Today we will be exploring XGBoost from a versatility standpoint and explaining why it can be used for both regression as well as classification.

Looking at the agenda I will provide an overview of XGB, talk about how it works in regression, how it works for classification then I will breakdown the mathematics behind the algorithm

For background XGB was greated by tea-on-key chen at while attending the university of Washington as a masters student

As of 2022 he works as an assistant professor in the machine learning department at Carnegie mellon university

XGB is a tree based algrethm. As we can see here it has built upon other tree based algrethms from the past. One distinct feature of xgb is its aggressive puning of trees helps avoid overfitting

Regression

First off xgb is meant for large and complex data. However, for visual sake we will be using a simple data set made up of 4 points

For starters we will graph the data

Step one for xgb is to make an initial prediction of where the datapoints that is represented by this red dotted line

It then calculates the residules which is the distance between the guess and the actual value these are represented by the black dotted lines

Then a tree will start with one leaf

A similarity score is calculated for the leaf which is sum of residules squared over the number of residules plus lamda

Lamda is an important varible for reducing overfitting by lowering the similarity score by increasing denominator in the similarity score. Lamda helps account for leaves with a low quatity of residules in it. Lamda can equal 0

After the similarity score is calculated for one leaf we then create more leaves by splitting the data

This split is represented by the green line

We then calculate the similarity score for the new leaves

Once all the leaves are created and we calculate the simulairy score for each we then calculate gain for each of our leaves. This score will help us determine overfitting

Gain is = similarity left +similarity right – the root (score at the top of the two leafs)

This helps us determine weather or not we gain anything from splitting our data and going down a level in the tree

next we prune the tree starting with the lowest leaf.

Note – the fariable of depth can be important for xgb at it determines when the algorithm stops making leaves thus controlling the lowest leaf values

A branch is removed if the root gain is less than the gamma starting at the lowest branch meaning we did not gain enough from having this branch based on the hyper parameters passed into our algorithm

Gamma is another important hyperparameter that shows the miniumn quality that a branch must have to stay in the model

After the trees are purned we calculate the Output value for the leaves to give us new predictions and new residlus

With that we can calculate the new prediction for each value

output value equals = sum of residules divided by number of residules plus lamda

new preditciton = original prediction + learning rate(default .3) times the output value

now that we have our new predictions we repeat this process over and over again with our new residules until we reach our max iteration or come up with a extremely low residule

I would like to highlight that Models like xgb show the importance of selecting hyperparameters in order to get the most out of the model

Classification

We will speed through classification since it is a very similar process to regression

Once again it starts with the innital prediction represtned by the red line with our residules represented by the black line

By default xgb will select .5 since that is between 1 and 0 for our binary classification

We will calculate the simulairy score where the top of the equation is the same however the bottom has changed to pervious probablty \* ( 1- previous prob) + lamda (used for overfitting) where once again lamda helps with overfitting

We start with one leaf and calculate the similarity scores then move to splitting the data into multiple leaves

We then calculate gain however we must be aware of a varible called cover in regression it is by default equal to 1 that is why we didn’t cover it

Cover is the minimum number of residules allowed in a leaf

Once we have simulariry scores for each of the leaves we can calulate the output values which is…

Since we are doing classification we now need to use the log odds where regression didn’t need to use log odds

Log odds equal

Next we add log odds to our original prediction + the learning reate at .3 \* the output value

This gives us our new guess for each point

Repeat this process until we have enough iterations or our residule gets too small

Math: