

Forest of Gradient Boosted Decision Tree

Ashokkumar Thakur (2051207)¹, Arjun Rao (2197557)², and Nilesh Mohan Dhikale
(2001224)³

¹Department of Computer Science, University of Houston

²Department of Computer Science, University of Houston

³Department of Computer Science, University of Houston

1 Abstract

Bagging and boosting both are considered very efficient approaches in machine learning. Both approaches are used to solve difficult problems. But there are very less work done to analyse the effect of these two approaches combined together. For this project we tried to combine two different algorithm that are based on bagging and boosting approach. We created a forest of gradient boosted trees (GBT) which is a boosting algorithm. This forest is using bootstrapped data sets for training different GBT trees. Our proposed approached gave the accuracy better than the popular classification methods.

Keywords – bagging, boosting, gradient boosted trees, bootstrap

2 Introduction

Machine learning has become very advanced in solving complex problems. Currently there are many algorithms which gives better results for different needs either for classification or regressions. One of the approaches which is widely popular is ensemble learning. Ensemble learning methods are machine learning paradigm where we aim to produce a model approximation for regression and classification where multiple weak model are trained to solve same problem, and when these weak models are combined correctly we can obtain more accurate result and models. Majorly there are two classifications in ensemble learning bagging and boosting.

Bagging

Bagging also known as bootstrap aggregation is applied to enhance the ensemble-based algorithm. In bootstrapping we generate sample of certain size from the data-set of some size randomly. The data distribution are elected independently in each bootstrap samples.

While generating the bootstrap samples, we need to consider 2 points, First the initial data-set should be large enough to capture the complexity of the distribution in order to get a good approximation of the sampling data. There should not be too much of correlation between the data-set of the bootstrap samples. Bootstrap samples are often used to evaluate the variance or confidence intervals of the estimators. The main idea in bagging is to generate a series of independent samples which are of same size from the original data-set. Bagging adds two more steps in the original model generation. First, we generate multiple bootstrap samples which will act as an independent data-set and passing each set of samples to the base model. Second, combining the predictions of multiple predictors.

For classification we can use majority voting technique to combine the output of the base predictors. Whereas, in regression we can average the predictions to get an ensemble output. Averaging sample outputs reduces the variance. One of the advantage of bagging is that it can be parallelised. As different models are fitted independently from each other, these models can be generated using intensive parallelisation techniques.

One of the famous bagging dependent algorithms is Random Forest which is an improved version of decision tree that uses bagging technique for improving the predictions of the base model. The difference between random forest and bagging is that in random forest while generating each bootstrap samples, only a subset of features is randomly selected. The main reason for selecting few features out of the total features is to reduce the correlation between different bootstrap samples. This way of sampling also makes the decision-making process more robust by using the concepts of bagging and random feature subset selection.

Boosting

Boosting works in the same way as of bagging method, we build multiple models and aggregate them to get a strong model. Boosting follows a sequential approach to fit weak models in an adaptive way. In each iteration, data is fitted in the weak classifier. Examples which are misclassified by the model, in this case the weights are increased, whereas, the examples which are classified correctly their weights are decreased. A new sample is generated according to the updated weights.

The main motive is to generate models that focus on the feature space with more misclassification. Boosting can be used for regression and classification problems. There are two important algorithms of boosting, first is Adaboost and second is gradient boosting.

AdaBoost or Adaptive Boosting uses an iterative optimization approach that is much more traceable. In this approach the weak learners are added one by one to find the best possible pair to the current model. Adaboost learns an effective ensemble classifier as it leverages the incorrectly classified sample at each age of the learning. It minimizes the loss function.

In **gradient boosting** we try to build ensemble model from a weighted sum of weak learners. Gradient boosting casts the problem into a gradient descent one, at each iteration we fit a weak learner to the opposite of the gradient of the current fitting error with respect to the current model. [9]

3 Literature Review

In [1] they built an ensemble using a voting system of boosting, bagging, and dagging ensemble with 8 sub-classifier in each one. In their work they performed comparison with simple bagging, boosting and dagging ensemble with 25 sub-classifiers and also other well known combining methods. They achieved better accuracy in most cases using their proposed technique.

In [2] authors studied binarization techniques that deals with multi-class classification problems by combining several binary classifiers. In their work, they studied if there is also some improvement possible when using ensemble of decision trees. They tried it over 67 multiclass datasets. They found out that some combinations of binarization technique and ensemble method improve the results of the ensemble method without binarization.

