XG BOOST

Umang Jain
20csu116

DEFINITION & TYPES OF BOOSTING ALGORITHM

- The term ‘Boosting’ refers to a family of algorithms which converts weak learner to strong learners.

There are many boosting algorithms which use other types of engine such as:

AdaBoost (Adaptive Boosting)

Gradient Tree Boosting

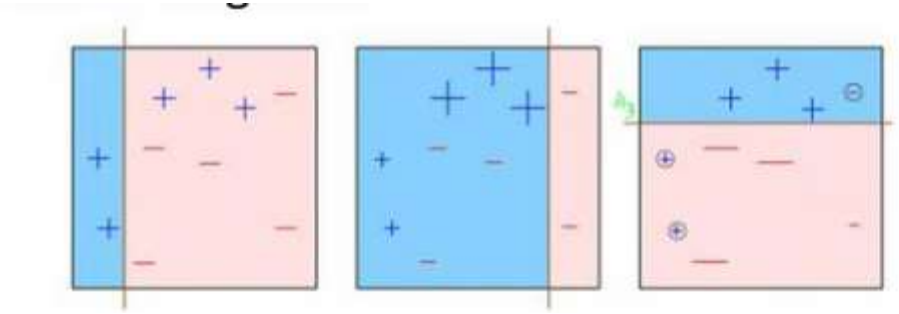
XGBoost

ADABOOST (ADAPTIVE BOOSTING)

- It fits a sequence of weak learners on different weighted training data. It starts by predicting original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy.
- Mostly, we use decision stamps with AdaBoost. But, we can use any machine learning algorithms as base learner if it accepts weight on training data set. We can use AdaBoost algorithms for both classification and regression problem.

LET'S TRY TO VISUALIZE ONE CLASSIFICATION PROBLEM

Look at the below diagram :



We start with the first box. We see one vertical line which becomes our first week learner. Now in total we have 3/10 mis-classified observations. We now start giving higher weights to 3 plus mis-classified observations. Now, it becomes very important to classify them right. Hence, the vertical line towards right edge. We repeat this process and then combine each of the learner in appropriate weights.

GRADIENT BOOSTING

- In gradient boosting, it trains many models sequentially. Each new model gradually minimizes the loss function (y = ax + b + e, e needs special attention as it is an error term) 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.
- The principle idea behind this algorithm is to construct new base learners which can be maximally correlated with negative gradient of the loss function, associated with the whole ensemble.

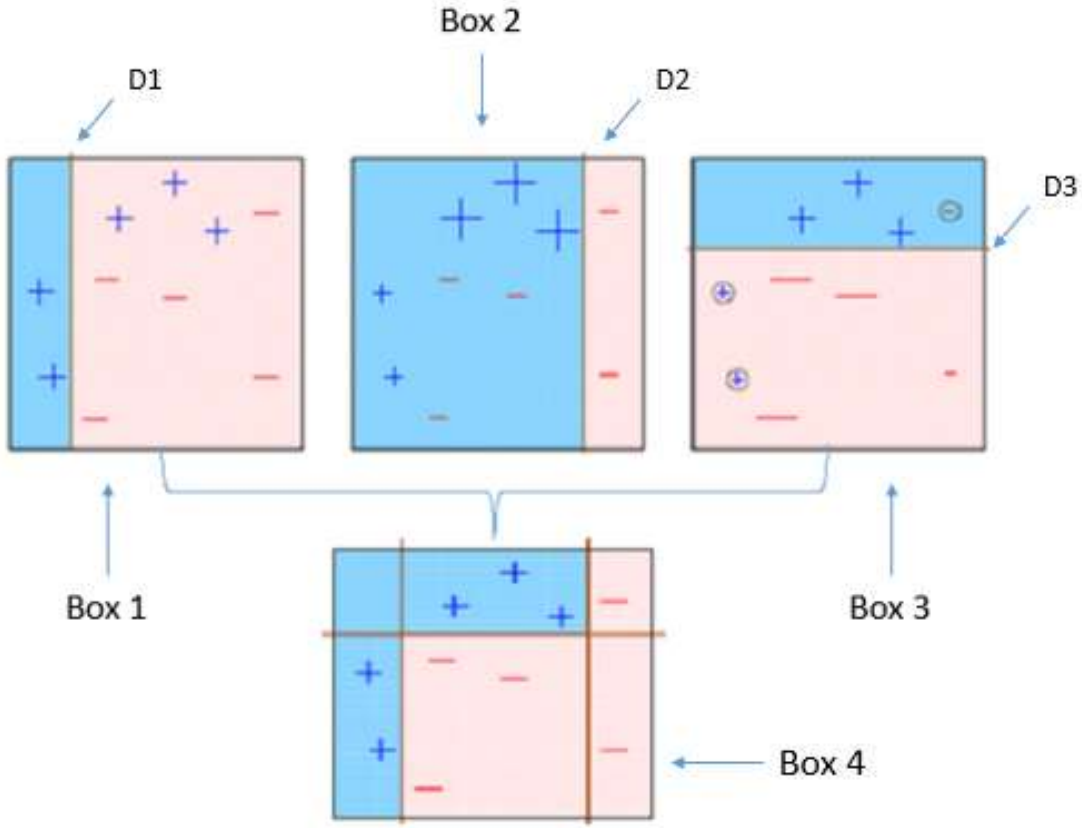
WHAT IS XGBOOST?

