ASSESSMENT COVER SHEET

Assignment Details

Course: 3920_COMP_SCI_X_0009

Semester/Academic Year: Semester 2 - 2019

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.

MAX VERHAGEN - 11/10/2019

SIGNATURE AND DATE

Algorithmic description

The Mathematically the classification of Adaboosting is defined as:

$$\boldsymbol{H(x) = sign\left(\sum_{t=1}^{T} a_t h_t(x)\right)}$$

Breaking down this formula we fine that h_t is the weak classification of attributes which separating them into two difference groups $X \rightarrow \{-1, +1\}$. The goal of the selected weak hypothesis is to produce a split that give the lowest error between groups. This hypothesis can be calculated in a range of ways however is typically done using decision trees.

Where total error is the sum of weights within a stun that incorrectly classifies a sample meaning it can either be between one and zero, where a total error of zero would indicate a perfect stump. For example, if the adabooster had produced the following stump from a sample size of 10 with all the same weights. For $D_1$the first value of distribution is given as 1/m where m is the number of attributes y_m within the selected data set, meaning within this case would be 1/10.

Active Smoker	Correct	Incorrect
Cancer	4	1
No Cancer	3	2

As we can observe the total error calculates to 3/10 and therefor this stump would have a say of would be 0.42. The amount of say is also logarithmically adjusted meaning that the closer you get to zero or one the more drastic difference there is between each interval. Gini Index is calculated for feature stump and the stump with the lowest index is selected to be added to the forest.

Next $\boldsymbol{a_t}$ is classifies as the "amount of say" at Interval t= 1,…,T. The Amount of say defines how well a stump defines a classifying sample and provide a larger voice to stumps that more accurately separate data compared to other stumps. This is calculated by the formula

$$a_t = \frac{1}{2} ln\left(\frac{1 - Total\ Error}{Total\ Error}\right)$$

The sample weights are then updated after each iteration as the adabooster trains weak learners using the new distribution D_{t+1}. This is calculated by multiplying the current sample weight D_t with the e to the power of the inverted amount of say, normalised by the normalization factor Z_t . This calculation takes the form of for the new sample weight that were correctly classified.

$$D_{t+1} = \frac{D_t e^{(-a_t)}}{Z_t}$$

Understanding of method

Adaboosting also knowing as adaptive boosting is a supervised machine learning algorithm that iteratively creates and classifies weak learners In addition to iteratively corrects mistakes and improves accuracy of weak classifiers by combining weak learners in order to create a strong classifier.

As adaboosters typically only use very weak learners such as 1-level stump decision trees, means that the algorithm is less likely to be affected by overfitting as stumps are very flexible and are more resilient to noise. However, if within the adabooster is the decision tree size was increase then overfitting would more become more likely to occur.

However on the flip side adabooster can become highly effected by outliers as they can be given massive weighting when miss classified causing it to have more effect on the end result then it would on other types of learners.

In addition, there is also two ways the sample weights can be re-calculated after a stump is selected the first being weighted gini indexes which is a variable that is factored into making the following stumps. However the method that will be used within this implementation is creating a new collection of samples after each iteration by normalising the new sample weights and then randomly selecting values which correspond the normalise weighted values. meaning that the classes with larger weight have more likely to be included multiple times in the new collection which ultimately makes there collective weight larger.

Analysis of implementation

For this project C++11 was choices as it is what I am most familiar and comfortable with. The compile command is as following: g++ -std=c++11 main.cpp -o program And then once compiled the cvs file is taking as an argument as followed: ./program filename.cvs

Within segment 1 of the main, the program converts the cvs features into a 2d double array, the diagnosis is also split into a second array. although not implemented in the final submission the ‘printinput’ function prints the separated cvs file to check for correct formatting.

unfortunately, the code is incomplete and could not be fixed before the submission date. However the overall structer of the adabooster has been implemented the stump building class isn’t full functioning and therefore prevents the system from working.

Performance

Accuracy of code: 0%

Sklearn Accuracy with 1 estimators : 80%

Sklearn Accuracy with 10 estimators : 93%

Sklearn Accuracy with 100 estimators : 88%

Sklearn Accuracy with 1000 estimators : 88%

Sklearn Accuracy with 10 estimators and Decision Tree Classifier max depth 1 : 93%

Sklearn Accuracy with 10 estimators and Decision Tree Classifier max depth 2 : 86%

Sklearn Accuracy with 10 estimators and Decision Tree Classifier max depth 10 : 75%

Sklearn Accuracy with 10 estimators and Decision Tree Classifier max depth 100 : 84%

Sklearn Accuracy with 10 estimators and Decision Tree Classifier max depth 1000 : 75%