

Gradient boosting from scratch



Course: ALY 6020 Predictive Analytics

Instructor: Dr. Marco Montes de Oca

Submitted by: Nithish Saravanan

Decision tree is one of the category of supervised machine learning that can be used on all continuous and categorical data where a sample or population will be subdivided into partitions based upon the most outstanding partition parameter.

Boosting a decision tree is a phenomenon in which we try to reduce the error and increase the accuracy with help of control parameters.

The given function for the assignment is :

f(x) =0.8cos(3.2πx) + 0.64cos(10.24πx) + 0.51cos(32.77πx)

The first step is to generate the data set from the given function other parameters.

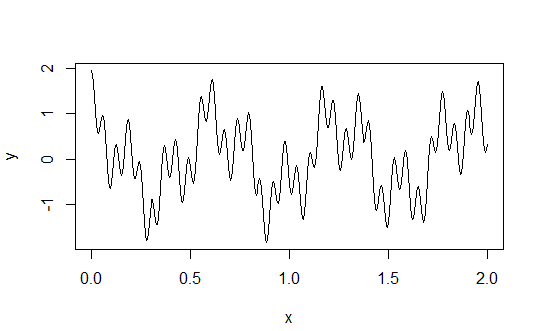
> #generating dataframe from function

> x<-seq(0,2,0.005)

> y<- 0.8\*cos(3.2\*3.14\*x) + 0.64\*cos(10.24\*3.14\*x) + 0.51\*cos(32.77\*3.14\*x)

> df <- data.frame(x,y)

Now the data frame is created for performing further calculation that is required to get the expected outcome. We plot the data frame to get the actual function.



The next step is to create the first regression tree to get the predicted values.

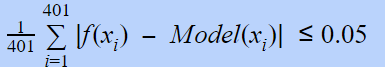
> df$pred <- predict(tree, data = df$x)

> p1<-df$pred

> error <- sum(abs(df$y - df$pred))

> abs\_error <- c(abs\_error,error)

Now we have to gradient boost the given values based upon the accuracy by



This equation will help us to minimize the error values when we try implementing the gradient boosting.

Now we need to implement step wise gradient boosting.

> while(error > 20.05){

+ df$res <- df$y - df$pred

+ error\_table[,paste0("Error after ",increment\_counter," tree",sep="")] <- df$res

+

+ # Incrementing the counter by 1

+ increment\_counter <-increment\_counter + 1

+ #print(df$res)

+ tree1 <- rpart(df$res~df$x, data = df, control = rpart.control(cp=0.00000000000001))

+ df$pred1 <- predict(tree1, data = df$x)

+ df$pred <- df$pred + alpha\*df$pred1

+ error <- sum(abs(df$y - df$pred)) #accuracy function

+

+ abs\_error <- c(abs\_error, error)

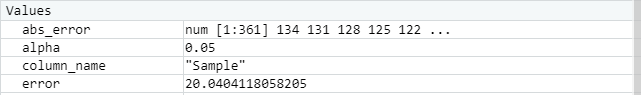
+ p2 <- rbind(p2,df$pred)

+

+ }

Since our accuracy is given to be <=0.05 with the equation when we bring the divisor value of 401 and multiply with 0.05 we tend to get the value 20.05 where the error stops when it is less than the mentioned value.

Upon execution of the loop we get.



Now we can plot the predicted value with the actual equation and boosted values to verify

> colnames(p2) <- seq(1,nrow(df))

> plot(df$x, df$y, type = 'line', col="Green", lwd=2)#Final Prediction Plot

> lines(df$x, p1, col="Red", lwd=2)#Initial Prediction plot

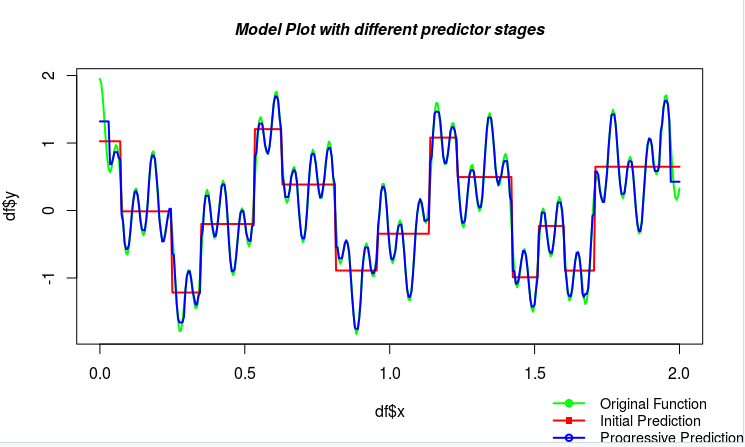
> lines(df$x, p2[359,], col="Blue", lwd=2)#Progressive Prediction plot

> legend(1.55, -2.6, legend = c("Original Function","First Prediction", "Prediction at Different Level"),

+ bty = "n", xpd=TRUE, lwd=c(2,2), col=c("green","red", "blue"), cex = 0.90, pch = c(10, 15, 1))

>

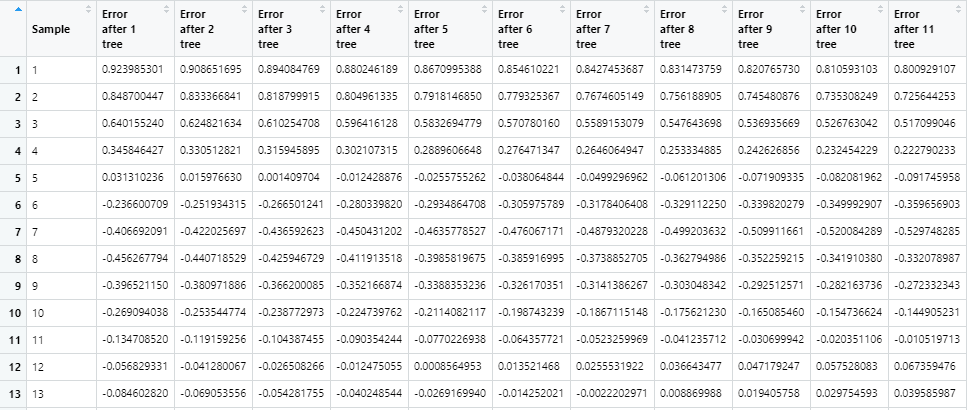
> title(main='Model Plot with different predictor stages', cex.main = 1, font.main= 4, col.main= "black", outer = FALSE)



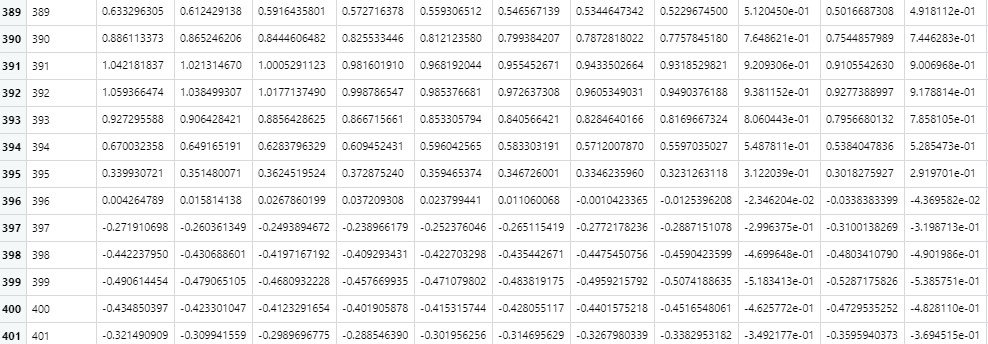
From the plot we see that

* The red line indicates the plot of our model with our initial predictor.
* The blue line indicates the plot of our model with a progressive predictor.
* The green line indicates the plot of our original function.

The error table for the trees are as below:



.

.

Boosting has helped us to understand how the model can predict the function how it is supposed to be mathematically than a single predictor model. Accuracy of any function can be attained with the help of boosting. While changing the learning parameters we were able to find that the error values remained the same.