# Predicting Cover Types with Adaboost

Machine Learning

MSc in Data Science and Economics

Nazli Begum Cirpanli
942345

November, 2021

# Contents

# 1 Introduction

Ensemble methods are machine learning methods that combine many classifiers to have a one strong classifier in order to achieve more accurate predictions. In the case of binary classification with zero-one loss, assuming the training errors of the classifiers are independent of each other,majority vote classifier f is defined as follows:

$$f(\mathbf{x}) = sgn(\sum_{x=1}^{T} h_i(\mathbf{x}))$$

Where ($x_1$,$y_1$),...,($x_m$,$y_m$) is the training set, and $h_1$,...,$h_t$ is the ensemble of classifiers. It is easily seen that if there are more $h_i$ predicting label 1 than $h_i$ with label -1 then majority vote classifier f chooses label 1 as the final prediction.

Training error of ensemble methods is bounded as seen below:

$$l_s(f) = exp(-2T(\frac{1}{2} - l_{ave})^2) = exp(-2T(\frac{1}{T}\sum_{i=1}^{T}\gamma_i)^2) \leq e^{-2T\gamma^2}$$

where T is the number of learners, $\gamma_t$ is $\epsilon_t - \frac{1}{2}$.

Training error of the majority vote classifier decreases exponentially with respect to $\gamma$. $\gamma$ value represents how better than a random guessing each classifier $h_i$ is on the training set. [1]

Bagging, random forest and boosting are all examples of ensemble learning. Bagging and random forest require that each classifier in the ensemble has independent training errors. Majority vote classifier decides on the label according to the majority of the labels predicted by each classifier. Boosting, on the other hand, does not require individual independency of the classifiers on the training errors to achieve the same exponential bound. It uses weighted majority of the classifiers $h_i$, and make prediction as a weighted majority vote.

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

[1] From the lecture notes, Boosting and Ensemble Methods,Nicolo Cesa-Bianchi, p.2 .

where $\alpha_t$ is the weight of each classifier $h_i$ generated by some learning algorithm A which is slightly more accurate than random guessing.

Each $h_i$ belongs to H and is called base classifier. Base classifiers are simple classifiers and usually are decision stumps. Decision stumps are decision trees having one level and 2 leaves. In other words, a decision stump is a decision tree with one splitting criterion and one attribute.
Here is how boosting algorithm works:

1. For t = 1... T
    - Set $d_t$ over 1,...,n
    - Run A on $d_t$ and output $h_i$ : X $\rightarrow$ {1,-1}
    - Calculate error
    - End for
2. Output H

In this analysis, Adaboost will be used as it is a boosting algorithm to solve a multiclass classification problem. However, brief information about Adaboost is given considering the binary case for the sake of brevity.

Adaboost (short for Adaptive Boosting) is a learning algorithm which initially assigns weights to every instance in the training set and adjusts those weights in each consecutive iteration. Algorithm adjusts the weights by looking at the errors of the weak classifiers ($\epsilon_t$) output by the algorithm at each round. Examples misclassified by the weak classifier in iteration t get more weight so that the algorithm would work harder on them the next iteration t+1. In order to produce the final prediction, algorithm also assigns weights ($\alpha$) to the weak classifiers based on their calculated errors and gives more weights to the ones with lower errors.

Initial weight assignment to m examples in the training set is

$d_{1,i} = \frac{1}{m}$ for all i

Adjusting weights of the instances based on the correctness of the prediction. Misclassified examples get more weight.

$$d_{t+1,i} = \frac{d_t,i}{Z_t} \times \begin{cases} e^{-\alpha_t} : \text{ if} y_i = h_t(x_i) \\ e^{\alpha_t} \ : \text{ if} y_i \neq h_t(x_i) \end{cases}$$

Assigning the normalized weights to the examples for the next itearation

$d_{t+1,i} = \frac{d_{t,i}}{Z_t} e^{-y_i \alpha_t h_t(x_i)}$

where $Z_t$ is a normalization factor, making the probability distribution sums up to 1.

Final output

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

where $\alpha_t = \frac{1}{2} \ln(\frac{\epsilon_t}{1-\epsilon_t})$ and $\epsilon_t = P_i \mathbf{x}[h_t(x_i) \neq y_i] = \sum_i d_{t,i} \mathbf{1}_{[h_{(t)}(x_i) \neq y_i]}$

Pseudocode of Adaboost algorithm is as follows:[2]

**AdaBoost**

**input:**
training set $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
weak learner WL
number of rounds $T$

**initialize** $\mathbf{D}^{(1)} = (\frac{1}{m}, \ldots, \frac{1}{m})$.

**for** $t = 1, \ldots, T$:
invoke weak learner $h_t = \mathrm{WL}(\mathbf{D}^{(t)}, S)$
compute $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[y_t \neq h_t(\mathbf{x}_t)]}$
let $w_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$
update $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_t h_t(\mathbf{x}_t))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$ for all $i = 1, \ldots, m$

**output** the hypothesis $h_s(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} w_t h_t(\mathbf{x})\right)$.

[2]From the chapter "Boosting" , Shai Shalev-Shwartz and Shai Ben-David,2014, Cambridge University Press, Understanding Machine Learning: From Theory to Algorithms, p. 135 .

