Hey there, this is prem Swaroop.

Summary of XG Boost with examples

Boosting - Boosting is a process that uses set of machine learning algorithms to combine weak learner to form strong learners in order to increase the accuracy of the model.

Under Ensemble learning we have types – we have

i) Sequential ensemble learning - boosting
ii) Parallel ensemble learning – bagging

Boosting – Here the weak learners are sequentially produced during the training phase.

The performance of the model is improved by assigning a higher weightage to the previously incorrectly classified samples.

In boosting we feed the entire dataset into the algorithm and the algorithm will make some predictions

Let's say the algorithm misclassified some of your data

Now what we do is we pay more attention to the misclassified data points

We increase the weightage of misclassified datapoints

We keep doing this until all our wrongfully predicted or misclassified samples are correctly predicted.

The basic principle here is to generate multiple weak learners and combine their predictions to form one strong rule.

Now these weak learners are generated by applying base machine learning algorithms on different distributions of the dataset.

These base machine learning algorithms are usually decision trees by default in a boosting algorithm.

So what these base learners do is they generate weak rules for each iteration

After multiple iterations, the weak learners are combined and they form a strong learner that will predict a more accurate outcome.

XGBoost – Extreme Gradient Boosting

Boosting

- Boosting builds models from individual 'weak learners' in an iterative way.

First of all a weak learner

For ex: we have a decision tree

For ex: would you like to take a job offer

- Well is it above certain salary – yes
- Well is it close to work – yes

Well do they offer great benefits – no

So probably u don't want to take that job.

So this is basically a decision tree

Well these are just 3 features we talked about

In reality we may have 10's or 100's of features for a given dataset.

So an individual weak learner or a tree is not enough

You put multiple trees together, this is where random forest comes into picture.

So in random forest we are building multiple trees but while building it all the features are not available

We are taking only subset of features to build a tree

Boosting builds (very similar to random forest) models from individual weak learners

Now unlike random forests , in boosting these models are not completely built on random subsets

Instead they sequentially more weights on instances with wrong predictions this is just

****<u>This is just like us , if we want to get better we look at where we fail and we try to fix it and we get better in life and at work or anywhere</u> ****

So that's exactly what actually does

Based on weak learners it says ok now I know why you have failed and then it gets better

So, it puts more emphasis on successful models and of course on the wrong predictions and it learns from this - that's what boosting is.

And coming to term Gradient Boosting - The name gradient comes from the fact that it uses Gradient Descent Algorithm (The same thing deep learning uses to find the minima in the loss function)

XG Boost is related to Boosting - it's an advanced version of boosting aka optimized version of boosting

It was developed in 2016 by University of Washington

Not every dataset requires deep learning , in fact before thinking of deep learning think of traditional machine learning

XG boost performs equally well if not outperform certain deep learning algorithms

And for very complicated cases where we have lot of annotated data deep learning probably a good approach.

But XG does an amazing job most of the time.

i) Q. What is a Gradient?

A gradient is a derivative. What does it tell u?

For a straight line it gives u the slope, or for any curve it gives u the slope at that specific point

What does that mean - Well it just tells u the slope

Now the 2nd order tells u with the direction of that slope

So in gradient descent if u r trying to find the minima – so if u r heading the wrong direction

