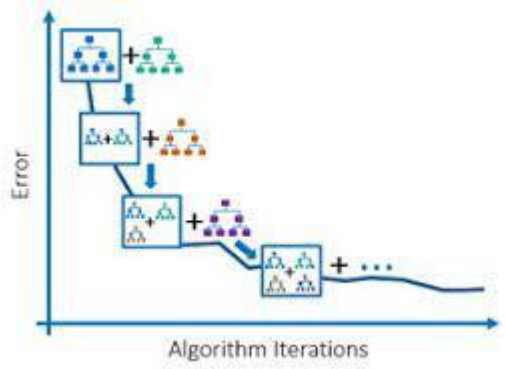
Boosting Algorithm: Gradient Boosting

Each new model gradually minimizes the loss function of the whole system using Gradient Descent method.
The learning procedure consecutively fit new models to provide a more accurate estimate of the response variable.

There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).



XGBoost algorithm was developed as a research project at the University of Washington. Tianqi Chen and Carlos Guestrin presented their paper at a Conference in 2016 and caught the Machine Learning world by fire.

XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

A **model** in supervised learning usually refers to the mathematical structure of by which the prediction Yi is made from the input xi.

A common example is a *linear model*, where the prediction is given as $\hat{y}\, i=\sum j\theta jxij$,
a linear combination of weighted input features

The **parameters** are the undetermined part that we need to learn from data. In linear regression problems, the parameters are the coefficients θ. Usually we will use θ to denote the parameters

θ will be identified by minimizing the loss which is an objective function.

Objective Function : Training Loss + Regularization

We need to find a way to find the best parameters given the training data. In order to do so, we need to define objective function, to measure the performance of the model given a certain set of parameters.

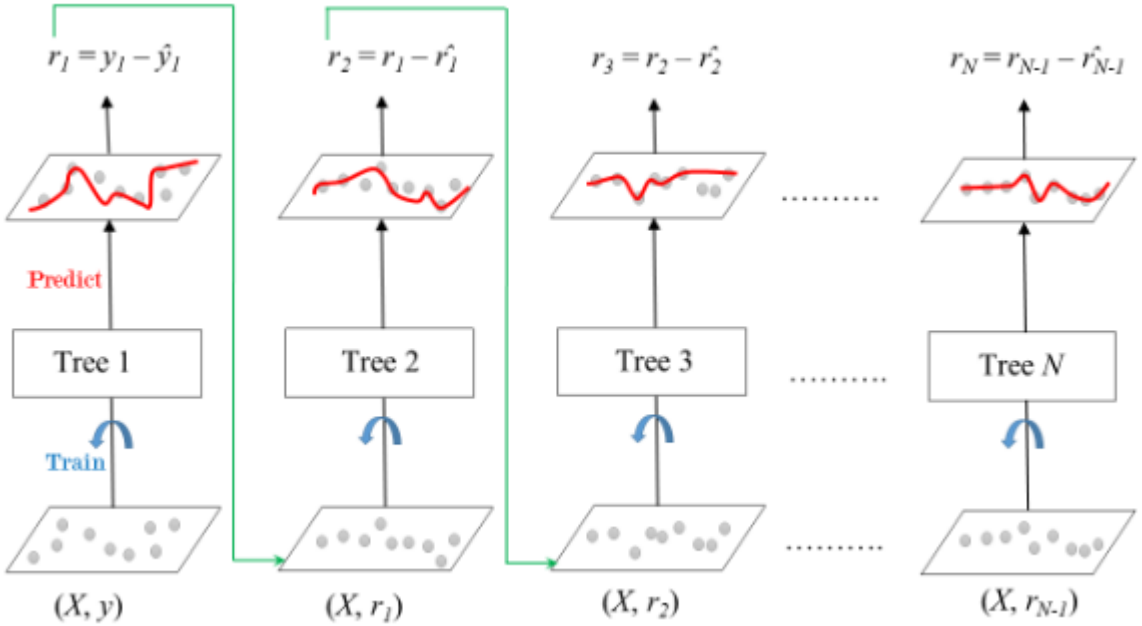
A very important fact about objective functions is they must always contain two parts: training loss and regularization

$$Obj(\Theta) = L(\theta) + \Omega(\Theta)$$

where L is the training loss function, and Ω is the regularization term. The training loss measures how predictive our model is on training data.

For example, a commonly used training loss is mean squared error

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2$$



$$Residual\ standard\ error = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{df}}$$

The ensemble consists of N trees. Tree1 is trained using the feature matrix X and the labels y. The predictions labelled y1(hat) are used to determine the training set residual errors r1. Tree2 is then trained using the feature matrix X and the residual errors r1 of Tree1 as labels. The predicted results r1(hat) are then used to determine the residual r2. The process is repeated until all the N trees forming the ensemble are trained.

XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data.
XGBoost is an implementation of gradient boosted **decision trees** designed for speed and performance.

Advantage

•Boosting technique takes care of the weightage of the higher accuracy sample and lower accuracy sample and then gives the combined results.
•Net error is evaluated in each learning steps. It works good with interactions.
•Boosting technique helps when we are dealing with bias or underfitting in the data set.
•Multiple boosting techniques are available. For example: AdaBoost, LPBoost, XGBoost, GradientBoost, BrownBoost

Disadvantage

• It increases the complexity of the classification.
•Time and computation can be a bit expensive.
.

