**Ace your Machine Learning Interview — Part 8**

**Dive into Ensemble Learning with AdaBoost from scratch using Python**

In this article of my series “Ace your Machine Learning Interview” I continue to talk about Ensemble Learning and in particular, I will focus on Boosting algorithms with special reference to **AdaBoost**. I hope that this series in which I review the basics of Machine Learning will be useful to you in facing your next interview!😁

In case you are interested in the previous articles in this series, I leave the links here:

1. [*Ace your Machine Learning Interview — Part 1*](https://towardsdatascience.com/ace-your-machine-learning-interview-part-1-e6a5897e6844)*: Dive into Linear, Lasso and Ridge Regression and their assumptions*
2. [*Ace your Machine Learning Interview — Part*](https://towardsdatascience.com/ace-your-machine-learning-interview-part-1-e6a5897e6844) *2: Dive into Logistic Regression for classification problems using Python*
3. [*Ace your Machine Learning Interview — Part*](https://towardsdatascience.com/ace-your-machine-learning-interview-part-1-e6a5897e6844) *3: Dive into Naive Bayes Classifier using Python*
4. [*Ace your Machine Learning Interview — Part 4*](https://medium.com/p/e30b695ce63/edit)*: Dive into Support Vector Machines using Python*
5. [*Ace your Machine Learning Interview — Part*](https://medium.com/p/e30b695ce63/edit) *5: Dive into Kernel Support Vector Machines using Python*
6. [*Ace your Machine Learning Interview — Part 6*](https://medium.com/p/5f0d84e435a1/edit)*: Dive into Decision Trees using Python*
7. [*Ace your Machine Learning Interview — Part 7*](https://medium.com/p/2688de34805f/edit)*: Dive into Ensemble Learning with Hard Voting Classifiers using Python*

**Introduction**

We talked in the last article in general about what Ensemble Learning is and we have seen and implemented simple Ensmble methods based on Majority Voting.

Today we talk more in detail about an Ensemble method called Boosting by making special reference to **Adaptive Boosting or AdaBoost**. You may have heard of this algorithm before, it is often used to win Kaggle competitions for example.

The basic idea of AdaBoost was first stated in the paper [*“The Strength of Weak Learnability by Robert E. Schapire”*](http://rob.schapire.net/papers/strengthofweak.pdf).

The idea is to use in a combined way various ML algorithms that are called weak learners since by themselves they are not very good, they can do little better than random guessing. By using these weak learners sequentially, very strong results can be obtained.

**Original AdaBoost Idea**

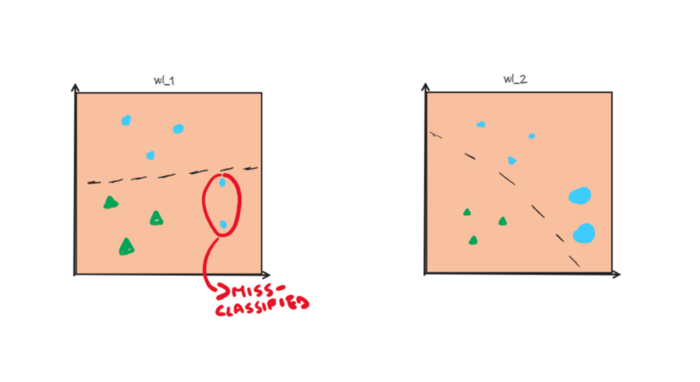
As originally conceived, **Boosting was originally based on the technique we have already seen called Pasting**.

Therefore, several weak learners were implemented and **each weak learner was trained on a subset of the train set data without replacement**. The weak learners that we will call ***wl\_1, wl\_2, …, wl\_n* were trained sequentially** and each *wl\_i+1* was trained on a random subset of the train set to which **50% of the misclassified data points from *wl\_i* were added.** In this way, more weight was given to the misclassified data points.

In the end, they would combine all the weak learners so trailed in a majority voting function.

