AdaBoost Algorithm: Understand, Implement and Master AdaBoost

Anshul Saini

16 Jan, 2024 • 8 min read

Introduction

AdaBoost algorithm, introduced by Freund and Schapire in 1997, revolutionized ensemble modeling. Since its inception, AdaBoost has become a widely adopted technique for addressing binary classification challenges. This powerful algorithm enhances prediction accuracy by transforming a multitude of weak learners into robust, strong learners

The principle behind boosting algorithms is that we first build a model on the training dataset and then build a second model to rectify the errors present in the first model. This procedure is continued until and unless the errors are minimized and the dataset is predicted correctly. Boosting algorithms work in a similar way, it combines multiple models (weak learners) to reach the final output (strong learners).

Learning Objectives

- To understand what the AdaBoost algorithm is and how it works.
- To understand what stumps are.

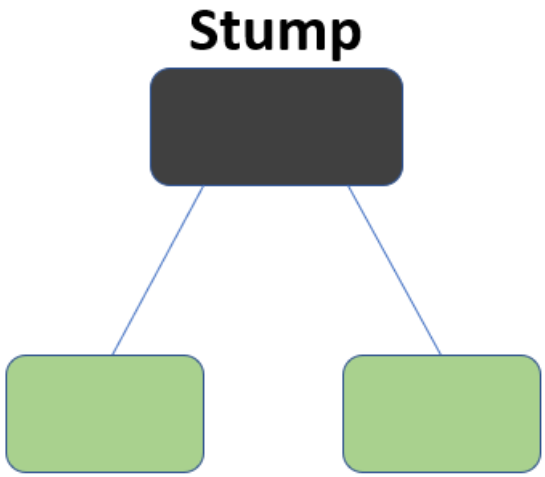
This article was published as a part of the [Data Science Blogathon]

Table of contents

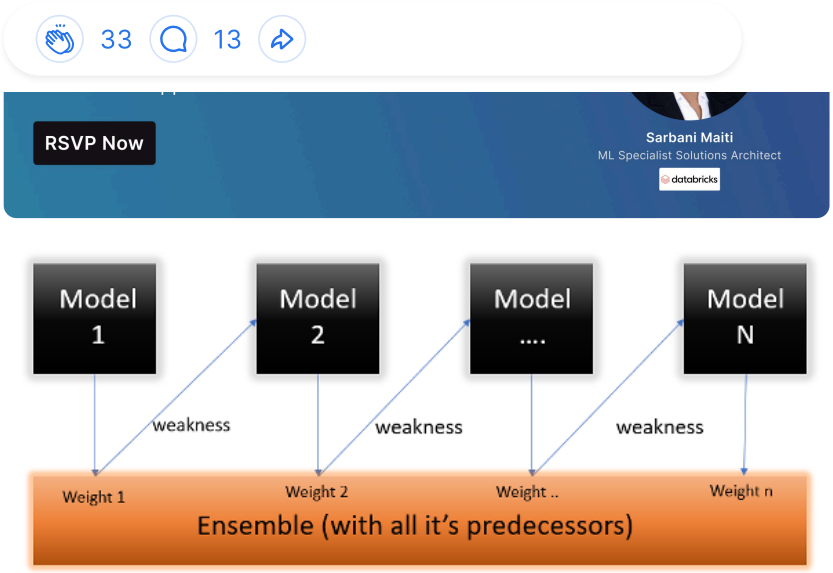
What Is the AdaBoost Algorithm?

There are many machine learning algorithms to choose from for your problem statements. One of these algorithms for predictive modeling is called AdaBoost.

AdaBoost algorithm, short for Adaptive Boosting, is **a Boosting technique used as an Ensemble Method in Machine Learning**. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances.



What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points with higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.



Let's take an example to understand this, suppose you built a decision tree algorithm on the Titanic dataset, and from there, you get an accuracy of 80%. After this, you apply a different algorithm and check the accuracy, and it comes out to be 75% for KNN and 70% for Linear Regression.

When building different models on the same dataset, we observe variations in accuracy. However, leveraging the power of AdaBoost, we can combine these algorithms to enhance the final predictions. By averaging the results from diverse models, Adaboost allows us to achieve higher accuracy and bolster predictive capabilities effectively.

If you want to understand this visually, I strongly recommend you go through this article.

Here we will be more focused on mathematics intuition.

There is another ensemble learning algorithm called the gradient boosting algorithm. In this algorithm, we try to reduce the error instead of wights, as in AdaBoost. But in this article, we will only be focussing on the mathematical intuition of AdaBoost.

Let's understand what and how this algorithm works under the hood with the following tutorial.

