

# Machine Learning With Boosting

A Beginner's Guide

By Scott Hartshorn

Sample Book – First 10% Of Content

## What Is In This Book

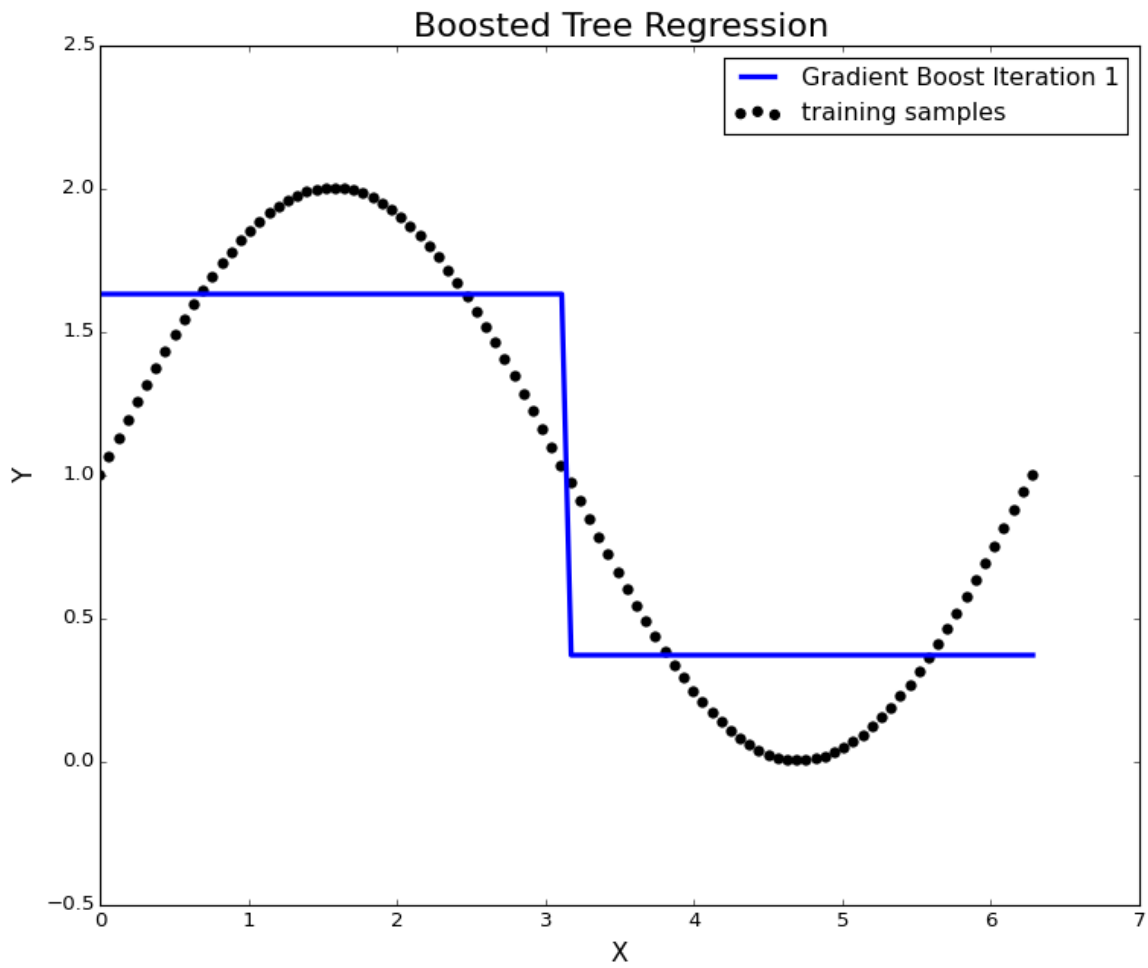
The goal of this book is to provide you with a working understanding of how the machine learning algorithm “Gradient Boosted Trees” works. Gradient Boosted Trees, which is one of the most commonly used types of the more general “Boosting” algorithm is a type of supervised machine learning. What that means is that we will initially pass the algorithm a set of data with a bunch of independent variables, plus the solution that we care about. We will use the known solution and the known independent variables to develop a method of using those variables to derive that solution (or at least get as close as we can). Later on, after we train the algorithm, we will use the method we derived to calculate solutions for unknown results from different independent variables.

This is an example driven book, rather than a theory driven book. That means we will be showing the actual algorithms within the code that executes gradient boosted trees, instead of showing the high level equations about which loss functions are being optimized.

