ASSESSMENT COVER SHEET

Assignment Details

Course: Introduction to Statistical Machine Learning_____

Semester/Academic Year: Semester 2, 2019____________

Assignment title: A2: Implementation of AdaBoost______

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.

Luoyang Xing

Oct 5, 2019

SIGNATURE AND DATE

Adaboost algorithm

The Adaboost algorithm is a boosting method to train a weak classifier with low accuracy repeatedly to get a final classifier with a very high accuracy.

The Adaboost algorithm uses a training set $(x_1, y_1), ..., (x_m, y_m)$ where xi is a data point vector containing the values of all features, and yi is the classification label. We have m data points in the training set. In here, we only let yi be 1 or -1 to represent two different classes. So, first, we need to initialize the distribution set, which contains the weights of all training data points. At the beginning, every data point has the same weight. So D1(i) = 1/m. And we have T rounds.

In each round, we use the weak classifier to get labels on each training data point. Then we compare them with the correct labels of training set. We compute the misclassification error e using sum of weights of mis-classified data points based of Dt. After that, we use e to get the weight of this round's weak classifier in the final classifier. The weight alpha is $\alpha_t = \frac{1}{2}\ln(\frac{1-e_t}{e_t})$. Then, we update the distribution set D_{t+1} to lower the weight of correctly classified data points and make the weight of misclassified data points higher. So it is more focused on the hard-to-classify data points. We update the distribution set by $D_{t+1}(i) = \frac{D_t(i)}{Z_t}\exp(-\alpha_t y_i h_t(x_i))$, where Zt is a normalization factor. Then we add $\alpha_t h_t(x)$ to the final classifier.

Then, after T rounds, we get the final classifier $H_{final}(x) = sign(\sum_t \alpha_t h_t(x))$.

More relevant things

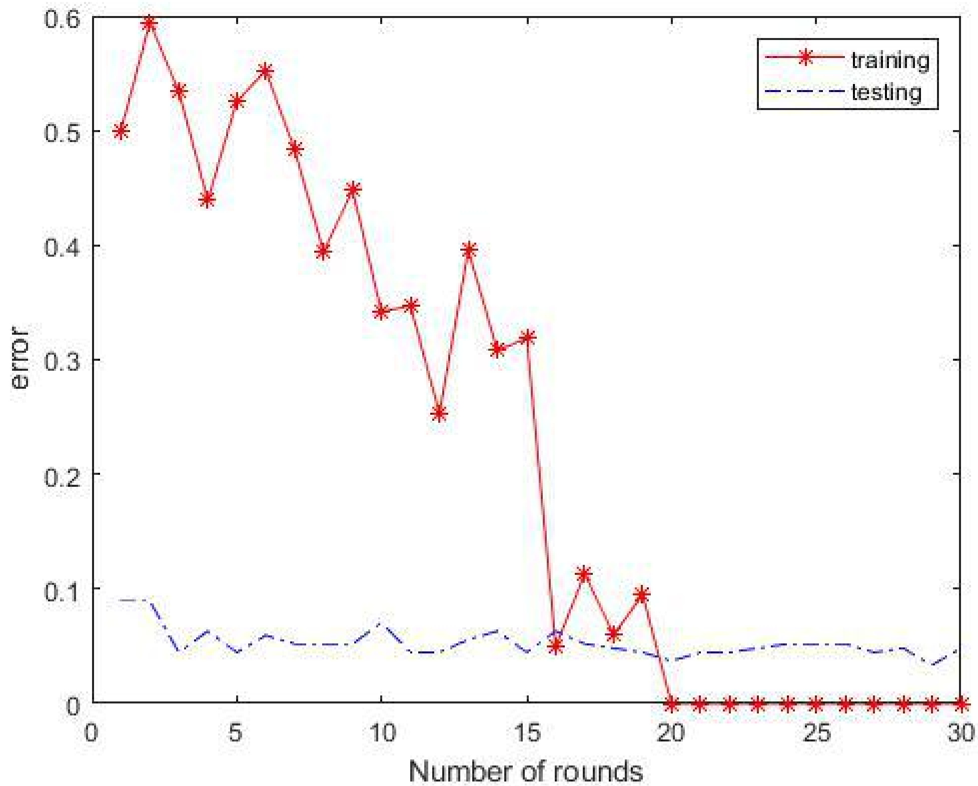
In the process of updating D, the normalization factor Zt is $2\sqrt{e_t(1-e_t)}$. When updating D, we can simplify the equation depending on the correctness of classification of data points. So, when the data point is misclassified, $y_i h_t(x_i) = -1$, $D_{t+1}(i) = \frac{D_t(i)}{2e_t}$. When the data point is correctly classified,

