Introduction

Welcome to the module on Boosting!

Boosting is one of the most powerful ideas introduced in the field of machine learning in the past few years. It was first introduced in 1997 by Freund and Schapire in the popular algorithm, AdaBoost. It was originally designed for classification problems. Since its inception, many other boosting algorithms have been developed that tackle regression problems also and have become famous as they are used in the top solutions of many Kaggle competitions. We will go through the concepts of the most popular boosting algorithms - AdaBoost, Gradient Boosting and XGBoost in this module.

In the previous course of Tree models, we had gone through different ensemble models. In this particular course we will focus on one such Ensemble model - Boosting.
Prof. Raghavan will introduce you to the overall structure of this course:

In this video, the professor mentions the different boosting methods we will discuss in the course:

1. Adaptive Boosting
2. Gradient Boosting
3. XGBoost

In this session

This session will introduce you to the following topics:

- Ensemble models
- Introduction to Boosting
- Building blocks of Boosting
- AdaBoost procedure

Guidelines for in-module questions

The in-video and in-content questions for this module are not graded.

Prof G Srinivasaraghavan

Professor, IIIT-B

The International Institute of Information Technology, Bangalore, commonly known as IIIT Bangalore, is a premier national graduate school in India. Founded in 1999, it offers Integrated M.Tech., M.Tech., M.S. (Research), and PhD programs in the field of information technology.

Anjali Rajvanshi

Sr. Subject Matter Expert, upGrad

Anjali has around 11 years of experience and has worked as a software engineer, data scientist, project lead for companies like Infosys, Evalueserve etc across geographies. She is currently working as a Senior Subject Matter Expert with upGrad.

Snehansu Sekhar Sahu

Sr. Research Engineer in ML & AI at American Express

Snehanshu has more than 6 years of experience and has worked with companies like Qualcomm Inc, Infoedge solutions & American Express. He is currently part of the ML & AI Research Group @ AmEx.

Report an error

NEXT
Ensemble models

Ensemble models

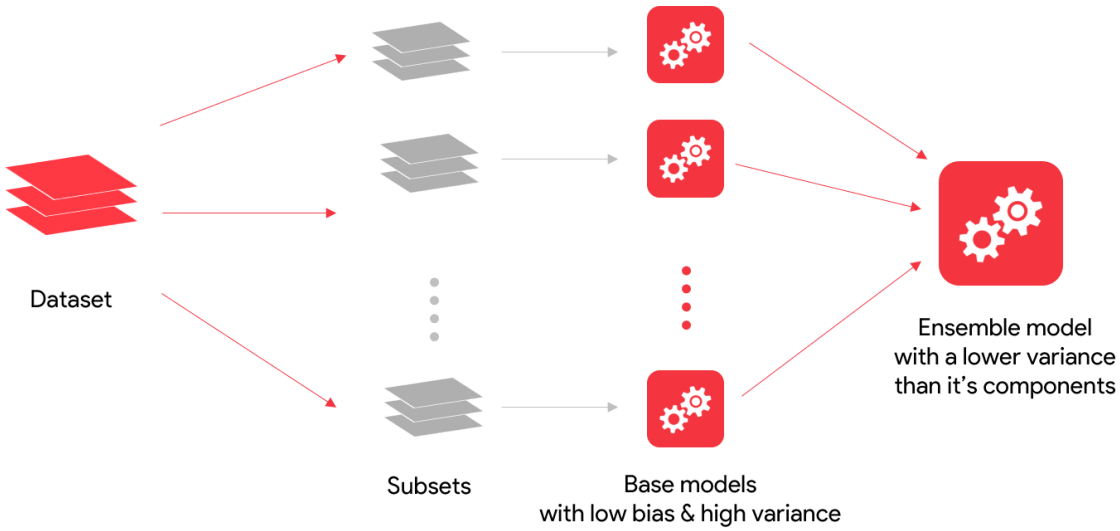
Before proceeding further please revisit the content on Ensemble models explained by Rahim.
For the purposes of this module we will be focusing on Boosting, but we will start with the difference between bagging & boosting.

"The strength of unity lies in the diversity", this saying holds the same meaning in the world of machine learning. Ensemble models bring us that flavour of diversity to create powerful models that can handle complex problems. It is a combination of models each of which are trained to solve the same problem to provide the best result at the end.

