Predicting Deaths on the Titanic with Gradient Boosted Trees

Ethan Reese

December 12, 2018

1 INTRODUCTION

In this experiment, gradient boosted trees, through the XGBoost library, are used to build the model. Gradient boosted trees have recently gained prevalence because of both their efficiency and effectiveness for building models around simple data sets, both for categorization and regression [**?**]. Many of the top scoring models on the data science competition website Kaggle use XGBoost as the underlying model. Gradient Boosted Trees are applicable to real world problems that use big data to predict future results and behavior. Notably, gradient boosted trees are aggressively used by major Wall Street firms to assess risk when offering a loan or decide whether securities are properly valued.

2 BACKGROUND

There are many methods that can be used to derive binary classifications from tabular data. These alternates methods have different relative benefits in terms of runtime speed and accuracy, which can vary with different kinds of data.

2.1 BOOSTED TREES

The boosted tree method uses a method based on gradient descent to minimize error. Gradient descent is a method to efficiently minimize a function $f(x)$. This can be done by selecting any x and iterating using the equation:

$$x_i = x_{i-1} - \gamma \frac{d}{dx} f(x_{i-1}) \tag{1}$$

γ is a constant that can be tuned to increase either precision or runtime speed. This sequence can continue iteratively until $x_i = x_{i-1}$, resulting ultimately in the x value for the minima of the function (the y value of this minima can be trivially calculated by plugging x_i into $f(x)$).

Likewise, a similar principle can be applied to boosting gradient trees to minimize the error. For binary classification problems, this error is generally calculated using the %

error. Gradient boosting algorithms are ensemble methods, meaning that they generally contain many seperate weak predictors (algorithms that predict results with less than 50% accuracy) and combine those results into a stronger predictor. The stronger predictor should have significantly lower error.

2.2 RANDOM FOREST

Like Boosted Trees, Random Forests are an ensemble method for deep learning. However, rather than boosting, random forests use a bagging method to combine the given results. Instead of making a weighted combination of the values, a random forest uses a more basic averaging method. Some studies indicate that random forests can approach error as low as that found in boosted tree algorithms, however it is able to run more quickly [**?**].

The method is trained by creating various decision trees based on a random subset of the features in the data set. It then selects the best points to split the data based on those randomly selected features to minimize the error. This continues until the minimum node count is reached, yielding many independent decision trees. Once this model has been trained, the test data can be run through each decision tree and thus has many predictions. For a classification problem, the ultimate prediction $P(x)$ is related to the set of individual tree predictions $T(x)$ as $P(x) = mode(T(x))$.

Random Forests achieve less error than boosted trees when only a small number of trees are generated, however gradient boosted trees become much more effective at their limit [**?**]. Further, random forests are particularly vulnerable to correlation bias [**?**].

2.3 NEURAL NETWORKS

Another method could be used for the binary classification of tabular data through neural networks. Neural networks work by creating hidden layers that perform operations on the data. It can effectively take many sets of inputs and create hidden layers that perform operations on this data to produce categorized results. For more complex data sets, neural networks are a very effective method because they can make use of all of the data available [**?**]. However, for simpler data sets, these benefits are diminished. Further, neural networks are signficantly more difficult to implement and computationally expensive to train and run.

3 IMPLEMENTATION

The goal of the model was to output a value of 0 or 1 based on details about the passenger. This value 0 or 1 would indicate whether the passenger died or survived. Two sets of data were provided. Firstly, a large set that had both information about the passengers and whether they survived. This data would be used to train the model. Also, a set of data that had the same factors about the passengers, but not whether they lived or survived. The goal of the model was to predict whether or not these passengers survived. The XGBoost model was selected to make predictions.

With XGBoost, the first step is to create a large set of decision trees. Decision trees can be constructed quickly by splitting up the data along artibrary lines into branches that

ultimately lead to leafs. A singular leaf will then select a prediction that minimizes the mean square error for all of the points of test data that fit into the leaf, which will be the mean of the test data for simpler models. Trees use an imperfect greedy model to assign predictions to leafs because creating a perfect decsion tree is an np-complete problem [**?**].

XGBoost is an ensemble method, meaning that it combines the predictions dictated by many decision trees into a much more accurate result. It does this by summing the predictions from multiple separate trees [**?**]. For example, if it was using four trees, then the prediction $P(x)$ would be found with the following equation where $p_n(x)$ is the prediction of tree n:

$$P(x) = p_1(x) + p_2(x) + p_3(x) + p_4(x) \tag{2}$$

Then, the difference between the prediction $P(x)$ and the actual outcome $f(x)$, or the residual value, is calculated [**?**].

$$R(x) = f(x) - P(x) \tag{3}$$

An additional decision tree is created once $R(x)$ is calculated that aims to minimize the distance between its $p(x)$ and $R(x)$ throughout the data set. Then the predictions from this new tree are added to the preexisting $P(x)$:

$$P_{i+1}(x) = P_i(x) + p_n(x) \tag{4}$$

In theory, $\lim_{i\to\infty} P_i(x) = 0$ [**?**]. However, in practical purposes, infinity cannot be reached, so the algorithm is continued until error does not appreciably decrease with each iteration. However, this result is a number, rather than a classification, so for classification problems there is an additional step. Because the regression method used is tree based, the results are bounded on $[0, 1]$ [**?**]. Further, because only two classifications are used, the numerical predictions can simply be mapped onto a classification, yielding classified results from the probabilities.

3.1 FEATURES

References

[1] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[2] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[3] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

[4] Paulo JG Lisboa, Alfredo Vellido, and H Wong. Bias reduction in skewed binary classification with bayesian neural networks. *Neural Networks*, 13(4-5):407–410, 2000.

[5] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518, 2000.

[6] Alex Rogozhnikov. Gradient boosting explained. 2016.

[7] Robert E Schapire. The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer, 2003.

[8] Laura Tolo̧si and Thomas Lengauer. Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27(14):1986–1994, 2011.