Stands for:

- e**X**treme **G**radient **B**oosting.

XGBoost is a powerful iterative learning algorithm based on gradient boosting.

Robust and highly versatile, with custom objective loss function compatibility.



XGBOOST = EXTREME GRADIENT BOOSTING

A machine learning library built around an efficient implementation of boosting for tree models (like GBM) and was developed by Tianqi Chen (Uni. Washington) in 2014.

► Core library in C++, with interfaces for many languages/platforms ▪ C++, Python, R, Julia, Java, etc. ▪ Distributed version for Hadoop + Spark

► Engineering goal: “Push the limit of computational resources for boosted tree algorithms” ▪ Parallelizable, cheap on memory, scales to large data sets

► Very powerful and flexible – lots of (hyper)parameters

► Huge succsess ▪ «Winning practically every prediction competiton on Kaggle»

XGBOOSTING (EXTREME GRADIENT BOOSTING)

What is the difference between the R gbm (gradient boosting machine) and xgboost (extreme gradient boosting)?

Both xgboost and gbm follows the principle of gradient boosting. There are however, the difference in modeling details. Specifically, xgboost used a more regularized model formalization to control over-fitting, which gives it better performance.

Objective Function : Training Loss + Regularization

The regularization term controls the complexity of the model, which helps us to avoid overfitting.

WHY USE XGBOOST?

All of the advantages of gradient boosting, plus more.

Two main reasons for use:

1. Low Runtime
2. High Model Performance
3. Easy to use
4. Easy to install.
5. Highly developed R/python interface for users.
6. Efficiency
7. Automatic parallel computation on a single machine..
8. Can be run on a cluster.
9. Accuracy:- Good result for most data sets.

PARAMETER INTRODUCTION

XGBoost has plenty of parameters. We can group them into

1. General parameters
   - Number of threads
2. Booster parameters
   - Stepsize
   - Regularization
3. Task parameters
   - Objective
   - Evaluation metric

PARAMETER INTRODUCTION

General parameters:

- nthread
  - Number of parallel threads.

- Booster
  - -gbtree : tree-based model.
  - gblinear : linear function.

PARAMETER INTRODUCTION

1. Parameter for Tree Booster

- max_depth

  -Maximum depth of a tree.

  -Range [1, co], default 6

- min_child_weight

  -Minimum sum of instance weight needed in a child.

  -Range [0, ∞o], default 1

- max_delta_step

  -Maximum delta step we allow each tree's weight estimation to be.