For a machine learning task (classification or regression), we need to have a model which identifies the necessary patterns in the data and does not overfit. In other words 'the models should not be so simple as to not identify even the important patterns present in the data on one hand and on the other hand, they should not be so complex so as to even learn the noise present in the dataset'.

We can arrive at this solution either through a single model or an ensemble i.e., a collection of models. By combining several models, **ensemble learning methods create a strong learner thus reducing the bias and/or variance of the individual models.**

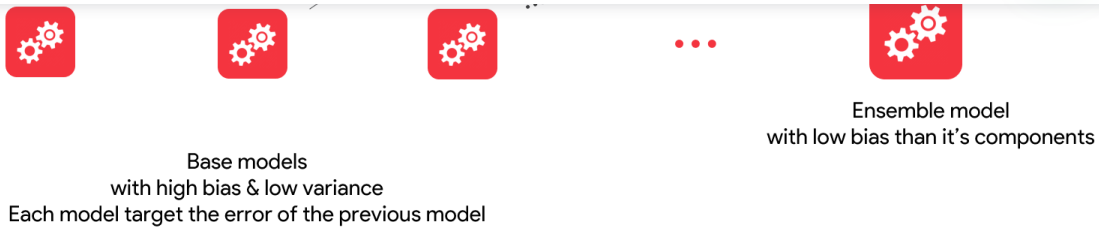
different subsets of the training data, resulting in a slight difference between the individual models. The predictions of these individual models are combined by taking the average of all the values for regression or a majority vote for a classification problem. Random forest is an example of the bagging method.



Bagging

Bagging works well when the algorithm we use to build our model has high variance. By this we mean the model built changes a lot even with slight changes in the data. As a result this, these algorithms overfit easily if not controlled. Recall, that decision trees are prone to overfitting if we don't tune the hyperparameters well. Bagging works very well for **high-variance models like decision trees.**

Boosting is an another popular approach to ensembling. This technique combines individual models into a strong learner by creating sequential models such that the final model has higher accuracy than the individual models. Let us understand this.



Boosting

These individual models are connected in such a way that the subsequent models are dependent on errors of the previous model and and each subsequent model tries to correct the errors of the previous models.

Question 1/1 Mandatory

Ensemble models

With respect to base models in Ensemble learning, which of the following statements is/are true?

1. In boosting the base models are parallelly connected to reduce the overall bias of the resulting strong model.
2. In bagging the base models are sequentially connected to reduce the overall variance of the resulting strong model.
3. In boosting the base models are sequentially connected to reduce the overall bias of the resulting strong model.

- Option 1 & 2
- Option 2 & 3

In boosting the base models are sequentially connected to reduce the overall bias of the resulting strong model.

Option 1 only ✕ Incorrect

Feedback:
In boosting, the base models are sequentially connected.

Your answer is Wrong. Attempt 2 of 2 Continue

Report an error

PREVIOUS
Introduction

NEXT
Weak Learners

Weak Learners

Before proceeding further on boosting, let's first look at the foundation of boosting algorithms — **Weak Learners.**
As learned earlier, Boosting is an approach where the individual models are connected in such a way that they correct the mistakes made by the previous model. Here, these individual models are called Weak learners.

Till now, all the models we have learned are strong learners where each model performs well on whatever task it is assigned to do - classification or regression.
Weak learner, on the other hand, refers to a simple model which performs at least **better than a random guesser** (the error rate should be lesser than 0.5). It primarily identifies only the prominent pattern(s) present in the data and thus is not capable of overfitting.
In Boosting, we use such weak learners to build our ensemble.

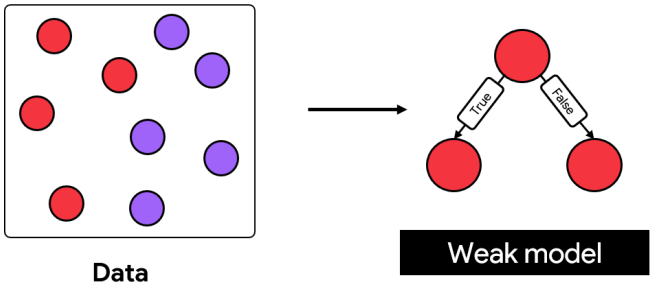
In Boosting, any model can be a weak learner – Linear regression, Decision Tree or any other model, but more often than not, tree methods are used here.
Note: Decision stump is one such weak learner when we are talking about a shallow decision tree having a depth of only 1.

