55 (http://l kaggle- master- explain gradien boostin

A Kaggle Master Explains Gradient Boosting

Ben Gorman (http://blog.kaggle.com/author/bengorman/) | 01.23.2017

This tutorial was originally posted here (https://gormanalysis.com/gradient-boosting-explained/) on Ben's blog, GormAnalysis (https://gormanalysis.com/).

If linear regression was a Toyota Camry, then gradient boosting would be a UH-60 Blackhawk Helicopter. A particular implementation of gradient boosting, XGBoost (https://github.com/dmlc/xgboost), is consistently used to win machine learning competitions on Kaggle (https://www.kaggle.com/). Unfortunately many practitioners (including my former self) use it as a black box. It’s also been butchered to death by a host of drive-by data scientists’ blogs. As such, the purpose of this article is to lay the groundwork for classical gradient boosting, intuitively *and* comprehensively.

Motivation

We'll start with a simple example. We want to predict a person's age based on whether they play video games, enjoy gardening, and their preference on wearing hats. Our objective is to minimize squared error. We have these nine training samples to build our model.

PersonID	Age	LikesGardening	PlaysVideoGames	LikesHats
1	13	FALSE	TRUE	TRUE
2	14	FALSE	TRUE	FALSE
3	15	FALSE	TRUE	FALSE
4	25	TRUE	TRUE	TRUE
5	35	FALSE	TRUE	TRUE
6	49	TRUE	FALSE	FALSE
7	68	TRUE	TRUE	TRUE
8	71	TRUE	FALSE	FALSE
9	73	TRUE	FALSE	TRUE

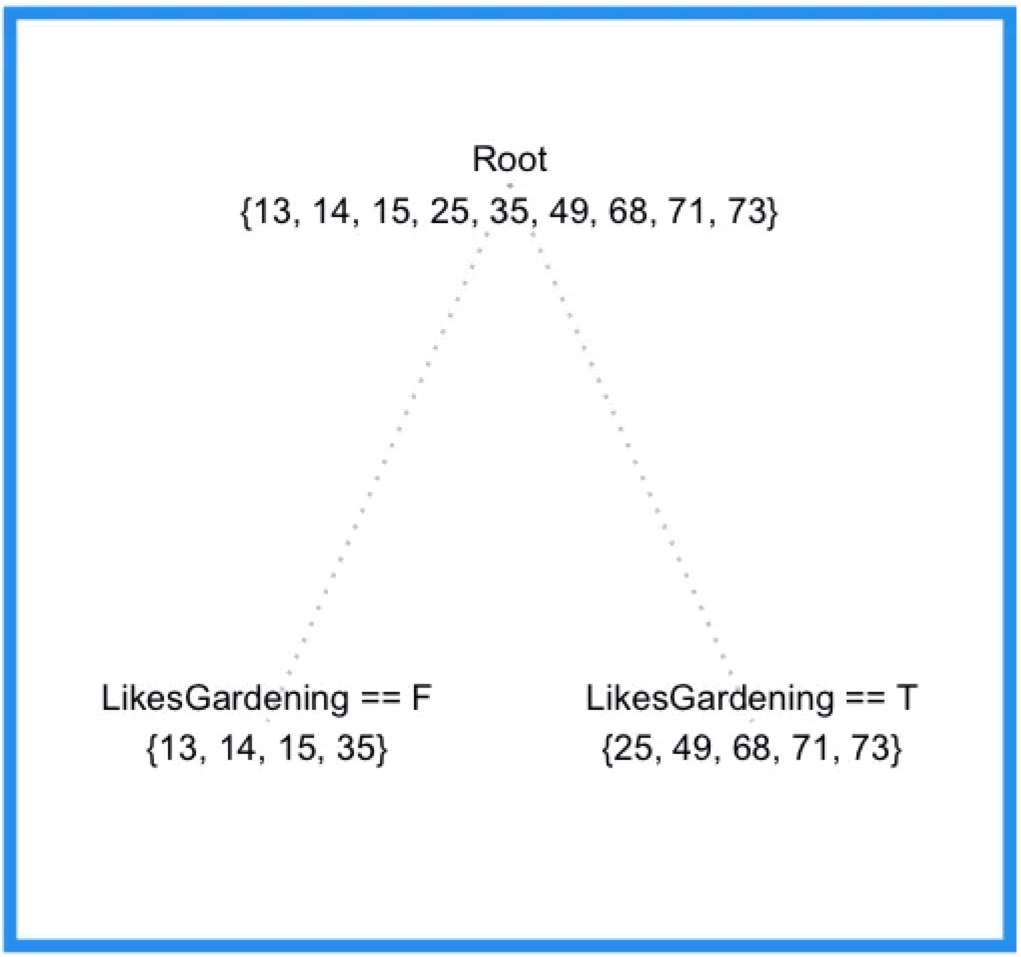
Intuitively, we might expect
– The people who like gardening are probably older
– The people who like video games are probably younger
– *LikesHats* is probably just random noise

We can do a quick and dirty inspection of the data to check these assumptions:

Feature	FALSE	TRUE
LikesGardening	{13, 14, 15, 35}	{25, 49, 68, 71, 73}
PlaysVideoGames	{49, 71, 73}	{13, 14, 15, 25, 35, 68}
LikesHats	{14, 15, 49, 71}	{13, 25, 35, 68, 73}

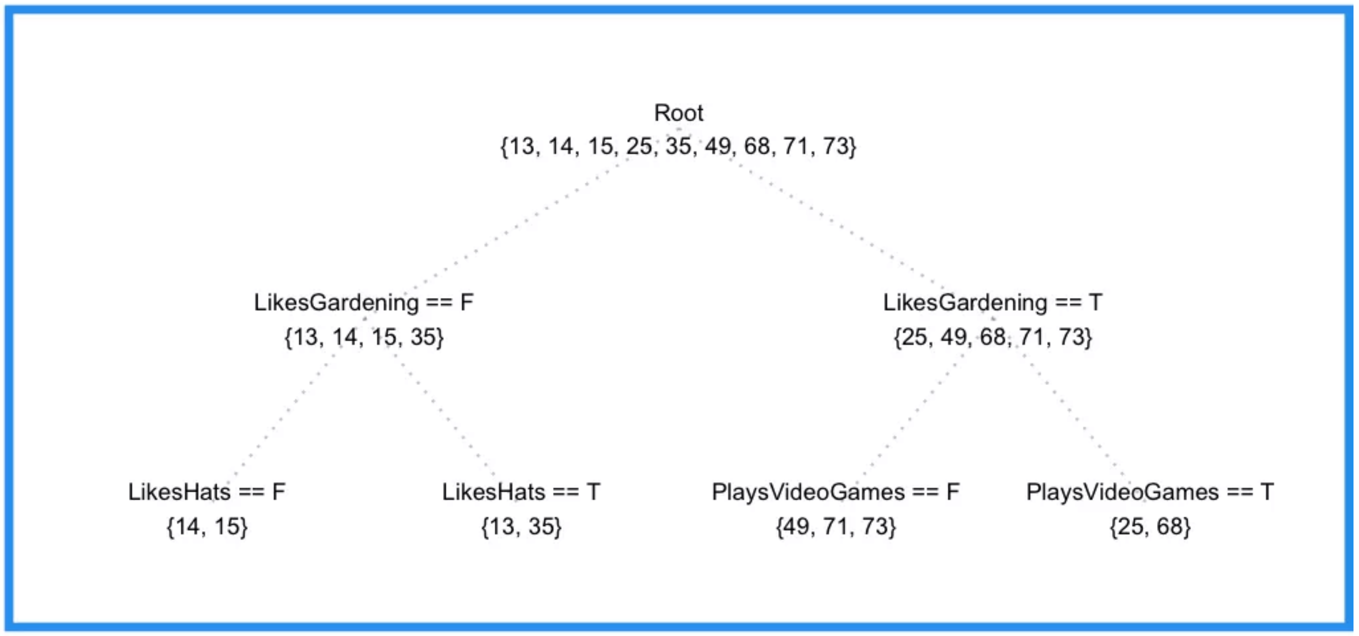
Now let's model the data with a regression tree. To start, we'll require that terminal nodes have at least three samples. With this in mind, the regression tree will make its first and last split on LikesGardening.

