Boosting and Boostability: Weak Learning Implies Strong Learning

Dhruv Srikanth[1]

[1]*Dept. of Computer Science*
University of Chicago
Chicago, USA
dhruvsrikanth@uchicago.edu

In this paper, I explore the design and analysis of boosting algorithms and the necessary and sufficient conditions on the weak learners to ensure boostability. Having gone through some papers on the topic, I will discuss my learning and understanding of boosting as well as what I found difficult and what I would like to do in the future to gain a deeper understanding of the topic. Additionally, I will talk about some of the boosting algorithms I have come across. I will also look at the binary classification problem and extensions to the multi-class problem.

I. INTRODUCTION

Boosting is an algorithmic approach to improving the accuracy of any given learning algorithm. This is accomplished by the use of an ensemble of different models and ensuring that, for any given example, there exists a combination of models that can, through our boosting algorithm, produce an accurate result. We explore algorithms such as **Adaboost** [3] and discuss a form of boosting that is different from adaptive boosting, namely gradient boosting. We now formally introduce the term of a weak learner, as a prediction rule that does better than random guessing, however, when used as a standalone learner, proves to be too weak to be used. In 1990, Schapire [5] first introduced the concept of boosting as an algorithmic approach to combining weak learners to produce strong learning, thereby driving down the true error on the distribution of data. We will take a game-theoretic approach in analyzing the framework of boosting algorithms and explore the weak learning conditions that must be imposed on the weak learners to provide a guarantee of driving down the true error on the distribution. Formally, this can be seen as the following two conditions that must both be met, to ensure that the framework is boostable:

1. Each weak learner must do better than random guessing on the distribution.

2. There exists a combination of weak learners, such that this combined prediction rule has an error $\leq \epsilon$.

We may then view this as a game being played by two players, the booster and weak learner, for a fixed number of rounds. This can be analyzed as a minimax game, where the optimal strategy lies in finding a good combination of prediction rules that can drive down the training error over the distribution of data. This can be done by focusing on learning the "hard" cases. It is important to note here that there exist several polynomial-time algorithms such as **Adaboost** [3], **XGBoost** [1] and several other boosting frameworks that are capable of improving the accuracy of a given learning algorithm, provided the conditions for weak learning over a set of m training examples $\{(x_i, y_i) \dots (x_m, y_m)\} \sim D \in \mathbb{R}^n$, are met. However, the problem of finding the best possible combination of prediction rules is a problem that still remains computationally hard.

We will also explore the boosting framework used in the **Adaboost** [3] algorithm. Finally, we will analyze how such boosting frameworks can be used on the binary classification problem and its extension to the multi-class classification problem. We will see that in the binary classification problem, a sufficient condition for weak learners is to do better than random guessing, meaning that each prediction rule must produce a training error of at most 50%. A natural extension of this would mean that for the multi-class problem, each weak learner must produce a training error of at most $1/C$, where C is the number of classes in the problem. However, this turns out to be too weak of a condition to produce strong learning [4]. Similarly, the conditions that make each weak learner too strong, can result in overfitting [4]. Finding a balance between the two is the key to designing a boosting framework for the multi-class classification problem [4].

II. BOOSTABILITY AND WEAK LEARNING CONDITIONS

In this section, we take a look at what are the necessary and sufficient conditions to ensure strong learning through the use of an ensemble of weak learners. We can understand boosting as the algorithmic approach of combining several weak classifiers to produce a strong classifier, thereby leading to the popular phrase, "weak learning implies strong learning". We may formally introduce boostability in the following way. An algorithm is said to be boostable if there exists a set of weak learning conditions that guarantee:

1. Each weak learner must do better than random guessing on the distribution $D \in \mathbb{R}^n$.

2. There exists a combination of weak learners, such

that this combined prediction rule produces an error $\leq \epsilon$.

We will formalize this under the term, boostability assumptions which is analogues to the weak learning conditions required for an algorithm to be boostable.

As mentioned in the introduction, if the weak learning condition is too weak, this results in underfitting and if the condition is too strong, the result is overfitting on the distribution D. Therefore, designing effective weak learning conditions will lead to a robust boosting framework. Furthermore, this provides a guarantee to drive down training error, whilst having an acceptable out-of-sample error, given that the weak learning conditions can be met over the distributions D.

As done so by Mukherjee and Schapire [4], I will present a game theoretic approach to finding the weak learning conditions. Consider a two player game between the booster and the weak learner. This game is played for T rounds. We can imagine in each round that the booster outputs a distribution and the weak learner outputs a prediction rule that does better than random guessing on the distribution. For the problem of binary classification, this means that the error on the sample is less than 50%. In the multi-class classification problem, in each round t over T rounds, a cost matrix C is computed. This cost matrix specifies to the weak learner, the cost associated with classifying a label as l, where l belongs to the set of labels present in our multi-class setting. Upon seeing this, the weak learner outputs a hypothesis $h \in H$, where H is our hypothesis class. Finally, the classification label is chosen by taking a plurality or weighted majority vote over all hypotheses produced by each weak learner, where the weight of a hypothesis produced by a weak learner is based on the total cost incurred by the weak learner during the current round t.

In viewing the boosting framework as the game described above, we can see that the maximum cost that a weak learner can experience in any given round must be the minimum conditions for weak learning. This shows us the importance in the design and construction of an appropriate, and certainly not arbitrary, cost matrix C. In order to do better than random guessing, the cost $c_{i,l}$ for example $x_i \in D^n$ classified as label l to be classified correctly must be less than the cost of an incorrect classification. This is to focus on learning the "hard" cases. As this is a two player game, the analog of this for the weak learner is on the hypothesis h that must be generated. The constraint on h is that it must be prediction rule that does better than random guessing on an example x_i. This can be shown to produce the following condition on weak learners for the multi-class classification problem. When weak learners are weighted by the cost associated with their misclassifications, to be boostable in the normal setting, each weak learner is subjected to the condition that they must do better than, meaning that they must have a lower cost than the average cost of the weak learners, weighted by the misclassification cost setting described previously. It is important to note here that the weight associated with any prediction rule must always be greater than 0, meaning the cost must be greater than 0.

The equivalence condition between weak learners is that each weak learner must satisfy the same condition in order to enable boostability. We can further analyze the equivalence condition between the weak learners by considering the following game theoretic analysis. Note that when we look at the equivalence between weak learning conditions, this does not mean that they are the same subset of the hypothesis class H. The traditional view of equivalence considers that for boostability, if we take two weak learners, either both weak learning conditions are satisfied, or neither is satisfied. The role of the weak learning condition is to impose a restriction on the response provided by the weak learner, which holds even in an adversarial setting. The optimal strategy of the booster in each round lies in imposing the strongest restriction to the weak learners, thereby restricting the space of prediction rules chosen from the hypothesis class H. In the next section we will explore the **Adaboost** [3] algorithm and its application to the binary and multi-class classification problems.

III. BOOSTING METHODS

A. Adaptive Boosting - Adaboost

In this section, we will explore the **Adaboost** [2] algorithm proposed by Freund and Schapire in 1997. Following the framework described in the previous section, we will run the weak learning algorithm between the booster and weak learners, for T rounds. This is in the online learning setting, where the algorithm will be run over a set of m training examples $\{(x_1, y_i), \ldots, (x_m, y_m)\}$. For round $t = 1$, the weights on each of the weak learners are set to be equal. For each of the subsequent rounds $t = 2, \ldots, T$, the weights are modified based on the correctness of classification. Considering the binary case of $\{-1, +1\}$, if an incorrect classification is made, the weights associated with the weak learners that produced the predictions rules that misclassified the training example i, that is the weak learners that produced the prediction rule h_t, for example i such that $h_t(x_i) \neq y_i$, is increased in order to force the algorithm to learn the "hard" cases. One approach used for weight modification is to only update the weights of the weak learners that misclassified the training example i by increasing them and not to modify the weights of the weak learners that correctly classified the training example i, as is done in the perceptron learning algorithm. Another approach

is to do the same for the weak learners that misclassified example i, and to decrease the weights associated with the weak learners that correctly classified the example i. Both approaches follow the same intuition, that is, to focus on learning the "hard" cases, thereby collectively driving the training error. This can be similarly extended to the multi-class problem in the same online setting. The most common method of extension is finding a reduction of the multi-class problem to the binary-class problem or a combination of binary-class problems. A reduction formulated by Schapire and Singer [6] is to ask the question of, for a training example x_i, is the correct label y_i or another label. Another reduction formulated by Freund and Schapire [2] is to ask the question of, for a training example x_i, is the correct label y_i or y_i', where $y_i \neq y_i'$. The approach described in the previous section can be applied to the multi-class problem in order to design the weak learning conditions to ensure boostability.

B. Gradient Boosting - XGBoost

In this next section, we will discuss gradient boosting and how it is different from an adaptive boosting algorithm such as the one described in the previous section. In adaptive boosting algorithms, the weights associated with different weak learners are adaptively modified based on misclassifications made on a training example. In gradient boosting algorithms, each weak learner of the ensemble-based algorithm will be sequentially trained using the gradient descent method, thereby driving down error on the data distribution. Gradient boosting has been shown to be more robust to outliers in the distribution of data. Adaptive boosting was designed to minimize classification loss, however, gradient boosting was designed to solve a differentiable loss function and therefore, can be used for classification as well as regression problems. Additionally, a key difference between adaptive boosting and gradient boosting is that the former performs up-weighting on the observations that were misclassified, whereas the latter identifies the "hard" cases to focus on by their larger residuals. Gradient descent can be augmented through the use of trees. In gradient tree boosting algorithms, decision trees are used as the weak learners in order to capture different features of the distribution being learned. The **XGBoost** [1] or extreme gradient boosting algorithm is an extension of gradient tree boosting which utilizes parallelized decision tree construction to improve training time and regularization techniques to reduce overfitting.

IV. FUTURE WORK

In this section, I will describe any future work that I would like to do on the topic of boosting for the binary and multi-class classification problems. There are several variants of analysis used by Mukherjee and Schapire in their paper, *A theory of multiclass boosting* [4]. They analyze their boosting framework as applied to different multi-class algorithms such as **Adaboost** [3]. I would like to spend more time understanding the mathematical proofs they provide for weak learning conditions and weak learner equivalence. In addition, I would like to gain an understanding of how decision trees can be employed as weak learners to further drive down the training error of a boosting framework. Finally, I would like to look at its extension to the **XGBoost** [1] algorithm for the multi-class problem.

V. CONCLUSION

Through this short paper, I gained a deeper understanding of boosting, the necessary and sufficient conditions to guarantee boosting (weak learning conditions), and a few boosting algorithms. Additionally, from a game-theoretic point of view, I developed an understanding of these algorithms and the boosting framework in general. I also learned about the application of boosting to the binary and multi-class setting.

VI. ACKNOWLEDGMENTS

I'd like to thank Dr. Avrim Blum for making the course, *TTIC 31250: Introduction to the Theory of Machine Learning*, so interesting.

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *CoRR* abs/1603.02754 (2016). arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754.

[2] Yoav Freund and Robert E Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.1997.1504. URL: https://www.sciencedirect.com/science/article/pii/S002200009791504X.

[3] Yoav Freund and Robert E. Schapire. "Experiments with a New Boosting Algorithm". In: *ICML*. 1996.

[4] Indraneel Mukherjee and Robert E. Schapire. *A theory of multiclass boosting*. 2011. DOI: 10.48550/ARXIV.1108.2989.

[5] Robert E. Schapire. "The Strength of Weak Learnability". English (US). In: *Machine Learning* 5.2 (June 1990), pp. 197–227. ISSN: 0885-6125. DOI: 10.1023/A:1022648800760.

[6] Robert E. Schapire and Yoram Singer. "Improved Boosting Algorithms Using Confidence-Rated Predictions". In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. COLT' 98. Madison, Wisconsin, USA: Association for Computing Machinery, 1998, pp. 80–91. ISBN: 1581130570. DOI: 10.1145/279943.279960. URL: https://doi.org/10.1145/279943.279960.