Step 1: Assigning Weights

The Image shown below is the actual representation of our dataset. Since the target column is binary, it is a classification problem. First of all, these data points will be assigned some weights. Initially, all the weights will be equal.

Row No.	Gender	Age	Income	Illness	Sample Weights
1	Male	41	40000	Yes	1/5
2	Male	54	30000	No	1/5
3	Female	42	25000	No	1/5
4	Female	40	60000	Yes	1/5
5	Male	46	50000	Yes	1/5

The formula to calculate the sample weights is:

$$w(x_i, y_i) = \frac{1}{N}, \; i = 1, 2, \ldots . n$$

Where N is the total number of data points

Here since we have 5 data points, the sample weights assigned will be 1/5.

Step 2: Classify the Samples

We start by seeing how well "*Gender*" classifies the samples and will see how the variables (Age, Income) classify the samples.

We'll create a decision stump for each of the features and then calculate the ***Gini Index*** of each tree. The tree with the lowest Gini Index will be our first stump.

Step 3: Calculate the Influence

We'll now calculate the **"Amount of Say"** or **"Importance"** or **"Influence"** for this classifier in classifying the data points using this formula:

$$\frac{1}{2}\log\frac{1-Total\ Error}{Total\ Error}$$

The total error is nothing but the summation of all the sample weights of misclassified data points.

Here in our dataset, let's assume there is 1 wrong output, so our total error will be 1/5, and the alpha (performance of the stump) will be:

$$Performance\ of\ the\ stump\ =\ \frac{1}{2}\log_e(\frac{1-Total\ Error}{Total\ Error})$$

$$\alpha\ =\ \frac{1}{2}\log_e\left(\frac{1-\frac{1}{5}}{\frac{1}{5}}\right)$$

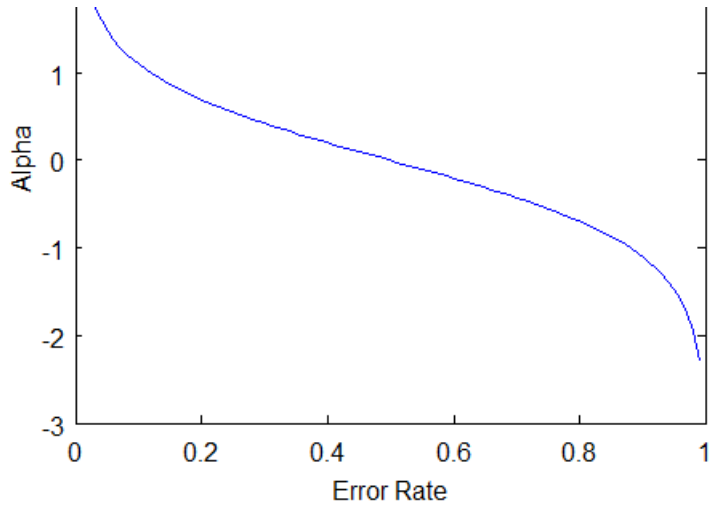
$$\alpha\ =\ \frac{1}{2}\log_e\left(\frac{0.8}{0.2}\right)$$

$$\alpha\ =\ \frac{1}{2}\log_e(4)\ =\ \frac{1}{2}*(1.38)$$

$$\alpha\ =\ 0.69$$

Note: Total error will always be between 0 and 1.

0 Indicates perfect stump, and 1 indicates horrible stump.



From the graph above, we can see that when there is no misclassification, then we have no error (Total Error = 0), so the “amount of say (alpha)” will be a large number.

When the classifier predicts half right and half wrong, then the Total Error = 0.5, and the importance (amount of say) of the classifier will be 0.

If all the samples have been incorrectly classified, then the error will be very high (approx. to 1), and hence our alpha value will be a negative integer.

Step 4: Calculate TE and Performance

You might be wondering about the significance of calculating the Total Error (TE) and performance of an Adaboost stump. The reason is straightforward – updating the weights is crucial. If identical weights are maintained for the subsequent model, the output will mirror what was obtained in the initial model.

The wrong predictions will be given more weight, whereas the correct predictions weights will be decreased. Now when we build our next model after updating the weights, more preference will be given to the points with higher weights.

we use the following formula:

$$New\ sample\ weight\ =\ old\ weight\ *\ e^{\pm Amount\ of\ say\ (\alpha)}$$

The amount of, say (alpha) will be ***negative*** when the sample is **correctly classified**.

The amount of, say (alpha) will be ***positive*** when the sample is **miss-classified.**

There are four correctly classified samples and 1 wrong. Here, the ***sample weight*** of that datapoint is *1/5,* and the ***amount of say/performance of the stump*** of ***Gender*** is *0.69*.