In [3] the authors studied techniques to deal with imbalanced data. Authors proposed a novel data balancing method by employing clustering technique with SVM. This method will select most informative majority and minority class and combines them to establishes a novel method. They have tested their approach with the popular techniques such as decision trees and naïve bayes and their proposed idea generates improved accuracy when classifying both majority and minority class instances.

In [4] authors studied ensemble learning methods combining bagging and boosting. In this work they compare the online version of their ensemble method with their own batch method that they created previously. They also stated that their online method require less running time than the batch mode because the online method require fewer passes through the training data.

In [5] the authors proposed a bagging-boosting based semi-supervised multi-hashing with query-adaptive re-ranking. They trained an individual hash table of multi-hashing using boosting. Their experiments results shows that the proposed technique gives better recall and precision for given data.

In [6] authors proposed a boosted version of the random forest classifier, which fits an additive model that is made of multiple random forest. Opposite to the traditional methods, they have used stronger out-of-bag (OOB) error estimates. They experimented with several datasets including high-dimensional and noisy domains and it improve the classification effectively.

In [7] the authors proposed a novel Federated Random Forest (BOFRF), that increases the predictive power of all participating sites, and provides significantly high improvement on predictive power. They introduced weight calculation and aggregation methods to assign weights to local classifiers. This method has improved the predictive power in all cases and perform significantly better than other models.

In [8] authors proposed vector quantization method using ensembles learning algorithms bagging and boosting. Initially data are randomly selected and a weak learner is trained on it. For later weak learners the probability distribution of learning data is modified due to which each weak learner focus on high error data. The proposed method gave good performance in shorter learning time than the traditional algorithms like K-mean.

4 Implementation

To implement our idea, that is combining bagging and boosting are combining Random Forest and Gradient Boosted Trees together. There are multiple algorithms that can be combined for this purpose but random forest gives better flexibility with the algorithm that we can use in it and this flexibility makes it easier to add Gradient Boosted Trees in it instead of regular Decision Trees.

4.1 DataSet


To implement this project and to test our application we needed a dataset that should be open-source and widely available. Along with this the dataset should be easy to understand and reliable, so that we can test it easily and also trusted for the project.

The dataset for this project that we selected is Titanic Survival dataset from Kaggle. It is one of the widely used and popular dataset among the data science community. This dataset is the go-to dataset for beginners and for the early testing of algorithms.

The dataset has following fields:

Fields	Description	Details
survival	survival	0 = No, 1 = Yes
pclass	Ticket Class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age	in years
sibSp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Below is the snippet from the Titanic dataset:

✓  titanic_data.head()

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Figure 1: Titanic data

4.2 Transformation on data

4.2.1 Handling Missing Values

There are certain missing values present in the dataset. Following are the count of missing values in the respective fields

Age	177
Cabin	687
Embarked	2

As we can see, fields like age and cabin which can be very important attributes for deciding the survival of passenger. To handle the missing value related to age, we replace the missing values with the mean of the age field. For cabin, as the value of cabin is not very descriptive and will not help in prediction, we will drop this field.

4.2.2 Combine Siblings and Parent Count

One of the important feature while predicting the survival of passenger can be how many siblings or parent they have. These things can be a deciding factors, for example, a person with many siblings will be having less chances of survivability as he might be thinking and helping his siblings while a person with less or no sibling will only think about himself or herself.

4.2.3 Encode Sex Field

As sex field is having a string value. It has to be encoded if we want to use it in our model. We encoded with men having value 0 while the female having value 1.