  -Range [0, co], default 0

PARAMETER INTRODUCTION

1. Parameter for Tree Booster

- Eta

-Step size shrinkage used in update to prevents overfitting.

-Range in [0,1], default 0.3

- gamma

-Minimum loss reduction required to make a split.

-Range [0, ∞o], default 0

PARAMETER INTRODUCTION

1. Parameter for Tree Booster

- Subsample

  -Subsample ratio of the training instance.

  -Range (0, 1), default 1

- colsample_bytree

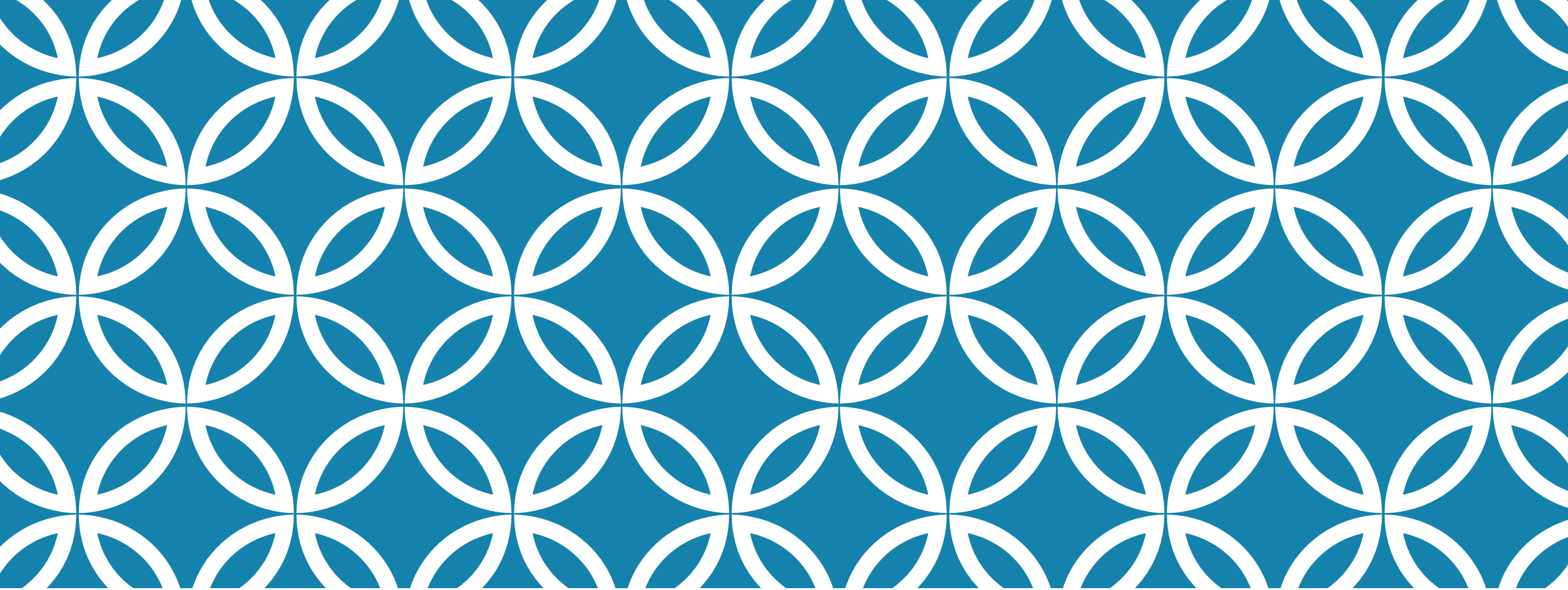
  -Subsample ratio of columns when constructing each tree.

  -Range (0, 1], default 1

PARAMETER INTRODUCTION

2. Parameter for Linear Booster

- Lambda
  -L2 regularization term on weights
  -default 0
- Alpha
  -L1 regularization term on weights
  - default 0
- lambda_bias
  -L2 regularization term on bias
  - default 0



THANK YOU

By :- Umang20csu116