In the next video, Anjali will help you in understanding the concept of weak learners.

To summarize: Weak learners are combined sequentially such that each subsequent model corrects the mistakes of the previous model, resulting in a strong overall model that gives good predictions.

Through weak learners we can:

- Reduce the variance of the final model, making it more robust(generalisable)
- Train the ensemble quickly resulting in faster computation time



- **A weak model/learner is not capable of overfitting**
- **it has to be at least better than a random guesser.**
- **Any model can be a weak learner – Linear regression, Decision Tree**
- **Decision Stump is a very shallow decision tree having a depth of 1**

Question 1/1 Mandatory

Weak learners

Boosting uses weak learners as classifiers because _____

1. The error is greater than 0.5
2. The error is lesser than 0.5
3. Each weak learners is independent of the previous model
4. Each weak learner is dependent on the previous model

1 & 3 only ✗ Incorrect

Feedback:
The weak learners in boosting algorithm are better than a random guesser

2 & 4 only ✓ Correct

Feedback:
The weak learners in boosting algorithm have an error rate less than 5 and each model correct the results of the previous tree.

Your answer is Wrong. Attempt 1 of 1 Continue

Report an error

PREVIOUS
Ensemble models

NEXT
Introduction to Adaboost

Weak Learners

Before proceeding further on boosting, let's first look at the foundation of boosting algorithms — **Weak Learners.**
As learned earlier, Boosting is an approach where the individual models are connected in such a way that they correct the mistakes made by the previous model. Here, these individual models are called Weak learners.

Till now, all the models we have learned are strong learners where each model performs well on whatever task it is assigned to do - classification or regression.
Weak learner, on the other hand, refers to a simple model which performs at least **better than a random guesser** (the error rate should be lesser than 0.5). It primarily identifies only the prominent pattern(s) present in the data and thus is not capable of overfitting.
In Boosting, we use such weak learners to build our ensemble.

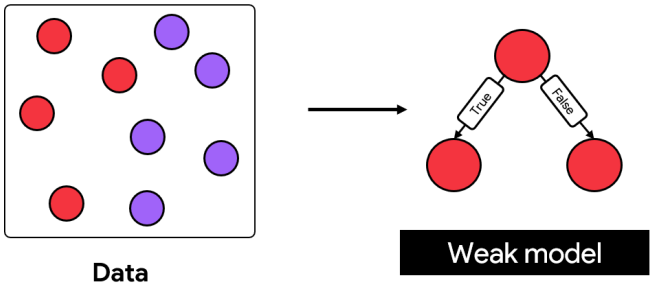
In Boosting, any model can be a weak learner – Linear regression, Decision Tree or any other model, but more often than not, tree methods are used here.
Note: Decision stump is one such weak learner when we are talking about a shallow decision tree having a depth of only 1.

In the next video, Anjali will help you in understanding the concept of weak learners.

To summarize: Weak learners are combined sequentially such that each subsequent model corrects the mistakes of the previous model, resulting in a strong overall model that gives good predictions.

Through weak learners we can:

- Reduce the variance of the final model, making it more robust(generalisable)
- Train the ensemble quickly resulting in faster computation time



- **A weak model/learner is not capable of overfitting**
- **it has to be at least better than a random guesser.**
- **Any model can be a weak learner – Linear regression, Decision Tree**
- **Decision Stump is a very shallow decision tree having a depth of 1**

Question 1/1 Mandatory

Weak learners

Boosting uses weak learners as classifiers because _____

1. The error is greater than 0.5
2. The error is lesser than 0.5
3. Each weak learners is independent of the previous model
4. Each weak learner is dependent on the previous model

1 & 3 only ✗ Incorrect

Feedback:
The weak learners in boosting algorithm are better than a random guesser

2 & 4 only ✓ Correct

Feedback:
The weak learners in boosting algorithm have an error rate less than 5 and each model correct the results of the previous tree.

