An introduction to XGBoost and Gradient Boosting Machines

Review by William Steimel

Sources

- XGBoost – Introduction to Boosted Trees
  - http://xgboost.readthedocs.io/en/latest/model.html
- Tree Boosting With XGBoost- Why Does XGBoost Win "Every" Machine Learning Competition? –Didrik Nielsen
  - https://brage.bibsys.no/xmlui/bitstream/handle/11250/2433761/16128_FULLTEXT.pdf
- Awesome XGBoost
  - https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions
- Gradient Boosting: Distance to Target (Tutorial by Terrence Parr and Jeremy Howard of University of San Francisco)
  - http://explained.ai/gradient-boosting/L2-loss.html
- XGBoost Parameters
  - http://xgboost.readthedocs.io/en/latest/parameter.html

Table of Contents

What is XGBoost?

- Today I will talk about XGBoost which is short for eXtreme Gradient Boosting.
- According to the GitHub release, XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable.
  - XGBoost was authored by Tianqi Chen and I suggest you read more about it from the original paper if you want to study it further.
  - https://arxiv.org/abs/1603.02754
- It implements machine learning algorithms under the Gradient Boosting framework.
  - Original Paper - Proposed by Friedman (1999) - Greedy Function Approximation: A Gradient Boosting Machine
- Recent Competition to XGBoost
  - LightGBM (Jan 2017) - Microsoft
  - CatBoost (April 2017)

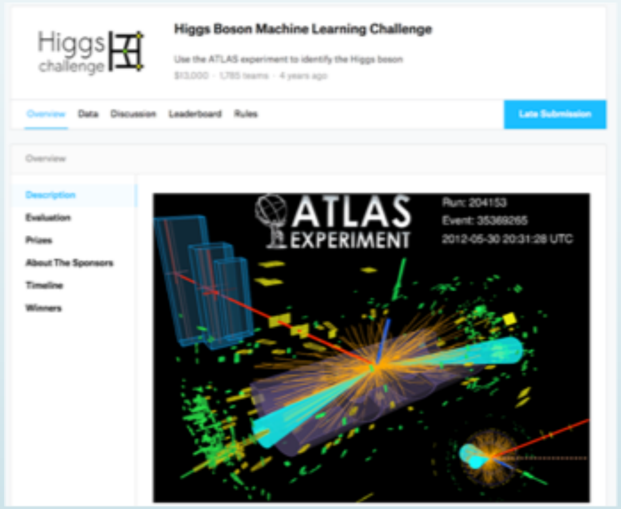
XGBoost

Benefits of XGBoost

- **Easy to use-** Installation of XGBoost is easy and you can get up to speed very quickly. The XGBoost website even provides a sample code where you can utilize XGBoost within 5 seconds.
- **Efficiency -** XGBoost is optimized for performance
- **Accuracy -** as evidenced by XGBoost's superior performance in Kaggle competitions it is also quite accurate on a wide variety of datasets.
- **Customization -** XGBoost has many hyperparameters and the ability to customize things like the objective and evaluation function. This ability to customize allows for superior performance and control.

Practical Uses of XGBoost

- The first time XGBoost was used was in 2014 during the Higgs Boson Machine Learning Challenge in which the goal was to use the ATLAS experiment to identify the Higgs boson.
- It was discovered that XGBoost returned extremely competitive results and more people began to use it.

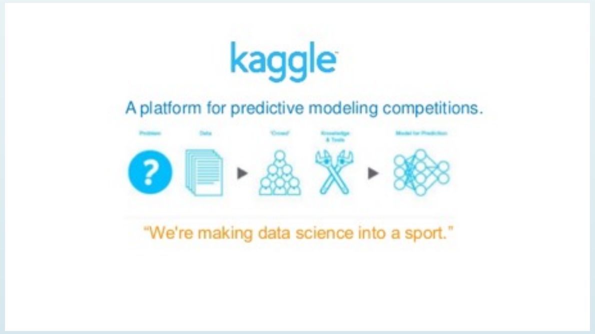

Higgs Boson Machine Learning Challenge

Use the ATLAS experiment to identify the Higgs boson

$13,000 · 1,785 teams · 4 years ago

Practical Uses of XGBoost

- In addition- research has shown that it has become somewhat popular since its inception in 2014:
  - 2015 Kaggle Competitions - There were 29 Kaggle Competitions in which 17 used XGBoost. (Chen and Guestrin, 2016)
  - KDD Cup 2015 - All of the top 10 solutions used XGBoost


kaggle
A platform for predictive modeling competitions.
"We're making data science into a sport."