# 2 Dataset

Forest cover type dataset, taken fromhttps://www.kaggle.com/, contains 54 attributes all of which are cartographic measures representing four wilderness areas located in the Roosevelt National Forest of northern Colorado. There are 10 quantitative and 44 binary variables. Target variable is cover type which is also a quantitative variable ranging from 1 to 7. Forest cover is the amount of land area that is covered by forest. Forest cover types are encoded as below:

1. SpruceFir
2. Lodgepole Pine
3. Ponderosa Pine
4. CottonwoodWillow
5. Aspen
6. Douglas-fir
7. Krummholz

There are 581,012 examples and the dataset does not have any missing values.

**Brief information about attributes:**

- Elevation - meters
- Aspect - azimuth
- Slope - degrees
- Horizontal_Distance_to_Hydrology - Horizontal distance to closest surface water features, meters
- Vertical_Distance_To_Hydrology - Vertical distance to closest surface water features, meters
- Horizontal_Distance_To_Roadways - Horizontal distance to n closest roadway, meters
- Hillshade_9am (0 to 255 index) - Hillshade index at 9am, summer solstice (0 to 255)

- Hillshade_Noon (0 to 255 index) - Hillshade index at noon, summer solstice (0 to 255)
- Hillshade_3pm (0 to 255 index) - Hillshade index at 3pm, summer solstice (0 to 255)
- Horizontal_Distance_To_Fire_Points - Horizontal distance to closest wildfire ignition points, meters
- Wilderness_Area - Wilderness area (4 binary columns, 0 = absence , 1 = presence)
- Soil_Type - Soil type (40 binary columns, 0 = absence ,1 = presence)
- Cover_Type - Forest Cover Type (integers 1 to 7)

Given these variables, the purpose is to find out what types of trees grow in an area.

The distribution of cover types is shown below. It is clear that the cover types are not in balance.

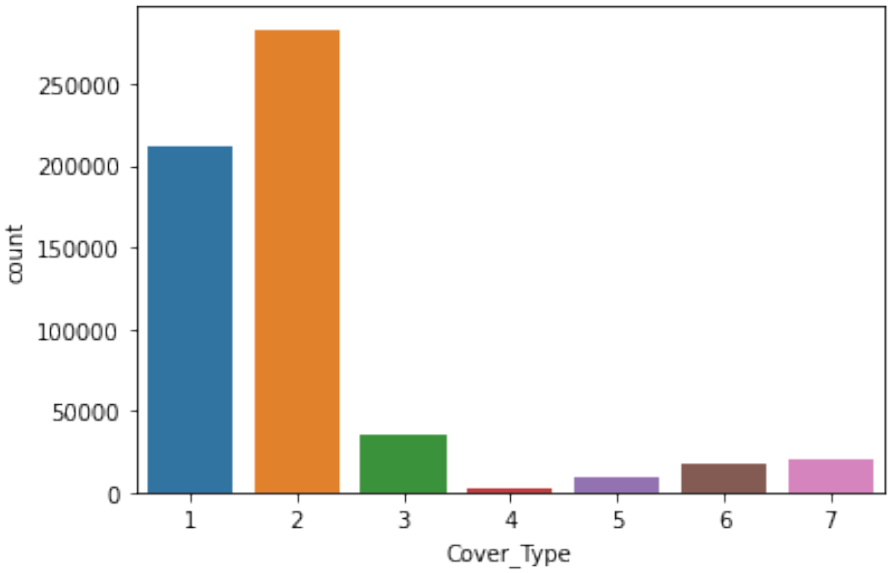



Figure 1: Cover Types

# 3 Predicting Cover Types with Adaboost

Since there are more than 2 cover types to be predicted, it is a multiclass classification problem. In this analysis, multiclass classification problem has been reduced to binary classification problem. Adaboost algorithm has been implemented using decision stumps as base classifiers. There are 7 different classes, hence for each class a binary classifier has been trained on the training set. Final prediction for each examle has been made by using argmax function..

First, the target column has been separated from the original data then both datasets have been divided into training and test sets by 30%. In order to keep the same distribution of each class in both sets, examples have been stratifed.

