as the weak learner, so the algorithm would be less prone to overfitting. I have tried to train the weak learner 100,000 times to look the overfitting.

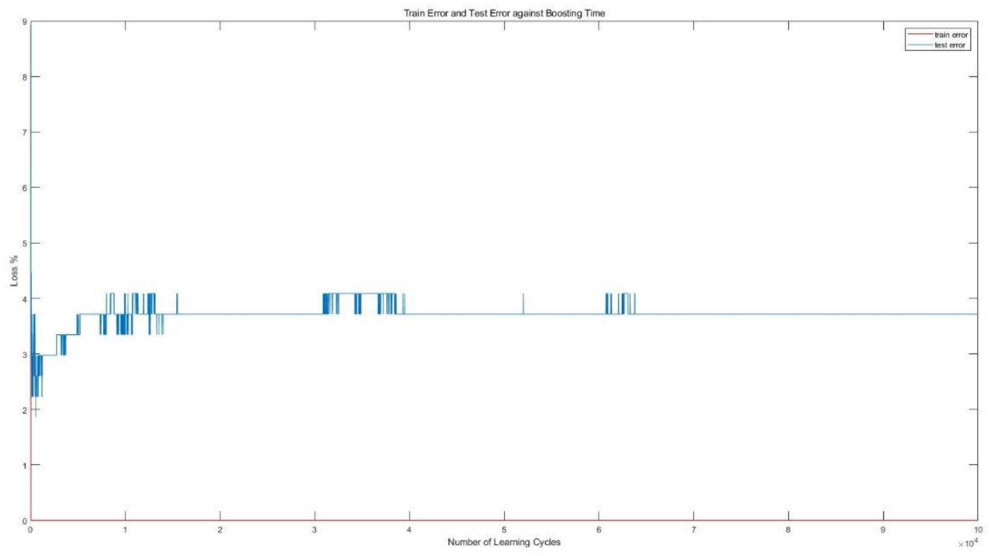

Figure 1 Training/test Error Curve against The Number of Iterations (100,000 times)

After 5000 iterations, the training error become stable around 3.717%. Before 5000 iterations, the loss is surely increasing. And the training loss was already 0%. I think overfitting occurs after about 500 iterations, and it might because there are noises in the datasets and training model was trying to regard them as normal samples.

4.3 Training/Testing Error

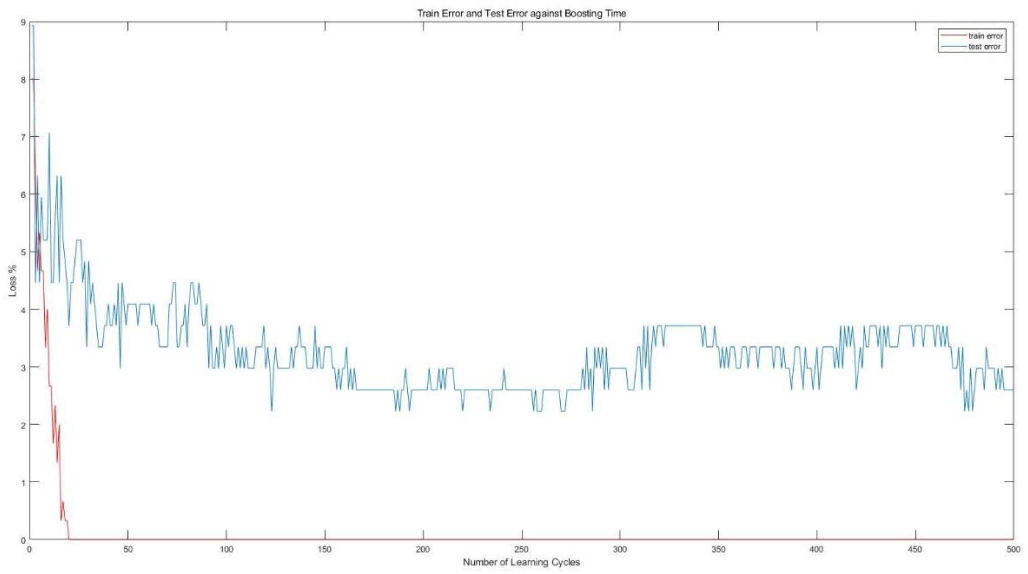

Figure 2 Training/test Error Curve against The Number of Iterations (500 times)

Figure 2 demonstrates the training and testing error during 500 times of training weak learners. At the round 20, train error reached 0% (never increase again) and test error was 3.717% at same time. And at round 500 (the end of test iteration), test error is 2.602%. Both of two curves show the tendency of decreasing even around iteration 500, it proves that Adaboost is hard to overfit when using a not complex model as its weak learner.