New weights for *correctly classified* samples are:

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \exp(-0.69)$$

$$New\ sample\ weight\ =\ 0.2\ *\ 0.502\ =\ 0.1004$$

For *wrongly classified* samples, the updated weights will be:

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \exp(0.69)$$

$$New\ sample\ weight\ =\ 0.2\ *\ 1.994\ =\ 0.3988$$

Note

See the sign of alpha when I am putting the values, the **alpha is negative** when the data point is correctly classified, and this *decreases the sample weight* from 0.2 to 0.1004. It is **positive** when there is **misclassification**, and this will *increase the sample weight* from 0.2 to 0.3988

1	Male	41	40000	Yes	1/5	0.1004
2	Male	54	30000	No	1/5	0.1004
3	Female	42	25000	No	1/5	0.1004
4	Female	40	60000	Yes	1/5	0.3988
5	Male	46	50000	Yes	1/5	0.1004

We know that the total sum of the sample weights must be equal to 1, but here if we sum up all the new sample weights, we will get 0.8004. To bring this sum equal to 1, we will normalize these weights by dividing all the weights by the total sum of updated weights, which is 0.8004. So, after normalizing the sample weights, we get this dataset, and now the sum is equal to 1.

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	0.1004/0.8004 =0.1254
2	Male	54	30000	No	1/5	0.1004/0.8004 =0.1254
3	Female	42	25000	No	1/5	0.1004/0.8004 =0.1254
4	Female	40	60000	Yes	1/5	0.3988/0.8004 =0.4982
5	Male	46	50000	Yes	1/5	0.1004/0.8004 =0.1254

Step 5: Decrease Errors

Now, we need to make a new dataset to see if the errors decreased or not. For this, we will remove the “sample weights” and “new sample weights” columns and then, based on the “new sample weights,” divide our data points into buckets.

Row No.	Gender	Age	Income	Illness	New Sample Weights	Buckets
1	Male	41	40000	Yes	0.1004/0.8004= 0.1254	0 to 0.1254
2	Male	54	30000	No	0.1004/0.8004= 0.1254	0.1254 to 0.2508
3	Female	42	25000	No	0.1004/0.8004= 0.1254	0.2508 to 0.3762
4	Female	40	60000	Yes	0.3988/0.8004= 0.4982	0.3762 to 0.8744
5	Male	46	50000	Yes	0.1004/0.8004= 0.1254	0.8744 to 0.9998

Step 6: New Dataset

classified records have higher sample weights, the probability of selecting those records is very high.

Suppose the 5 random numbers our algorithm take is 0.38,0.26,0.98,0.40,0.55.

Now we will see where these random numbers fall in the bucket, and according to it, we'll make our new dataset shown below.

Row No.	Gender	Age	Income	Illness
1	Female	40	60000	Yes
2	Male	54	30000	No
3	Female	42	25000	No
4	Female	40	60000	Yes
5	Female	40	60000	Yes

This comes out to be our new dataset, and we see the data point, which was wrongly classified, has been selected 3 times because it has a higher weight.

Step 7: Repeat Previous Steps

Now this act as our new dataset, and we need to repeat all the above steps i.e.

1. Assign ***equal weights*** to all the data points.
2. Find the stump that does the ***best job classifying*** the new collection of samples by finding their Gini Index and selecting the one with the lowest Gini index.
3. Calculate the ***"Amount of Say"*** and ***"Total error"*** to update the previous sample weights.
4. Normalize the new sample weights.

Iterate through these steps until and unless a low training error is achieved.

a ***sequential manner.*** If we send our **test data** now, it will pass through all the decision trees, and finally, we will see which class has the majority, and based on that, we will do predictions
for our test dataset.

Conclusion

You have finally mastered this algorithm if you understand each and every line of this article.

We started by introducing you to what Boosting is and what are its various types to make sure that you understand the Adaboost classifier and where AdaBoost algorithm falls exactly. We then applied straightforward math and saw how every part of the formula worked.

In the next article, I will explain Gradient Descent and Xtreme Gradient Descent algorithm, which are a few more important Boosting techniques to enhance the prediction power.

If you want to know about the python implementation for beginners of the AdaBoost machine learning model from scratch, then visit this complete guide from analytics vidhya. This article mentions the difference between bagging and boosting, as well as the advantages and disadvantages of the AdaBoost algorithm.

Key Takeaways

- In this article, we understood how boosting works.
- We understood the maths behind adaboost.
- We learned how weak learners are used as estimators to increase accuracy.