As mentioned earlier, for each class in range (1,7) decision stumps have been constructed as base classifiers. For each class, label of the each instance in the training set has been replaced with 1 if the label equals to that particular class and -1 otherwise. With this encoding, weak classifier belonging to that particular class has been trained on the training set for T rounds. For example, if the class currently being examined is 1, training set has been changed like this:

$$(x,1) \rightarrow \begin{cases} (x,1):1 \\ (x,2):-1 \\ (x,3):-1 \\ (x,4):-1 \\ (x,5):-1 \\ (x,6):-1 \\ (x,7):-1 \end{cases}$$

At end the end of each round t, the weak classifier produced by the algorithm, predicts the label which receives the most weighted votes. That prediction has been compared with the true value of that particular instance. According to the correctness of the prediction, error of the classifier has been calculated and based on that error, weights of the examples have been adjusted for the next round. This process has been continued for T rounds for each class.Each 7 binary classifier has the form $h(x) = sgn(g(x))$. In the end, 7 binary classifiers output by the Adaboost algorithm have been used to make the final prediction. For each example in the test set, labels have been predicted by using $H(x) = argmax_i h_i(x)$, where $h_i$ is the classifier corresponding to class i.

Model has been trained and tested first by employing validation approach and then, 5-fold stratified cross validation. With stratification, purpose is to keep the original proportions of the target values in each train and test set.

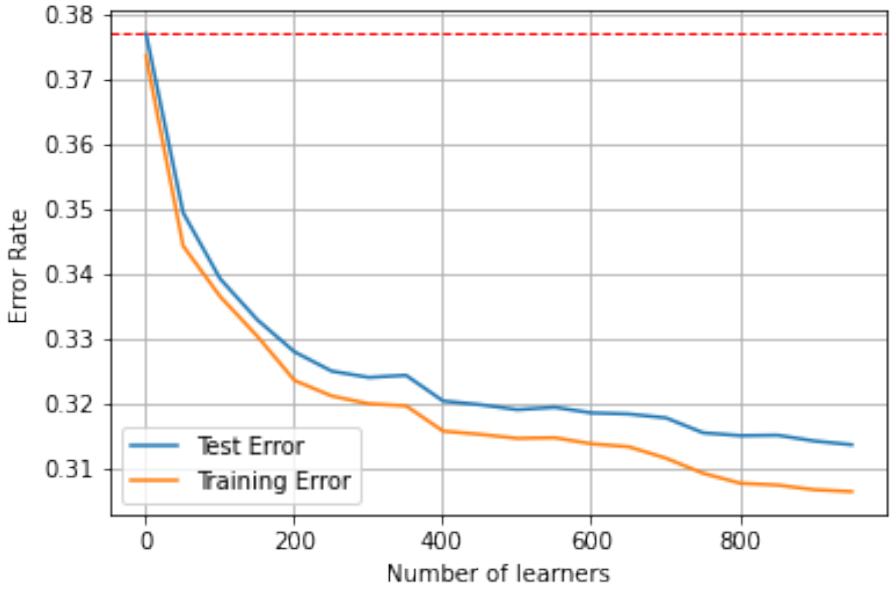
## 3.1 Different Values of the Number T of AdaBoost Rounds

T (number of learners) is the only parameter that can be tuned in Adaboost algorithm. Therefore, in order to examine the changes in the error rate with respect to the number of learners, algorithm has been run multiple times with different numbers of T.

Forest Cover Type dataset has 581,012 rows, thus executing the algorithm with higher number of learners takes a lot of time. To avoid long runtimes, a sample has been taken from the original data and used for examining the accuracy rates of the classifiers with respect to changing numbers of T. 100,000 rows have been taken randomly from the data with replacement and split into train and test sets by 30% again.

Accuracy rates and respective error rates with respect to number of learners T have been plotted below. Assuming the error of each base classifiers is at most $\frac{1}{2} - \gamma$, training error of the classifer produced by Adaboost algorithm is at most $e^{-2T\gamma^2}$. This means that as T increases, emprical risk of the classifier goes to zero.


(a) **Error Rate vs Number of Learners**




(b) **Accuracy Rate vs Number of Learners**

# 4 Conclusion

In this analysis a specific boosting algorithm, namely Adaboost, has been studied and used to make multiclass classification on forest cover type dataset. Algorithm has been implemented from scratch by employing one vs all encoding, thus seven different classifiers have been produced for each class. Predictions have been made based on these seven binary classifiers. With the validation approach 66.80% accuracy rate has been reached with 105 learners. Then, performing stratified 5 fold cross validation has given 63.94% accuracy with the same number of learners.

In order to see how the error rate changes with the number of learners, algorithm has been run on a sample taken from the original dataset for different number of learners. It has been shown that with increasing number of learners, empirical risk of the classifier output by Adaboost algorithm approaches to zero.

*"I declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study."*