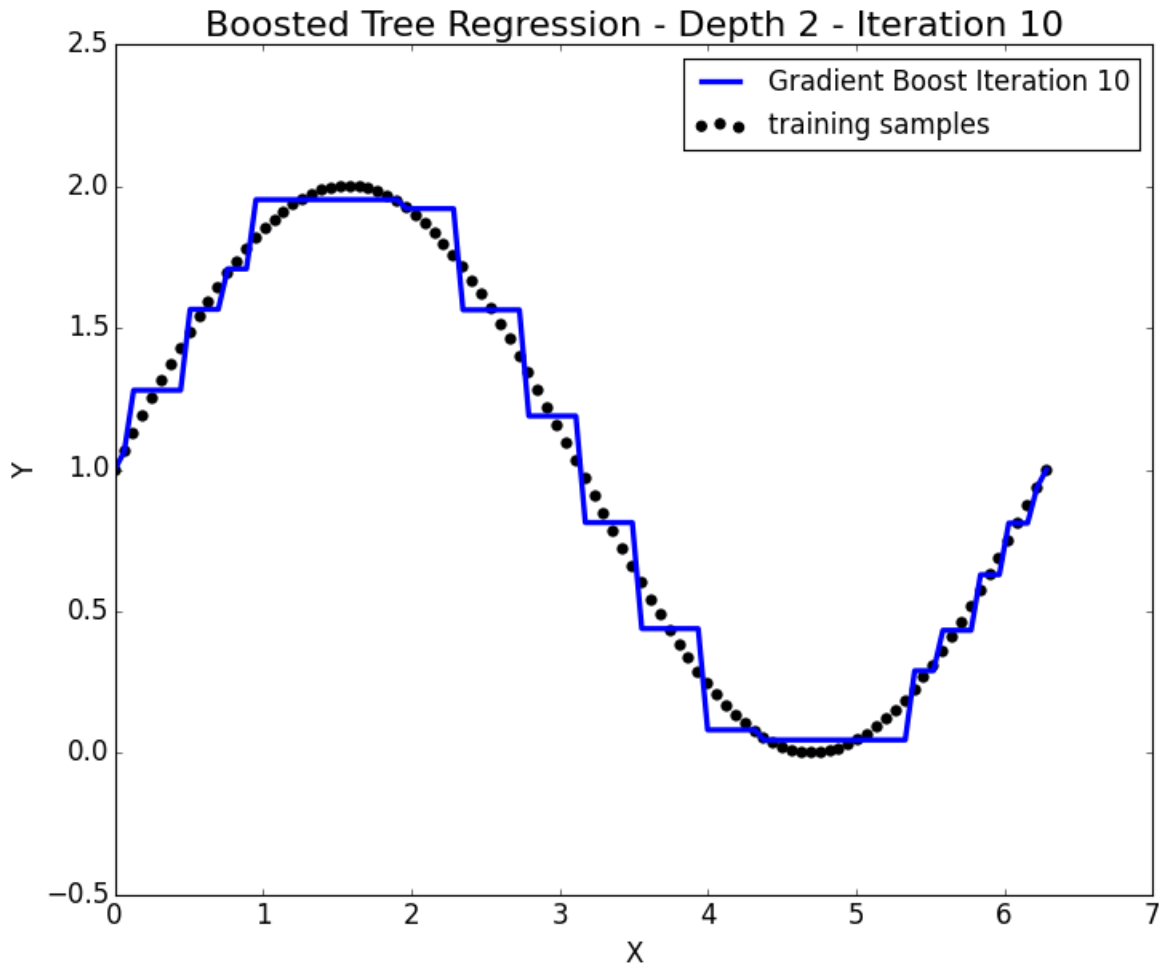
The most common explanation for boosting is “Boosting is a collection of weak learners combined to form a strong learner”. The goal of this book is to provide a more tangible and intuitive explanation than that. This book starts with some analogies that provide a rough framework of how boosting works. And then goes into a step by step explanation of gradient boosted trees. It turns out that the actual boosting algorithms are a straightforward application of algebra, except for the decision trees that are one part of the process for most boosting algorithms. (The decision trees are reasonably straight forward, but are not algebra.)