Source:Tree Boosting With XGBoost- Why Does XGBoost Win "Every" Machine Learning Competition? –Didrik Nielsen

Practical Uses of XGBoost

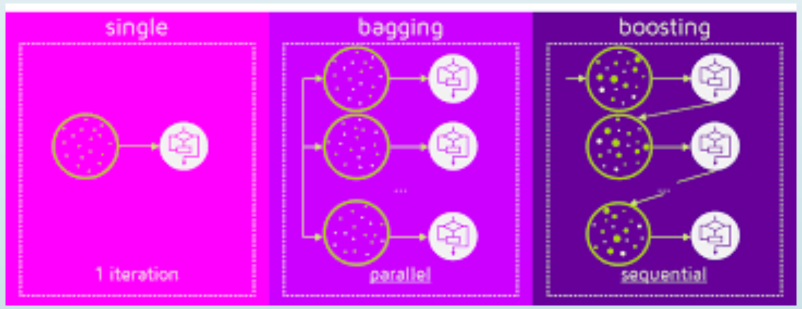
- XGBoost has also been successful at the below competitions:
  - Maksims Volkovs, Guangwei Yu and Tomi Poutanen, 1st place of the 2017 ACM RecSys challenge.
  - Vlad Sandulescu, Mihai Chiru, 1st place of the KDD Cup 2016 competition.
  - Marios Michailidis, Mathias Müller and HJ van Veen, 1st place of the Dato Truely Native? competition.
  - Vlad Mironov, Alexander Guschin, 1st place of the CERN LHCb experiment Flavour of Physics competition.
  - Josef Slavicek, 3rd place of the CERN LHCb experiment Flavour of Physics competition.
  - Mario Filho, Josef Feigl, Lucas, Gilberto, 1st place of the Caterpillar Tube Pricing competition.
  - Qingchen Wang, 1st place of the Liberty Mutual Property Inspection.
  - Chenglong Chen, 1st place of the Crowdflower Search Results Relevance.
  - Alexandre Barachant ("Cat") and Rafał Cycoń ("Dog"), 1st place of the Grasp-and-Lift EEG Detection.
  - Halla Yang, 2nd place of the Recruit Coupon Purchase Prediction Challenge.
  - Owen Zhang, 1st place of the Avito Context Ad Clicks competition.
  - Keiichi Kuroyanagi, 2nd place of the Airbnb New User Bookings.
  - Marios Michailidis, Mathias Müller and Ning Situ, 1st place Homesite Quote Conversion.
- Source: https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

What is Boosting/Gradient Boosting?

- This next section will talk a little about Ensemble Methods before discussing boosting.
- Ensemble Models are the combination of multiple machine learning models which often can outperform just using a single model when it comes to predictive performance.
- An example of a simple ensemble model would be the creation of 100 regression decision trees and averaging the results together to get a final prediction.

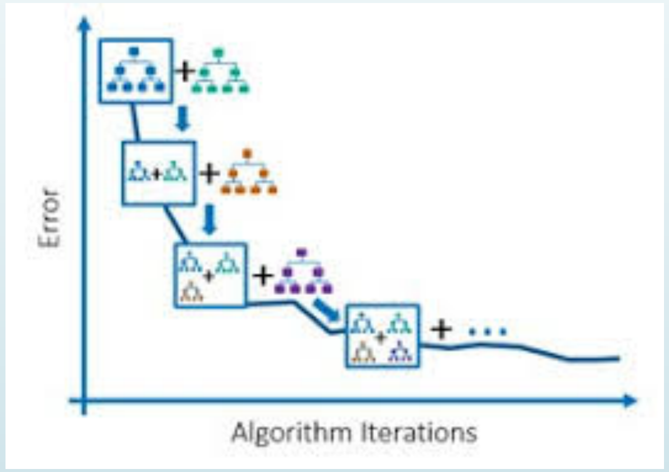
What is Boosting/Gradient Boosting?

- There are three main ensemble methods including Bagging, Boosting, and Stacking.
- Ensemble Methods are techniques generally used to reduce variance and bias in the case of Machine Learning Modeling.
- We use ensembles to turn one weak learner into a strong learner by combining multiple learners in conjunction with each other. This is why it is called ensemble learning.



What is Boosting/Gradient Boosting?

- Boosting is a general ensemble technique in which the predictors are not made independently, but sequentially which means each model learns from the mistakes of the previous model.



How to explain Gradient Boosting:

Terrence Parr and Jeremy Howard

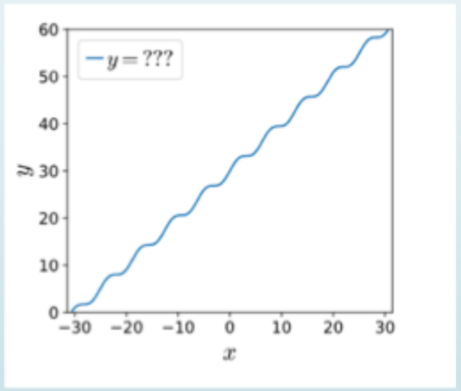
University of San Francisco

Gradient Boosting / XGBoost

- To understand XGBoost you need to further understand Gradient Boosting.
- The next section will cover the basic mathematics behind Gradient Boosting
- This tutorial contains 3 sections but I will only review section 1 of 3
  - **Gradient Boosting: Distance to target**
  - Gradient Boosting: Heading in the right direction
  - Gradient Boosting performs gradient descent
- This tutorial examines the most common form of GBM that optimizes the mean squared error (MSE) and L_2 loss or cost

