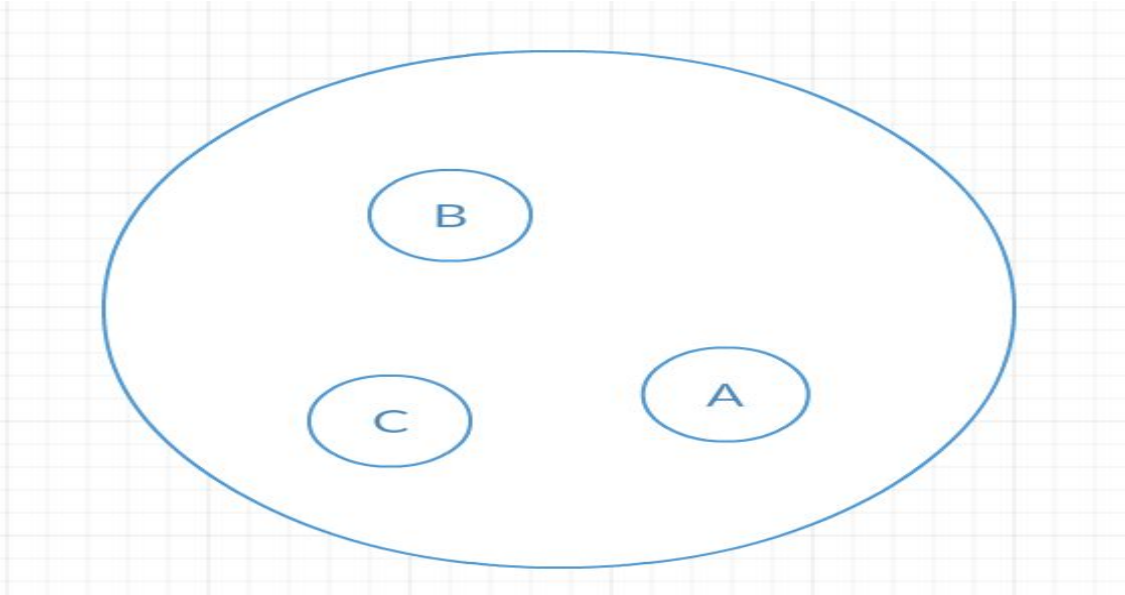
What is Boosting ?

Boosting is a method of converting a set of weak learners into strong learners.

Suppose we have a binary classification task. A weak learner has an error rate that is slightly lesser than 0.5 in classifying the object, i.e the weak learner is slightly better than deciding from a coin toss. A strong learner has an error rate closer to 0. To convert a weak learner into strong learner, we take a family of weak learners, combine them and vote. This turns this family of weak learners into strong learners

The idea here is that the family of weak learners should have a minimum correlation between them.



Here, let A, B and C be different classifiers. Their area A represents where the classifier A misclassifies(goes wrong) and area B represents where the classifier B misclassifies and area C represents where the classifier C misclassifies. Since there is no correlation between the errors of each classifier, combining them and using a technique of democratic voting to classify each object, this family of classifiers will never go wrong. I guess this would’ve provided a basic understanding of boosting.

The different types of boosting algorithms are:

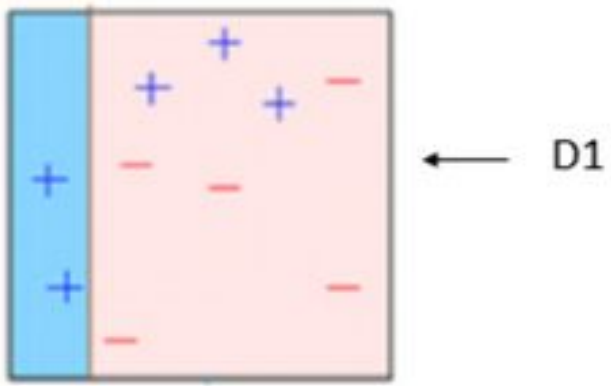
- AdaBoost
- Gradient Boosting
- XGBoost

AdaBoost(Adaptive Boosting)

