# Gradient Boosting

## Gradient Boosting

- Gradient boosting involves three elements:
    - A weak learner to make predictions.
    - An additive model to add weak learners to minimize the loss function.
    - A loss function to be optimized

# Weak Learner

- Decision trees are used as the weak learner in gradient boosting.
- Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and “correct” the residuals in the predictions.
- Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss.
- Initially, such as in the case of AdaBoost, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels.
- It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes.
- This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

## Additive Model

- Trees are added one at a time, and existing trees in the model are not changed.
- A gradient descent procedure is used to minimize the loss when adding trees.
- Traditionally, gradient descent is used to minimize loss and optimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.
- Instead of parameters, we have weak learner decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss).
- Generally this approach is called functional gradient descent or gradient descent with functions
- The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model.
- A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset

## Loss Function

- The loss function used depends on the type of problem being solved.
- It must be differentiable, but many standard loss functions are supported and you can define your own.
- For example, regression may use a squared error and classification may use logarithmic loss.
- A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

## XGBoost

# XGBoost

- XGBoost stands for e**X**treme **G**radient **B**oosting
  - *The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms*
- It is an implementation of gradient boosting machines created by Tianqi Chen, now with contributions from many developers
- XGBoost is a software library that you can download and install on your machine, then access from a variety of interfaces
- **The library is laser focused on computational speed and model performance**, as such there are few frills. Nevertheless, it does offer a number of advanced features

## System Features

- The library provides a system for use in a range of computing environments, not least:
  - **Parallelization** of tree construction using all of your CPU cores during training.
  - **Distributed Computing** for training very large models using a cluster of machines.
  - **Out-of-Core Computing** for very large datasets that don’t fit into memory.
  - **Cache Optimization** of data structures and algorithm to make best use of hardware.

# Algorithm features

- The implementation of the algorithm was **engineered for efficiency of compute time and memory resources**. A design goal was to make the best use of available resources to train the model. Some key algorithm implementation features include:
  - **Sparse Aware** implementation with automatic handling of missing data values.
  - **Block Structure** to support the parallelization of tree construction.
  - **Continued Training** so that you can further boost an already fitted model on new data.
- XGBoost is free open source software