Your answer is Wrong. Attempt 1 of 1 Continue

Report an error

PREVIOUS
Ensemble models

NEXT
Introduction to Adaboost

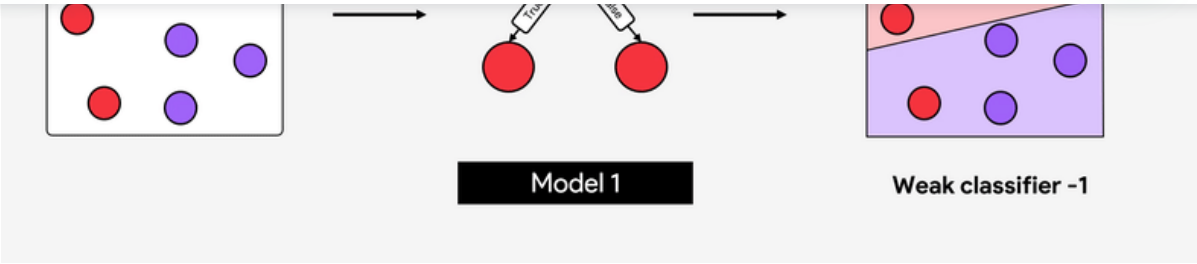
Introduction to Adaboost

Now that we have looked at the overall understanding of what boosting is, let's start with the original boosting algorithm - Adaboost.

AdaBoost stands for Adaptive Boosting and was developed by Schapire and Freund, who later on won the 2003 Godel Prize for their work.

In the next video, Anjali will give you an overview of Adaboost and why we use it.

- Adaboost starts with a uniform distribution of weights over training examples i.e. it gives equal weights to all its observations. These weights tell the importance of each datapoint being considered.
- We start with a single weak learner to make the initial predictions.
- Once the initial predictions are made, patterns which were not captured by previous weak learner is taken care of by the next weak learner by giving more weightage to the misclassified data points.
- Apart from giving weightage to each observation, the model also gives weightage to each weak learner. More the errors in the weak learner, lesser is the weightage given to it. This helps when the ensembled model makes final predictions.
- After getting these two weights - for the observations and the individual weak learners, the next model(weak learner) in the sequence trains on the resampled data (data sampled according to the weights) to make the next prediction.
- The model will iteratively continues the steps mentioned above for a pre-specified number of weak learners.
- In the end, we take a weighted sum of the predictions from all these weak learners to get an overall strong learner.



A strong learner is formed by the combination of multiple weak learners which are trained on the mistakes of the previous model

Now that you have an intuitive understanding of Adaboost, let's take a look at the inner working of the algorithm with the help of a numerical example.

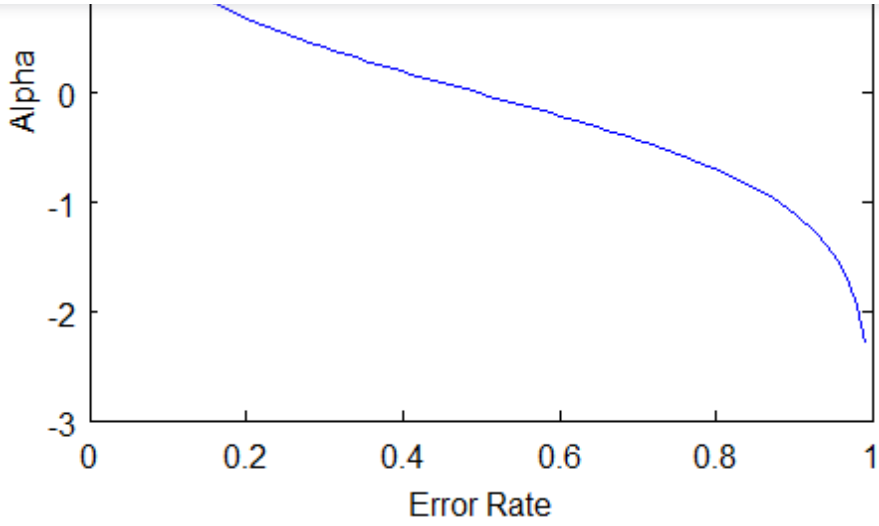
In Adaboost, we start with a base model with equal weights given to every observation. In the next step, the observations which are incorrectly classified will be given a higher weight so that when a new weak learner is trained, they will give more attention to these misclassified observations.

In the end, we get a series of models that have different say according to the predictions each weak model has made. If the model performs poorly and makes many incorrect predictions then it is given less significance, whereas if the model performs well and does correct predictions most of the time, then it is given more significance in the overall model.

The say/importance each weak learner, in our case the decision tree stump, has in the final classification depends on the total error it made.

$$\alpha = 0.5 \ln((1 - \text{Total error})/\text{Total error})$$

You can see the relation of **α**(**alpha**) & **error** with this graph:



The value of error rate lies between 0 & 1 so let's see how alpha & error is related.

