Title: An Optimized, Enhanced XGBoost Algorithm on Ramanujan Graphs (Ramanujan XGBoost) has a Significantly Higher Accuracy than the XGBoost Algorithm on Decision Trees
Author: Elinor Velasquez
Date: 10 January, 2025

Introduction

Gradient Boosting is a well known ensemble machine learning method that creates more accurate models by combining individual (weaker) models. After training the initial model on the full dataset (the initial model may be taken to be a single decision tree with only one split), weights are assigned to the misclassified observations in the next iteration (to focus on and correct these errors). The split is based on the optimum of a loss function, using the gradient descent algorithm. To achieve the final prediction, a weighted sum of the individual model predictions is computed, with weights higher for the more accurate models. Note that the full dataset is originally split into a “training” subset and a “testing” subset, so by full dataset, it actually is meant the full training subset.

Extreme Gradient Boosting (XGBoost) is a gradient boosting algorithm [Chen 2016], with a collection of decision trees, in which each new tree sequentially corrects the errors of previous trees. Namely, it evaluates the previous model’s errors and assigns more weight to misclassified samples, then corrects those errors when building the next model. It combines predictions using a weighted sum, where better models have more influence.

To lessen overfitting, a function (called omega) was added to the XGBoost algorithm to penalize complexity [Deka 2024]. In addition the omega function caused the algorithm to be easily parallelizable (speeding up computation). Also, subsampling and weight scaling were used to make the algorithm more flexible. Sparse data is often a result of one-hot encoding of categorical features; in the algorithm, sparsity is dealt with by use of appropriate default nodes. XGBoost uses buckets to handle memory challenges and speed up the greedy algorithm used in the node splitting. For speeding up the sorting of data, data blocks are employed along with a compressed format, and needed statistics by block. While a random forest is more directly interpretable than XGBoost, clusters of features are discovered in XGBoost. For example, in genomics, gene clusters are discovered.

There is a risk of greater overfitting when using XGBoost instead of a random forest, if not well tuned. However, XGBoost performs better on imbalanced data than a random forest as well as a better handling of missing values, and is overall more accurate than a random forest.

The Ramanujan XGBoost Algorithm

The algorithm discussed in this work, namely, “Ramanujan XGBoost,” uses Ramanujan graphs instead of decision trees in the XGBoost algorithm. Ramanujan graphs are well known to be most useful for construction of optimal communication networks [Bien in Murty 2020].

The mixing time of a random walk equals the time it takes for a random walk to be spread over the graph. The faster mixing time of random walks on Ramanujan graphs ensures that the entire feature space is explored more quickly than on decision trees. This is important in gradient boosting, because each random walk is helpful for determining how gradients behave across different nodes. The algorithm learns global feature interactions more efficiently, in particular, for datasets with nonlinear patterns and nonseparated classes.

The Ramanujan XGBoost algorithm, implemented in C++, has the following characteristics: parallel processing for gradient updates and random walks, weighted random walks, precise floating-point operations, adjacency lists, and feature importance weighting.

The algorithm was tested on a complex toy dataset. The dataset C++ implementation was based on the python sklearn function ``generate_dataset." The features of the dataset were: Twenty noisy, redundant features with nonseparated binary class (target) boundaries, and were nonlinear. The size of the dataset was 1000 samples (observations). Five features were linear combinations of the 15 other features, thus redundant. The noise came about by having 10% of the observations' target values randomly flipped from their original values. The dataset was balanced, with no missing values.

Performance and Limitations

Table 1: Area under the ROC curve: AUC of relevant machine-learning algorithms for complex toy dataset

Algorithm	AUC
Random Forest	0.91
XGBoost	0.94
Ramanujan XGBoost	0.99

The C++ implementation of the Ramanujan XGBoost algorithm is hosted at:
https://github.com/elinorv21/xgboost/

References

[Murty 2020] M. Ram Murty, "Ramanujan Graphs: An Introduction". Indian J. Discrete Math., Vol. 6, No.2 (2020) pp. 91–127.

[Chen 2016] Chen, Tianqi; Guestrin, Carlos (2016). "XGBoost: A Scalable Tree Boosting System". In Krishnapuram, Balaji; Shah, Mohak; Smola, Alexander J.; Aggarwal, Charu C.; Shen, Dou; Rastogi, Rajeev (eds.). *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM.

[Deka 2024] Deka, Partha Pritam.; Weiner, Joyce.; Zicari, Prof. Roberto V., XGBoost for Regression Predictive modelling and Time Series Analysis: Learn How to Build, Evaluate, and Deploy Predictive Models with Expert Guidance. Birmingham : Packt Publishing, Limited, 2024.