Predicting Deaths on the Titanic with Gradient Boosted Trees

Ethan Reese

January 6, 2019

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 [8]. 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

Boosted trees use an ensemble method of deep learning based on gradient descent. They use an ensemble method for machine learning, which consists of generating a multitude of weak predictors (or models that predict with only slightly greater than 50% accuracy) and combining the results to create one stronger result. The method is moderately fast to train and produces very strong results [2].

It works by producing trees that aim to minimize the error of the sum of the results of all subsequent trees. Basing the new weak predictors on the results of the previous predictors allows the model to effeciently minimize error and produce a better result. Boosted trees are a good compromise with fast training time and robust accuracy [2].

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 [2].

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 [2]. Further, random forests are particularly vulnerable to correlation bias [9].

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 [1]. For more complex data sets, neural networks are a very effective method because they can make use of all of the data available [5]. 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 APPROACH

Boosted trees were used for this model because of their ability to both train with reasonable speed and also predict effectively. 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.

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 [4].

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 [7]. 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 [7].

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

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, $\lim_{i\to\infty} P_i(x) = 0$ [6]. 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 $[0, 1]$ [3]. Further, because only two classifications are used, the numerical predictions can simply be mapped onto a classification, yielding classified results from the probabilities.

4 IMPLEMENTATION

In the direct construction of my model, my two primary decisions were to pick the features of the given data that were used to make a prediction and the parameters of the model.

4.1 FEATURES

The model used five of the provided data's features (gender, age, immediate family, fare, ticket class, extended family on board). The model used these data to create dividing lines within the simple decision trees, which are then combined to form the boosted tree. The gender data gave a binary indication of whether a passenger was male or female. The age data indicated the age of every passenger as an integer. The immediate family count represented the number of parents or children that the passenger had on board. The fare indicates as an integer the amount that the passenger paid for passage on the Titanic. The ticket class was the class of cabin that the passenger was located in. Extended family on board was the total number of siblings and spouses that each passenger had on board.

4.2 PARAMETERS

The parameters used to construct the XGBoost tree influenced the number of basic decision trees made and the size of those trees. All of the parameters were optimized through a process of cross validation aiming to maxmize the negative mean square error, thus minimizing the absolute value of the error. To do this, the data were randomly split into ten distinct groups. Then, ten experiments are run, each one using nine of the sets as training data and one set of data as test data. The singular set used as test data is different for each of the ten experiments. The error is calculated for each of these experiments and then the arithmetic mean is taken of the set.

The parameters were optimized through experimentation. To optimize the individual parameters, a parameter was first isolated for optimiztion. Then, this isolated parameter was varied by values an order of magnitude apart. The model was run one time for each of these values and a cross validation score was generated for each. The parameter value that yielded the highest cross validation score (smallest error) was identified and more values were tested for the parameter in the same order of magnitude, ultimately revealing the most effective value for the parameter. Then, the parameter was locked at this value and the subsequent paramter was isolated. This process was repeated for every parameter relevant to forming the decision trees involved in the model.

4.2.1 Learning Rate (`eta`)

The learning rate of a gradient boosted tree impacts the scaling of the features in each tree. A lower learning rate leads to a higher number of decision trees, meaning a potentially more accurate result but a substantially slower runtime. However, a lower learning rate can result in an overfit model. The learning rate used was 0.127.

4.2.2 Maximum Depth (`max_depth`)

The maximum depth is the most number of layers that any decision tree will have as part of the model. A higher value means that the model constructs more detailed trees, leaving a vulnerability to overfitting. For the model, the maximum depth was reduced to 3 from the default value of 6.

4.2.3 Minimum Child Weight (`min_child_weight`)

As the respective decision trees are being constructed, the weight of each individual leaf decreases with each partition made. Once a child reaches the minimum child weight, the tree discontinues the partitioning process. A higher value results in a less complex model and decreases the chances of overfitting. For the model, the minimum child weight was lowered from the default value of 1 down to 0, meaning that the minimum child weight was not a factor that indicated when to stop forming the trees. The lower value for maximum depth provided this stopping point in a sensible location.

4.2.4 Subsampling (`subsample`)

The subsampling value is the amount of data randomly selected before growing trees, a lower value is useful to prevent overfitting because less of the training data is used. This was lowered for 0.947 from the dfault value of 1 in the model, thereby reducing the model's tendency to overfit.

4.2.5 Estimators (`n_estimators`)

The number of estimators is the number of individual decision trees used in the formation of the ultimate boosted tree. However, XGBoost has a built in method that will stop the model from continuing to add trees after each additional tree is no longer making an appreciable difference to the error (a sign that the model is beginning to overfit). It stops the model to continue to grow even if the number of estimators defined in the model has not yet been

reached. Thus, the number of estimators was set to 1000, a number of estimators that would never be reached. The model generally used between 4 and 10 trees in practice.

5 EVALUATION

The model proved to be relatively ineffective at predicting which passengers would die in the test data set. The model was scored as a percentage of the number of passengers in the test data set that had their status of survival guessed correctly. The most effective run was able to produce a dataset with 76.555% accuracy. This places the model in the lower 50% of models entered in the Kaggle competition.

Other models using boosted trees and neural networks were able to perform far better because of more robust feature engineering. I did not include any feature engineering in my model. Having more features would have allowed the model to form more trees and take greater advantage of the power of gradient boosted trees. In the future, many more features could have been engineered on the data set to allow more calculations to be done and resulting a far more robust model.

Additionally, my method of optimiziation was adequate for the problem but not scalable to a problem with more features. The parameters of the model are interconnected and have interaction between them. Thus, varying them one at a time in isolation may result in slightly flawed values. For a model on a small scale this doesn't make a major difference, but as more features are introduced, the difference becomes more substantial. Grid Search based optimization methods offer a way around this by optimizing multiple variables simultaneously, thus taking the interconnectedness into account.

Ultimately, the model requries more features and a more robust optimization method to be effective.

References

[1] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

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

[3] 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.

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

[5] 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.

[6] 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.

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

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

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