XGBoost Evaluation

Beixuan Yang

December 22,2021

Abstract

XGBoost is a highly effective and widely used tree boosting machine learning method. This course project intends to evaluate the XGBoost algorithm on both classification and regression tasks.

1 Introduction

XGBoost is a very popular machine learning algorithm. It builds multiple trees as weak learners and combine them as a strong learner. Unlike random forest that looks at the average/votes of each tree, instead, it takes the sum of all the trees' output to compute the final result. In order to form this "additive" method, the trees are targeting to predict the residual of the previous output(i.e. the ground truth subtract the sum of the previous trees). Ideally, the absolute value of the target (the residual) will gradually decrease and finally converge to 0 or some predefined threshold as the number of trees grows. In this project. XGBoost algorithm is tested on four datasets, the "US-Accident-Oct2021" & the "fashion mnist" for classification; the "synchronous machine" & the "Beijing PM2.5" for regression. The authors of [1] introduced a high scalable XGBoost system which can deal with billions of examples using fewer resources than many systems. Due to the limitation of time and hardware, this project will focus on the performance of the basic algorithm only.

2 Related work

As a favored method, XGBoost has been widely evaluated and compared with other machine learning algorithms. For instance it is often compared with Random Forest. In [2], the author compared XGBoost and Random Forest in details. Neutral Network is famous for it's capacity of handling huge amount of data, so the authors of [1] proposed a scalable system which makes XGBoost become competitive in this era that available data size soars fast.

3 Approach & Experimental results

The main objective of this project is to test XGBoost with 4 datasets, of which two for classification and two for regression. XGBoost is compared with algorithms (kNN, Neural Network, Linear Regression, Decision Tree, Random Forest and AdaBoost) with regards to accuracy, R2 score, running time (training time plus testing time) as well as the performance on unbalanced dataset.

3.1 Classification

3.1.1 4 class classification on US-Accident dataset

This is a countrywide car accident dataset collected from February 2016 to December 2020. There are about 1.5 million accident records covers 49 states in the US [3]. Only the data from California are used for evaluation in this project. The task is to predict the severity level of the accidents. The data is unbalanced, among the 4 levels, the most interested level (level 4) occupies only 1.8% of the dataset. Therefore, the samples are selected in purpose such that the percentage of level 4 increased to 15%.Three different settings of preprocessing are performed on this dataset in order to test how will the performance of algorithms impacted when data forms are changed.

3.1.2 US-Accident setting 1

Data size: 42210 samples with 109 attributes(unscaled). It is tested with kNN, XGBoost and Random Forest. The results are shown in table 1.

3.1.3 US-Accident setting 2

Data size: 42210 samples with 15 selected attributes (unscaled). The attributes are ranked by Random Forest It is tested with kNN, XGBoost and

Table 1: Results of US-Accident setting1

	XGBoost	kNN	Random Forest
Overall Accuracy	0.798	0.676	0.808
Accuracy of level 4	0.455	0.224	0.486
Running time	29s	6.9s	9s

Random Forest. Results are shown in table 2.

Table 2: Results of US-Accident setting2

	XGBoost	kNN	Random Forest
Overall Accuracy	0.812	0.678	0.817
Accuracy of level 4	0.532	0.194	0.512
Running time	16s	0.7s	26s

3.1.4 US-Accident setting 3

Data size: 42210 samples with 15 selected attributes(scaled to (0,1)). It is tested with kNN, XGBoost and Neural Network. Results are shown in table 3.

Table 3: Results of US-Accident setting3

	XGBoost	kNN	Neural Network
Overall Accuracy	0.663	0.676	0.677
Accuracy of level 4	0.422	0.242	0
Running time	17s	0.9s	88.9s

3.1.5 Summary of the 3 US-Accident settings

The overall accuracy of XGBoost is good but not best in all three settings. In two of them, XGBoost accived the best performance on predicting the most interested class even though the dataset is unbalanced and there are only 15% samples belongs to the interested class. In setting 3, Neural Network has higher overall accurcy than XGBoost. However, it doesn't classify any data into the interested class, whereas XGBoost achieved 42.2% accuracy in classifiy the interested class.

3.1.6 10 class classification on Fashion Mnist

This test is aimed to test running accuracy between XGBoost and Neural Network (20% data from the full dataset are randomly selected to use in this task). I used the code frame of my previous assignment because the hyperparamter of Neural Network was tuned previously. The results are listed in table 4. To reach 85% of accuracy, XBGoost is slightly slower, while to achieve 87%, XGBoost is much faster than Neural Network.I could not get higher accuracy for both the methods.

Table 4: Results of XGBoost vs Neural Network Fashion Mnist

	Running time of XGBoost	Running time of Neural Network
Accuracy 0.85	52s	26s
Accuracy 0.87	179s	391s

3.2 Regression

3.2.1 Excitation current regression of synchronous machine

synchronous machine [4] is a small dataset consisted of 557 samples and 5 attributes. 5-fold cross-validation are applied to compare 7 algorithms. This dataset is selected because it is recently released. However, there is no significant difference in terms of R2 score and mean squared error between this 7 methods. As we can see in table 5, all the methods got high score. The average performance of XGBoost is in the 3rd place.

Table 5: R2 score comparison on synchronous machine dataset

	XGBoost	AdaBoost	NN	kNN	LinearR	RandomF	DecisionT
1	0.99981	0.99780	0.99832	0.99179	1.0	1.0	0.99982
2	0.99977	0.99777	0.99715	0.98788	1.0	1.0	0.99979
3	0.99996	0.99768	0.99481	0.99017	1.0	1.0	0.99989
4	0.99995	0.99764	0.99765	0.99223	1.0	1.0	0.99980
5	0.99993	0.99727	0.99707	0.99183	1.0	1.0	0.99988

3.2.2 PM2.5 regression of Beijing PM2.5 dataset

Beijing PM2.5 dataset [5] is a hourly collected PM2.5 values of Beijing along with 13 attributes. Since the dataset is large, only the data form station Wanshouxigong (there are totally 9 data collecting stations) are used as the training set. This is an unbalanced dataset for the highest PM2.5 value are around 690 while 50% of the values are under 50. The interested values are in the higher end. In this project, we keep the original distribution of PM2.5 values to train the models. These models are tested in both natural distributed dataset as well as the selected data which contains high PM2.5 values only.

3.2.3 Beijing PM2.5 setting 1

The attributes are not scaled. 5-fold cross-validation is used to compare the performance of predicting the PM2.5 value. As shown in table 6,7 and 8, XGBoost surpasses the other 6 methods in terms of R2 score and MSE. With regards to the running time, it is in the slow end but not the worst.

Table 6: R2 score comparison on predicting the PM2.5 setting1

	XGBoost	AdaBoost	NN	kNN	LinearR	RandomF	DecisionT
1	0.9603	0.64757	0.90509	0.89265	0.8599	0.94741	0.88864
2	0.96053	0.5615	0.91547	0.90151	0.89016	0.94506	0.89016
3	0.95547	0.54515	0.90681	0.8901	0.84994	0.93992	0.88141
4	0.95942	0.52221	0.91026	0.89048	0.86053	0.9442	0.88587
5	0.95132	0.60226	0.90798	0.8906	0.85238	0.93569	0.87442
ave	0.957354	0.575738	0.909122	0.893068	0.856476	0.9424556	0.8841

Table 7: Running time (s) comparison on predicting the PM2.5 setting1

	XGBoost	AdaBoost	NN	kNN	LinearR	RandomF	DecisionT
1	42	3	18	4	0.05	79	0.4
2	16	3	17	4	0.03	63	0.4
3	16	3	11	4	0.03	63	0.4
4	15	3	25	4	0.03	64	0.5
5	18	3	23	5	0.03	70	0.4