- When the base model performs with less error overall then, as you can see in the plot above, the α is a large positive value, which means that the weak learner will have a high say in the final model.
- If the error is 0.5, it means that it is not sure of the decision, then the α =0, i.e the weak learner will have no say or significance in the final model.
- If the model produces large errors(i.e close to 1), then α is a large negative value, meaning that the predictions it makes is incorrect most of the time. Hence this weak learner will have a very low say in the final model.

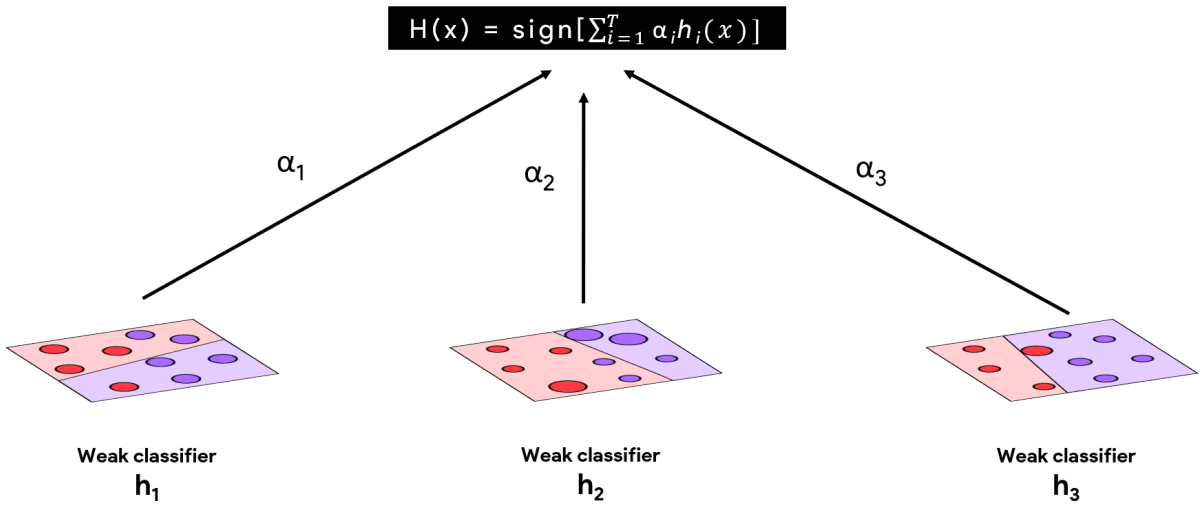
After calculating the say/importance of each weak learner, we find the new weights of each observation present in the training dataset. The following formula is used to compute the new weight for each observation:

new sample weight for the incorrectly classified = **original sample weight** * e^{α}

new sample weight for the correctly classified = **original sample weight** * $e^{-\alpha}$

The samples which the previous stump incorrectly classified will be given higher weights and the ones which the previous stump classified correctly will be given lower weights. Next, a new stump will be created by randomly sampling the weighted observations. Due to the weights given to each observation, the new dataset will have a tendency to contain multiple copies of the observations that were misclassified by the previous tree and may not contain all observations which were correctly classified. This will essentially help the next weak learner to give more importance to the incorrectly classified sample so that it can correct the mistake and correctly classify it now. This process will be repeated till a pre-specified number of trees are built i.e. the ensemble is built.

The AdaBoost model makes predictions by having each tree in the ensemble classify the sample. Then, we split the trees into groups according to their decisions. For each group, we add up the significance of every tree inside the group. The final prediction made by the ensemble as a whole is determined by the sign of the weighted sum.



NOTE: Weight is mentioned in two different contexts - one for each observation & the other is in context of the importance/say given to each weak learner in the model.

Question 1/2 Mandatory

Adaboost

Suppose you are performing a binary classification problem using Adaboost. If there are total of 10 samples and the first tree that the model predicted has total 2 misclassifications.

With this understanding, what will be the say/importance of this tree

(Round off the value up to 3 decimal places)

- (○) 0.693 ✓ Correct

  Feedback:
  The say/importance of a decision tree = 0.5*ln((1-0.2)/0.2)

- ( ) 0.301

Your answer is Correct. Attempt 1 of 1 Continue

Report an error

NOTE: Weight is mentioned in two different contexts - one for each observation & the other is in context of the importance/say given to each weak learner in the model.

Question 2/2 Mandatory

Adaboost

Suppose you are performing a binary classification problem using Adaboost. If there are a total of 10 samples and the first tree that the model predicted has total 2 misclassifications.

With this understanding, what will be the total updated weights of all incorrect predictions?

(Round off the value up to 3 decimal places)

0.1 ✗ Incorrect

Feedback:
The new weights of incorrect predictions = 0.2 * $e^{0.693}$

0.399 ✓ Correct

Feedback:
The new weights of incorrect predictions = 2*0.1 * $e^{0.693}$

Your answer is Wrong. Attempt 1 of 1 Continue

AdaBoost Algorithm

Now that we have understood the intuition behind the algorithm, let us see the pseudo-code behind it.

In the next video, Anjali will explain how the pseudo-code in conjunction with the numerical example we saw earlier.

Here is the summary of the AdaBoost algorithm we have studied until now:

2. Compute $\epsilon_t = \sum_i^n D_i[h_t(x_i) \neq y_i]$

3. Compute $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$

4. Update $D_{t+1}(i) = \frac{D_t(i) * e^{-\alpha_t y_i h_t(x_i)}}{z_t}$ where, $z_t = \sum_{i=1}^{n} D_i * e^{-\alpha_t y_i h_t(x_i)}$

3. Final Model: $H(x) = sign(\sum_{t=i}^{T} \alpha_t h_t(x))$

We see here that with each new weak learner, the distribution of the data changes i.e. the weight given to each observation changes.

Observe the factor: $e^{-\alpha_t y_i h_t(x_i)}$

If there is a misclassification done by the model, then the product of $y_i * h_t(x_i)$ = -1

So the power of the exponential will be positive (growing exponential weight).

This indicates that the weight will increase for all misclassified points.

Otherwise, if it is correctly classified then a product of $y_i * h_t(x_i)$ = 1.

So it will have a decaying weight because of the negative term in the power of the exponential term. This indicates that the weight will decrease for all correctly classified points.

The point here is **to force classifiers to concentrate on observations that are difficult to classify correctly.**

=~0.1*3
=~0.3

Growing weight

=~0.1*.3
=~.03

Decaying weight

The model continues adding weak learners till a preset number of weak learners have been added.

We then make the final prediction by adding up the weighted prediction for every classifier.

$$H(x) = sign(\sum_{t=i}^{T} \alpha_t h_t(x))$$

Note: Summarizing the notations in the lecture - At an iteration t, we have a distribution D_t of the training data T on which we fit a model h_t and then use the results to create a new distribution D_{t+1}.

The final model $H(x)$ we built is an ensemble of all the individual models h_i with weights α_i.

In the next segment, you will learn how we perform these two steps. Before that, try thinking about some ideas we just learnt through the following questions.

Question 1/1 Mandatory

Probability Boosting

In a binary classification into +/-1, at an iteration t, F_t predicts the values for certain data points as follows:

By looking at the table, probabilities of which of the points will be boosted?

(More than one option can be correct)

☐ x_1 ✓ Correct

Feedback:
This point is incorrectly predicted, hence the probability for the next iteration will be increased.

☐ x_2

☐ x_3

☐ x_4 ✓ Correct

Feedback:
This point is incorrectly predicted, hence the probability for the next iteration will be increased.

Your answer is Correct. Attempt 2 of 2 Continue

Question 3/3 Mandatory

Question 1

Practical advice: Before you apply the AdaBoost algorithm, you should remove the **Outliers**. Since AdaBoost tends to boosts up the probabilities of misclassified points and there is a high chance that outliers will be misclassified, it will keep increasing the probability associated with the outliers and make the progress difficult. Some of the ways to identify outliers are:

- Boxplots
- Cook's distance
- Z-score.

Additional Reading:

- Please refer to the link for 'A Short Introduction to Boosting' by Freund and Schapire.

Report an error

Error value

Why is the error ϵ_t always less than 1/2 in the above AdaBoost setting?