The examples that will be shown will focus on two types of problems. One is a regression analysis, where we are trying to predict a real value given a set of data. One real life example of regression is Zillow predicting a house’s value based on publicly available data. The example regression analysis we will show isn’t that complicated. We will try to predict the value of a sine wave, shown below as the black dots

**This book is a sample of the full book. Per the licensing agreement I have when publishing on Amazon, I can only provide 10% of the content via a free online distribution. That 10% is included below. The remaining content is available exclusively through [Amazon](#), and can be had at [this link](#). The book Kindle version of the book is less than three dollars (or the equivalent in other currencies) on Amazon**

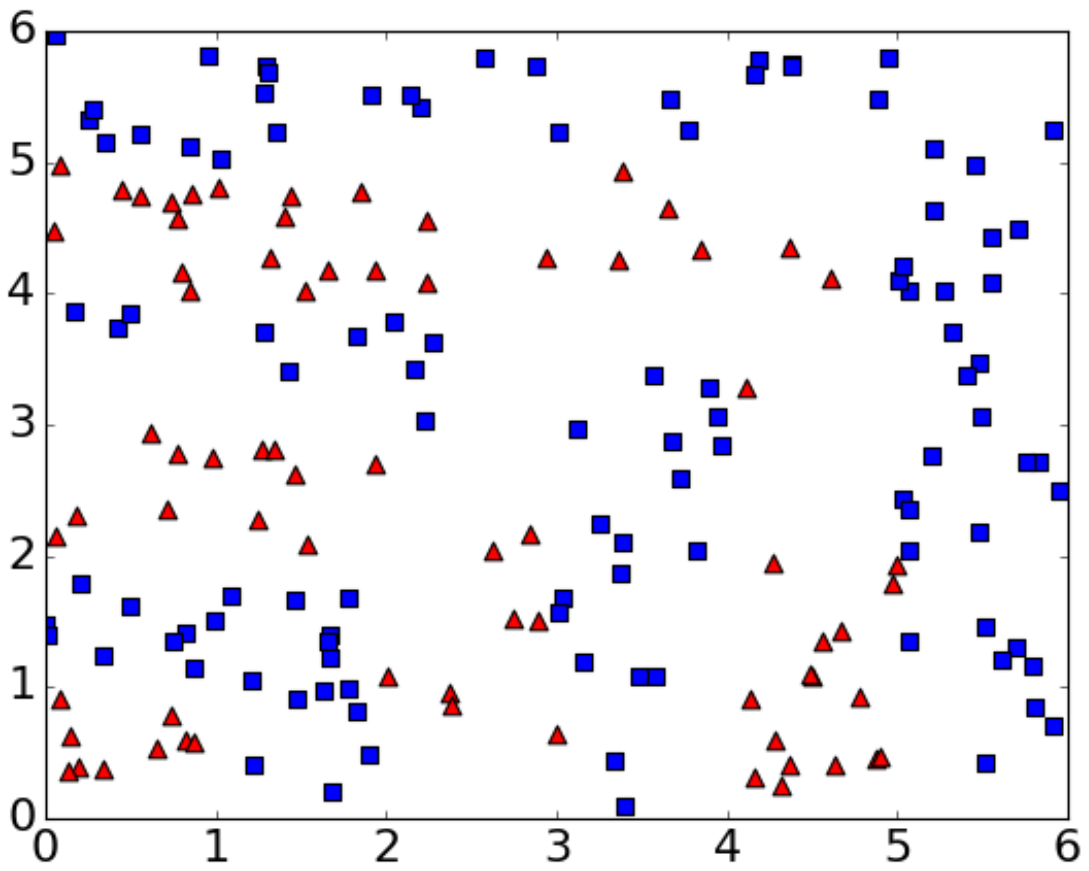


And we will show how that single blue line, which results from a decision tree, can be improved using boosting to match the sine wave values more closely, as shown below