Table 8: MSE comparison on predicting the PM2.5 setting1

	XGBoost	AdaBoost	NN	kNN	LinearR	RandomF	DecisionT
1	17	51	26	28	32	19	28
2	16	56	24	26	31	19	28
3	17	56	25	27	32	20	28
4	17	59	25	28	31	20	28
5	18	53	25	27	32	21	29

3.2.4 Beijing PM2.5 setting 2

The attributes are scaled (normalized) and also, continuous attributes has their z score added as extra attributes. Again, 5-fold cross-validation is used to compare the performance of predicting the PM2.5 value. Under this setting all the algorithm predicts almost perfect, XGBoost ranks 2 in terms of R2 score (table 9).

Table 9: R2 score comparison on predicting the PM2.5 setting2

	XGBoost	AdaBoost	NN	kNN	LinearR	RandomF	DecisionT
1	0.99999	0.98995	0.99927	0.92555	1.0	0.99998	0.99996
2	1.0	0.98952	0.99928	0.92481	1.0	1.0	0.99999
3	1.0	0.98898	0.99833	0.921281	1.0	1.0	0.99999
4	1.0	0.99126	0.99893	0.92290	1.0	0.99999	0.99998
5	1.0	0.98988	0.99906	0.92244	1.0	0.99998	0.9999

3.2.5 Beijing PM2.5 setting 3

The third test is used to test the performance of predicting only the data with high PM2.5 values with the models trained in setting 1. Over 27000 samples with PM2.5 value greater than 75 (the pollution line) from 2 other data collecting stations are used in this task. Form table 10, it is clear that XGBoost achieves the highest R2 score (85%) in this setting which indicates that it tends to perform well on the small percentage of classes (usually the interested ones)when the training data are unbalanced.

Table 10: R2 score comparison on predicting the PM2.5 setting3

	XGBoost	AdaBoost	NN	kNN	LinearR	RandomF	DecisionT
R2	0.852	0.703	0.83	0.764	0.78	0.844	0.707
MSE	32	46	34	41	39	33	45

4 Conclusions

Based on the limited work done in this project, XGBoost is a great algorithm in terms of accuracy in classification tasks and R2 scores in Regression under different settings of preprocessed data. Unlike linear Regression, kNN's etc., XGBoost's performance doesn't change dramatically with variance settings of data reprocessing. The down side is that it is relatively slow due to the structure that a set of weak learners(trees) need to be built. But still, it is usually faster than Neural Network and Random Forest in this experiment. Other than the overall accuracy, on both the unbalanced datasets, it performs better than all the other methods when predicting the data that are interested but with lower percentage in the full set. This is under the situation that no cost matrix are applied while building the models. The existing xgb libs/packages do not take cost matrix. Thus, a potential future work could be to implement a version of XGBoost that takes cost matrix, this may even boosting its performance on predicting interested classes/values.

[6] [7] [8] are some reference code that I learned from for this project.

References

[1] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," 2016. [Online]. Available: https://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf

[2] A. Gupta. (2021) Xgboost versus random forest. [Online]. Available: https://medium.com/geekculture/xgboost-versus-random-forest-898e42870f30

[3] S. Moosavi. (2021) Us accidents (updated). [Online]. Available: https://www.kaggle.com/sobhanmoosavi/us-accidents

[4] R. BAYINDIR and H. T. KAHRAMAN. (2021) Synchronous machine data set data set. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Synchronous+Machine+Data+Set

[5] S. X. Chen. (2017) Beijing pm2.5 data data set. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data

[6] K. Lemagnen. (2017) Hyperparameter tuning in xgboost. [Online]. Available: https://blog.cambridgespark.com/hyperparameter-tuning-in-xgboost-4ff9100a3b2f

[7] A. Xiao. (2021). [Online]. Available: https://github.com/Aaron-X-93

[8] A. SINGH. (2021) Synchronous motors best result 97.1% r2 score rf. [Online]. Available: https://www.kaggle.com/akarshsinghh/synchronous-motors-best-result-97-1-r2-score-rf