4.4 Comparison with Inbuild Package (Fitensemble)

Fitensemble is an inbuild package of MATLAB that can boost some given weak learners for classification and regression, for training the model with AdaBoost with 500 learning cycles of decision tree, the instruction is

```
ClassTreeEns = fitensemble (X_train, Y_train, 'AdaBoostM1', 500, 'Tree');
```

The instruction for recording training loss is

```
rsLoss = resubLoss (ClassTreeEns, 'Mode', 'Cumulative');
```

And predict a label via the instruction

```
predicted_label = predict (ClassTreeEns, X_test);
```

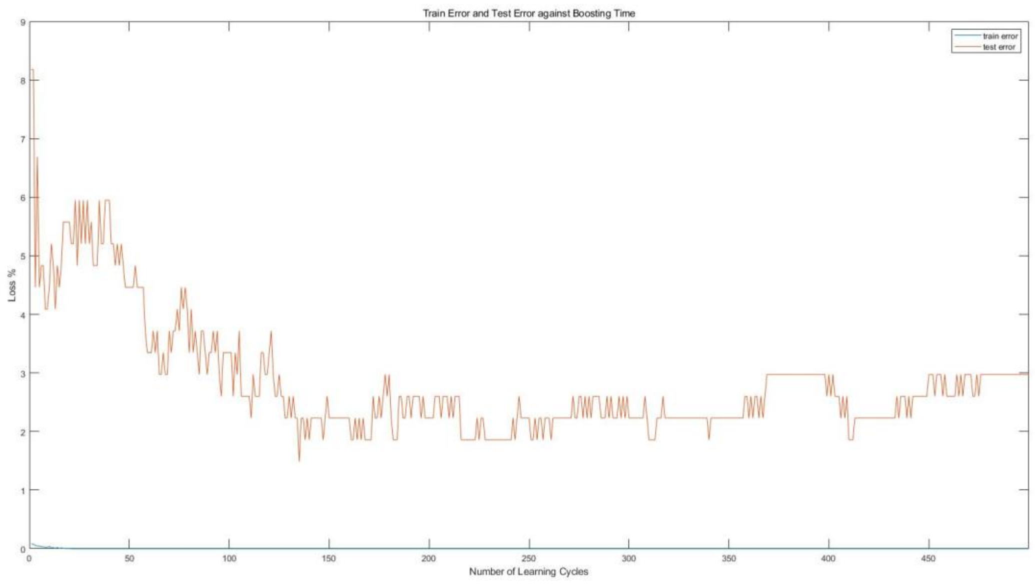

Figure 3 Training/test Error (Fitensemble) Curve against The Number of Iterations (500 times)

Figure 3 demonstrates the training and testing loss Fitensemble's model, compare with my model, the training loos is reached 0 at very early stage, it shows the weak learner decision tree it chose is stronger than decision stump I used in fitting the training set.

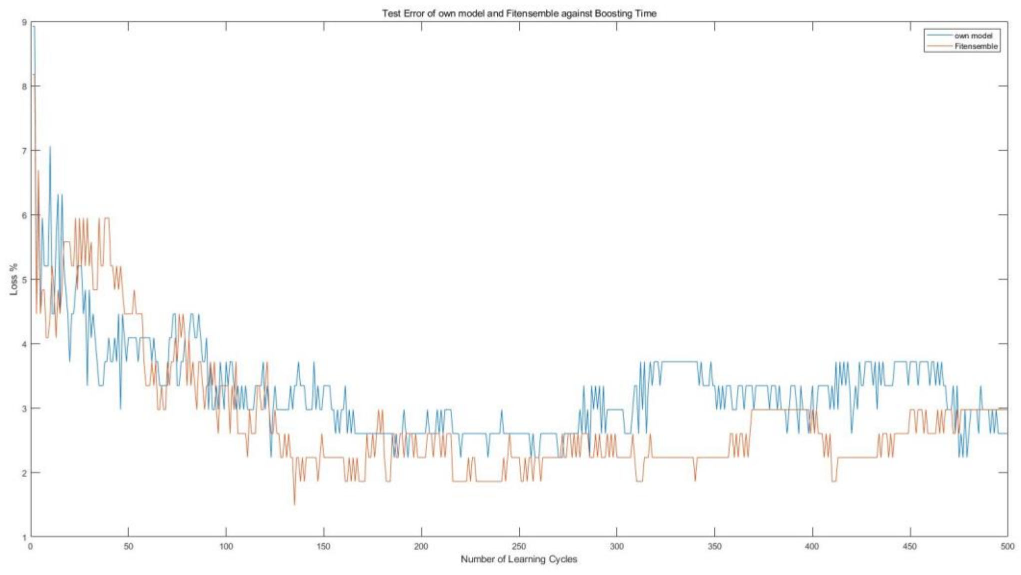

Figure 4 Test Error of own model and Fitensemble against Boosting Time

Figure 4 demonstrates testing loss of both Fitensemble’s model and my own model, the Fitensemble’s test loss is generally less than my model’s test loss. It shows the performance of Adaboost is also related to the strength of weak learner.

5. Conclusion

Adaboost is a very sufficient method that converts weak learners to a stronger learner and it is hard to overfit, but it is very important in choosing weak learner, the weak learner need to be sufficient but should not be complex. And the performance of Adaboost is also related to datasets, because Adaboost is sensitive to noises and outliners.