Introduction to Statistical Machine Learning
Assignment 2

by Deepesh Kumar Malik
a1804938
a1804938@student.adelaide.edu.au

Report submitted for **4020_COMP_SCI_X_0009 Introduction to Statistical Machine Learning** at, University of Adelaide towards the Master of Data Science

Introduction

In recent years, a massive gain in popularity of boosting algorithms is seen in data science or machine learning. Boosting algorithms basically combines number of weak learners/ low accuracy models to form a high accuracy model. It is utilized in various places like insurance, sales and marketing, and credit. AdaBoost, Gradient Boosting, etc. are few widely used algorithms.

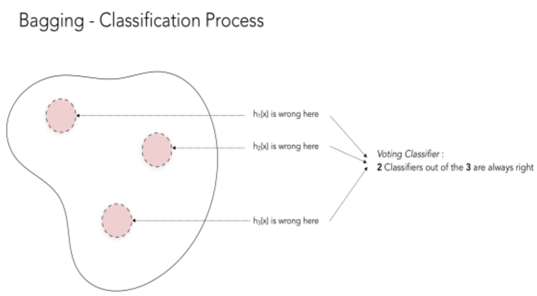
Adaboost was introduced by Freund and Schapire in 1995. Adaboost or Adaptive boosting is a type of ensembling technique. An ensemble model is a composition of number of low performing classifiers to build an improved classifier.

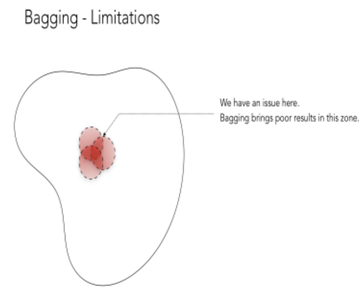
A weak classifier is one which performs poorly, and usually is better than random guess. For example, classifying whether a person is male or female just by their height. It can be assumed that anyone over height 5'9" is a male and rest are females. A lot of misclassification will happen by this way, but our accuracy will still be above 50%.

For a binary classification problem, we classify an observation as 0 or 1. Bagging stands for "Bootstrap Aggregating". In bagging we select T bootstrap samples, and then classifier is fitted on each samples and the model is trained on it parallel. Typically, in Random Forest, parallel training of decision trees is done. The output of all the classifiers are then averaged in bagging classifier:

$$H_T(x) = 1/T \sum_t h_t(x)$$

As seen in the below image, consider 3 classifiers which gives classification result. The result can be right or wrong. If these 3 classifiers are plotted, there will be regions in which these classifiers may be wrong which are represented in red. This example will work nicely, if one classifier is wrong, then others two will be correct. Bagging doesn't work properly, if all classifiers are wrong i.e. mistaken in same region.

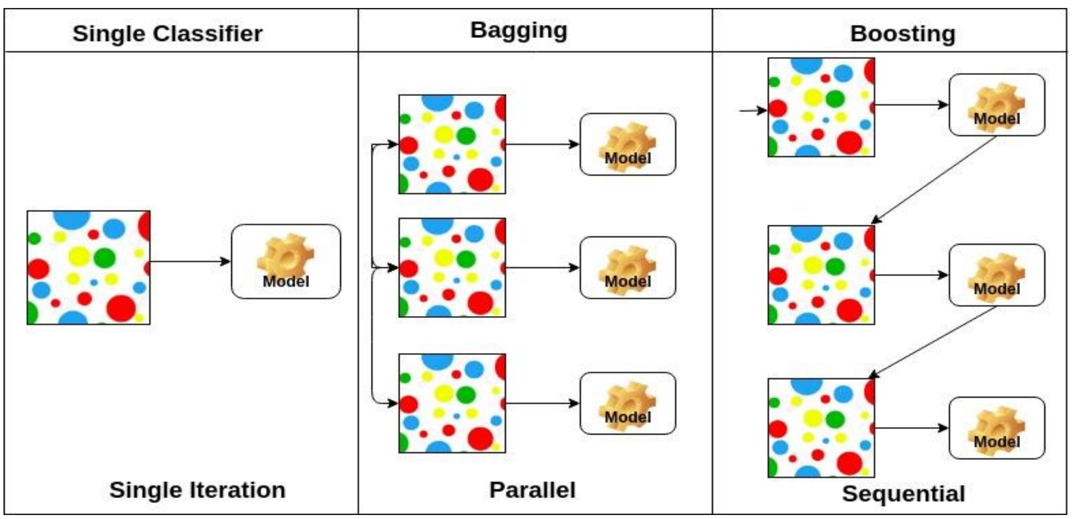




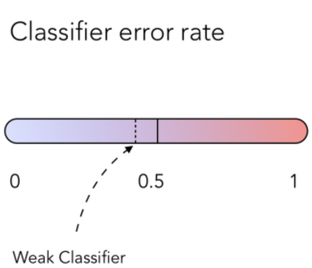
Because of this reason Boosting was the introduced. Adaboost follows 2 main rules:

a) Models are trained sequentially instead of training them parallel.

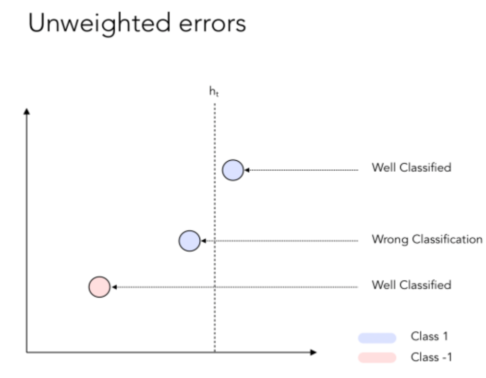
b) Focus on previous classifier poor performance needs to be taken into consideration by each model by minimizing training error.



Boosting trains number of low performing algo, called weak learners. Weak learners have error rate slightly under 50%

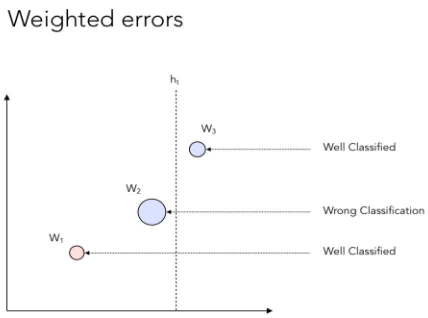


By weighting errors through all the iterations, it will give higher weight to samples which were poorly classified previously. Initially, weight assigned to each data sample while calculating the error rate is 1/n where n is total number of samples to classify.



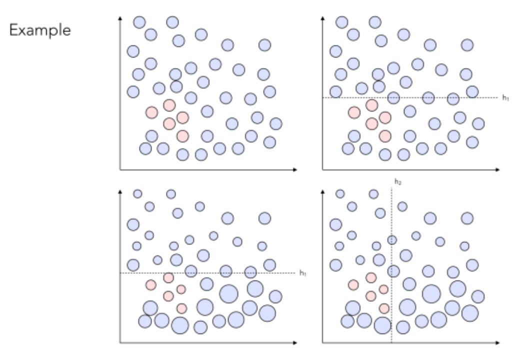
$$E = \sum_{y_i \neq h(x_i)} 1/n$$

After applying weight to the errors

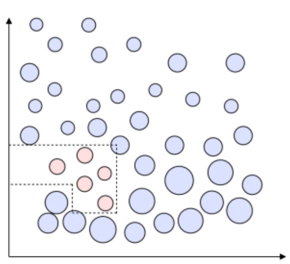


$$E = \sum_{y_i \neq h(x_i)} w_i$$

Another pictorial explanation of classifier where blue regions are wrongly classified and then in the last graph even more blues are wrongly classified making their weight even higher than the previously wrong classifications.



In the end, a a strong classifier may look like this



Tree Stumps ->

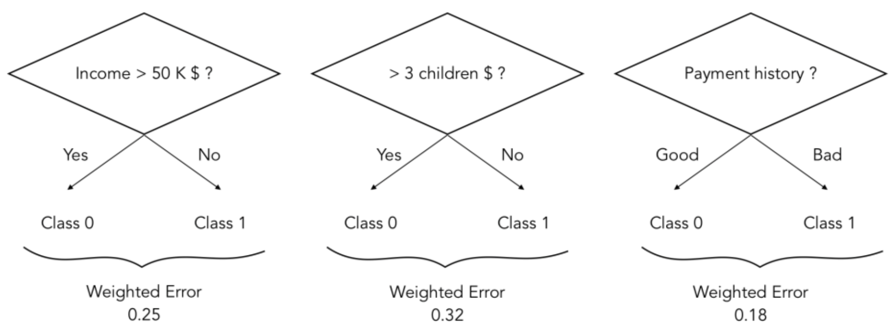
Tree stumps are basically 1-level decision tree. At each step, our motive is to find the best stump, i.e. the line which will split the data in the best possible way to minimize the overall error. A stump can be considered as a basic test, where the assumption is anything lying on one side belongs to class 0, and anything lying on other side is belonging to class 0.