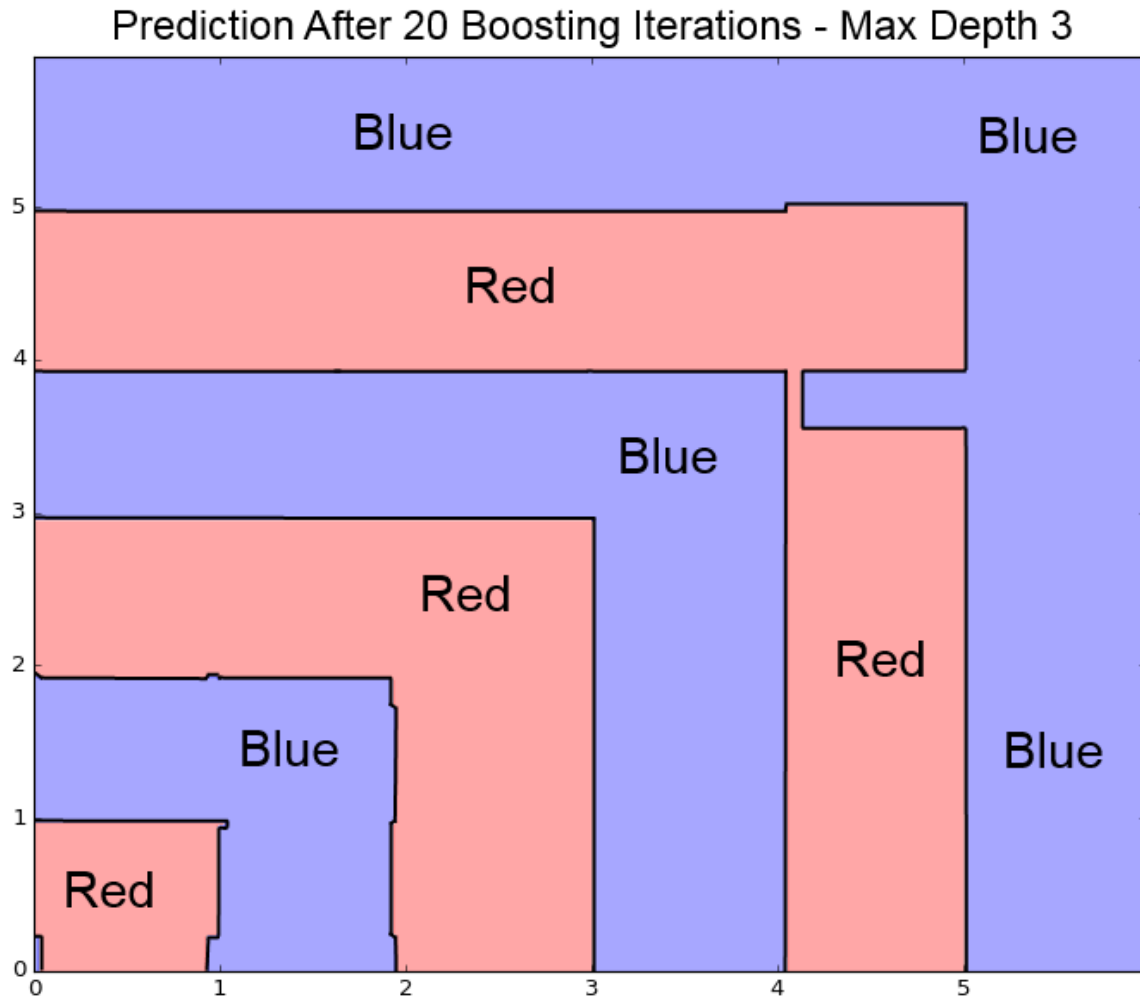


Obviously this sine wave isn't as complicated as Zillow's house price prediction, but it turns out that once we understand how the boosting algorithm works, it is simple to increase the complexity with more data or more layers of boosting.

The other example we will show is a categorization problem. With categorization we are trying to predict discrete results. I.e. instead of Zillow predicting a house's value, it could be an investment company trying to determine should they invest in this asset, yes or no. In that example we will show how we can take categorical data shown below as either red triangles or blue squares



And make predictions about the values of the entire design space, shown below



Once we understand how to group two different categories using boosting, we will extend that to how to work with any number of categories.

### Get The Data And Examples Used In This Book

The algebra of the boosting algorithms can be duplicated in Excel, and I've included an Excel file that does just that in the free downloadable bonus material here <http://www.fairlynerdy.com/boosting-examples> That bonus material also includes all of the Python code used to generate the examples shown.

If you want to help us produce more material like this, [then please leave a positive review](#) for this book on Amazon. It really does make a difference!

If you spot any errors in this book, think of topics that we should include, or have any suggestions for future books then I would love to hear from you. Please email me at

ImFairlyNerdy (at) gmail (dot) com

~ Scott Hartshorn

## Your Free Gift

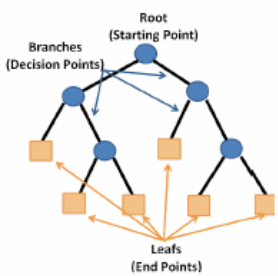
As a way of saying thank you for your purchase, I'm offering this free cheat sheet on Decision Trees that's exclusive to my readers.