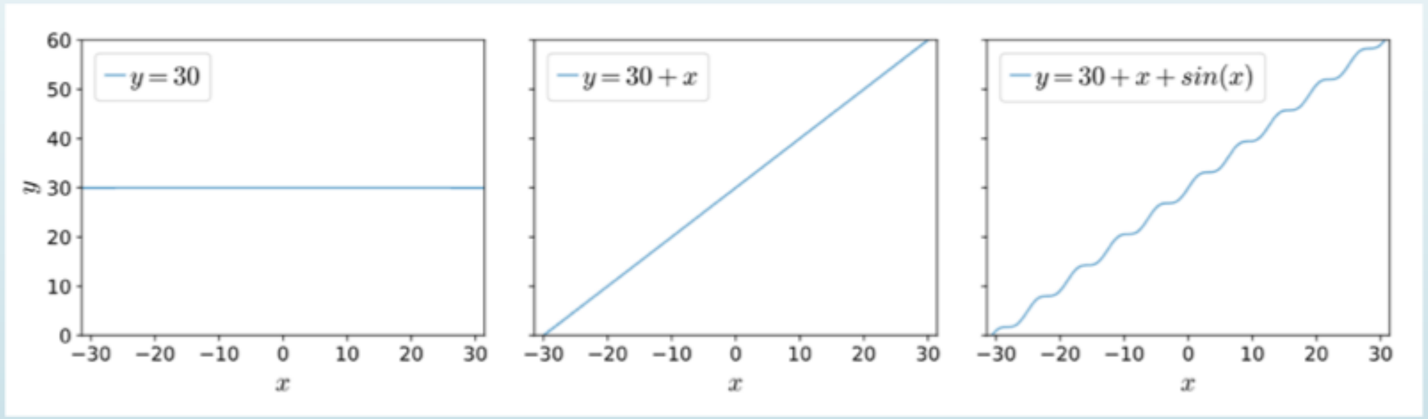
An Introduction to Additive Modeling

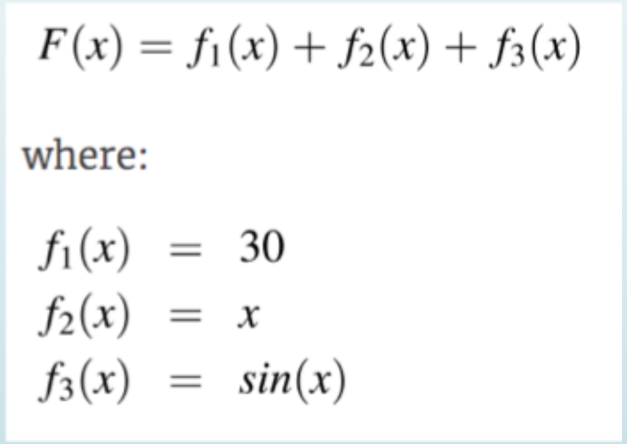
- What mathematicians call Additive Modeling is a foundation of boosting.
  - Essentially - Adding simple terms together to create a more complicated expression
- Consider the following curve that shows *y* as some unknown function of x.
- Lets assume that the function is composed of several simple terms and we are trying to guess what they are:



An Introduction to Additive Modeling

- A function can be subdivided into simpler more manageable parts
- The function on the previous slide can be described as an addition or combination of 3 sub-functions
  - 30
  - x
  - $\sin(x)$





$$F(x) = f_1(x) + f_2(x) + f_3(x)$$

where:

$$\begin{aligned} f_1(x) &= 30 \\ f_2(x) &= x \\ f_3(x) &= sin(x) \end{aligned}$$

An Introduction to Additive Modeling

- Mathematicians describe the decomposition of a function into the addition of M subfunctions through the below formula:

$$F_M(\mathbf{x}) = f_1(\mathbf{x}) + \ldots + f_M(\mathbf{x}) = \sum_{m=1}^{M} f_m(\mathbf{x})$$

- In Machine Learning, the goal is to model a function that draws the best curve through a set of data points. (x, y)
- "Adding up a bunch of subfunctions to create a composite function that models some data points is then called *additive modeling*"
  - *Gradient Boosting Machines use additive modeling to sequentially build a model that more closely fits the data through the combination of simple sub-models*

An Introduction to Boosted Regression

- **Boosting**- combines multiple simple models into a single composite model
  - The combination of many “weak learners” in a step wise/sequential fashion creates an overall stronger model/predictor
- In Regression problems, Given feature vector x we are typically trying to learn a scalar target value y for a bunch of (x_i, y_i) pairs
- Given a single feature vector x and scalar target value y for a single observation, we can express a composite model that predicts (as the addition of M weak models :

$$\hat{y} = \sum_{m=1}^{M} f_m(\mathbf{x})$$

An Introduction to Boosted Regression

- However, the models are not only decision trees and can be anything from K-nearest neighbors, regression trees etc.
- Boosting adds weak models in a step wise fashion improving on the previous model at each step
  - Boosting is considered greedy as choosing $f_m(x)$ never alters the previous function.
  - We choose to stop adding models when the performance of $\hat{y} = f_m(x)$ is good enough or at a designated threshold or when $f_m(x)$ does not add any more benefit
  - M is usually a hyperparameter of number of stages
    - (In the case of XGBoost it is the number of boosted trees)
- Because greedy strategies often choose one weak model at a time Boosting Models can also be expressed in the below format:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + f_m(\mathbf{x})$$