Tree 1



This is nice, but it's missing valuable information from the feature LikesVideoGames. Let's try letting terminal nodes have 2 samples.

Overfit Tree

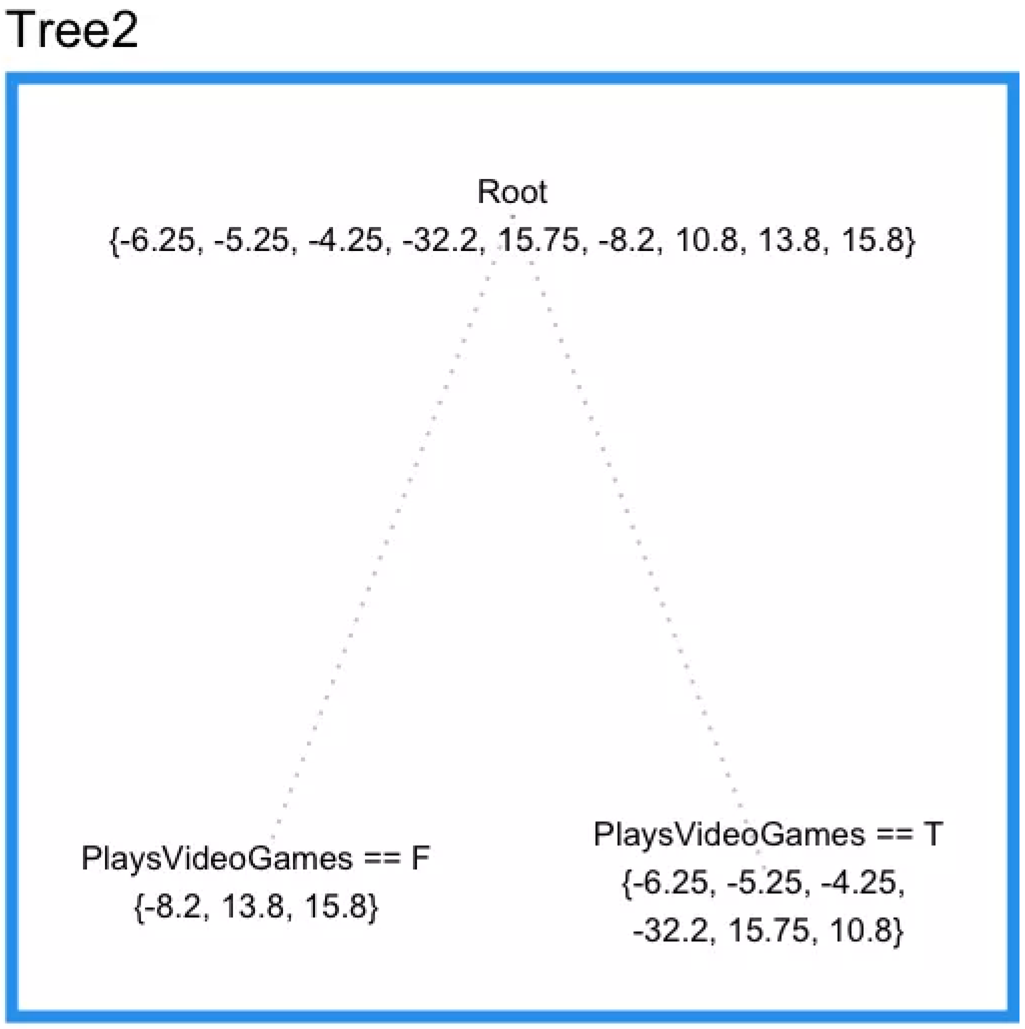


Here we pick up some information from *PlaysVideoGames* but we also pick up information from *LikesHats* – a good indication that we're overfitting and our tree is splitting random noise.

Here in lies the drawback to using a single decision/regression tree – **it fails to include predictive power from multiple, overlapping regions of the feature space**. Suppose we measure the training errors from our first tree.

PersonID	Age	Tree1 Prediction	Tree1 Residual
1	13	19.25	-6.25
2	14	19.25	-5.25
3	15	19.25	-4.25
4	25	57.2	-32.2
5	35	19.25	15.75
6	49	57.2	-8.2
7	68	57.2	10.8
8	71	57.2	13.8
9	73	57.2	15.8

Now we can fit a second regression tree to the residuals of the first tree.



Notice that this tree does **not** include *LikesHats* even though **our overfitted regression tree above did**. The reason is because this regression tree is able to consider LikesHats and PlaysVideoGames with respect to all the training samples, contrary to our overfit regression tree which only considered each feature inside a small region of the input space, thus allowing random noise to select *LikesHats* as a splitting feature.

Now we can improve the predictions from our first tree by adding the "error-correcting" predictions from this tree.