At each iteration t, we choose the weak classifier that splits the data best way, by minimizing the overall error rate.

In Adaboost, Decision Stump is the most common weak learner which consists a decision tree of depth 1. Basically it is a model that returns an output on a single condition. It can be said as "If (condition) then A else B"

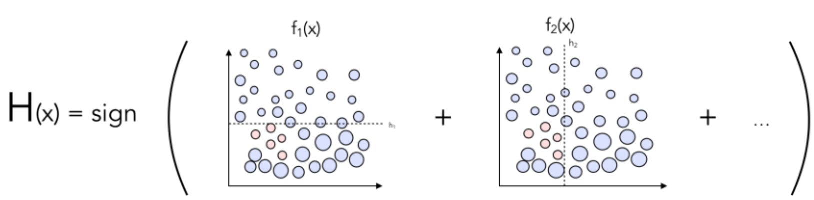
Finding the best split ->

At each iteration t, the best weak classifier ht is identified which gives the best fit, which is a decision tree with 1 node and 2 leaves (a decision stump). The weighted error resulting of this split should be minimal.

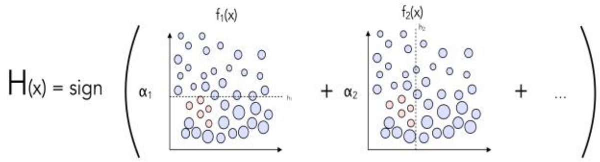


Combining classifiers ->

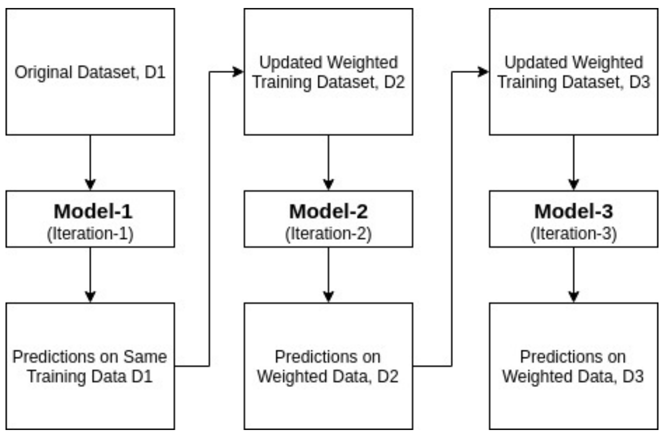
Next step combines all the classifiers in a Sign classifier, and it will be classified as 0 or 1.



Weights are added on each classifier, so that same importance is not given to all the different classifiers.



Adaboost flow will look like as below:



Mathematical explanation of the logic behind adaboost ->

---

1. **Input:** $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of Iterations T
2. **Initialize:** $d_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$
3. **Do for** $t = 1, \ldots, T$,
   (a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto \{-1, +1\}$, i.e. $h_t = \mathcal{L}(S, \mathbf{d}^{(t)})$
   (b) Calculate the weighted training error ε_t of h_t:
   $$\varepsilon_t = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h_t(\boldsymbol{x}_n)) \ ,$$
   (c) Set:
   $$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}$$
   (d) Update weights:
   $$d_n^{(t+1)} = d_n^{(t)} \exp\left\{-\alpha_t y_n h_t(\boldsymbol{x}_n)\right\} / Z_t \ ,$$
   where Z_t is a normalization constant, such that $\sum_{n=1}^{N} d_n^{(t+1)} = 1$.
4. **Break if** $\varepsilon_t = 0$ or $\varepsilon_t \geq \frac{1}{2}$ and set $T = t - 1$.
5. **Output:** $f_T(\boldsymbol{x}) = \sum_{t=1}^{T} \frac{\alpha_t}{\sum_{r=1}^{T} \alpha_r} h_t(\boldsymbol{x})$

---

1) Given (x_1,y_1),…..,(x_m,y_m) where x_i ∈ X, y_i ∈ {-1, +1}

“A training set consisting of m samples where all x inputs are an element of the total set X and where y outputs are an element of a set comprising of only two values, -1 and 1

2) Initialize all weights of samples to 1/(number of training sample).