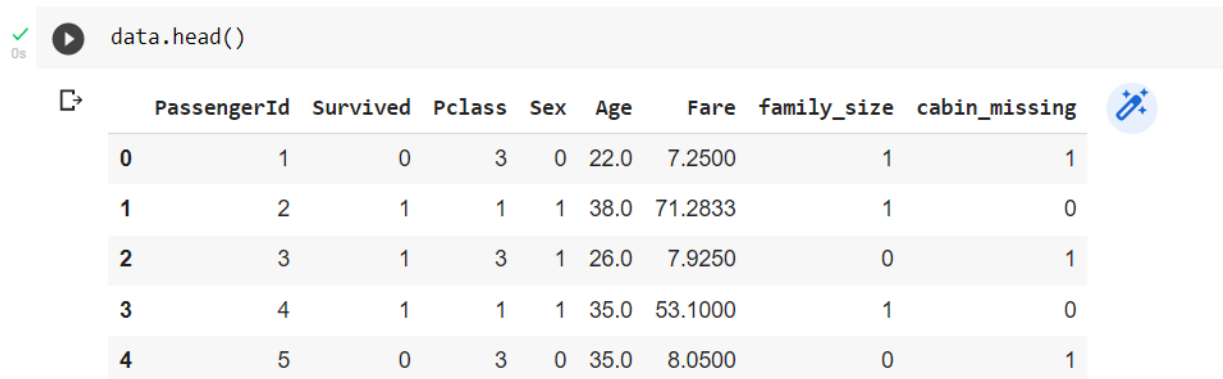
4.2.4 Data set Split

As we are having only this dataset, we have to split it to train and test, so that we can train our model and also have a dataset to test on it. We have used `train_test_split` method from `scikit-learn` to split the data.

We kept 70% of the data for training and remaining 30% for testing.

The above training dataset will be given to library models directly for training. While for the forest of gradient boosted trees, we will create bootstrap samples from this training data-set and further this data-set will be used for the training the custom created model.

Below is the sample from the data after transformation:



	PassengerId	Survived	Pclass	Sex	Age	Fare	family_size	cabin_missing	
0	1	0	3	0	22.0	7.2500	1	1	
1	2	1	1	1	38.0	71.2833	1	0	
2	3	1	3	1	26.0	7.9250	0	1	
3	4	1	1	1	35.0	53.1000	1	0	
4	5	0	3	0	35.0	8.0500	0	1	

Figure 2: Titanic data after transformation

4.3 Algorithm

For this project, we are creating a forest of Gradient Boosted Trees. There are three main components for this project, bagging, random forest, and gradient boosted trees.

One of most important steps in implementing this project is to create modules for combining these two approaches via, Random Forest and Gradient Boosted Trees. For bagging purpose, we have created our own module which will take the dataset and create bootstrapped datasets from it. Creating the module from scratch and customized will makes it flexible to use it with any different module in our project.

To implement this, we have created a random forest module from scratch. Creating this module from scratch will give us freedom to incorporate it with any other algorithms. Using the algorithm from the library will come with some restrictions that might be not suitable for our purpose so we decided to implement our own Random Forest module.

For Gradient Boosted Trees module, this we are using from library. The Gradient Boosted Trees will not need to modify or customized for our purpose. The algorithm provided by the library is well suited implementing this project. We used this library algorithm without created Random Forest module.

As mentioned above the project has three main components, but the most important part is to combine all these three components together to come up with the desired algorithm.

The algorithm of the project is as follows:

1. Clean the dataset and apply transformation on it (if require)
2. Pass the data to created random forest module
3. This module will create bootstrapped dataset from the dataset from step 1
4. Gradient Boosted dataset will run on each bootstrapped dataset that was created in previous step for specified number of times and result will be stored
5. After completing the above step, combined result will be averaged as per the number of Gradient Boosted Trees used together
6. This averaged result will be output of the module and returned as result

4.4 Process

To test whether the our implementation is performing well with the traditional algorithms, we have also created and tested Scikit-Learn implementation of Random Forest and Gradient Boosted Trees viz, `RandomForestClassifier` and `GradientBoostingClassifier`.

These will act as baseline for our implementation. We can verify and test our algorithm and see how good or bad it is performing. First we created and tested the library implementations on this dataset. It is giving 79.9% and 78% accuracy respectively.

5 Conclusion

The Result that we got is showing that for this case the forest of gradient boosted trees are showing promising results. The random forest from Scikit-Learn library is giving an accuracy of 79.9%, while the Scikit-Learn implementation of Gradient Boosted Trees is giving the accuracy of 78.0%. Both methods are giving the similar results.

The implementation for this project i.e. forest of gradient boosted trees is also giving the similar result with accuracy of 85.6%. This accuracy is bit more than the accuracy of previous two models. As the model performance is dependent on the data on which model is trained, but for this case forest of GBT is showing performance improvement compared to the Random Forest and regular Gradient Boosted Trees.