Word Count 35 | Word Limit 5 - 100

- At an iteration
t
, we have a distribution
D
t
of the training data
T

Suggested Answer

The models used in the Adaboost setting are weak learners but the prediction error they make is less than a random guess. A random guess error is 50% or 0.5. Hence, $\epsilon_t < 0.5$

Attempt 1 of 1 | Continue

Practical advice: Before you apply the AdaBoost algorithm, you should remove the **Outliers**. Since AdaBoost tends to boosts up the probabilities of misclassified points and there is a high chance that outliers will be misclassified, it will keep increasing the probability associated with the outliers and make the progress difficult. Some of the ways to identify outliers are:

- Boxplots
- Cook's distance

Error value

In a binary classification into +/-1, at an iteration t, the first model h_0 predicts the values for certain data points as follows:

Data points	Predicted Value	Actual Value
1	-1	+1
2	+1	+1
3	-1	-1
4	+1	-1

If we consider a hypothetical situation such that these data points constitute the complete dataset on which we build the model h_0, is h_0 be an acceptable starting model?

Yes ✗ Incorrect

Feedback:

We see that the model has misclassified 2 points and the misclassification error is 0.5. We cannot accept this model because the weak learners should have an error less than 0.5.

No ✓ Correct

Feedback:

We see that the model has misclassified 2 points and the misclassification error is 0.5. We cannot accept this model because the weak learners should have an error less than 0.5.

Your answer is Wrong. Attempt 1 of 1 Continue

Alpha value

The following table shows the actual value and the predicted value by an intermediate tree h_t for 5 sample points :

Data points	Actual	Predicted
1	+1	-1
2	-1	-1
3	+1	+1
4	-1	+1
5	+1	+1

If we consider a hypothetical situation such that these data points constitute the complete dataset on which we build the model h_t, what weight α_t will be attached to h_t ?

- 1.5
- 0.2027 ✓ Correct

  Feedback:
  We see that the error ϵ_t = 0.4. Setting this value into the formula, we get 1/2ln(0.6/0.4) = 0.2027
- 0.6012
- 0.4076

Your answer is Correct. Attempt 1 of 2 Continue

AdaBoost Lab

In the following video, Snehanshu will give you a walkthrough on the notebook on how to implement the AdaBoost algorithm in python. In this exercise, you have to go through the notebook attached to the page and answer the questions that follow.

AdaBoost Lab Download

answer the following questions.

Refer to the documentation of AdaBoostClassifier if needed.

You can try changing the number of trees in 'estimators = list(range(1, 50, 3))' to 'estimators = list(range(1, 200, 3))' and see if the acuracy increases.

Note that we have used 'accuracy_score' as the evaluation metric here. We can use other evaluation metrics also like 'roc_auc_curve'.

Question 1/2 Mandatory

AdaBoost Accuracy

What is the accuracy of the initial weak learner we choose?

Please round off to the nearest decimal.

- 0.96
- 0.94 ✓ Correct

Feedback:
The shallow decision tree we fit at the start gives an accuracy of 0.9385964912280702, which on rounding off gives 0.94.

Let's summarize all we have learned until now in the next segment.

Report an error

PREVIOUS
AdaBoost Algorithm

NEXT
Summary

answer the following questions.

Refer to the documentation of AdaBoostClassifier if needed.

You can try changing the number of trees in 'estimators = list(range(1, 50, 3))' to 'estimators = list(range(1, 200, 3))' and see if the acuracy increases.

Note that we have used 'accuracy_score' as the evaluation metric here. We can use other evaluation metrics also like 'roc_auc_curve'.

Question 2/2 Mandatory

AdaBoost Accuracy

As we increase the number of estimators in the AdaBoostClassifier, what happens to the accuracy of the model?

- Increases ✓ Correct

  Feedback:
  We see in the graph that as we increase the number of trees, the accuracy increases in a zigzag way

- Decreases ✗ Incorrect

In this lab, you'll explore the breast cancer dataset and try to train the model to predict if the person is having breast cancer or not. We will start off with a weak learner, a decision tree with maximum depth = 2.

We will then build an adaboost ensemble with 50 trees with a step of 3 and compare the performance with the weak learner.

Let's get started by loading the libraries.

In [1]:

```
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.datasets import load_digits
from sklearn import metrics
%matplotlib inline

import os
import warnings
warnings.filterwarnings('ignore')
```

We will use the breast cancer dataset in which the target variable has 1 if the person has cancer and 0 otherwise. Let's load the data.