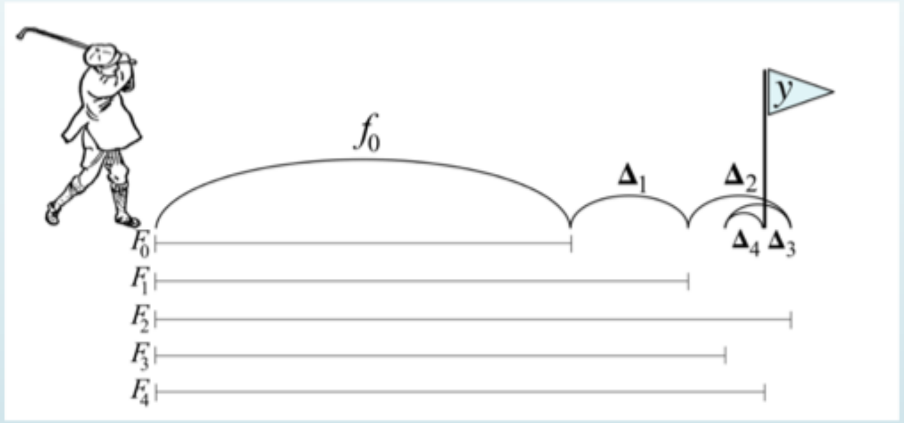
The Intuition behind gradient boosting

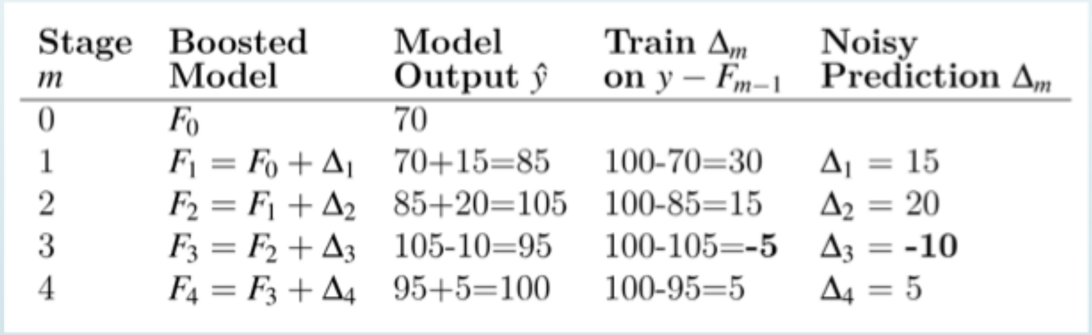
- To construct a boosted regression model we start by creating a model that approximates y based on a given feature vector x
- Then we gradually nudge (move) the overall $F_m(x)$ model towards the known target value by adding one or more tweaks/changes $\Delta_m(x)$.

$$\begin{aligned} \hat{y} &= f_0(\mathbf{x}) + \Delta_1(\mathbf{x}) + \Delta_2(\mathbf{x}) + \ldots + \Delta_M(\mathbf{x}) \\ &= f_0(\mathbf{x}) + \sum_{m=1}^{M} \Delta_m(\mathbf{x}) \\ &= F_M(\mathbf{x}) \end{aligned}$$

The Intuition behind gradient boosting

- The author presents this by example of a golfer whacking a golf ball towards a hole to more easily understand gradient boosting.
- At each swing (step) the golfer is trying to get closer to the hole.
- The chart/table represents the model at each step

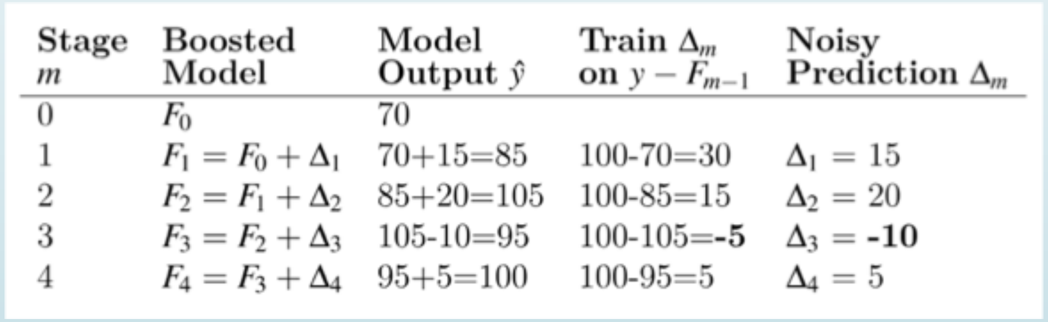




Stage m	Boosted Model	Model Output $\hat{y}$	Train Δ_m on $y - F_{m-1}$	Noisy Prediction Δ_m
0	F_0	70		
1	$F_1 = F_0 + \Delta_1$	70+15=85	100-70=30	$\Delta_1 = 15$
2	$F_2 = F_1 + \Delta_2$	85+20=105	100-85=15	$\Delta_2 = 20$
3	$F_3 = F_2 + \Delta_3$	105-10=95	100-105=**-5**	$\Delta_3 =$ **-10**
4	$F_4 = F_3 + \Delta_4$	95+5=100	100-95=5	$\Delta_4 = 5$

