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

XGBoost (Extreme Gradient Boosting) is a boosting algorithm, 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. It minimizes the loss function (RMSE for regression, log loss for classification) by gradient descent.

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 algorithm, "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.

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

Algorithm	AUC
Random Forest (sklearn)	0.91
XGBoost (sklearn)	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.