

KDnuggets

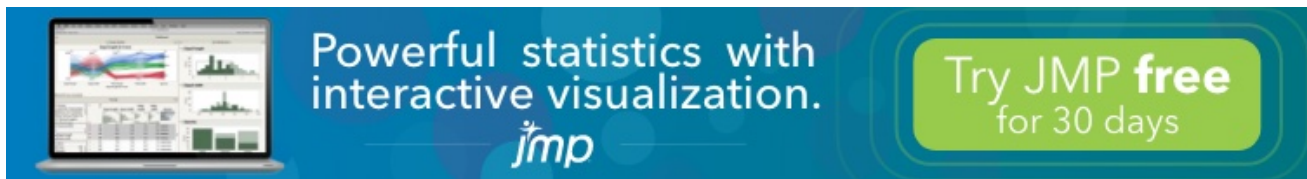
[Subscribe to KDnuggets](#)



[Submit a blog](#)
[Win a Reward!](#)



- [Blog](#)
- [Opinions](#)
- [Tutorials](#)
- [Top stories](#)
- [Courses](#)
- [Datasets](#)
- [Education: Online](#)
- [Certificates](#)
- [Events / Meetings](#)
- [Jobs](#)
- [Software](#)
- [Webinars](#)



[Powerful Statistics with interactive visualization. Try JMP free for 30 days](#)

[Topics:](#) [AI](#) | [Data Science](#) | [Data Visualization](#) | [Deep Learning](#) | [Machine Learning](#) | [NLP](#) | [Python](#) | [R](#) | [Statistics](#)

[KDnuggets Home](#) » [News](#) » [2018](#) » [Aug](#) » [Tutorials, Overviews](#) » Unveiling Mathematics Behind XGBoost ([18:n31](#))

Unveiling Mathematics Behind XGBoost

[<= Previous post](#)

[Next post =>](#)



Like 121

Share 121

Tweet

Share

Share

88

Tags: [Gradient Boosting](#), [Mathematics](#), [XGBoost](#)

Follow me till the end, and I assure you will atleast get a sense of what is happening underneath the revolutionary machine learning model.



[KNIME](#)
[Intro to](#)
[Deep Learning](#)
[Apr 26-30](#)
[Register Now](#)

 [comments](#)