The Intuition behind gradient boosting

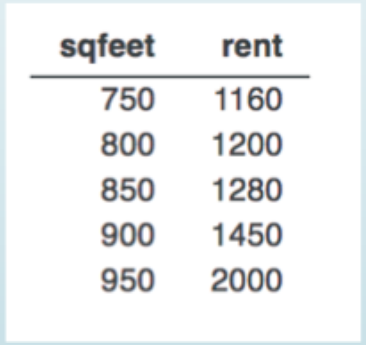
- This example only works with one observation for illustrative purposes:
  - After the initial stroke, the golfer determines the appropriate nudge by computing the difference between y and the first approximation $y - f_0(x)$,
- The difference is usually called the residual or residual vector
  - The residual helps push us from the current prediction toward the actual y value
- As an example, let's say that the hole is at y=100 yards, $f_0(x) = 70$ Manually boosting, we might see a sequence like the below table, depending on the imprecise Δ_m strokes made by the golfer:



Stage m	Boosted Model	Model Output $\hat{y}$	Train Δ_m on $y - F_{m-1}$	Noisy Prediction Δ_m
0	F_0	70		
1	$F_1 = F_0 + \Delta_1$	70+15=85	100-70=30	$\Delta_1 = 15$
2	$F_2 = F_1 + \Delta_2$	85+20=105	100-85=15	$\Delta_2 = 20$
3	$F_3 = F_2 + \Delta_3$	105-10=95	100-105=**-5**	$\Delta_3 =$ **-10**
4	$F_4 = F_3 + \Delta_4$	95+5=100	100-95=5	$\Delta_4 = 5$

Gradient Boosting regression by example

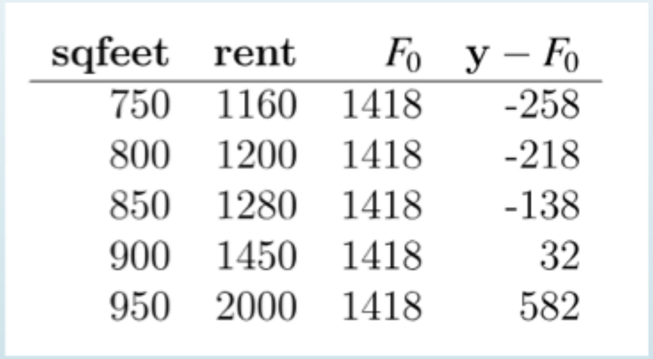
- The next section illustrates Gradient Boosting by through a practical problem that tries to predict rent based on apartment sq. feet. (in Japan meters squared)
- Training data:
  - From this data, we'd like to build a GBM to predict rent price given square footage
- To move towards y from $\hat{y}$, we need a direction vector.



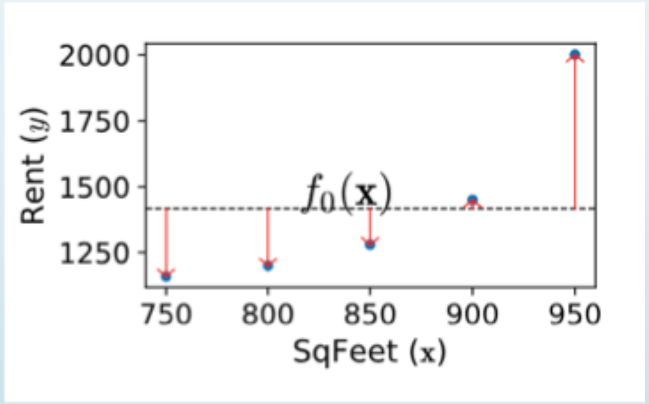
sqfeet	rent
750	1160
800	1200
850	1280
900	1450
950	2000

Gradient Boosting regression by example

- The author uses the mean/average of all 5 examples as the initial model:
  - $F_0(x_i) = f_0(x_i) = 1418$
  - The mean is used as it is the single value that minimizes the the mean squared error between it and the y_i values.
- Once we have F_0, we compute $y - F_0$, which is the residual between the target and the previous estimate:



sqfeet	rent	F_0	$y - F_0$
750	1160	1418	-258
800	1200	1418	-218
850	1280	1418	-138
900	1450	1418	32
950	2000	1418	582