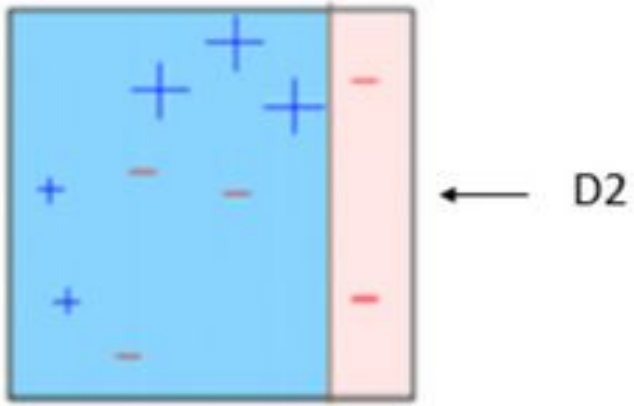
The Adaptive Boosting technique was formulated by Yoav Freund and Robert Schapire, AdaBoost works on improving the areas where the base learner fails. The base learner is a machine learning algorithm which is a weak learner and upon which the boosting method is applied to turn it into a strong learner.

We take the training data and randomly sample points from this data and apply decision stump algorithm to classify the points. After classifying the sampled points we fit the decision tree stump to the complete training data. This process iteratively happens until the complete training data fits without any error or until a specified maximum number of estimators.

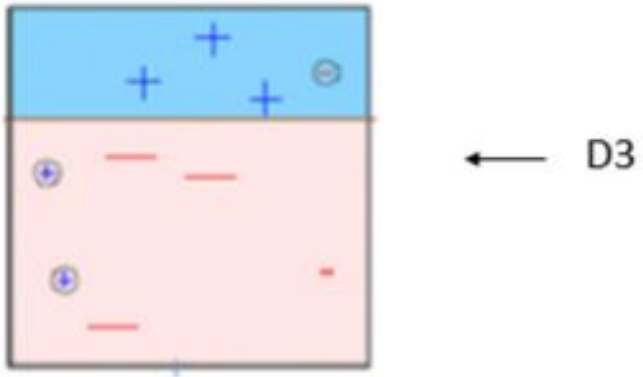
After sampling from training data and applying the decision stump, the model fits as showcased below.



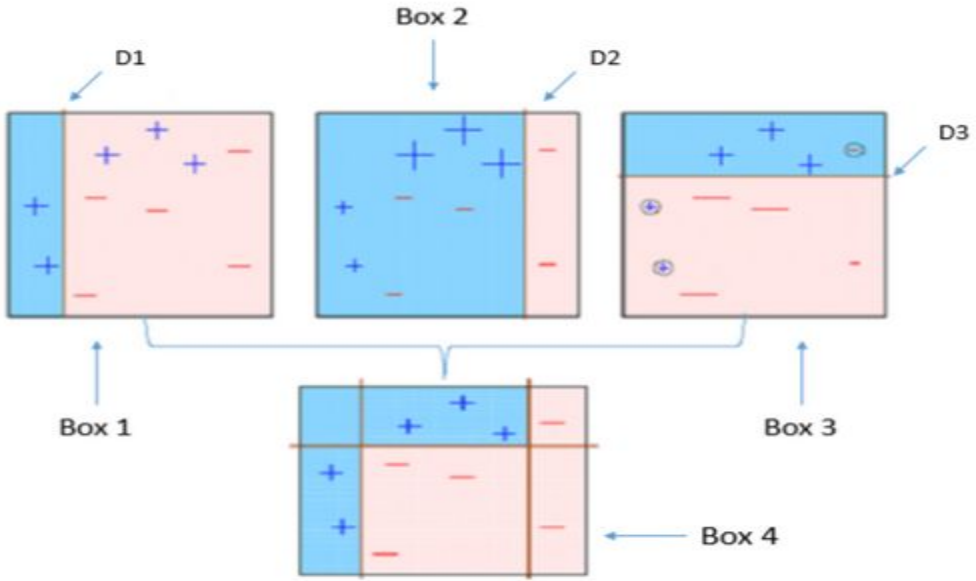
We can observe that three of the positive samples are misclassified as negative. Therefore, we exaggerate the weights of these misclassified samples so that that they have a better chance of being selected when sampled again.We can observe that three of the positive samples are misclassified as negative. Therefore, we exaggerate the weights of these misclassified samples so that that they have a better chance of being selected when sampled again.



When data is sampled next time, the decision stump 2 is combined with decision stump 1 to fit the training data. Therefore we have a miniature ensemble here trying to fit the data perfectly. This miniature ensemble of two decision stumps misclassifies three negative samples as positive. Therefore, we exaggerate the weights of these misclassified samples so that that they have a better chance of being selected when sampled again.