PersonID	Age	Tree1 Prediction	Tree1 Residual	Tree2 Prediction	Combined Prediction	Final Residual
1	13	19.25	-6.25	-3.567	15.68	2.683
2	14	19.25	-5.25	-3.567	15.68	1.683
3	15	19.25	-4.25	-3.567	15.68	0.6833
4	25	57.2	-32.2	-3.567	53.63	28.63
5	35	19.25	15.75	-3.567	15.68	-19.32
6	49	57.2	-8.2	7.133	64.33	15.33
7	68	57.2	10.8	-3.567	53.63	-14.37
8	71	57.2	13.8	7.133	64.33	-6.667
9	73	57.2	15.8	7.133	64.33	-8.667

Tree1 SSE	Combined SSE
1994	1765

Gradient Boosting – Draft 1

Inspired by the idea above, we create our first (naive) formalization of gradient boosting. In pseudocode

1. Fit a model to the data, $F_1(x) = y$
2. Fit a model to the residuals, $h_1(x) = y - F_1(x)$
3. Create a new model, $F_2(x) = F_1(x) + h_1(x)$

It's not hard to see how we can generalize this idea by inserting more models that correct the errors of the previous model. Specifically,

$$F(x) = F_1(x) \mapsto F_2(x) = F_1(x) + h_1(x) \ldots \mapsto F_M(x) = F_{M-1}(x) + h_{M-1}(x)$$

where $F_1(x)$ is an initial model fit to y

Since we initialize the procedure by fitting $F_1(x)$, our task at each step is to find $h_m(x) = y - F_m(x)$.

Stop. Notice something. h_m is just a "model". Nothing in our definition requires it to be a tree-based model. This is one of the broader concepts and advantages to gradient boosting. It's really just a framework for iteratively improving any weak learner. So in theory, a well coded gradient boosting module would allow you to "plug in" various classes of weak learners at your disposal. In practice however, h_m is almost always a tree based learner, so for now it's fine to interpret h_m as a regression tree like the one in our example.

Gradient Boosting – Draft 2

Now we'll tweak our model to conform to most gradient boosting implementations – we'll initialize the model with a single prediction value. Since our task (for now) is to minimize squared error, we'll initialize F with the mean of the training target values.

$$F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^n L(y_i, \gamma) = \underset{\gamma}{\arg\min} \sum_{i=1}^n (\gamma - y_i)^2 = \frac{1}{n}\sum_{i=1}^n y_i.$$

Then we can define each subsequent F_m recursively, just like before

$F_{m+1}(x) = F_m(x) + h_m(x) = y$, for $m \geq 0$

where h_m comes from a class of base learners $\mathcal{H}$ (e.g. regression trees).

At this point you might be wondering how to select the best value for the model's hyper-parameter m. In other words, how many times should we iterate the residual-correction procedure until we decide upon a final model, F? This is best answered by testing different values of m via cross-validation (https://en.wikipedia.org/wiki/Cross-validation_(statistics)).

Gradient Boosting – Draft 3

Up until now we've been building a model that minimizes squared error, but what if we wanted to minimize absolute error? We *could* alter our base model (regression tree) to minimize absolute error, but this has a couple drawbacks..

1. Depending on the size of the data this could be very computationally expensive. (Each considered split would need to search for a median.)
2. It ruins our "plug-in" system. We'd only be able to plug in weak learns that support the objective function(s) we want to use.

Instead we're going to do something much niftier. Recall our example problem. To determine F_0, we start by choosing a minimizer for absolute error. This'll be $median(y) = 35$. Now we can measure the residuals, $y - F_0$.

PersonID	Age	F0	Residual0
1	13	35	-22

PersonID	Age	F0	Residual0
2	14	35	-21
3	15	35	-20
4	25	35	-10
5	35	35	0
6	49	35	14
7	68	35	33
8	71	35	36
9	73	35	38

Consider the first and fourth training samples. They have F_0 residuals of -22 and -10 respectively. Now suppose we're able to make each prediction 1 unit closer to its target. Respective squared error reductions would be 43 and 19, while respective absolute error reductions would be 1 and 1. So a regression tree, which by default minimizes squared error, will focus heavily on reducing the residual of the first training sample. But if we want to minimize absolute error, moving each prediction one unit closer to the target produces an equal reduction in the cost function. With this in mind, suppose that instead of training h_0 on the residuals of F_0, we instead train h_0 on the gradient of the loss function, $L(y, F_0(x))$ with respect to the prediction values produced by $F_0(x)$. Essentially, we'll train h_0 on the cost reduction for each sample if the predicted value were to become one unit closer to the observed value. In the case of absolute error, h_m will simply consider the sign of every F_m residual (as apposed to squared error which would consider the magnitude of every residual). After samples in h_m are grouped into leaves, an average gradient can be calculated and then scaled by some factor, γ, so that $F_m + \gamma h_m$ minimizes the loss function for the samples in each leaf. (Note that in practice, a different factor is chosen for each leaf.)

Gradient Descent

Let's formalize this idea using the concept of gradient descent (https://en.wikipedia.org/wiki/Gradient_descent). Consider a differentiable function we want to minimize. For example,

$$L(x_1, x_2) = \frac{1}{2}(x_1 - 15)^2 + \frac{1}{2}(x_2 - 25)^2$$

The goal here is to find the pair (x_1, x_2) that minimizes L. Notice, you can interpret this function as calculating the squared error for two data points, 15 and 25 given two prediction values, x_1 and x_2 (but with a $\frac{1}{2}$ multiplier to make the math work out nicely). Although we can minimize this function directly, **gradient descent will let us minimize more complicated loss functions** that we *can't* minimize directly.

Initialization Steps:
Number of iteration steps $M = 100$
Starting point $s^0 = (0, 0)$
Step size $\gamma = 0.1$

For iteration $m = 1$ to M:
1. Calculate the gradient of L at the point $s^{\text{(m-1)}}$
2. "Step" in the direction of greatest descent (the negative gradient) with step size γ. That is, $s^{\text{m}} = s^{\text{(m-1)}} - \gamma \nabla L(s^{\text{(m-1)}})$

If γ is small and M is sufficiently large, s^{M} will be the location of L's minimum value.

A few ways we can improve this framework:
– Instead of iterating a fixed number of times, we can iterate until the next iteration produces sufficiently small improvement.
– Instead of stepping a fixed magnitude for each step, we can use something like line search (https://en.wikipedia.org/wiki/Line_search) smartly choose step sizes.

If you're struggling with this part, just google gradient descent (https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=gradient%20descent). It's been explained many times in many ways.

Leveraging Gradient Descent

Now we can use gradient descent for our gradient boosting model. The objective function we want to minimize is L. Our starting point is $F_0(x)$. For iteration $m = 1$, we compute the gradient of L with respect to $F_0(x)$. Then we fit a weak learner to the gradient components. In the case of a regression tree, leaf nodes produce an **average gradient** among samples with similar features. For each leaf, we step in the direction of the average gradient (using line search to determine the step magnitude). The result is F_1. Then we can repeat the process until we have F_M.

Take a second to stand in awe of what we just did. We modified our gradient boosting algorithm so that it works with any differentiable loss function. (This is the part that gets butchered by a lot of gradient boosting explanations.) Let's clean up the ideas above and reformulate our gradient boosting model once again.