Decision trees form the heart of Gradient Boosted Trees, so it is important to understand how they work. They are also a useful machine learning technique in their own right. This is a 2 page PDF document that I encourage you to print, save, and share. [You can download it by going here](#)

# DECISION TREES

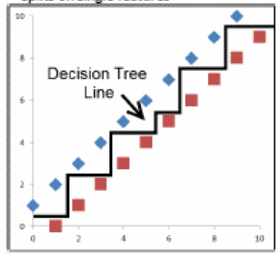
### GENERAL INFO

- Decision trees are a type of supervised machine learning
- They use known training data to create a process that predicts the results of that data. That process can then be used to predict the results for unknown data
- A decision tree processes data into groups based on the value of the data and the features it is provided
- At each decision gate, the data gets split into two separate branches, which might be split again
- The final grouping of the data is called Leaf Nodes

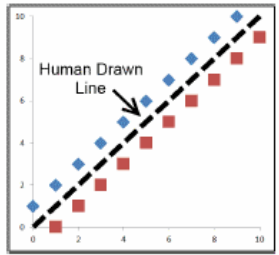


- Decision trees can be used for regression to get a real numeric value. Or they can be used for classification to split data into different distinct categories.

volume and mass, the decision tree will split the data as a series of splits on single features



- It will not do what a human might do, shown below



### STOPPING CRITERIA

- Additional splits can occur until a stopping criteria is reached
- Common stopping criteria are
  - Maximum Depth: the maximum number of splits in a row has been reached

$$leaves = 2^{depth}$$

- Maximum Leaves: don't allow any more leaves
- Min Samples Split: Only split a node if it has at least this many samples
- Min Samples Leaf: Only split a node if both children resulting have at least this many samples
- Min Weight Fraction: Only split a node if it has at least this percentage of the total samples

### PICKING THE SPLITS

- Decision trees look through every possible split and pick the best split between two adjacent points in a given feature
- If the feature is in the real range, the split occurs halfway between the two points
- The split are always selected on a single feature only, not any interaction between multiple features
- That means if there is a relationship between multiple features, for instance, if density is the critical feature, but the data is provided in

### RANDOMIZATION

- When a single decision tree is run, it usually looks at every point in the available data.
- However some algorithms combine multiple decision trees to capture their benefits while mitigating over fitting
- Two widely used algorithms that use multiple decision trees are Random Forests and Gradient Boosted Trees
- Random Forests use a large number of slightly different decision trees run in parallel and averaged to get a final result.
- Gradient Boosted Trees use decision trees run in series, with later trees correcting errors on previous trees.
- If multiple trees are combined, it can be advantageous to have a random element in the creation of the trees, which can mitigate over fitting
- Bagging** – (short for **Bootstrap aggregating**) is drawing samples from the data set with replacement.
- If you had 100 data points, you would randomly draw 100 points and on average get 63.2% unique points and 36.8% repeats

### Time Complexity

- For a single decision tree, the most expensive operation is picking the best splits
- Picking the best split requires evaluating every possible split
- Finding every possible split requires sorting all the data on every feature
- That means for M features and N points, this takes  $M * N \lg(N)$  time
- For machine learning algorithms that use multiple decision trees, those sorts can be done a single time and cached

### Overfitting

- Over fitting, which means too closely matching the training data at the expense of the test data, is a key concern for decision trees
- Different stopping criteria should be evaluated with cross-validation to mitigate over fitting

<http://www.fairlynerdy.com/decision-trees-cheat-sheet/>



# Table of Contents

## Introduction

- [A Quick Example Of Boosting](#)
- [Why Do People Care About Boosting?](#)
- [An Analogy of How Boosting Works](#)
- [How Boosting Really Works – Skipping Over The Details](#)

The material below here is not included in the sample, but is included in the full book on [Amazon here](#)

## Gradient Boosted Trees For Regression

- [Regression vs. Classification](#)
- [Regression Decision Trees](#)
- [Gradient Boosted Trees Regression](#)
- [Regression Boosting Example](#)
- [Learning Rate: How Fast Boosting Incorporates Improvements](#)

## Gradient Boosted Trees For Classification

- [Classification](#)
- [Numbering Ranges – 0 to 1 & Negative Infinity to Positive Infinity](#)
- [Classification Example](#)
- [Predicting With A Model](#)
- [Boosting With 3 Or More Categories](#)

## Model Tuning And Improvement

- [Gradient Boosted Tree Parameters](#)
- [Feature Engineering](#)

## Final Thoughts

- [If You Found Errors Or Omissions](#)
- [More Books](#)
- [Thank You](#)

## A Quick Example Of Boosting

There are several different boosting algorithms that exist. This book focuses on one of them, gradient boosted trees. The exact math differs between different boosting algorithms, but they are all characterized by two key features.

1. Multiple iterations
2. Each subsequent iteration focuses on the parts of the problem that previous iterations got wrong.

A real life example of a boosting algorithm in progress might be a high school band teacher teaching a class of 20 students. This specific band teacher wants to make the average quality of his class as good as possible. So what does he do?

On day 1 he knows nothing about the quality of the musicians that he is teaching, so he simply teaches a standard class. After that, he knows exactly how well each student is doing. So he then tailors his instruction to focus on whichever students he can help the most that day. Typically that will be the worst students in the class. After all, his goal isn't to make his best students perfect, he is trying to bring up the average. It is usually easier to turn Bad into Acceptable than it is to turn Good into Exceptional. So this music teacher will tend to ignore the advanced students and spend more time with the worst students.

How is this an example of boosting put into practice? Simple, the iterations take place each subsequent day at each new class. And the requirement that the algorithm focuses on fixing the errors from previous iterations is satisfied since the teacher is finding the students with the largest amount of error and working to improve them.

## Why Do People Care About Boosting?

Machine learning is a field that has an increasing number of applications, large companies like Google and Amazon are using it for their personal assistant products (i.e. Alexa), and it will likely revolutionize a number of different fields, such as when autonomous cars become available. That is machine learning on a grand scale done with large dedicated teams, and is more advanced than we will cover in this book. On an individual's scale, machine learning has a number of interesting applications as well. One of the easiest places to see them applied is on the machine learning competition site Kaggle.

On Kaggle, individuals or small teams compete to take different sets of data and extract the most information possible out of them using whatever techniques they choose. The winners frequently receive cash, as well as bragging rights. But the important point here is not the competitors, but the companies who are generating the data sets. They have real problems where a better analysis of data can open up new business opportunities, and they are willing to give away cash prizes (typically in the tens of thousands of dollars, sometimes more) to get better answers.

It is difficult to know which machine learning algorithm will work best for any given problem. However in recent problems people have found that boosting (especially XGBoost which uses gradient boosted trees along with other improvements) has done very well. Here are some competitions that have been won using boosting, or boosting in conjunction with other techniques

- [Liberty Mutual Property Inspection](#) – Use some property information to predict the hazards in a home, for insurance purposes.
- [Caterpillar Tube Pricing](#) – Attempt to predict how much a supplier will charge for different orders of metal tubing.
- [Avito Duplicate Ads Detection](#) – Identify duplicate ads from an online marketplace so that they can be removed.
- [Facebook Robot Detection](#) – An online auction site (likely a penny auction site) has been flooded with robot bidders which is causing the real customers to leave the site. Can you identify the robots?
- [Otto Product Classification](#) – Use a set of provided features to figure out what category different products should be grouped into.

The type of boosting shown in this book, Gradient Boosted Trees, uses multiple decision trees (which are a series of if-then questions that split the data into two different branches, shown in detail later) sequentially in order to improve on one of the main limitations that decision trees have, which is overfitting.

## An Analogy of How Boosting Works

Boosting, when applied to computer science, has a more formal definition than the music teacher analogy listed above. One of the most common descriptions of boosted learning is that a group of “weak learners” can be combined to form a “strong learner”.

Applying that description to the music teacher analogy would be that a group of daily instructions (the weak learners) can be combined to produce a good quality course (the strong learner)

The combination of weak learners resulting in a strong learner description is accurate, and it does give some information. However, it leaves out some important points, such as

- What is a weak learner?
- How are they combined?
- Are some weak learners better than others?

The following analogies are intended to build your intuition on those points above before getting into the actual math of how boosting algorithms work.

### What Is A Good Weak Learner?

A weak learner is any machine learning algorithm that gives better accuracy than simply guessing. For instance, if you are trying to classify animals at a zoo, you might have an algorithm that can correctly identify zebras most of the time, but it simply guesses for any other animal. That algorithm would be a weak learner because it is better than guessing.

If you had an algorithm that identified every animal as a zebra, then that probably is not better than guessing and so it would not be a weak learner.

For boosting problems, the best kinds of weak learners are ones that are very accurate, even if it is only over a limited scope of the problem. For instance, the algorithm that correctly identifies zebras would be good. It allows you to confidently identify at least most of the zebras, allowing other weak learners to focus on the remaining animals.

A weak learner that would not be as useful is one that simply counted the number of legs an animal has. If the animal has four legs it could be a zebra, horse, alligator, or panda. If it has zero legs, it could be a fish, snake, or a worm. That kind of identification helps some, but it doesn't really narrow the scope of the problem that much for future learners.

### How Are Weak Learners Combined?

Boosting algorithms typically work by solving subsections of the problem, by peeling them away so future boosting iterations can solve the remaining sections.

Here is another analogy. Imagine you are hiring people to build your house, and you have 10 different big jobs that need to be done. A great way of doing it would be to get someone who is really good at

foundations to build the foundation. Then hire a very good carpenter to focus on the framing. Then hire a great roofer and plumber to focus on their sections of the house. At each stage, a small subsection of the project is getting completely solved.

The roofer may be completely useless at laying foundations, but as long as you use him at the right time you will be in good shape.

The contrast to that method would be to hire 10 people who are all decent at most things, but not great at anything. None of them can build you a good foundation, and if you start with one, the next one will have to come in and fix some problems with it, while at the same time doing a shoddy job framing. The third person will have to come in and make corrections to the errors that the first two left behind. You **might** get a good product at the end, but more likely you will have adequate results that still have errors in them.

The takeaway is that weak learners are best combined in a way that allows each one to solve a limited section of the problem. Any machine learning routine can be used as a weak learner. Neural nets, support vector machines or any other would work, but the most commonly used weak learner is the decision tree.

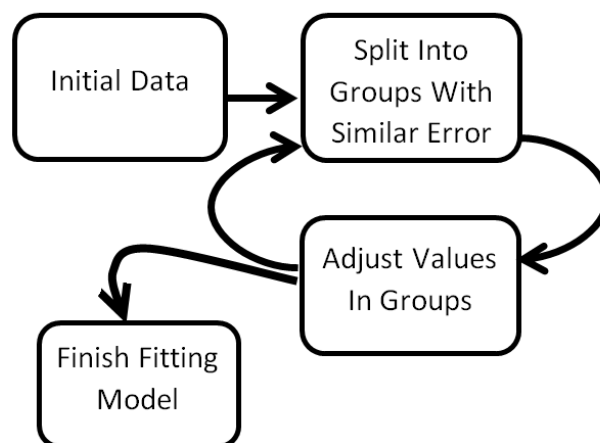
## How Boosting Really Works – Skipping Over The Details

With boosting, and with many other types of machine learning, there are two stages to using the algorithm. The first stage is to train the algorithm on data that you know the answer to. You train the model using data that has some “features” that you think will be useful in getting your desired results, as well as a final answer that you also know. This is known as “fitting” the model or the classifier using training data. By fitting the boosting model, it learns how to use the features of the data in order to create groups of data points with similar final answers. It also learns how to adjust the results within each group in order to get the known final answer.

The second stage is to use that fitted model on a set of data that has the same features as the training data, except that you don’t know the final answer. The machine learning algorithm will then operate on the unknown data the same way that it learned to based on the known data, and draw conclusions from that. The theory is that if it groups the second set of data using the same method as it did for the first set of data, and then adjusts the values in the second set of groups using the method developed by looking at the final answers in the first set of groups, that the algorithm can determine good estimated values for the second set of data.

The first stage, fitting the model, is somewhat more complicated than the second stage of predicting with a fit model. At a high level, the process we use to fit the boosting model is to start with the training data and make an initial estimate of a value for all the data points. Then we calculate how much error was in that initial estimate. (Which is only possible because we know the actual final value of the training data). Next, we attempt to group data points together, using their features to generate the groups, with the objective of making groups that have a similar amount of error within each group. Then, for each group, we calculate a *single* value and adjust all the data points in the group by that value. This creates a new value for every data point. That completes a boosting iteration. We could then be done with the training of the boosting algorithm, or we could use the new value to calculate new errors and go through the whole cycle repeatedly until the results stop improving.

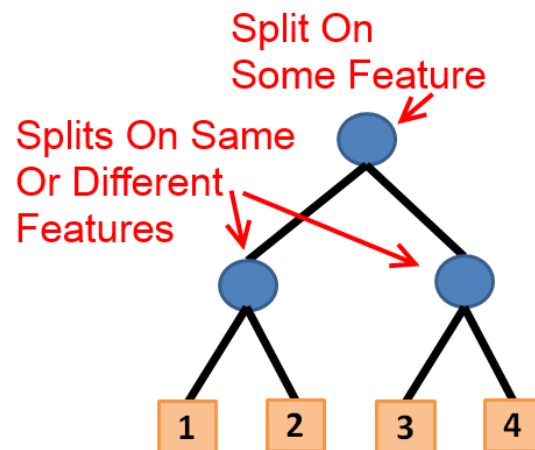
In the simplest terms, the cycle looks like this



## Splitting The Groups

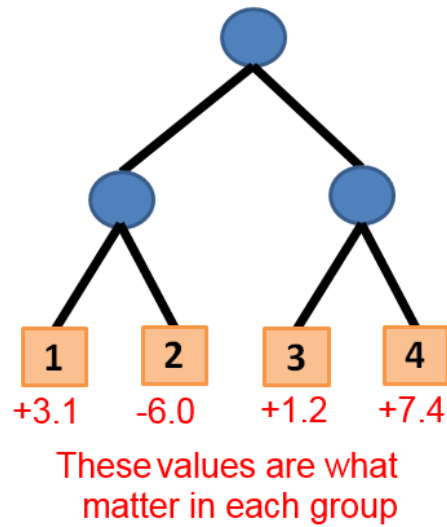
The most computationally expensive part of gradient boosting is determining the best way to group the data. The process used to group the data is regression decision trees.

The images below show an example of how the decision tree split the data into groups. What happens is that data starts at the head node (the top) and gets grouped according to how much error remains in each data point, based on the features that can be used to split the data up. The resulting groups are shown below as 1, 2, 3, 4. (Real problems could have a greater or fewer number of groups)



Each split occurs on a single feature at a specific value of that feature. Subsequent splits could use different features or different values of the same feature. At any given split, a data point will take one branch or the other based on the value it has for the feature being operated on. As a result, the splits result in multiple groups.

Each of those groups will calculate a value that will get added or subtracted from all of the data points that fall within that group. That is shown in the image below

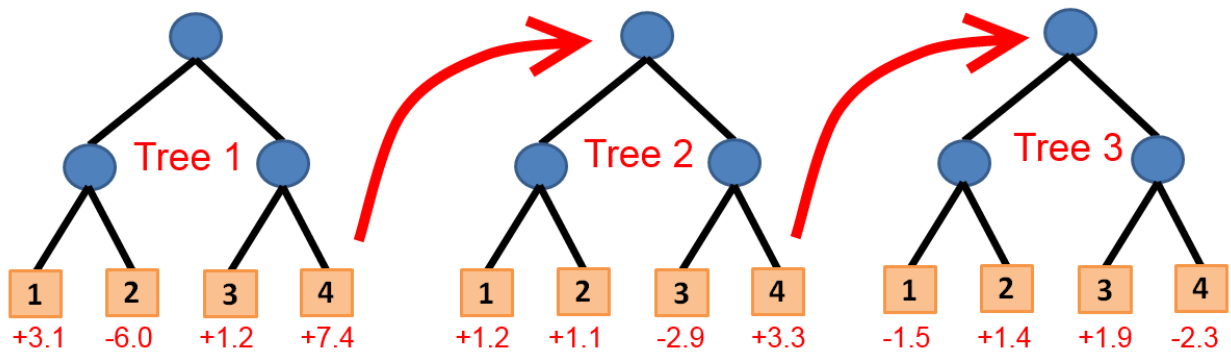


**This value is important.** It is, in fact, the whole point of making these groups. That value is a real number that will get added (or subtracted) from the current value of every member of the group, which updates the current value to a new value.

The change value will be calculated for each group in order to give the most improvement possible to the members of that group, but it will frequently be the case that some data points improve while others get worse on any given boosting cycle.

All the data points start with the same initial estimate value, but after the first boosting cycle they will have different values depending on the amount of change imposed by the group they are in. Since different points have different estimated values, they will have different error relative to their true values. As a result, in subsequent boosting cycles any given data point will not necessarily end up grouped together with the same points as it was in previous cycles. This means that, given enough boosting cycles, each data point in the training set can be improved to very closely match its true value.

Multiple sequential boosting cycles might look something like this





Where every data point in the training set follows its own path through each tree into the groups and changes its value depending on what group it lands in. Each tree will end up with groups using with different features, and the groups will have different amounts of change applied to their values. Different trees could end up with different numbers of groups.

The final results from a trained set of gradient boosted trees are the splits that created the groups for each tree, and the amount of change that gets applied to each group. Once the algorithm is trained and you have those results, you can take a different set of data, determine which groups each data point would fall into for each tree, and apply the appropriate change for each data point in the new set of data.

### That's It For The Free Sample

Per the licensing agreement with Amazon, I can only provide 10% of the book as a free sample, and this has reached the end of the 10%. The remaining 90% dives more deeply into boosting examples, and shows exactly how regression and classification works with gradient boosting.

The full book can be purchased here on Amazon [“Machine Learning With Boosting”](#) for less than \$3 (or the equivalent in other currencies.)

I gave some thought onto which 10% of the book to include as this free sample, and eventually decided to just keep the first 10%, as opposed to trying to select the section on a specific topic. The reason is that the analogies and general overview provided in the first 10% of the book can be useful by itself, even without the detailed technical content. But any given block of 10% of the detailed technical content is not as useful without the framework provided by the introduction.

If you like this book, the other pure machine learning book I have written is [Machine Learning With Random Forests And Decision Trees](#) Random Forests are another type of Machine learning algorithm where you combine a bunch of decision trees that were generated in parallel, as opposed to in series like we did with boosting. That book might also be useful to you.

If you have any questions simply write here:

ImFairlyNerdy (at) gmail (dot) com

~ Scott Hartshorn