LGBM

Light GBM uses a novel technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value

In AdaBoost, the sample weight serves as a good indicator for the importance of samples. However, in Gradient Boosting Decision Tree (GBDT), there are no native sample weights, and thus the sampling methods proposed for AdaBoost cannot be directly applied.
Here comes gradient-based sampling

Gradient represents the slope of the tangent of the loss function, so the idea is, if gradient of data points are large in some sense, these points are important for finding the optimal split point as they have higher error.

GOSS keeps track of all the instances with large gradients and performs random sampling on the instances with small gradients.

For example, let's say I have 500K rows of data where 10k rows have higher gradients. So the algorithm will choose (10k rows of higher gradient+ x% of remaining 490k rows chosen randomly).

Let's say x is 10%, total rows selected are 59k out of 500K based on which split value if found

This is a method that is employed exclusively in lightGBM. The essential observation behind this method is that not all data points contribute equally to training; data points with small gradients tend to be more well trained (close to a local minima). This means that it is more efficient to **concentrate on data points with larger gradients**

The most straightforward way to use this observation is to simply ignore data points with small gradients when computing the best split. However, this has the risk of leading to biased sampling, changing the distribution of data. For instance, if data that belonged to the "young" age group tended to be less well trained, the sampled data will have a much younger age distribution. This means that the split is likely to be younger than the optimal value

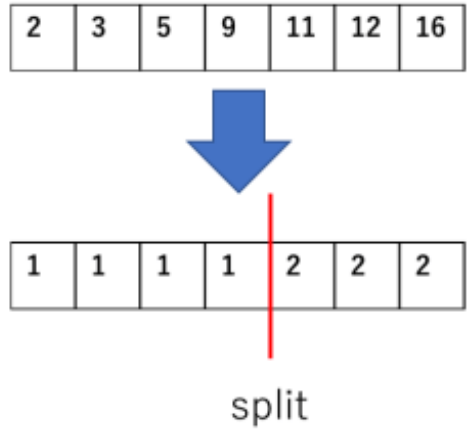
Finding the Best Split:

This step requires the algorithm to go through every feature of every data point. The computational complexity is **O(nxn)**

For instance, a tf-idf matrix of a million documents with a vocabulary size of 1 million would have a *trillion* entries.
Thus, a naive GBDT would take forever to train on such datasets

Histogram-based methods:

Firstly the number of bins creates a trade-off between speed and accuracy:
the more bins there are, the more accurate the algorithm is, but the slower it is as well.



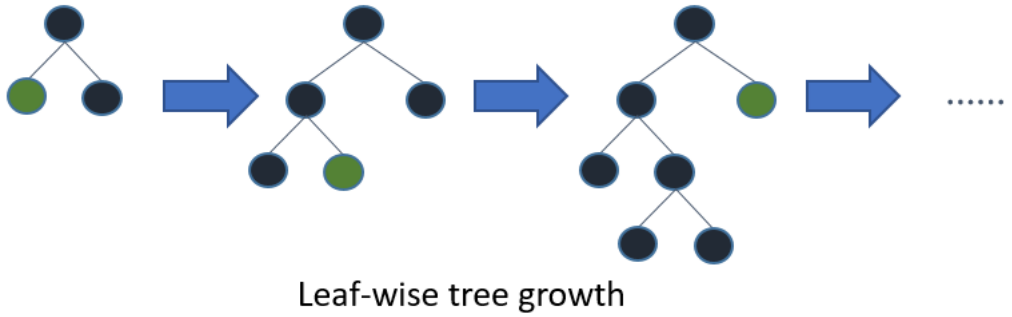
Ignoring sparse inputs:
lightGBM internal table data tend to be sparse. Since the vast majority of the values will be 0. These zeroes will be ignored.
This reduces the number of samples that have to be used when evaluating each split, speeding up the training process.

Exclusive Feature Bundling

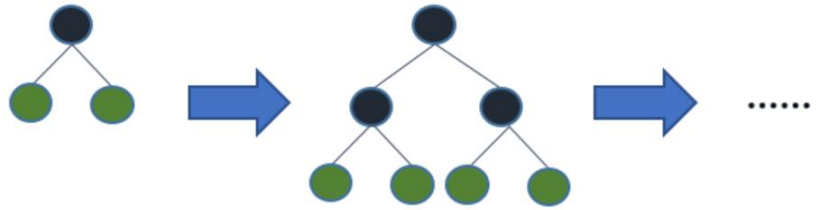
The essential observation behind this method is that the sparsity of features means that **some features are never non-zero together**.
For instance, the words "Python" and "protein" might never appear in the same document in the data.
This means that these features can be "bundled" into a single feature without losing any information.
Suppose the tf-idf score for "Python" ranges from 0 to 10 and the tf-idf score for "protein" ranges from 0 to 20.
In this case, the feature

$$\text{Python} \quad if \quad \text{protein} = 0 \quad else \quad \text{protein} + 10$$

Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree **leaf-wise** while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm



Leaf-wise tree growth



Level-wise tree growth