Initialize the model with a constant value:
$F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, \gamma)$

For m = 1 to M:
Compute *pseudo* residuals, $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$
Fit base learner, $h_m(x)$ to pseudo residuals

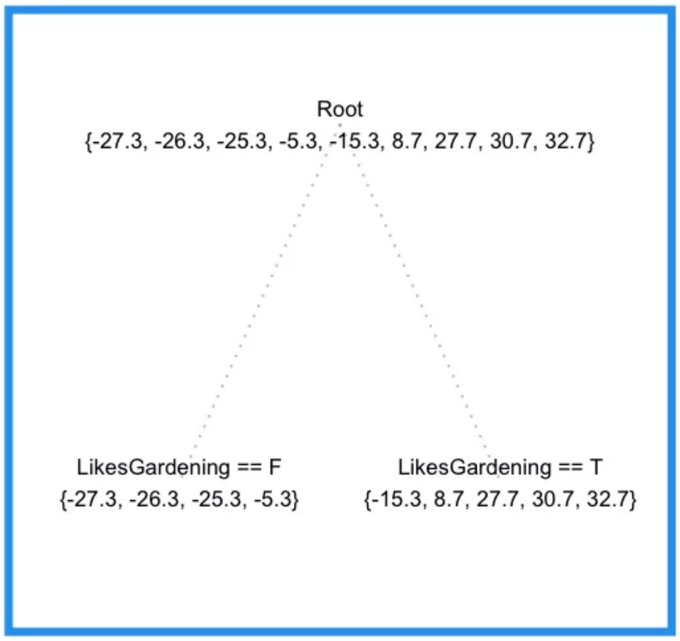
Compute step magnitude multiplier γ_m. (In the case of tree models, compute a different γ_m for every leaf.)
Update $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$

In case you want to check your understanding so far, our current gradient boosting applied to our sample problem for both squared error and absolute error objectives yields the following results.

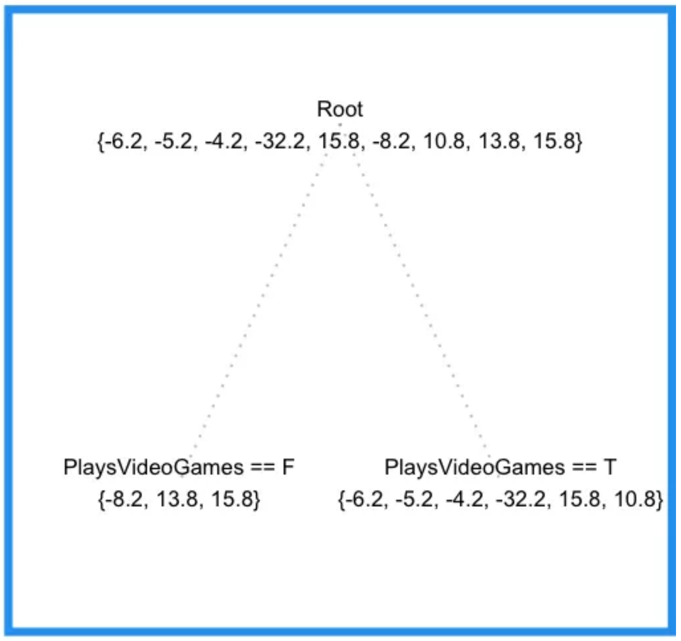
Squared Error

Age	F0	PseudoResidual0	h0	gamma0	F1	PseudoResidual1	h1	gamma1	F2
13	40.33	-27.33	-21.08	1	19.25	-6.25	-3.567	1	15.6
14	40.33	-26.33	-21.08	1	19.25	-5.25	-3.567	1	15.6
15	40.33	-25.33	-21.08	1	19.25	-4.25	-3.567	1	15.6
25	40.33	-15.33	16.87	1	57.2	-32.2	-3.567	1	53.6
35	40.33	-5.333	-21.08	1	19.25	15.75	-3.567	1	15.6
49	40.33	8.667	16.87	1	57.2	-8.2	7.133	1	64.3
68	40.33	27.67	16.87	1	57.2	10.8	-3.567	1	53.6
71	40.33	30.67	16.87	1	57.2	13.8	7.133	1	64.3
73	40.33	32.67	16.87	1	57.2	15.8	7.133	1	64.3

h0



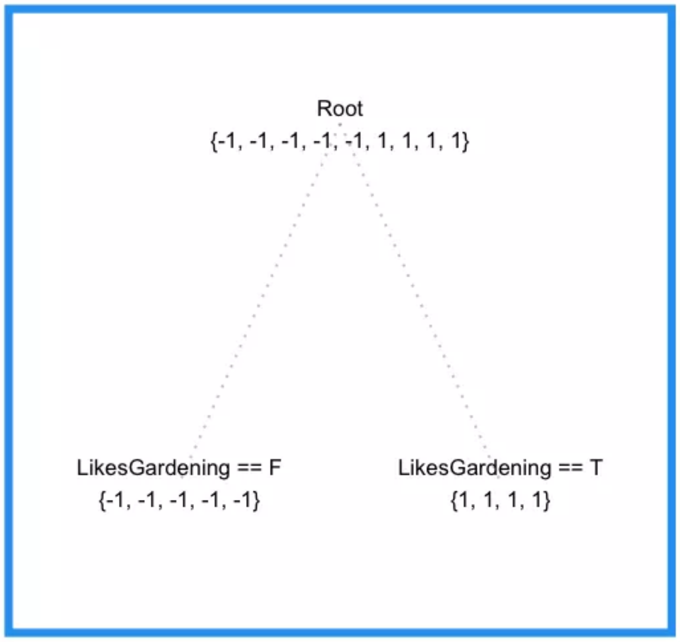
h1



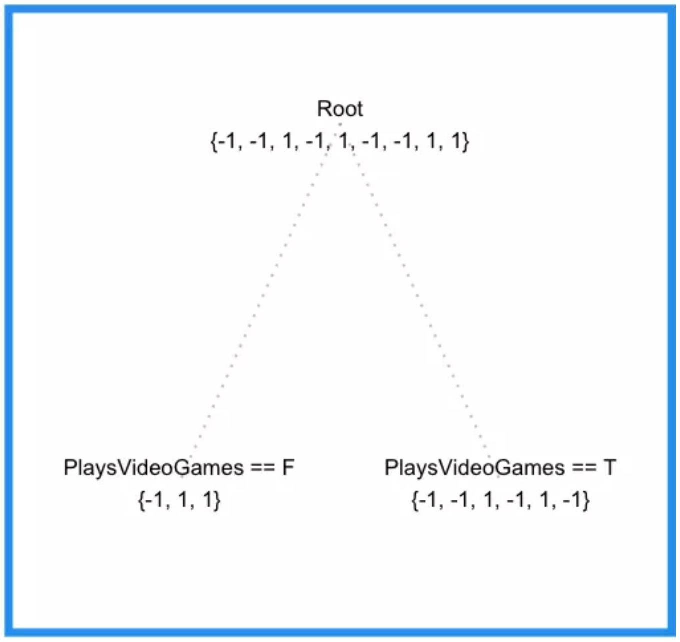
Absolute Error

Age	F0	PseudoResidual0	h0	gamma0	F1	PseudoResidual1	h1	gamma1	F2
13	35	-1	-1	20.5	14.5	-1	-0.3333	0.75	14.25
14	35	-1	-1	20.5	14.5	-1	-0.3333	0.75	14.25
15	35	-1	-1	20.5	14.5	1	-0.3333	0.75	14.25
25	35	-1	0.6	55	68	-1	-0.3333	0.75	67.75
35	35	-1	-1	20.5	14.5	1	-0.3333	0.75	14.25
49	35	1	0.6	55	68	-1	0.3333	9	71
68	35	1	0.6	55	68	-1	-0.3333	0.75	67.75
71	35	1	0.6	55	68	1	0.3333	9	71
73	35	1	0.6	55	68	1	0.3333	9	71