In [2]:

```
cancer = load_breast_cancer()
digits = load_digits()

data = cancer
```

In [3]:

```
df = pd.DataFrame(data= np.c_[data['data'], data['target']],
                  columns= list(data['feature_names']) + ['target'])
df['target'] = df['target'].astype('uint16')
```

In [4]:

```
df
```

Out[4]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	me symme
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.24
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.18
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.20
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.25
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.18
...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.17
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.17
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.15
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.23
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.15

569 rows × 31 columns

In [5]:

```
df.head()
```

Out[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 31 columns

In [6]:

```
# adaboost experiments
# create x and y train
X = df.drop('target', axis=1)
y = df[['target']]

# split data into train and test/validation sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=101)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(455, 30)
(455, 1)
(114, 30)
(114, 1)
```

In [7]:

```
# check the average cancer occurence rates in train and test data, should be comparable
print(y_train.mean())
print(y_test.mean())
```

```
target    0.626374
dtype: float64
target    0.631579
dtype: float64
```

In [8]:

```
# base estimator: a weak learner with max_depth=2
shallow_tree = DecisionTreeClassifier(max_depth=2, random_state = 100)
```

In [9]:

```
# fit the shallow decision tree
shallow_tree.fit(X_train, y_train)

# test error
y_pred = shallow_tree.predict(X_test)
score = metrics.accuracy_score(y_test, y_pred)
score
```

Out[9]:

```
0.9385964912280702
```

Now, we will see the accuracy using the AdaBoost algorithm. In this following code, we will write code to calculate the accuracy of the AdaBoost models as we increase the number of trees from 1 to 50 with a step of 3 in the lines:

'estimators = list(range(1, 50, 3))'

'for n_est in estimators:'

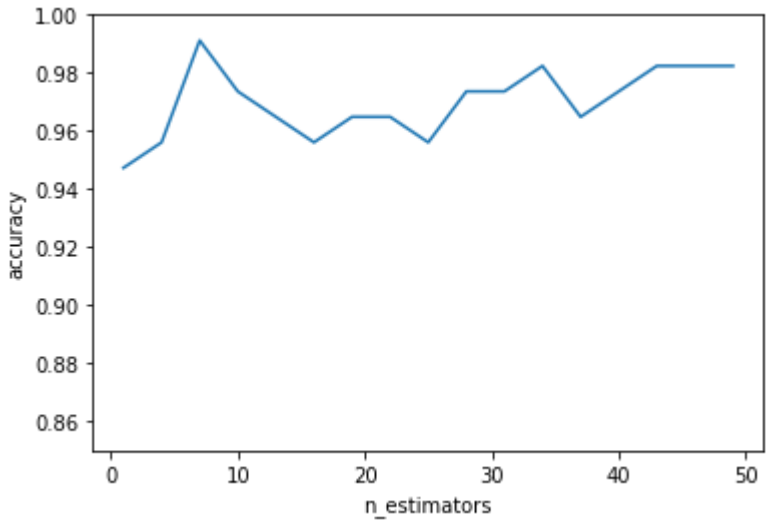
We finally end up with the accuracy of all the models in a single list abc_scores.

In [14]:

```
# adaboost with the tree as base estimator

estimators = list(range(1, 50, 3))

abc_scores = []
for n_est in estimators:
    ABC = AdaBoostClassifier(base_estimator=shallow_tree, n_estimators = n_est, random_stat

    ABC.fit(X_train, y_train)
    y_pred = ABC.predict(X_test)
    score = metrics.accuracy_score(y_test, y_pred)
    abc_scores.append(score)
```

In [15]:

```
plt.plot(estimators, abc_scores)
plt.xlabel('n_estimators')
plt.ylabel('accuracy')
plt.ylim([0.85, 1])
plt.show()
```



Summary

In this session, we learnt the intuition behind boosting and studied in detail the AdaBoost algorithm.

- Adaboost starts with a uniform distribution of weights over training examples.
- These weights tell the importance of the data point being considered.
- We first start with a weak learner $h_1(x)$ to create the initial prediction.
- Patterns which are not captured by previous models become the goal for the next model by giving more weightage.
- The next model(weak learner) trains on this resampled data to create the next prediction.
- In the end, we make a weighted sum of all the linear classifiers to make a strong classifier.

Additional Content - Soft Skills:

Navigate here to access the soft skills content related to Job Interviews.

Question 1/1 Mandatory

Key Takeaways

What are your 3 key takeaways from this session?

a weak learner h1(x) to create the initial prediction.
Patterns which are not captured by previous models become the goal for the next model by giving more weightage.
The next model(weak learner) trains on this resampled data to create the next prediction.

Attempt 1 of 1 Continue

Report an error

PREVIOUS
AdaBoost Lab

FINISH SESSION
Summary