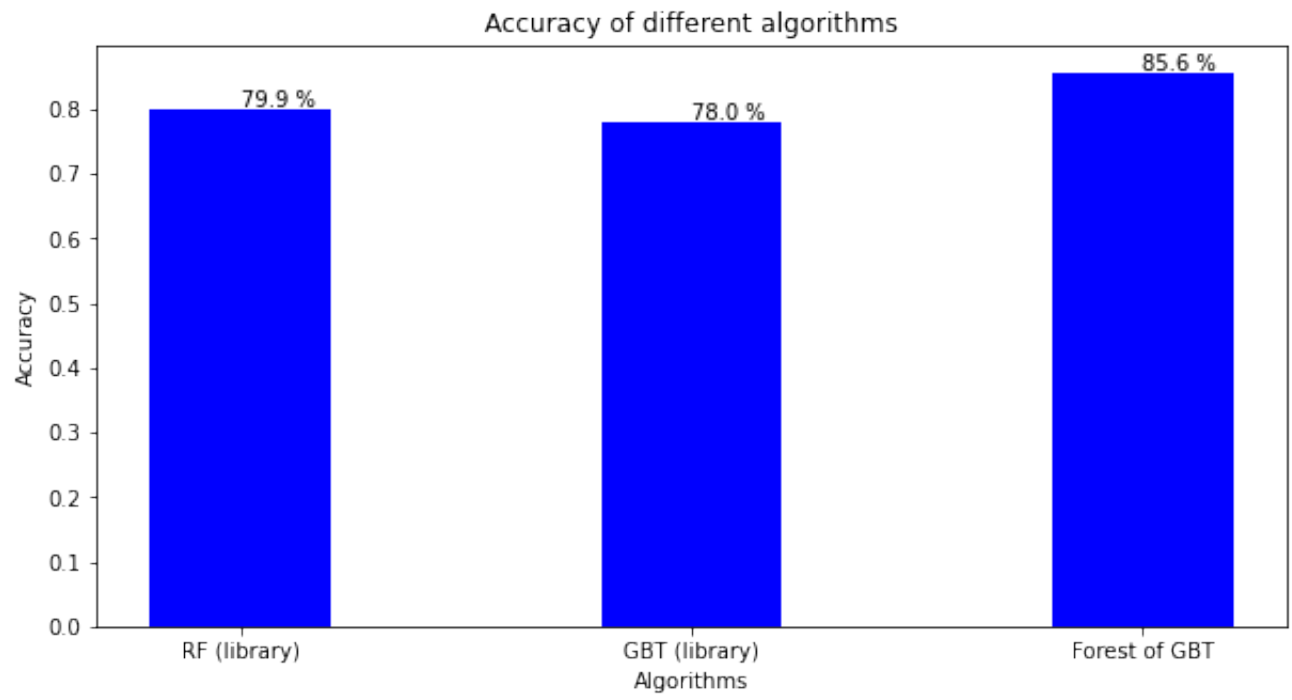


Figure 3: Accuracy Plot

References

- [1] S. B. Kotsianti and D. Kanellopoulos, “Combining Bagging, Boosting and Dagging for Classification Problems,” *Lecture Notes in Computer Science*, pp. 493–500, 2007
- [2] J. J. Rodríguez, J. F. Díez-Pastor, Á. Arnaiz-González, and C. García-Osorio, “An Experimental Study on Combining Binarization Techniques and Ensemble Methods of Decision Trees,” *Multiple Classifier Systems*, pp. 181–193, 2015
- [3] Md. Y. Arafat, S. Hoque, S. Xu, and D. Md. Farid, “Advanced Data Balancing Method with SVM Decision Boundary and Bagging,” *2019 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, Dec. 2019
- [4] N. C. Oza and S. Russell, “Experimental comparisons of online and batch versions of bagging and boosting,” *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, 2001
- [5] Ng, Wing W. Y., et al. “Bagging–boosting-based Semi-supervised Multi-hashing With Query-adaptive Re-ranking.” *Neurocomputing*, vol. 275, Elsevier BV, Jan. 2018, pp. 916–23. Crossref, <https://doi.org/10.1016/j.neucom.2017.09.042>.
- [6] T. Salles, M. Gonçalves, V. Rodrigues, and L. Rocha, “BROOF,” *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Aug. 2015
- [7] M. Gencturk, A. A. Sinaci, and N. K. Cicekli, “BOFRF: A Novel Boosting-Based Federated Random Forest Algorithm on Horizontally Partitioned Data,” *IEEE Access*, vol. 10, pp. 89835–89851, 2022
- [8] N. Shigei, H. Miyajima, M. Maeda, and L. Ma, “Bagging and AdaBoost algorithms for vector quantization,” *Neurocomputing*, vol. 73, no. 1–3, pp. 106–114, Dec. 2009
- [9] M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer, P.N. Suganthan, “Ensemble deep learning: A review”