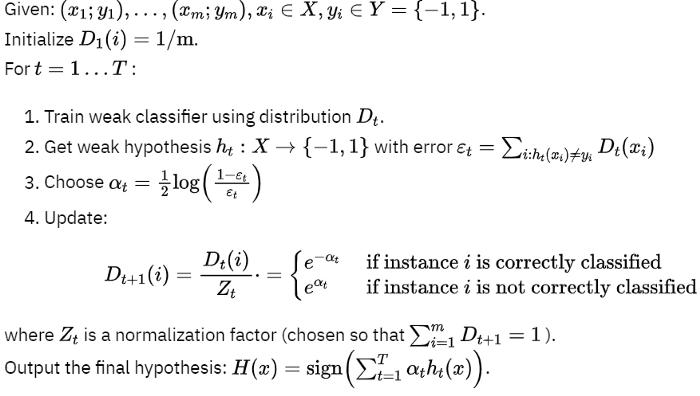
**AdaBoost**

The basic idea has remained the same but now the AdaBoost algorithm is slightly different. First, **Pasting is no longer used but each weak learner is trained over the entire available train set**. In addition, another mechanism is used to give more importance to the points misclassified by the previous weak learner. The weak learner ***wl\_i+1* will simply give more weight to the misclassified points of *wl\_i* and less weight to the well-classified ones**.



Weight Missclassified data point (Image By Author)

Before implementing this algorithm let’s look at the pseudo code to get an idea.



Pseudocode Adaboost (Image By Author)

**Let’s code!**

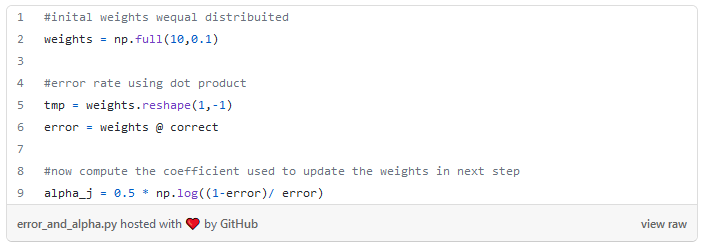
So first we initialize two arrays *y* and *y\_hat* which will be binary vectors of +1 and -1. These will represent the true classes and the predicted classes. After that, we create a list of booleans where each entry tells us whether the prediction was correct or not.

https://gist.github.com/March-08?page=1



Now let’s initialize the weights. At first, the weights will be the same for each data point, since we have 10 each weight will be 0.1.

Using these initial weights and predictions, we go on to calculate the error and coefficient alpha that we will need to calculate the newly updated weights as shown in the pseudo-code.



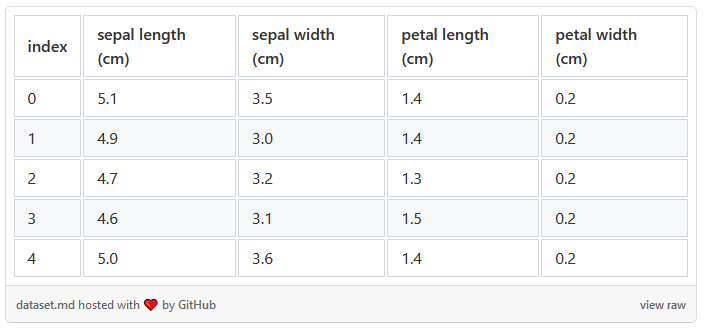
Now we are ready to calculate the new weights and then normalize them. Note that depending on whether the prediction was correct or incorrect we have two different formulas for updating the weights.



Now we have the new weights in step two. Y**ou can choose to do m steps by putting everything in a for loop**.

Fortunately, much more efficient implementations already exist with various libraries. So **let’s see how to use AdaBoost using sklearn!**

The following Iris dataset is provided by sklearn under an open license, it can be found [here](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html).



First we import the data and split it into train and test set. Note for simplicity we will use only two features in this dataset.



Now with just a few lines of code, we can create an AdaBoost based on 500 simple Decision Trees each with *max\_depth = 1*. And we see that we get much better results than just using decision trees.



That’s it! In a few simple steps, you can create your AdaBoost algorithm and get your predictions.

**Final Thoughts**

So in this second paper on Ensemble Learning, we saw a new algorithm called AdaBoost. So-called weak learners are trained sequentially, and this can be viewed badly since this sequential training will waste a lot of our time. We will then be able to **make each predictor correct its predecessor by paying attention to the underfitted cases**. **So new predictor focuses more on hard cases. For the new predictor, we increase the weight of the hard cases.**

Follow me for new articles of my series “Ace you Machine Learning Interview”!😁

**The End**

*Marcello Politi*

[Linkedin](https://www.linkedin.com/in/marcello-politi/), [Twitter](https://twitter.com/_March08_), [CV](https://march-08.github.io/digital-cv/)