The previously misclassified samples are chosen and decision stump 3 is applied to fit the training data. We can find that two positive samples are classified as negative and one negative sample is classified as positive. Then the ensemble of three decision stumps(1, 2 and 3) are used to fit the complete training data. When this ensemble of three decision stumps are used the model fits the training data perfectly.



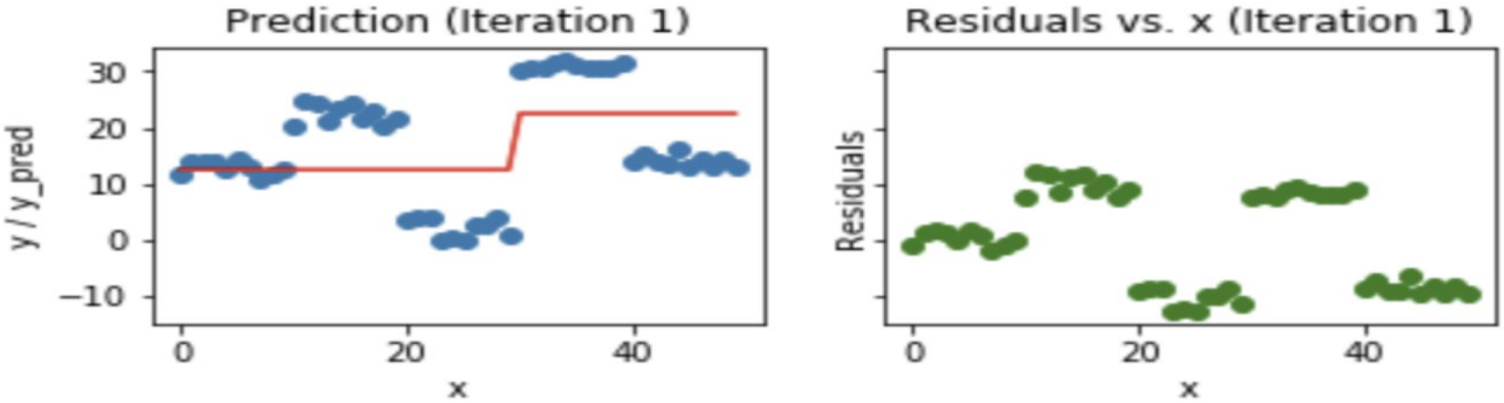
The drawback of AdaBoost is that it is easily defeated by noisy data, the efficiency of the algorithm is highly affected by outliers as the algorithm tries to fit every point perfectly.

Gradient Boosting:

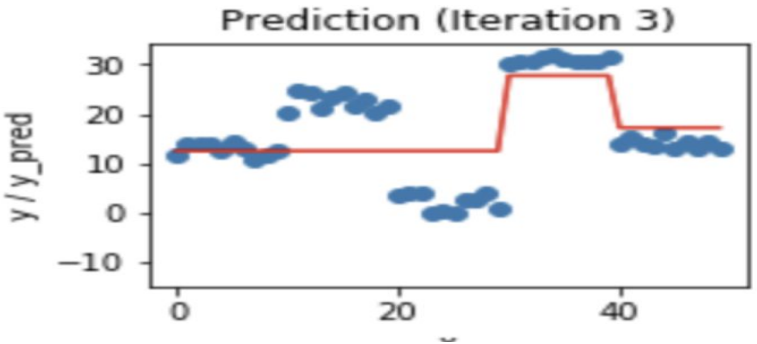
Gradient Boosting is also a boosting algorithm hence it also tries to create a strong learner from an ensemble of weak learners. This is algorithm is similar to Adaptive Boosting(AdaBoost) but differs from it on certain aspects. In this method we try to visualise the boosting problem as an optimisation problem, i.e we take up a loss function and try to optimise it. This idea was first developed by Leo Breiman.

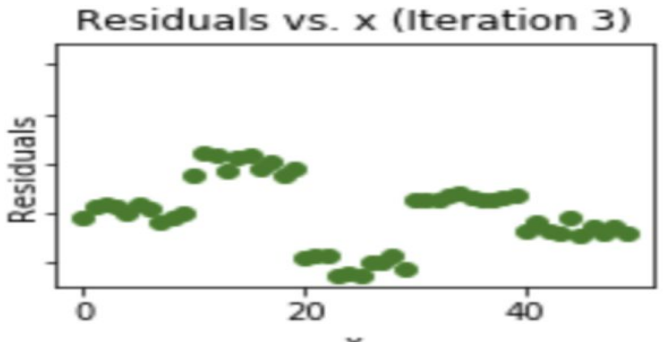
How is Gradient Boosting interpreted as an optimisation problem?

We take up a weak learner(in previous case it was decision stump) and at each step, we add another weak learner to increase the performance and build a strong learner. This reduces the loss of the loss function. We iteratively add each model and compute the loss. The loss represents the error residuals(the difference between actual value and predicted value) and using this loss value the predictions are updated to minimise the residuals.

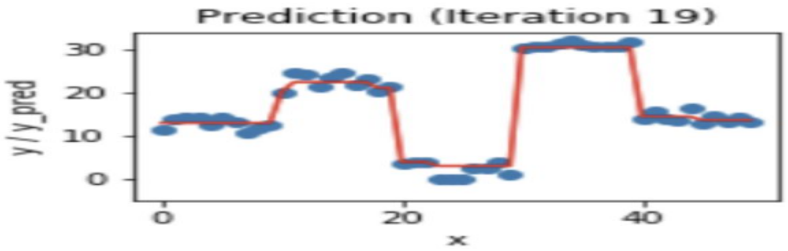


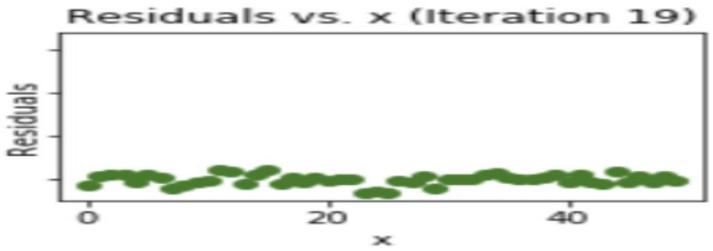
In the first iteration, we take a simple model and try to fit the complete data. You can from the above image that the prediction values of the model of the ground truth are different. The error residuals are plotted on the right side of the image. The loss function is trying to reduce these error residuals by adding more weak learners. The new weak learners are added to concentrate on the areas where the existing learners are performing poorly.

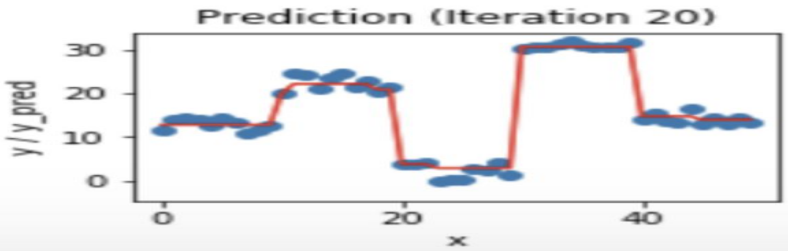


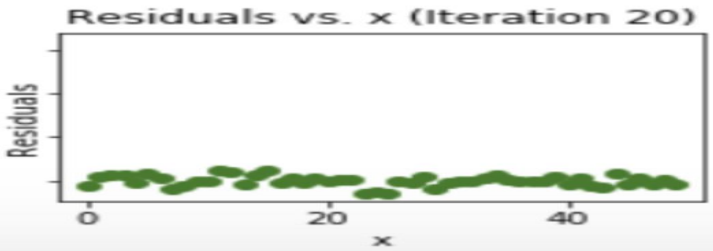


After three iterations, you can observe that model is able to fit the data better. This process is iteratively carried out until the residuals are zero.









XGBoost(Extreme Gradient Boosting):

Features of XGBoost are:

- Clever Penalisation of Trees
- A Proportional shrinking of leaf nodes
- Newton Boosting
- Extra Randomisation Parameter

In XGBoost the trees can have a varying number of terminal nodes and left weights of the trees that are calculated with less evidence is shrunk more heavily. Newton Boosting uses Newton-Raphson method of approximations which provides a direct route to the minima than gradient descent. The extra randomisation parameter can be used to reduce the correlation between the trees,the lesser the correlation among classifiers, the better our ensemble of classifiers will turn out. Generally, XGBoost is faster than gradient boosting but gradient boosting has a wide range of application