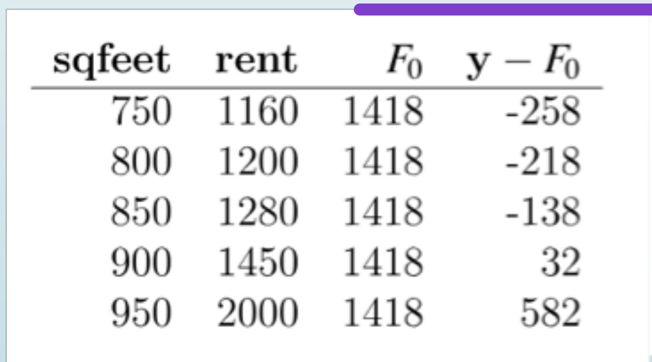
Residuals vs Rent Target Values

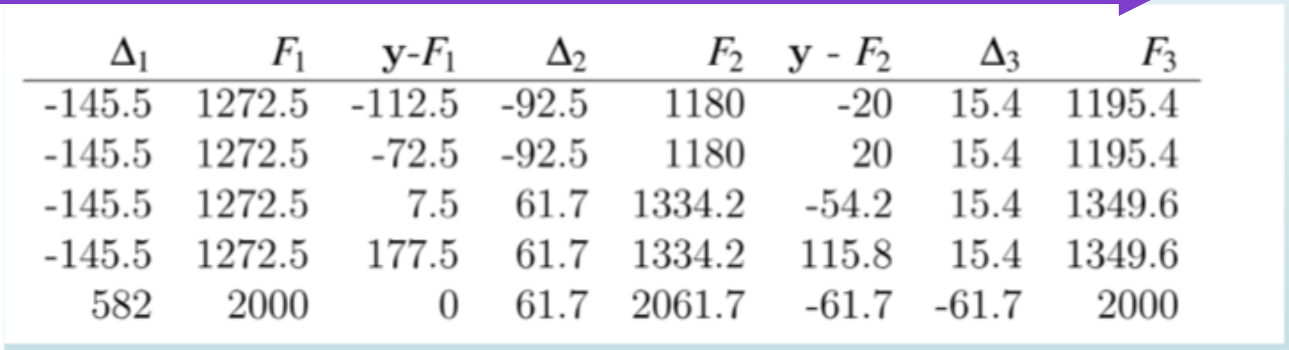
Gradient Boosting regression by example

- Next a weak model, Δ_1 is trained to predict the residual vector given x_1 for all observations.
- For simplicity this example assumes a learning rate of $\eta = 1.0$ which will be discussed in other tutorials.

$$F_m(X) = F_{m-1}(X) + \eta\Delta_m(X)$$

- Since we have a learning rate of 1 the above can be written like this :
  - $F_1 = F_0 + \Delta_1$, $F_2 = F_1 + \Delta_2$
- The table below summarizes these relationships in step wise order:





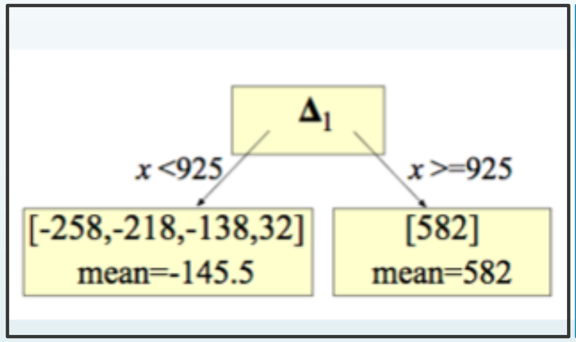
sqfeet	rent	F_0	$y - F_0$	Δ_1	F_1	y-F_1	Δ_2	F_2	y - F_2	Δ_3	F_3
750	1160	1418	-258	-145.5	1272.5	-112.5	-92.5	1180	-20	15.4	1195.4
800	1200	1418	-218	-145.5	1272.5	-72.5	-92.5	1180	20	15.4	1195.4
850	1280	1418	-138	-145.5	1272.5	7.5	61.7	1334.2	-54.2	15.4	1349.6
900	1450	1418	32	-145.5	1272.5	177.5	61.7	1334.2	115.8	15.4	1349.6
950	2000	1418	582	582	2000	0	61.7	2061.7	-61.7	-61.7	2000

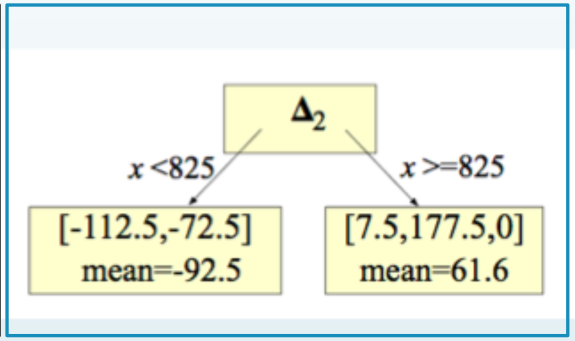
Gradient Boosting regression by example

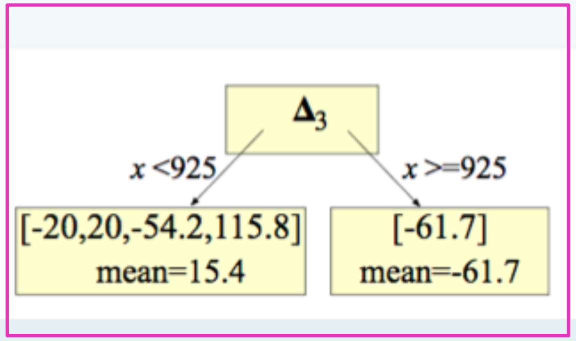
- These are the respective regression tree stumps for the previous slide:

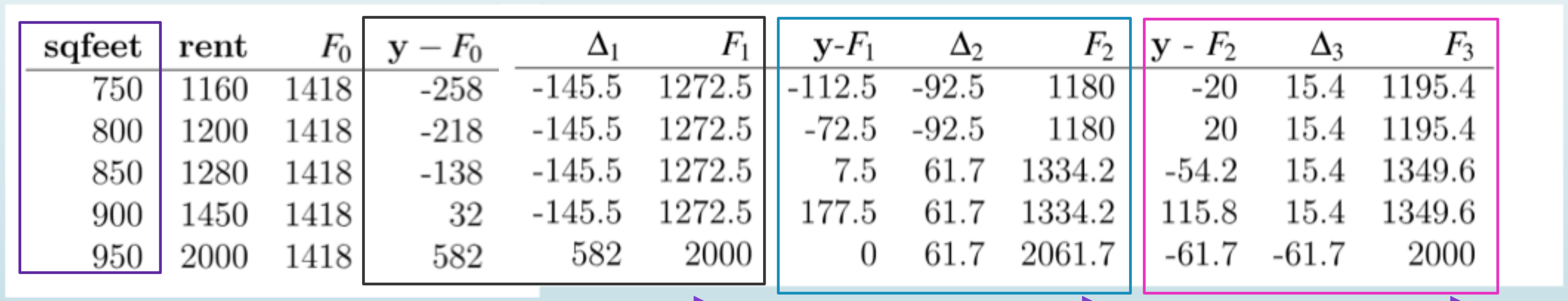
Decision Split Criteria: based on minimized variance

Mean of Residuals







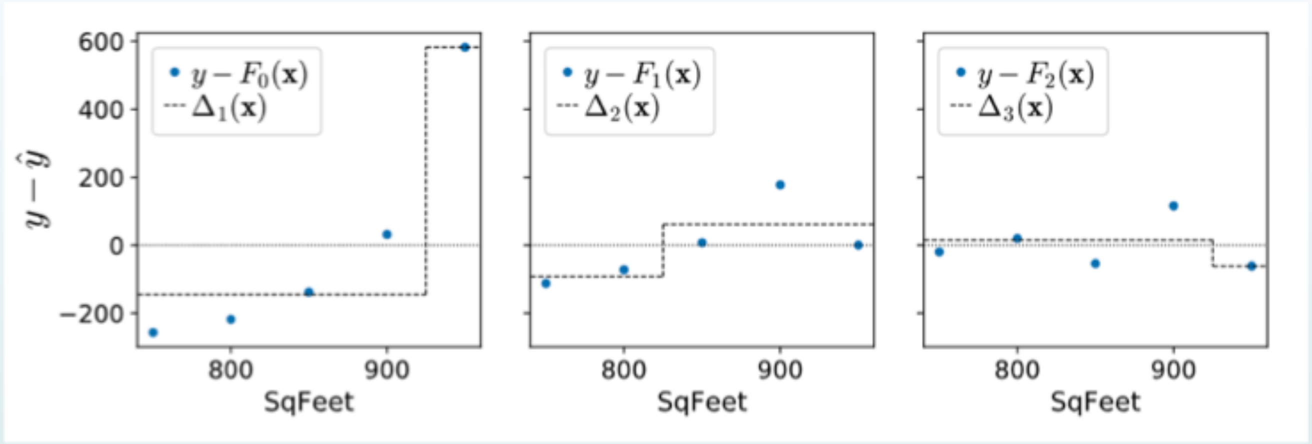


sqfeet	rent	F_0	$y - F_0$	Δ_1	F_1	y-F_1	Δ_2	F_2	y - F_2	Δ_3	F_3
750	1160	1418	-258	-145.5	1272.5	-112.5	-92.5	1180	-20	15.4	1195.4
800	1200	1418	-218	-145.5	1272.5	-72.5	-92.5	1180	20	15.4	1195.4
850	1280	1418	-138	-145.5	1272.5	7.5	61.7	1334.2	-54.2	15.4	1349.6
900	1450	1418	32	-145.5	1272.5	177.5	61.7	1334.2	115.8	15.4	1349.6
950	2000	1418	582	582	2000	0	61.7	2061.7	-61.7	-61.7	2000

Gradient Boosting regression by example

Key Notes:

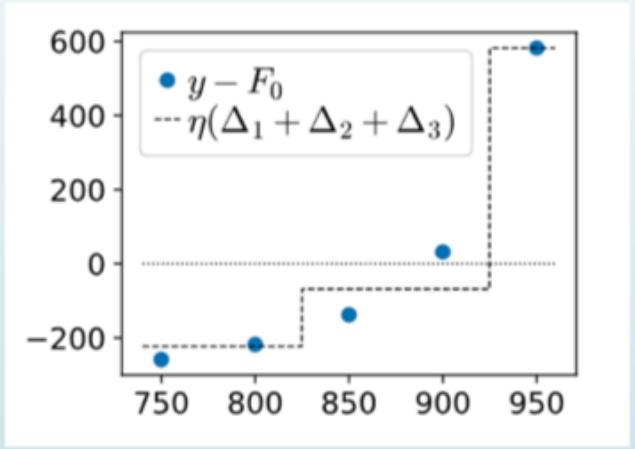
- We are always training on the residual vector $y - F_{m-1}$
- The best way to visualize the learning of $y - F_{m-1}$ by weak models Δ_m, is by looking at at the residual vectors and model predictions on the same y-axis
- With each model you can see the residuals get smaller as more weak models are added
  - Blue dots – residual vector used to train Δ_m weak models
  - Dashed lines – predictions made by Δ_m

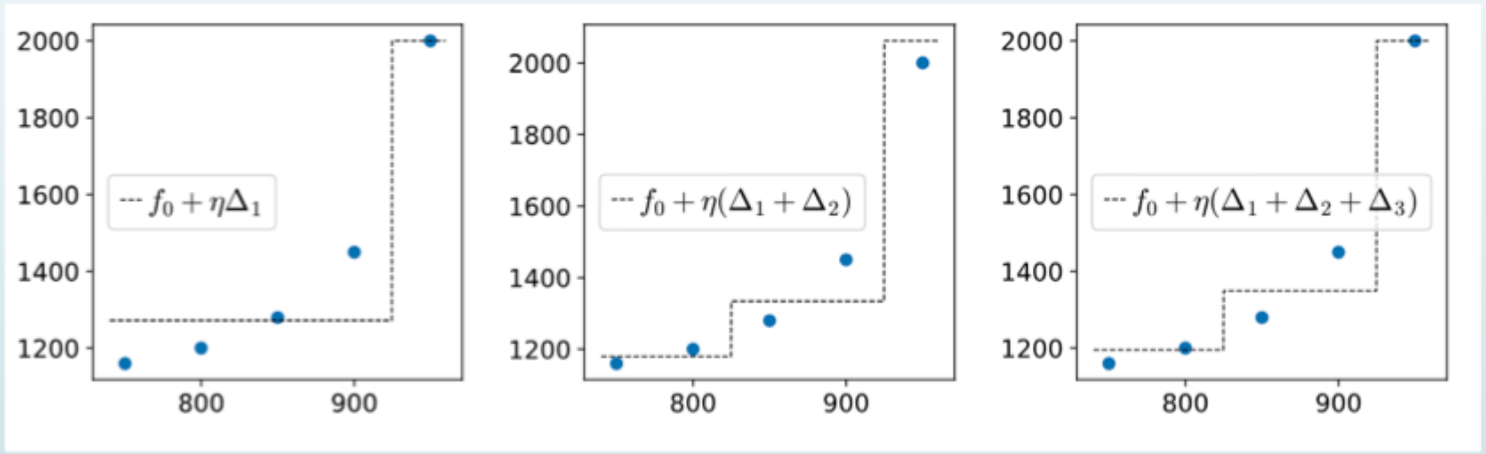


Gradient Boosting regression by example

- The composite model sums together all of the weak models

- If we add all of the weak models to the initial f_0 average model, the full composite model is a pretty good predictor of the actual rent values





Measuring Model Performance

- To measure model performance like in any machine learning problem we define a loss or cost function.
- This is typically the loss across all N observations of prediction $\hat{y}$ instead of y

$$L(\mathbf{y}, F_M(X)) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, F_M(\mathbf{x}_i))$$

- In this case we are using the mean squared error (MSE) which is common for regression problems – The goal is to minimize

$$L(\mathbf{y}, F_M(X)) = \frac{1}{N}\sum_{i=1}^{N} (y_i - F_M(\mathbf{x}_i))^2$$

Conclusion

- This tutorial covered some simple basics of Gradient Boosting
- The other two tutorials cover go into further details regarding gradient boosting which I will try to cover in the future.
- https://www.youtube.com/watch?v=0Xc9LIb_HTw

XGBoost Hyperparameters

Choosing Hyperparameters

- Firstly, Hyperparameters function like knobs or settings for machine learning algorithms and any modification to them can impact a model's performance positively or negatively.
- I would define Hyperparameter tuning to be the process used to find the optimal hyperparameters.
- If you read the below link you can see general XGBoost has many hyperparameters
  - http://xgboost.readthedocs.io/en/latest/parameter.html
- There are three types of parameters to be tuned with XGboost defined by the authors including general parameters, booster parameters and task parameters.
  - General Parameters - Guide the overall Function and Boost method
  - Booster Parameters - Parameters depending on which booster you have chosen (The tree booster is commonly used)
  - Task Parameters - Guide the optimization/Learning (learning task/objective)
- Common Methods for Hyperparameter Tuning
  - Grid Search
  - Random Search
  - Bayesian Optimization

Commonly Tuned Hyperparameters

- **1. Number of Trees**
  - n_estimators (Number of Boosted Trees)
- **2. Tree Complexity**
  - max_depth - is the maximum number of nodes from the root of the tree until the farthest leaf of the tree. A max depth that is too high can sometimes lead to overfitting as it leads to more granular tree splits.
  - min_child_weight- the minimum weight required to create a new node in a tree. A lower value leads to increased tree complexity which may lead to overfitting.
- **3. "Lagrangian multiplier" (complexity control- functions similar to regularization)**
  - Gamma - The Gamma parameter is another type of regularization for gradient boosting problems and considered a complexity control.
- **3. Data sampling at each boosting round.**
  - Subsample - fraction of rows to subsample at each boosting step (example: 1 uses all rows, .5 uses half)
  - colsample_bytree - fraction of columns to subsample at each boosting step (example: 1 uses all rows, .5 uses half)
- **4. Regularization**
  - reg_alpha
  - reg_lambda
- **5. Learning Rate**
  - (eta) learning_rate – Affects how our model converges to the local minima

Other Useful Resources

Gradient Boosting Playground

- I recommend the below link if you are interested in playing with gradient boosting and understanding each trees sequential formation
- Gradient Boosting Playground by Alex Rogozhnikov
  - http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html

Useful Resources to learn XGBoost/Gradient Boosting

- http://explained.ai/gradient-boosting/index.html
- http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html
- https://github.com/dmlc/xgboost
- https://xgboost.readthedocs.io/en/latest/model.html
- http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/