h0



h1



Gradient Boosting – Draft 4

Here we introduce something called shrinkage (https://en.wikipedia.org/wiki/Gradient_boosting#Shrinkage). (https://en.wikipedia.org/wiki/The_Hamptons_(Seinfeld)) The concept is fairly simple. For each gradient step, the step magnitude is multiplied by a factor between 0 and 1 called a learning rate. In other words, each gradient step is *shrunken* by some factor. The current Wikipedia excerpt on shrinkage doesn't mention why shrinkage is effective – it just says that shrinkage appears to be empirically effective. My personal take is that it causes sample-predictions to *slowly* converge toward observed values. As this slow convergence occurs, samples that get closer to their target end up being grouped together into larger and larger leaves (due to fixed tree size parameters), resulting in a natural regularization effect.

Gradient Boosting – Draft 5

Last up – row sampling and column sampling. Most gradient boosting algorithms provide the ability to sample the data rows and columns before each boosting iteration. This technique is usually effective because it results in more *different* tree splits, which means more overall information for the model. To get a better intuition for why this is true, check out my post on Random Forest (https://gormanalysis.com/random-forest-from-top-to-bottom/), which employs the same random sampling technique. Alas we have our final gradient boosting framework.

Gradient Boosting in Practice

Gradient boosting in incredibly effective in practice. Perhaps the most popular implementation, XGBoost (https://github.com/dmlc/xgboost), is used in a number of winning Kaggle solutions. XGBoost employs a number of tricks that make it faster and more accurate than traditional gradient boosting (particularly 2nd-order gradient descent) so I'll encourage you to try it out and read Tianqi Chen's paper about the algorithm (http://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf). With that said, a new competitor, LightGBM (https://github.com/Microsoft/LightGBM) from Microsoft, is gaining significant traction.

What else can it do? Although I presented gradient boosting as a regression model, it's also very effective as a classification and ranking model. As long as you have a differentiable loss function for the algorithm to minimize, you're good to go. The logistic function (https://en.wikipedia.org/wiki/Logistic_function) is typically used for binary classification and the softmax function (https://en.wikipedia.org/wiki/Softmax_function) is often used for multi-class classification.

I leave you with a quote from my fellow Kaggler Mike Kim (https://www.kaggle.com/mikeskim).

My only goal is to gradient boost over myself of yesterday. And to repeat this everyday with an unconquerable spirit.

Bio

(https://www.kaggle.com/ben519)I'm Ben Gorman (https://www.kaggle.com/ben519) – math nerd and data science enthusiast based in the New Orleans area. I spent roughly five years as the Senior Data Analyst for Strategic Comp (http://www.greatamericaninsurancegroup.com/Insurance/Strategic-Comp/Pages/default.aspx) before starting GormAnalysis (https://gormanalysis.com/). I love talking about data science, so never hesitate to shoot me an email if you have questions: bgorman@gormanalysis.com (mailto:bgorman@gormanalysis.com). As of September 2016, I'm a Kaggle Master ranked in the top 1% of competitors world-wide.

Share this:

(http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/?share=twitter&nb=1)

(http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/?share=facebook&nb=1) 236

Related

The Future of AI in Africa Looks Bright | A Winner Interview with Mhamed Jabri (http://blog.kaggle.com/2019...
February 26, 2019
In "Kernels"

Reviewing 2018 and Previewing 2019 (http://blog.kaggle.com/2019... 2018-and-previewing-2019/)
January 18, 2019
In "Kaggle News"

Data Notes: Surprising Insights Into When Crowdfunding Works (http://blog.kaggle.com/2019... notes-surprising-insights-into-when-crowdfunding-works/)
March 14, 2019
In "Data Notes"

TUTORIAL (HTTP://BLOG.KAGGLE.COM/TAG/TUTORIAL/)

XGBOOST (HTTP://BLOG.KAGGLE.COM/TAG/XGBOOST/)

Comments 55

REPLY (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13536#RESPOND)

Vladislavs Dovgalecs

JANUARY 23, 2017 AT 1:53 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13536)

Yes, UH-60 Blackhawk Helicopter is *way* much faster than Toyota Camry but it is also super expensive and very costly in maintenance 🙂

Carlos Aguayo

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13862#RESPOND)

MAY 20, 2017 AT 7:13 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13862)

"Don't miss the forest for the trees" 😉

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13537#RESPOND)

Agzamov Rustam

JANUARY 23, 2017 AT 11:24 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13537)

Is it correct that F_2(x) = F_1(x) + F_2(x) (step 3 in draft 1)? May be it should be F_2(x) = F_1(x) + h_1(x)?

Ben Gorman

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13542#RESPOND)

JANUARY 24, 2017 AT 7:34 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13542)

Whoops, that's a mistake. Thanks for catching. Will try to get it fixed ASAP.

Sijo Vm

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13540#RESPOND)
JANUARY 24, 2017 AT 1:01 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13540)

For personid 9 and age = 73, playing video games was false. However, in your overfit tree diagram, it falls in true. Please correct it and edit the document since it is very confusing. Thank you .