$$D_{t+1}(i) = \frac{D_t(i)}{2(1-e_t)}.$$

We can get training error and test error on each round. The error is computed by summing up the weights of all misclassified data points in training data set and testing data set. We put the error of each round in a vector, after the rounds end, we can plot the error with respect to the number of rounds to see the change.

Analysis of Adaboost Implementation

The Adaboost algorithm trained the weak classifier each round, and output a final strong classifier based on the training set. The following two graphs are showing the error against number of rounds.



So, in figure 1, the y axis is the misclassification error, x axis is the number of rounds. The training error is in red line. We can see as the number of rounds increases, the training error is dropping. At around 20 rounds, the training error reaches 0. And the blue line represents the testing error. It goes down gradually at a low speed. Finally it maintains an error at around 5%.

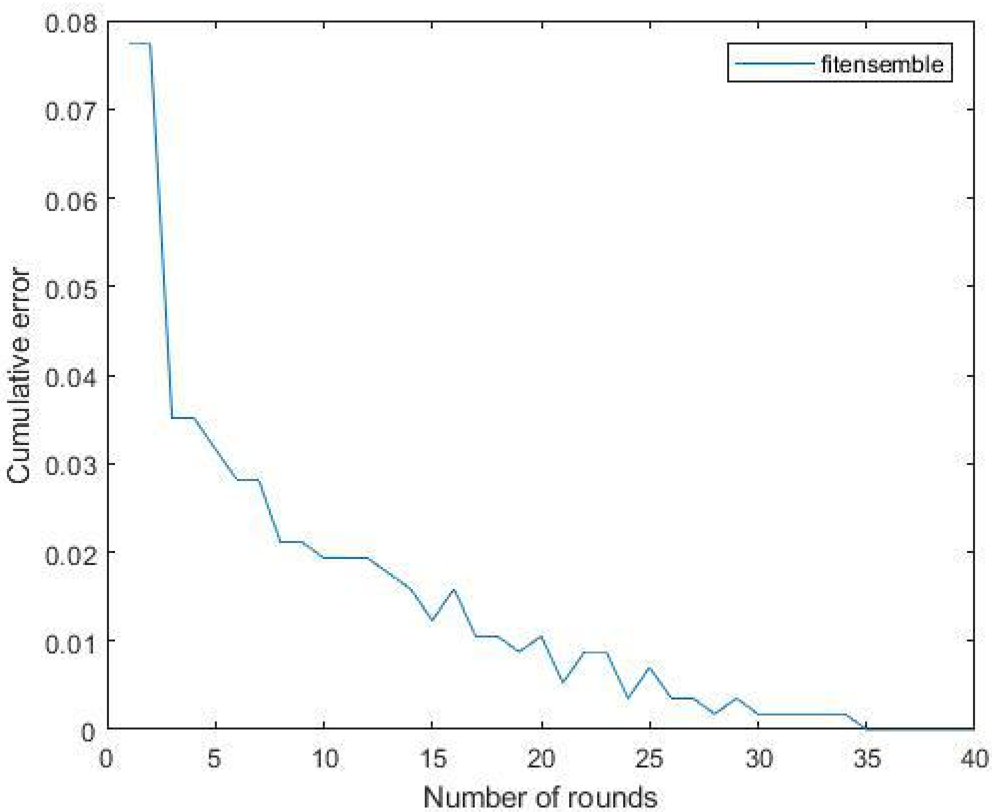


Figure 2 show the change of misclassification error of a in-build package fitensemble. The error goes down quickly. And at around 35 rounds, it reaches 0.

So, we can see that my implementation of Adaboost takes 20 rounds to make the final classifier while the in-build fitensemble takes 35 rounds. My implementation is faster. Maybe this is due to use of different weak learners in each round. We can see that as number of rounds goes up, the training error and testing error both go down. The final classifier has an accuracy at around 95% at prediction.