By [Ajit Samudrala](#), Aspiring Data Scientist

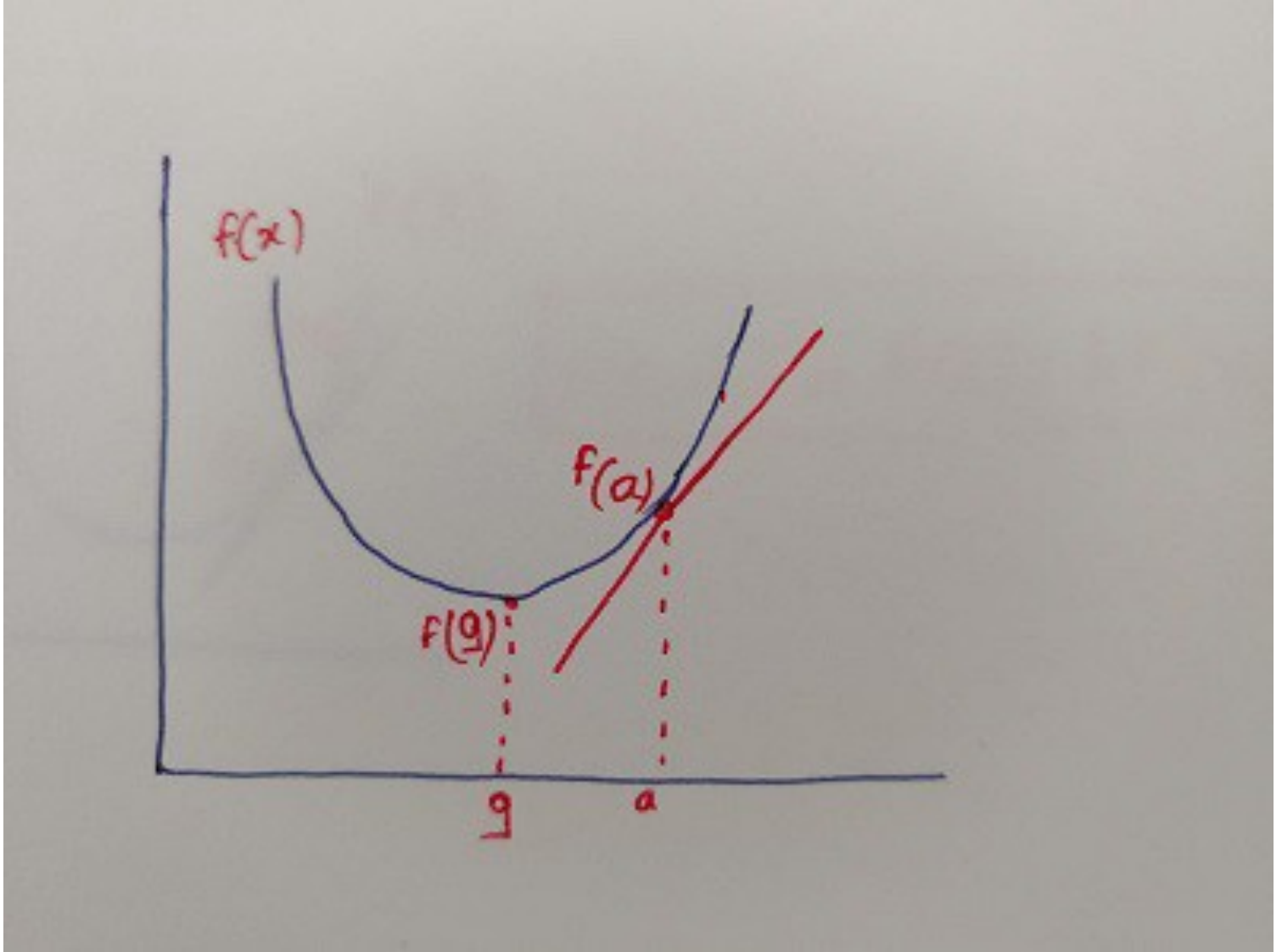
This article is targeted at people who found it difficult to grasp the original paper. Follow me till the end, and I assure you will at least get a sense of what is happening underneath the revolutionary machine learning model. Lets get started.