Ben Gorman

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13543#RESPOND)
JANUARY 24, 2017 AT 7:35 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13543)

Good catch. Fixing this now!

ldmtwo

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13579#RESPOND)
FEBRUARY 8, 2017 AT 4:09 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13579)

Looking at the table below tree 2, how are the two trees combined? Take row 1 for instance. We have regression tree2 that produces 19.25 and -3.567? Is tree 2 trying to predict the error or what? I'm lost. Thanks.
PersonID Age Tree1 Prediction Tree1 Residual Tree2 Prediction Combined Prediction Final Residual
1 13 19.25 -6.25 -3.567 15.68 2.683

ldmtwo

REPLY

(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14060#RESPOND)

SEPTEMBER 21, 2017 AT 7:18 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14060)

IDK It wasn't clear before, but to answer my question: each residual R in the earlier steps is made by 1) get the prediction for a base model, 2) with a 2nd model, predict the individual errors (residuals) that the 1st model will have, and 3) adjust base predictions with the residual. The first model would be fit with inputs X and labels Y. The 2nd model would be fit with inputs X and labels R.

huts

REPLY

(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13581#RESPOND)

FEBRUARY 10, 2017 AT 5:23 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13581)

Thanks for the awesome article Ben! I only had a general view of how boosting worked, but this makes it very clear for gradient boosting, though I did struggle to understand the paragraph "Gradient Boosting - draft 3". I think the fact that the gradient descent is only used to get residuals and NOT to optimize parameters could be stressed even more (I was misleaded by searching for those parameters being optimized with gradient descent, but I was being confused by the "classic" use of gradient descent in other models, such as in neural nets).

There is, however, one point that I did not get when you are giving the estimating process for the squared error and the absolute error loss functions. For the squared error, h0 and h1 forecasts are given by the mean of the samples contained in each leaf node. However, I did not get how you got the h0 and h1 forecasts for the absolute error function. Besides, the way gamma0 and gamma1 are estimated for the absolute error loss function is still obscure (though I understand from the line search article from wikipedia that we are looking for the step that minimizes the loss function given the gradient in one point).

That would be great if you could explicit those last details so we could understand the whole picture of your article 😉

Thanks again, and continue the great work!

Walter

REPLY
↰
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13920#RESPOND)
JUNE 15, 2017 AT 8:20 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13920)

Huts,

The way that h0 and h1 are computed in the absolute error loss function example is again by the mean of the samples contained in each leaf node. The problem is that the first decision tree has a drawing mistake. The left leaf (LikesGardening==F) should contain the values {-1,-1,-1,-1} corresponding to PersonID {1,2,3,5}. The second leaf (LikesGardening==T) should contain the values {-1,1,1,1,1} corresponding to PersonID {4,6,7,8,9}. Now, buy computing the mean in leaf1: (-1-1-1-1)/4 = -1, and for leaf2: (-1+1+1+1+1)/5=0.6. This values coincide with the values provided in the table.

In regards of gamma0 and gamma1, remember that the median is the statistic that minimizes the absolute error loss function. Therefore, gamma0 is obtained by getting the median over all values of (Age-F0)/h0, while gamma1 is obtained as median{(Age-F1)/h1}.

Please let me know if this answer your questions

rn27in .

REPLY
↰
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14147#RESPOND)
NOVEMBER 11, 2017 AT 10:06 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14147)

Thanks Walter, this really explains the last crucial part.

Thanks for your help again

Hung Nguyen

REPLY
↰
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14171#RESPOND)
NOVEMBER 27, 2017 AT 4:30 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14171)

I think we have to find the median{(Age-F0)}, and divide that result by the corresponding h0.

Shijie

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=16796#RESPOND)
APRIL 10, 2018 AT 2:21 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-16796)

This is some really useful clarification! Regarding the reason why we are using pseudo residual instead of real residual, can I understand it as it's more flexible for different loss functions?

Bo Yang

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13621#RESPOND)
MARCH 3, 2017 AT 9:19 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13621)

Excellent article, light on jargon and great explanations. I wish I came upon articles like this one when I was learning about gradient boosting, I would have learned it 10x faster.

António Góis

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13624#RESPOND)
MARCH 4, 2017 AT 5:05 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13624)

First of all, thanks for this awesome explanation. I found it very easy to read, and I think I finally grasped the concept of gradient boosting. All the other explanations I've seen just hit you with the formulas without giving an intuition.
I just didn't get your comparison with random forest in "Draft 5". The point in using only some samples per tree and only some features per node, in random forests, is that you'll have a lot of trees voting for the final decision and you want diversity

among those trees (correct me if I'm wrong here).
From what I understood of gradient boosting, you have a tree with only one node/feature voting for the direction of the gradient in each step, and you don't reuse the same feature in a future step. If you don't have several trees voting in parallel to decide one gradient direction, whats the point of limiting the features+samples that a tree has access to?

Probably I got something wrong here, but I don't know what...
Thanks a lot for the article and for your attention,
António

Brian

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13631#RESPOND)
MARCH 6, 2017 AT 1:16 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13631)

Any idea where the terminal node estimates come from, for the classic algorithm?

Yohan Obd

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=13700#RESPOND)
MARCH 19, 2017 AT 11:48 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-13700)

Thank you very much for this article !
I have one question however. If after the first model, the next ones are trained on the error of the one before, that means that to "feed" those models you need to have the true value in order to compute the error to use for the model. However once you start working on data outside of your train set, you don't have the true value anymore, it is then impossible to compute the error. Should the model 2 to m be rather fed with the prediction of the previous one instead of its error ?

b k

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14034#RESPOND)

AUGUST 26, 2017 AT 12:18 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14034)

In Draft '3', I question in following sentence "They have F_0 residuals of -22 and -10 respectively. Now suppose we're able to make each prediction 1 unit closer to its target. Respective squared error reductions would be 43 and 19, while respective absolute error reductions would be 1 and 1". Can anyone help me in understanding on values 43,19 are arrived?

Lee XA

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14161#RESPOND)

NOVEMBER 21, 2017 AT 8:00 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14161)

make each prediction closer to target , F_0 residuals will be -22 -> -21, -10 -> -9 . MSE will be 22**2 - 21**2 = 43 and 10**2 - 9**2 = 81

Sam Chen

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=26303#RESPOND)

JULY 4, 2018 AT 12:22 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-26303)

Making each prediction 1 unit closer to the target. Concretely, next time we make prediction for first and forth samples both 34(1 unit closer), then the squared error reduction for first sample is (35-13)^2 - (34-13)^2 = 43, while the absolute error reduction is |35-13| - |34-13| = 1. Hope you find it helpful.

ldmtwo

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14061#RESPOND)

SEPTEMBER 21, 2017 AT 7:29 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14061)

When updating predictions using a learning rate, would that be done as
1) p(i+1) = p(i) + r(i)*lr, which is a simple reduction,
or 2) p(i+1) = p(i) * (1-lr) + (p(i)+r(i))*lr, which is exponential moving average?

lr=learning rate, r=residual, p=predictions

Kumar

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14076#RESPOND)

Rajendran

SEPTEMBER 27, 2017 AT 9:48 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14076)

>What else can it do? Although I presented gradient boosting as a regression model, it's also very effective as a classification and ranking model. As long as you have a differentiable loss function for the algorithm to minimize, you're good to go. The logistic function is typically used for binary classification

It will be cool if we you can say what the logistic function is used for. Is it to convert the prediction to a probability?