With 2nd order gradient u already know that that's not the right direction

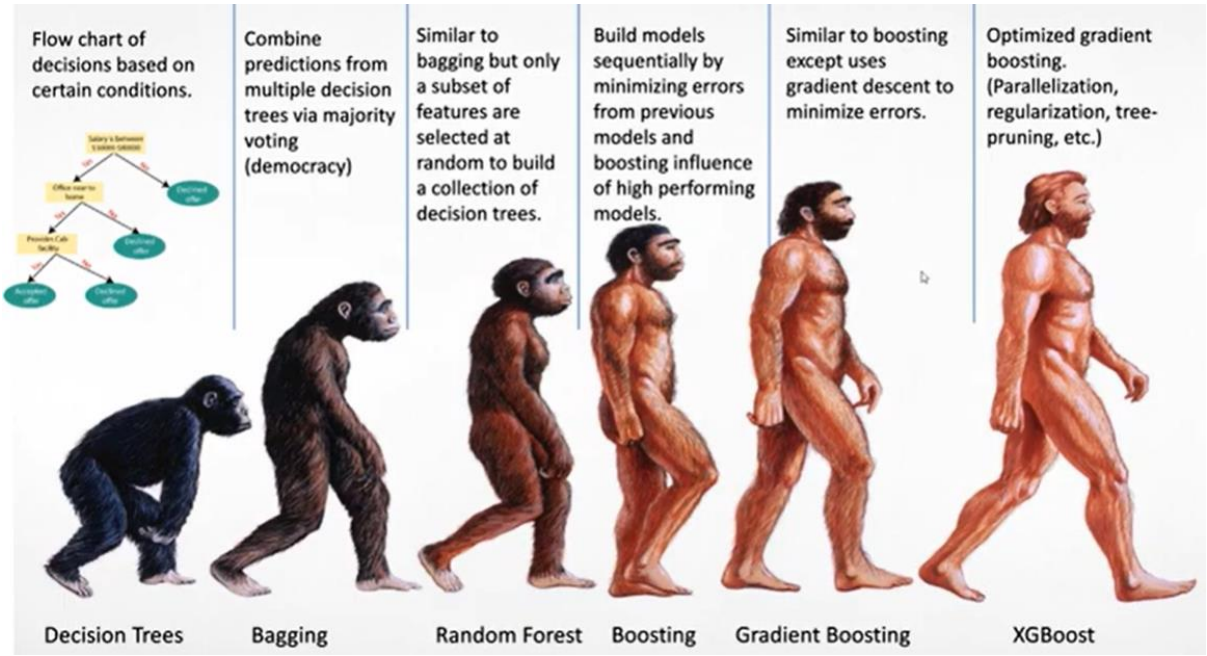
U have to go to the opposite direction

So this is a bit optimized by using the 2nd order gradient where it figures out the right direction to move

ii) It also uses advanced L1 and L2 regularization.
Again regularization prevents overfitting
In deep learning for ex. We do dropout as one of our regularization methods .
So here it uses L1 and L2 to prevent this overfitting
And random forest can depending on the number of trees and everything u can actually get a bit overfitting and L1 and L2 are built into XG Boost so that's taken care of.

iii) And finally parallelization is the best aspect of this. Its superfast
When we run this we can fire up our processors to see how all the processors are being efficiently used and that's why it is pretty fast

**** So its basically an optimized boosting algorithm ****



Starting with Decision tree all the way to XG Boost is really a true evolution

➔ **The decision tree is just a basic flowchart**

To use the same example, we used earlier,

Is the salary above certain thing – yes take the job

Is it near to the home - yes take the job

So this is basically a decision tree it works fine for these type of simple scenarios

But what if we have many features in a complicated example

This is where we get bagging where we combine the predictions from multiple decision trees

For ex: most of we are used to arranged marriages in India

Ur parents are looking for a groom or a bride and there are many criteria right okay

So the criteria is - do they work – what is the educational qualification – how much do they earn

-are they good looking – what is the height – what is the weight

There are so many parameters

Decision tree is basically okay is the salary above this, is this above that , is this above this

But often times life is not that simple – you cannot say salary is good and this is good and this is good

There are other attributes that different people look at

Probably your mom looks at lot more on a different side of the story than your dad. So, in this example u combine the prediction

➔ **bagging is bringing all the family members together. they all look at different attributes and they say based on that I vote for this girl and based on this I vote**

So this is what happens in bagging

So here what happens is bagging is basically all ur family members are going to vote and then u get 20 votes for candidate one, 10 votes for candidate two, and 3 votes for candidate three

And then so the candidate one wins - this is basically democracy that's exactly what bagging is.

It combines the predictions from multiple decision trees.

➔ **Random forest is very similar except it takes a subset of features**

so instead of saying height, weight, salary and job and all of these let's just take only three of these features and then vote.

And other family members take random three features and vote

So, it takes a subset of these and then voting it and again you're basically looking at this majority voting so that's what random forest is

So, this solves overfitting a little bit but u will still have some overfitting when u do random forest

➔ **Now boosting unlike random forest it's not going to take this random subset**

it actually does sequentially minimize the errors from previous model.

If we take the arrange marriage example think of this as ur uncle votes for something, your mom goes to ur uncle and says hey why did u vote for this and then learns from that and then based on that mom modifies her vote.

Basically, our model is influenced by the past models and then it boosts on high performing models, so if there is a model that's high performing u get that

So that's where the boosting term comes into picture

➔ **Gradient boosting is very similar to boosting except for the loss function it uses the gradient descent. That's pretty much it**

➔ **AND XG Boost is the most evolved version of this algorithm**
Which is it's optimized for both speed and also accuracy bcz of parallelization, regularization etc.