Curious case of John and Emily



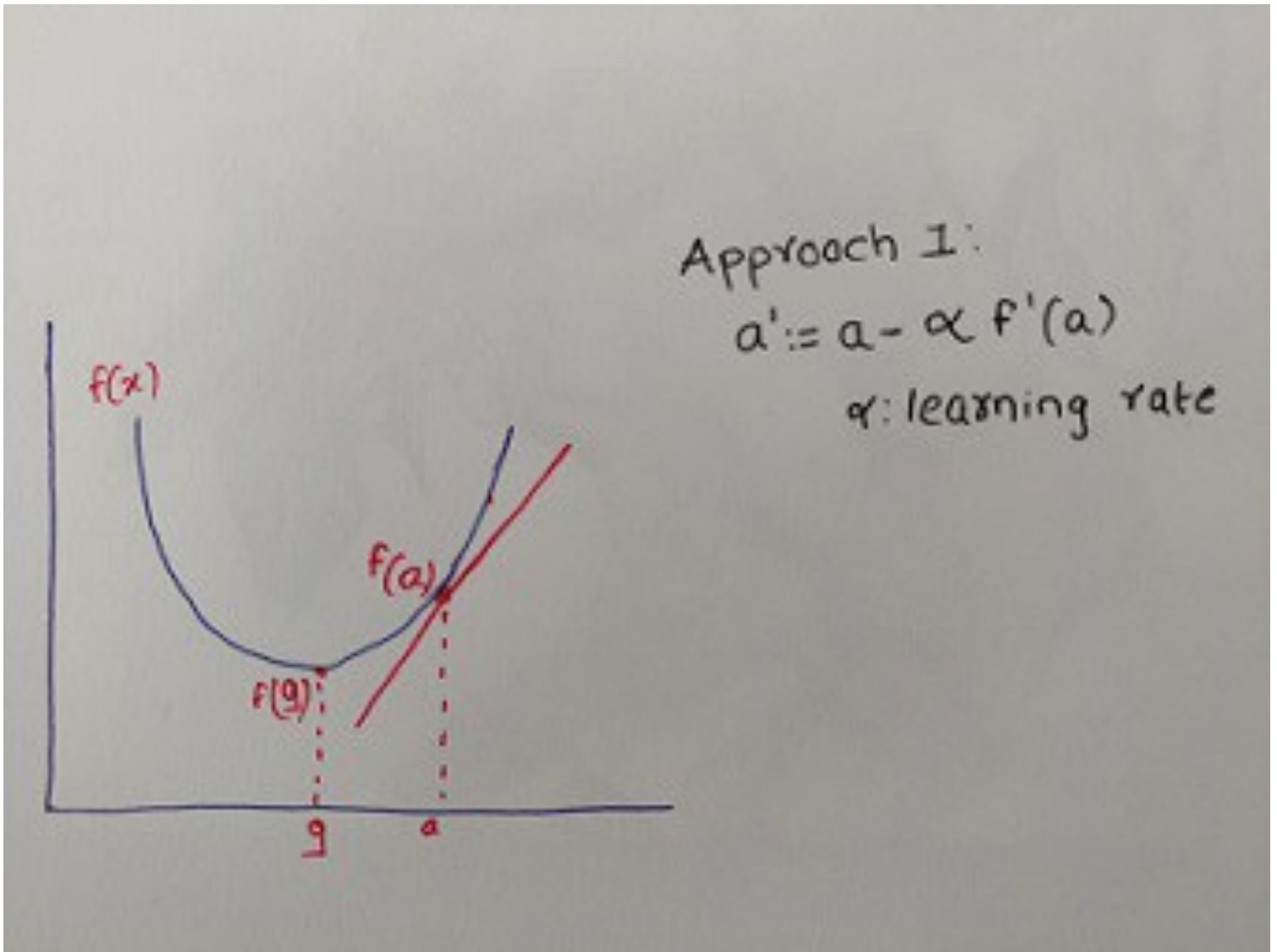
John and Emily embarked on an adventure to pluck apples from the tree located at the lowest point of the mountain. While Emily is a smart and

adventurous girl, John is a bit cautious and sluggish. However, only John can climb tree and pluck apples. We will see how they achieve their goal.



John and Emily start at point 'a' on the mountain and tree is located at point 'g'. There are two ways they can reach point 'g':

1. Emily calculates the slope of the curve at point 'a'; if the slope is positive, they move in the opposite direction or else in the same direction. However, slope gives direction but doesn't tell how much they need to move in that direction. Therefore, Emily decides to take small step and repeat the process, so they don't end up at the wrong point. While the direction of the step is controlled by the negative gradient, the magnitude of the step is controlled by the learning rate. If the magnitude is either too high or too low, they may end up at a wrong point or take way too long to reach to point 'g'.



John is skeptical about this approach and doesn't want to walk more if by chance they end up at the wrong point. So Emily comes up with a new approach:

2. Emily suggests that she will first go to next potential points and calculate the value of the loss function at those points. She keeps a note of the value of the function at all those points, and she will signal John to come to that point where the value is least. John loved this approach as he will never end up at a wrong point. However, poor Emily need to explore all points in her neighborhood and calculate the value of the function at all those points. The beauty of XGBoost is it intelligently tackles both these problems.

Please be advised I may use function/base learner/tree interchangeably.

Gradient Boosting

Many implementations of Gradient Boosting follow approach 1 to minimize the objective function. At every iteration, we fit a base learner to the negative gradient of the loss function and multiply our prediction with a constant and add it to the value from previous iteration.