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14117#RESPOND)

Shantanu Gangal

OCTOBER 21, 2017 AT 4:25 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14117)

@disqus_r9FLPaMgKX:disqus Thanks for writing this fantastic article.
I don't follow how the absolute error loss function is differentiable though. Maybe I am missing a simple thing -- but isn't the function non-differentiable at 0?

Ben Gorman

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14135#RESPOND)

NOVEMBER 2, 2017 AT 9:35 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14135)

Thanks Shantanu. We could define the derivative = 0 at x = 0 to get around this scenario (which is unlikely).

Chetan Bhat

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14125#RESPOND)

OCTOBER 26, 2017 AT 3:01 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14125)

Hey Ben, great article! I've used this write-up, along with other reference material in tandem to "boost" my own understanding of Gradient Boosting.

In the section "leveraging gradient descent" where you present a summary table to check understanding (for squared error), shouldn't the "PseudoResidual0" column contain gradients of residuals rather than the residuals themselves? Right now this column contains residuals only i.e. difference between age and F0. Or am I missing something?

Ben Gorman

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14136#RESPOND)

NOVEMBER 2, 2017 AT 9:42 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14136)

Thanks! In the case of squared error, the gradients of residuals = residuals. (derivative of 0.5(yhat - y)^2 w.r.t. yhat is yhat - y)

Chetan Bhat

REPLY ↰ (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14138#RESPOND)

NOVEMBER 3, 2017 AT 6:39 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14138)

Ah! of course. Thank you 🙂

Ali

REPLY ↰ (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=27599#RESPOND)

JULY 12, 2018 AT 6:21 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-27599)

I understand the explanation but i think there is an error i would say it's 2 yhat - y. From where comes the 0.5 ?

Thanks Ben

ben

REPLY ↰ (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14167#RESPOND)

NOVEMBER 24, 2017 AT 5:31 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14167)

Thank you for the great article! I am hung up on the "Final Residual," however. Why did you subtract the combined prediction from the actual age, when you subtracted actual age from Tree1 prediction and Tree2 prediction? I don't understand why you are reversing the order of the arithmetic here.

Sean

REPLY ↰ (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14175#RESPOND)

NOVEMBER 28, 2017 AT 3:48 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14175)

Excellent article,

Any chance you could (here or in another article) give us a sense of what the prediction step for new examples looks like? Is it an average or weighted average of the stump models? Or would we only use model 2? Something entirely different? thanks --Sean

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14213#RESPOND)

Octavian Tuchila

DECEMBER 17, 2017 AT 4:55 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14213)

Great article!

I just don't understand one thing:
How do you get numerical predictions from Tree 1, like `19.25` or `57.2`?

Tree1, because it's a decision tree, should be a binary classifier, right?
So it should return make predictions such as `True` or `False`, correct?

zekeriya

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14266#RESPOND)

JANUARY 25, 2018 AT 9:51 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14266)

İt is decision tree for regression problem. And you should calculate average age for each terminal node.
mean(13,14,15,35) = 19.25
mean(25,49,68,71,73) = 57.2

Igor

[REPLY ↩ (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=18094#RESPOND)]

APRIL 28, 2018 AT 2:20 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-18094)

Maybe it's a stupid question, but what I don't really understand is how to make this algorithm use all predictors (LikesGardening, PlaysVideoGames, LikesHats) in a systematic and robust way.

Ben, you showed clearly how one can improve the predictions from using only LikesGardening in Tree 1 to incorparate PlaysVideoGames and build better tree Tree 2. So, it's more or less clear how one can proceed with

Gradient Boosting – Draft 1

Inspired by the idea above, we create our first (naive) formalization of gradient boosting. In pseudocode

1. Fit a model to the data, F_1(x) = y
2. Fit a model to the residuals, h_1(x) = y - F_1(x)
3. Create a new model

in case of just one predictor. But what about the LikesHats variable? And are we done with LikesGardening? In other words, how can you proceed to have all predictors (what if there are dozens of them?) used systematically and efficiently and make sure the algorithm converges nicely?

Thanks in advance for your help

ST22

[REPLY ↩ (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14232#RESPOND)]

Bangalore

DECEMBER 31, 2017 AT 2:47 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14232)

Thank you Ben! This is awesome.

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14251#RESPOND)

ananiask8

JANUARY 13, 2018 AT 11:15 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14251)

Can you elaborate on this one:

"My personal take is that it causes sample-predictions to slowly converge toward observed values. As this slow convergence occurs, samples that get closer to their target end up being grouped together into larger and larger leaves (due to fixed tree size parameters), resulting in a natural regularization effect."

I'm not following unto which values exactly they converge... what are the observed values? Why does this grouping effect help regularization? Is it that samples that are intrinsically better explained via some sort of feature, eventually end up in a leaf described by it? Or something different?

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14521#RESPOND)

Rajalakshmi

MARCH 10, 2018 AT 7:23 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14521)

Hi, I m new to this topic. Can you please tell how you are computing tree 1 prediction and tree 2 prediction and its residue

Gorfball

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=14768#RESPOND)

MARCH 13, 2018 AT 4:31 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-14768)

Hey, Ben. Thanks for the article. Admittedly an amateur, I'm leaving a bit confused. I found this brief thread to be reasonably helpful, but the information in your article feels dissonant: https://stats.stackexchange.com/questions/186966/gradient-boosting-for-linear-regression-why-does-it-not-work (https://stats.stackexchange.com/questions/186966/gradient-boosting-for-linear-regression-why-does-it-not-work)

In particular, this quote: Boosting shines when there is no terse functional form around. Boosting decision trees lets the functional form of the regressor/classifier evolve slowly to fit the data, often resulting in complex shapes one could not have dreamed up by hand and eye. When a simple functional form is desired, boosting is not going to help you find it (or at least is probably a rather inefficient way to find it).

In the example given, I don't see why using GB is anything but an inefficient routine for mimicking the process of a standard least-squares regression. In the link provided, it alludes to the innate tendency to motivate regularization you mentioned, but also describes that other, better methods are available (e.g., ridge regression) for doing so.

Is representing this problem with a simple decision tree really a good example of how GB is more powerful than a least-squares regression? Am I missing something in this Example?

Diogo Pinto

REPLY ↰ (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=15478#RESPOND)

MARCH 23, 2018 AT 5:30 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-15478)

Hi,

Thank you for the awesome article!

I just want to point out that, from my understanding, the pictures for the Gradient Boosting using Absolute Error example do not match the information from the table, so it might be useful to recheck that.

Thank you again 🙂
Diogo Pinto

Nick

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=15586#RESPOND)
MARCH 24, 2018 AT 1:14 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-15586)

Good article, but there is a confusing error in e.g. "Gradient Boosting – Draft 1". In the first bullet point, you define y to be the predicted value. If you calculate the residuals in the 2nd bullet point, you want to calculate the original value - the predicted value. We have h(x) = y-F_1(x), where F_1(x) was defined to be y, thus always zero per the stated equations. It should be a ŷ or something similar.