3) Train weak learner using distribution Dt.

ε = minimum misclassification error for the model

α = weight for the classifier

Zt = normalization factor, for a true distribution

For t=1 to T classifiers, fit the model on training data and select classifier with the lowest weighted classification error.

The formula to compute ε is as follows

$$\widehat{\mathsf{MME}}_{\mathsf{emp}}^{(j)} = \frac{\sum_{i=1}^{N} w_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^{N} w_i}$$

yi not equal to hj will be = 1 if misclassified and 0 if correctly classified.

wi = weight

*Updated weights= old weights*alpha*y*predicted_y*

Implementation:

The dataset used in this experiment is Wisconsin Diagnostic Breast Cancer dataset consisting of 569 samples. The data is split into 300 training samples, and 269 testing samples. There are a total of 31 features which consists of 30 input features and a corresponding label (y), which have benign or malignant status. For our implementation, the values of output label y is updated to numeric values with M to -1, and B to +1.

Important hyper parameters of adaboost using decision trees are:

base_estimator: used to train the weak models. Decision tree is default argument.

n_estimators: is the number of models to iteratively train.

learning_rate: is the contribution of each model to the weights and defaults to 1.

Since AdaBoost is a sequential process, we get output from series of decision stumps, we’ve to check if performance can be improved by increasing the number of learners.

Training Accuracies for custom Adaboost →

Iteration/ Learning rate	5	25	50	75	100

0.1	0.936	0.973	0.98	0.986	0.993
0.5	0.956	1	1	1	1
1	0.93	0.99	0.993	1	1
1.5	0.916	0.94	0.123	0.94	0.123
2	0.926	0.94	0.11	0.94	0.11

Testing Accuracies for custom Adaboost →

Iteration/ Learning rate	5	25	50	75	100
0.1	0.932	0.947	0.962	0.955	0.958
0.5	0.955	0.94	0.955	0.962	0.966
1	0.917	0.902	0.936	0.914	0.94
1.5	0.917	0.91	0.085	0.91	0.085
2	0.917	0.91	0.097	0.91	0.097

Sklearn Accuracies->

Iteration/ Learning rate	5	25	50	75	100
0.1	0.921	0.936	0.962	0.958	0.962
0.5	0.947	0.932	0.958	0.947	0.962
1	0.932	0.97	0.970	0.97	0.966
1.5	0.925	0.962	0.951	0.966	0.977
2	0.917	0.91	0.496	0.839	0.485

Sklearn Accuracies for different depths ->

Depth	2	4	6	8	10
Iteration:100, LearningRate:1.5	0.970	0.976	0.880	0.906	0.906

Any ML learner can be used as base estimator provided it accepts sample weight such as SVC - Support Vector Classifier, Decision Tree etc.

Adaboost classifier object with different base learners →

Iteration:100	DecisionTree	RandomForest	SVC
LearningRate:1.5	95.895	95.149	92.537

Mean accuracies of different cross validation algorithms →

<table>
<tr><th colspan="2">Iteration:100, LearningRate:1.5</th><th>Training Accuracy</th><th>Testing Accuracy</th></tr>
<tr><td colspan="2">DecisionTree</td><td>92.000</td><td>93.827</td></tr>
<tr><td colspan="2">RandomForest</td><td>94.666</td><td>95.802</td></tr>
<tr><td rowspan="3">SVC</td><td>linear</td><td>94.888</td><td>95.061</td></tr>
<tr><td>rbf</td><td>87.333</td><td>94.074</td></tr>
<tr><td>sigmoid</td><td>44.000</td><td>77.0370</td></tr>
</table>

Run:

Place the dataset in the folder consisting of python code file.

Refrences:

http://rob.schapire.net/papers/explaining-adaboost.pdf

https://chrisalbon.com/machine_learning/trees_and_forests/adaboost_classifier/

https://www.datacamp.com/community/tutorials/adaboost-classifier-python

http://www.boosting.org/papers/MeiRae03.pdf

https://en.wikipedia.org/wiki/AdaBoost

https://towardsdatascience.com/boosting-and-adaboost-clearly-explained-856e21152d3e

http://www.cs.princeton.edu/~schapire/papers/FreundSc99.ps.gz

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

https://machinelearningmastery.com/adaboost-ensemble-in-python/