Input: training set $\{(x_i, y_i)\}_{i=1}^n$, a differentiable loss function $L(y, F(x))$, number of iterations M .

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma).$$

2. For $m = 1$ to M :

1. Compute so-called *pseudo-residuals*:

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \dots, n.$$

2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$.
3. Compute multiplier γ_m by solving the following *one-dimensional optimization* problem:

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

[Source: Wikipedia](#)

The intuition is by fitting a base learner to the negative gradient at each iteration is essentially performing gradient descent on the loss function. The negative gradients are often called as pseudo residuals, as they indirectly help us to minimize the objective function.

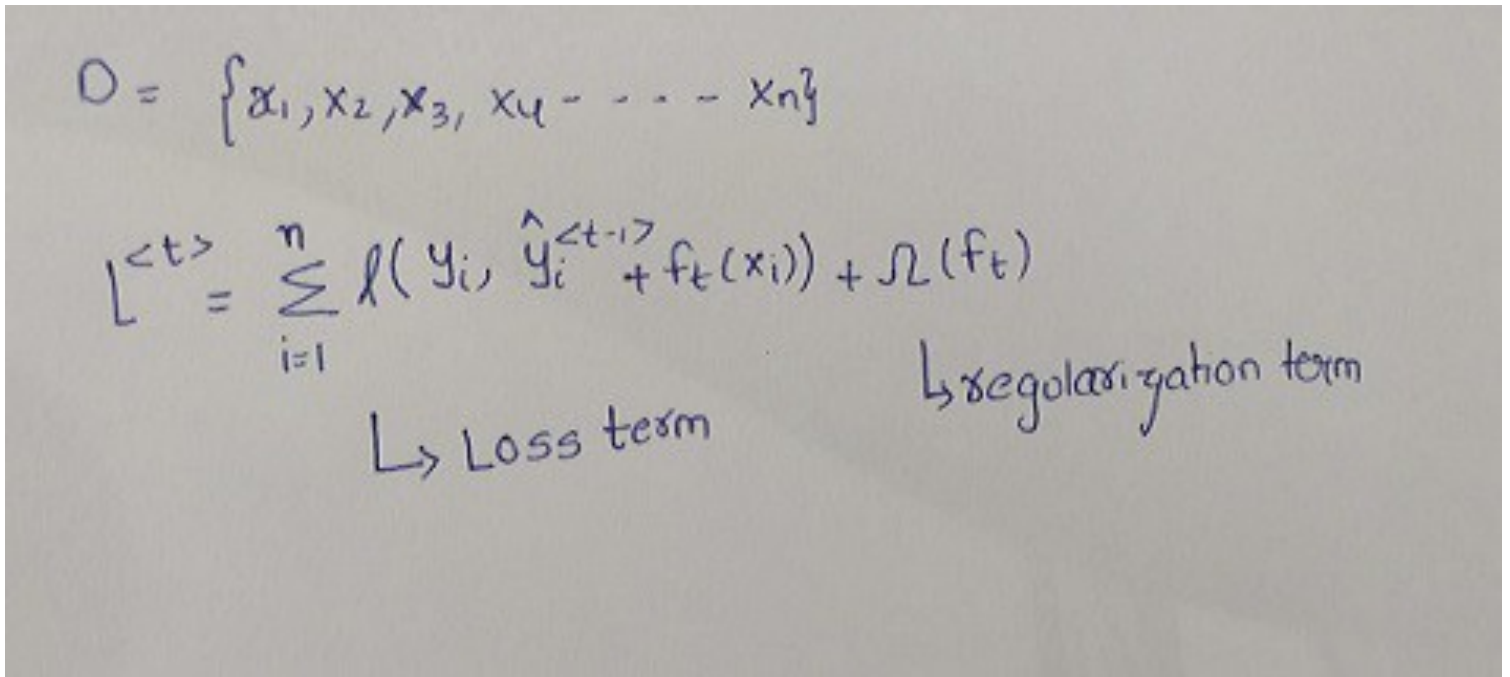
XGBoost

It was a result of research by Tianqi Chen, Ph.D. student at University of Washington. XGBoost is an ensemble additive model that is composed of several base learners.

$$F = \{f_1, f_2, f_3, f_4, \dots, f_m\} \text{ set of base learners}$$

Final Prediction: $\hat{y}_i = \sum_{t=1}^m f_t(x_i)$

How should we choose a function at each iteration? we choose a function that minimizes the overall loss.



$$D = \{x_1, x_2, x_3, x_4, \dots, x_n\}$$

$$L^{<t>} = \sum_{i=1}^n l(y_i, \hat{y}_i^{<t-1>} + f_t(x_i)) + \Omega(f_t)$$

\hookrightarrow Loss term
 \hookrightarrow regularization term

In the gradient boosting algorithm stated above, we obtained $f_t(x_i)$ at each iteration by fitting a base learner to the negative gradient of loss function with respect to previous iteration's value. In XGBoost, we explore several base learners or functions and pick a function that minimizes the loss (Emily's second approach). As I stated above, there are two problems with this approach:

1. exploring different base learners
2. calculating the value of the loss function for all those base learners.

XGBoost uses Taylor series to approximate the value of the loss function for a base learner $f_t(x_i)$, thus, reducing the load on Emily to calculate the exact loss for different possible base learners.

Taylor Expansion

$$f(a+h) = f(a) + f'(a)h + \frac{1}{2} f''(a)h^2 + \dots + \frac{f^{(n)}(a)}{n!}h^n$$

Here $a = \hat{y}_i^{(t-1)}$
 $h = f_t(x_i)$
 $f(a) = \ell(y_i, \hat{y}_i^{(t-1)})$

$$\therefore L^{(t)} = \sum_{i=1}^n \ell(y_i, \hat{y}_i^{(t-1)}) + \left(\frac{\partial \ell(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \right) f_t(x_i) + \left(\frac{\partial^2 \ell(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)^2}} \right) f_t(x_i)^2 + \dots$$

$\ell(y_i, \hat{y}_i^{(t-1)})$ is constant irrespective of any function

$$\therefore L^{(t)} = \sum_{i=1}^n (C + g_i f_t(x_i) + h_i f_t(x_i)^2) + \Omega(f_t)$$

Pick $f_t(x_i) \Rightarrow L^{(t)}$ is minimum. Removing constant as it is equal for any function.

$$L^{(t)} = \sum_{i=1}^n (g_i f_t(x_i) + h_i f_t(x_i)^2) + \Omega(f_t)$$

It only uses expansion up to second order derivative assuming this level of approximation would be suffice. The first term 'C' is constant irrespective of any $f_t(x_i)$ we pick. ' g_i ' is the first order derivative of loss at previous iteration with respect predictions at previous iteration. ' h_i ' is the second order derivative of loss at previous iteration with respect to predictions at previous iteration. Therefore, Emily can calculate ' g_i ' and ' h_i ' before starting exploring different base learners, as it will be just a matter of multiplications thereafter. Plug and play, isn't it?

So XGBoost has reduced one of the burdens on Emily. Still we have a problem of exploring different base learners.

$$L^{(t)} = \sum_{i=1}^n (g_i f_t(x_i) + h_i f_t(x_i)) + \Omega(f_t) \longrightarrow (2)$$

Let f_t has K leaf nodes. I_j be the set of instances belonging to node 'j'. ' w_j ' be the prediction for node 'j'.

$$\Omega(f_t) = \delta K + \frac{1}{2} \lambda \sum_{j=1}^K w_j^2$$

$$L^{(t)} = \sum_{j=1}^K \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \delta K \longrightarrow (3)$$

For each leaf 'j', $\frac{dL^{(t)}}{dw_j^*} = 0$

$$0 = \sum_{i \in I_j} g_i + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) \times w_j^*$$

$$w_j^* = \frac{- \sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

Assume Emily fixed a base learner f_t that has ' K ' leaf nodes. Let I_j be the set of instances belonging to node 'j' and ' w_j ' be the prediction for that node. So for an instance 'i' belonging to I_j , $f_t(x_i) = w_j$. Therefore, we obtained equation 3 from equation 2 by substituting $f_t(x_i)$ with respective predictions. After substituting the predictions, we can take the derivative of loss function with respect to each leaf node's weight, to get an optimal weight.

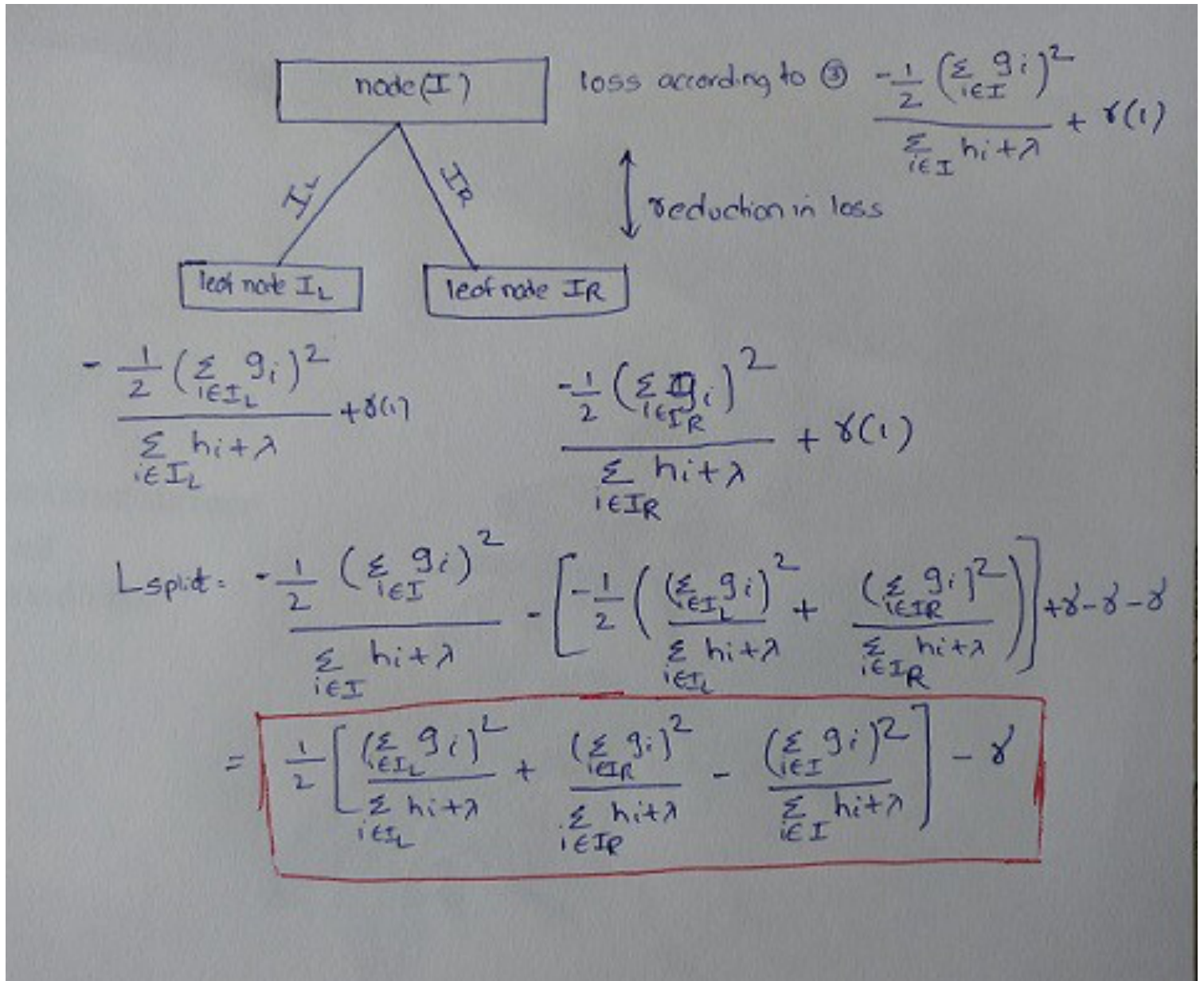
Substituting weights into ③

$$L^{(t)} = -\frac{1}{2} \sum_{j=1}^K \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma K \quad \text{--- ④}$$

This is the best loss for a fixed base learner with 'K' nodes. There will be several hundreds of possible tree structures. It is impossible to explore all of them.

Substituting the optimal weights back into equation 3 gives equation 4. However, this is the optimal loss for a fixed tree structure. There will be several hundreds of possible trees.

Let's formulate the current position of Emily. She now knows how to ease the burden of computing loss using Taylor expansion. She knows for a fixed tree structure what should be the optimal weights in the leaf node. The only thing that is still a concern is how to explore all different possible tree structures.



Instead of exploring all possible tree structures, XGBoost greedily builds a tree. The split that results in maximum loss reduction is chosen. In the above picture, the tree starts at Node 'I'. Based on some split criteria, the node is divided into left and right branches. So some instances fall in left node and other instances fall in right leaf node. Now, we can calculate the reduction in loss and pick the split that causes best reduction in loss.

But still Emily has one more problem. How to choose the split criteria? XGBoost uses different tricks to come up with different split points like histogram, exact, etc. I will not be dealing with that part, as this article has already grown in size.

Bullet Points to Remember

1. While Gradient Boosting follows negative gradients to optimize the loss function, XGBoost uses Taylor expansion to calculate the value of the loss function for different base learners. Here is the best video on the internet that explains [Taylor expansion](#).
2. XGBoost doesn't explore all possible tree structures but builds a tree greedily.
3. XGBoost's regularization term penalizes building complex tree with several leaf nodes.
4. XGBoost has several other tricks under its sleeve like Column subsampling, shrinkage, splitting criteria, etc. I highly recommend continue reading the original paper [here](#).

You can view the complete derivation [here](#).

Read my other article on PCA [here](#).

Bio: [Ajit Samudrala](#) is an aspiring Data Scientist and Data Science intern at SiriusXM.

[Original](#). Reposted with permission.

Related:

- [Intuitive Ensemble Learning Guide with Gradient Boosting](#)
- [Introduction to Python Ensembles](#)
- [CatBoost vs. Light GBM vs. XGBoost](#)

2 Comments KDnuggets  [Disqus' Privacy Policy](#)

 Login ▾

 Recommend 7  Tweet  Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name



Muralidharan Surendran • 2 years ago

Good but needs more explanation.

1. In the loss equation what happened to the term $f(x_i)^2$? it suddenly disappeared after the constant was removed?
2. Why add the regularisation term? not clear and needs explanation?
3. How was the equation for regularisation term arrived at?
4. The loss term after replacing it with the regularisation is not clear at all (it went from $i = 1$ to n to $j = 1$ to k)?

^ | ▾ • Reply • Share ›



Ajit Samudrala → [Muralidharan Surendran](#) • 2 years ago • edited

Thanks for reading the post.

1. In the loss equation what happened to the term $f(x_i)^2$? it suddenly disappeared after the constant was removed?
A.) It was my mistake. It should actually be $f(x_i)^2$ after the constant is removed
- 2.) Why add the regularization term? not clear and needs explanation?
A.) Regularization helps to limit the hypothesis space of your base functions at any iteration. The model prefers a simple function over a complex function at any iteration for same loss.
- 3.) How was the equation for regularization term arrived at?
A.) This term was introduced by the author. Intuitively, it is function of number of leaf nodes and weights assigned to each leaf node, which makes sense as you don't want the model to fit the residuals in a single iteration. You can come up with any other term that you think will make the model more conservative, thus heavily limiting your hypothesis space.
- 4.) The loss term after replacing it with the regularization is not clear at all (it went from $i = 1$ to n to $j = 1$ to k)?
A.) The loss is average across all samples in the data set. While fitting any tree you assign a sample to a leaf node 'K'. So summation over all leaf nodes is nothing but summation over all samples.

Don't hesitate to ask if you still have any questions.

^ | ▾ • Reply • Share ›

 Subscribe  Add Disqus to your siteAdd DisqusAdd  Do Not Sell My Data

[<= Previous post](#)

[Next post =>](#)

Top Stories Past 30 Days

Most Popular

1. [How To Overcome The Fear of Math and Learn Math For Data Science](#)
2. [More Data Science Cheatsheets](#)
3. [How to Succeed in Becoming a Freelance Data Scientist](#)
4. [Top 10 Python Libraries Data Scientists should know in 2021](#)
5. [Are You Still Using Pandas to Process Big Data in 2021? Here are two better options](#)

Most Shared

1. [How To Overcome The Fear of Math and Learn Math For Data Science](#)
2. [How to Succeed in Becoming a Freelance Data Scientist](#)
3. [More Data Science Cheatsheets](#)
4. [Top 10 Python Libraries Data Scientists should know in 2021](#)
5. [A Machine Learning Model Monitoring Checklist: 7 Things to Track](#)

Latest News

- [Interpretable Machine Learning: The Free eBook](#)
- [Deep Learning Recommendation Models \(DLRM\): A Deep Dive](#)
- [Deepfakes are now mainstream. What's next?](#)
- [Can Robots and Humans Combat Extinction Together? Find ...](#)
- [NoSQL Explained: Understanding Key-Value Databases](#)
- [Why machine learning struggles with causality](#)

Top Stories Last Week

Most Popular

1. [Top 10 Python Libraries Data Scientists should know in 2021](#)
2. [Shapash: Making Machine Learning Models Understandable](#)
3. [The 8 Most Common Data Scientists](#)
4. [Easy AutoML in Python](#)
5. [How to Succeed in Becoming a Freelance Data Scientist](#)



Most Shared

1. [Shapash: Making Machine Learning Models Understandable](#)
2. [What's ETL?](#)
3. [Easy AutoML in Python](#)
4. [Deep Learning Is Becoming Overused](#)
5. [The 8 Most Common Data Scientists](#)

More Recent Stories

- [Why machine learning struggles with causality](#)
- [A/B Testing: 7 Common Questions and Answers in Data Science In...](#)
- [Start a Career in a Growing Field with Google's Data Analyti...](#)
- [E-commerce Data Analysis for Sales Strategy Using Python](#)
- [How to Make Sure Your Analysis Actually Gets Used](#)
- [Microsoft Research Trains Neural Networks to Understand What T...](#)
- [KDnuggets 21:n13, Apr 7: Top 10 Python Libraries Data Scien...](#)
- [Working With Time Series Using SQL](#)
- [How Noisy Labels Impact Machine Learning Models](#)
- [KDnuggets Top Blogs Reward Program](#)

- [How to Dockerize Any Machine Learning Application](#)
- [Automated Text Classification with EvalML](#)
- [Top Stories, Mar 29 – Apr 4: Top 10 Python Libraries Dat...](#)
- [The Best Machine Learning Frameworks & Extensions for Ten...](#)
- [How to deploy Machine Learning/Deep Learning models to the web](#)
- [Awesome Tricks And Best Practices From Kaggle](#)
- [One Million KDnuggets Visitors in March. Wow.](#)
- [What did COVID do to all our models?](#)
- [Shapash: Making Machine Learning Models Understandable \[Gold Blog\]](#)
- [What's ETL? \[Silver Blog\]](#)

[KDnuggets Home](#) » [News](#) » [2018](#) » [Aug](#) » [Tutorials, Overviews](#) » Unveiling Mathematics Behind XGBoost ([18:n31](#))

© 2021 KDnuggets. | [About KDnuggets](#) | [Contact](#) | [Privacy policy](#) | [Terms of Service](#)

[Subscribe to KDnuggets News](#)

X