Pulkit Bansal

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=16289#RESPOND)
APRIL 1, 2018 AT 11:15 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-16289)

Hi Ben,

Great article. This gave me more clarity about gradient boosting.

I have one question though. Could you please explain how are pseudo-residuals computed? I mean how are we computing derivative of the loss function w.r.t. a F_0(x) which is a tree. I can understand computing gradient w.r.t a real number x1, x2 etc. , but partial derivative w.r.t. a regression tree and what that means is not clear to me.

Thanks,
Pulkit

Renato Ciani

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=16408#RESPOND)
APRIL 3, 2018 AT 7:10 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-16408)

Thanks! Thanks! You're great! Great article! Sometimes it's hard to dive into a raw paper like the original Friedman's. This kick off makes the understanding smooth.

I would suggest check the example of Absolute Error (maybe need a correction) , the h0 tree doesn't seems to show the right sets for LikesGardening pseudoResiduals0, ie. LikesGardening == F -> {-1,-1,-1,-1} and LikesGardening == T -> {-1,1,1,1,1}. So we can have the right avg({-1,-1,-1,-1}) = -1 and avg({-1,1,1,1,1}) =0.6 to match the h0 column.

Best Regards,

Renato

Michael Zeng

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=16475#RESPOND)
APRIL 4, 2018 AT 4:38 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-16475)

Thanks for the excellent explanation. How would you explain the superiorty of XGBoost when dealing with imbalanced sample?

Igor

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=18095#RESPOND)
APRIL 28, 2018 AT 2:24 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-18095)

Just to use chronological order to make my question visible today (I apologize if duplication was not needed).

Maybe it's a stupid question, but what I don't really understand is how to make this algorithm use all predictors (LikesGardening, PlaysVideoGames, LikesHats) in a systematic and robust way.

Ben, you showed clearly how one can improve the predictions from using only LikesGardening in Tree 1 to incorparate PlaysVideoGames and build better tree Tree 2. So, it's more or less clear how one can proceed with

Gradient Boosting – Draft 1

Inspired by the idea above, we create our first (naive) formalization of gradient boosting. In pseudocode

1. Fit a model to the data, F_1(x) = y
2. Fit a model to the residuals, h_1(x) = y - F_1(x)
3. Create a new model

in case of just one predictor. But what about the LikesHats variable? And are we done with LikesGardening? In other words, how can you proceed to have all predictors (what if there are dozens of them?) used systematically and efficiently and make sure the algorithm converges nicely?

Thanks

Igor

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=18097#RESPOND)
APRIL 28, 2018 AT 2:27 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-18097)

Another more general question for the community.

Does anyone know of a source to look up some GB algorithm examples coded in SAS (I mean Base/Stat, not Enterprise Miner)?

Thank you in advance

Dylan

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=22546#RESPOND)
JUNE 4, 2018 AT 8:38 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-22546)

I'd like to elaborate on the idea of gradient descent. You mentioned in part 4 that the wikipedia on stepping doesn't really explain in pragmatic terms what the benefit is and if I understand the context correctly, this is what I've understood:

As you take smaller steps toward a local minima, iteratively smaller steps guarantees accuracy by not overshooting your local minima into the other side of the concave (picture a valley in a graphed function). Smaller steps guarantee that you do not head toward the ramp back up with your stepping's "velocity".

I might be thinking of the wrong thing or conflating stochastic gradient descent without appreciating the differences here but this is how I've understood it.

Hande

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=25027#RESPOND)
JUNE 25, 2018 AT 7:50 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-25027)

Very nice and clear explanation. Thank you!

smriti

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=25223#RESPOND)
JUNE 27, 2018 AT 3:06 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-25223)

Hi Ben,thanks for this illustration. Would you have similar illustration available for binary dependent variable. In this case how do you compute the residuals.

Utkarsh Mishra

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=27287#RESPOND)
JULY 10, 2018 AT 5:40 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-27287)

"Notice that this tree does not include LikesHats even though our overfitted regression tree above did. The reason is because this regression tree is able to consider LikesHats and PlaysVideoGames with respect to all the training samples, contrary to our overfit regression tree which only considered each feature inside a small region of the input space, thus allowing random noise to select LikesHats as a splitting feature."

Can someone elaborate these line for me.
What does it mean ?

Cody

REPLY (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=30173#RESPOND)

Bushnell

AUGUST 1, 2018 AT 10:23 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-30173)

Ben, thanks for writing this. Best write up I've seem on the topic, hands down!

ken

REPLY (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=31042#RESPOND)

AUGUST 8, 2018 AT 1:21 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-31042)

This article is full of mistakes and wrong inference. Please, first learn your stuff thoroughly before writing about it and creating information pollution.

Chetan Bhat

REPLY (/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=31715#RESPOND)

AUGUST 14, 2018 AT 7:17 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-31715)

Could you point out which parts are problematic? Simply saying "there are mistakes" and not giving more clarity in itself is not helpful for anyone.

Puzzle

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=36524#RESPOND)

SEPTEMBER 16, 2018 AT 6:50 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-36524)

I feel the same way, using regression tree for classification problems, squared error for classification??? etc……

Chetan Bhat

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=31718#RESPOND)

AUGUST 14, 2018 AT 7:21 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-31718)

Hey Ben,

Thanks again for this. I noticed you had asked on StackOverFlow long ago a question on line search - how does this really work in Gradient Boosting, what is the use of moderating the step size (even when there is a separate learning rate parameter as well) etc.

Do you have any understanding on this as of now? Could you write about it a bit, or point to the right resources to get a better grasp?

Gautam

REPLY
(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=32055#RESPOND)

AUGUST 17, 2018 AT 6:50 PM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-32055)

I used your explanation in an interview and was told that it's the best explanation they've ever heard of Gradient boosted trees. Thanks!

Serge Mosin

REPLY

(/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/?REPLYTOCOM=32380#RESPOND)

(http://www.databrawl.com)

AUGUST 20, 2018 AT 9:48 AM (HTTP://BLOG.KAGGLE.COM/2017/01/23/A-KAGGLE-MASTER-EXPLAINS-GRADIENT-BOOSTING/#COMMENT-32380)

Hi Ben. Thanks for the great hands-on tutorial on this. You've left out some important points though. Could you please clarify (and maybe update the article as well) the following:
1. What are the squared error and absolute error loss functions? I assume the squared error is smth like 1/2 (y - F(x))^2, but I have really no idea what you mean by absolute error objective, nor Google does.
2. How do you calculate gammas here? For squared errors seems that you're just taking them as 1 instead of solving minimization problem for L(y, F - gamma * h). For the absolute error, I have really no idea how you came up with the values.

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

☐

Save my name, email, and website in this browser for the next time I comment.

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

Submit

(https://www.facebook.com/kaggle) (https://twitter.com/kaggle)