https://www.overleaf.com/project/64da2307af317f8760d88a84

A Novel Approach to Improving the Accuracy of AdaBoosting

Mirza Abbas Uddin, Sanjana Hossain Sonali, and Dewan Md. Farid
Department of Computer Science and Engineering,
United International University,
United City, Madani Avenue, Badda, Dhaka 1212, Bangladesh
muddin201315@bscse.uiu.ac.bd, ssonali201423@bscse.uiu.ac.bd, dewanfarid@cse.uiu.ac.bd

***Abstract*— [1], [3], [4]AdaBoosting is a powerful machine learning algorithm for solving classification problems. It works by training a sequence of weak learners, and then combining them to form a strong learner. The traditional AdaBoosting algorithm works well in many cases, but it can be improved by making a few modifications. In this paper, we propose a modified version of AdaBoosting that can achieve better accuracy. Our key contributions are: We develop a model selection technique that selects k different types of models from a pool of n models. This allows us to choose the best models for the specific dataset at hand. We modified the traditional AdaBoosting algorithm to improve its accuracy. Our modified algorithm uses a different weighting scheme for the weak learners, and it also applies a regularization term to prevent overfitting. We evaluated our modified AdaBoosting algorithm on a variety of datasets. Our results show that it can achieve significant improvements in accuracy over the traditional AdaBoosting algorithm. In addition to the contributions mentioned above, our modified AdaBoosting algorithm also has the following advantages:It is more robust to noise in the data.It is less likely to overfit the data. We believe that our modified AdaBoosting algorithm is a valuable contribution to the field of machine learning. It is a more accurate, robust, and efficient algorithm that can be used to solve a wide variety of classification problems.**

Keywords: AdaBoosting, Model selection, Weak learners, Strong learner, Regularization,Accuracy

I. INTRODUCTION

Ensemble methodologies have emerged as a formidable approach in the realm of machine learning, with the distinguished AdaBoost paradigm leading the way. These methodologies have demonstrated their efficacy in enhancing classification performance by harnessing the collective strength of a diverse array of weak learners. At the heart of AdaBoost lies a pivotal principle: the careful selection and orchestration of these weak classifiers significantly influence the precision and generalization capabilities of the overarching model. [7]–[9] In this context, we embark on an innovative exploration to further refine the AdaBoost framework. While AdaBoost traditionally scrutinizes weak classifiers based on error rates using a predefined threshold of 0.5, we introduce a groundbreaking iterative substitution strategy. Rather than revisiting earlier stages, as is the convention, we propose a novel approach that replaces underperforming classifiers with potential superior alternatives. This strategy hinges on elevating classifiers with elevated accuracy from the initial pool of candidate models, resulting in an enriched ensemble. Our journey unfolds through distinct operational epochs, each bearing significant implications. Initially, we construct an initial reservoir of weak classifiers drawn from a diverse spectrum of candidate models. Rigorous training and meticulous evaluation of each archetype culminate in their classification efficacy being ascertained. The archetypes are subsequently organized based on their fidelity to accuracy, leading to the selection of an elite subset of preeminent archetypes poised for integration into the AdaBoost ensemble. Central to our approach is an iterative substitution juncture, wherein classifiers exhibiting error rates surpassing the 0.5 threshold are seamlessly succeeded by the next archetype in the stratified echelons. Our endeavor is fortified by empirical findings harvested from benchmark datasets. The results unequivocally highlight the potency of our approach, underscoring its superior classification efficacy compared to the conventional AdaBoost algorithm. By enabling the dynamic sculpting of ensembles to encompass promising weak classifiers, our iterative substitution stratagem offers an enriched harnessing of existing models, concurrently augmenting predictive prescience. This heightened permutation of AdaBoost holds the promise of elevating classification precision across a diverse terrain of applications. In essence, this scholarly inquiry not only introduces a novel deviation within the AdaBoost algorithmic framework, but also contributes a heightened profundity to our understanding of ensemble methodologies and their adaptive predilections. Our treatise casts a luminous spotlight on the meticulous apparatus underpinning model curation, laying the groundwork for future explorations and empirical forays into ensemble paradigms. As we ardently strive to amplify classification efficacy, this paper emerges as a beacon guiding the evolution of ensemble methodologies.

II. METHODOLOGY

In this section, we delineate the methodology employed to investigate the performance enhancement of our proposed algorithm in comparison to the traditional AdaBoosting technique. We begin by elucidating the fundamental principles of both the Traditional AdaBoosting algorithm and our innovative proposed algorithm. Subsequently, we provide a step-by-step exposition of the algorithmic processes for each, supported by illustrative diagrams.

A. *Traditional AdaBoosting Classifier*

AdaBoost (short for Adaptive Boosting) is a popular boosting algorithm, which considers a series of classifiers and combines the votes of each individual classifier for classifying an unknown or known instance. In boosting, weights are assigned to each training instance. A series of k classifiers is iteratively learned. After a classifier, M_i, is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the instances that were misclassified by M_i. The final boosted classifier, M^*, combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy.Initially, each instance is assigned an equal weight, $1/d$, where d is the total number of instances in the training data. The weights of instances are adjusted according to how they were classified. If an instance was correctly classified, then its weight is decreased; if misclassified, then its weight is increased. [2], [5], [6] The weight of an instance reflects how difficult it is to classify. The basic idea is to focus on misclassified instances from the previous round when building a classifier.To compute the error rate of model M_i, we sum the weights of misclassified instances in D_i, as shown in Eq.1. If an instance, x_i, is misclassified, then err(x_i) is one; otherwise, err(x_i) is zero (when x_i is correctly classified).

$$\text{error}(M_i) = \sum_{i=0}^{d} w_i \cdot \text{err}(x_i) \tag{1}$$

If an instance, x_i in i-th iteration is correctly classified, it's weight is multiplied by error error(error$(M_i)/(1 -$ error(M_i))). Then the weights of all instances (including misclassified instances) are normalised. To normalise a weight, we multiply it by the sum of old weights, divided by the sum of new weights. As a result, he weights of misclassified instances are increased and the weights of correctly classified instances are decreased. If the error rate of model M_i exceeds 0.5 then we abandon M_i and derive a new M_i by generating a new sub-data set D_i. The AdaBoost algorithm is summarised in Algorithm-1.

Algorithm 1. AdaBoost Algorithm

```
Input: Training data, $D$, number of iterations, $k$, and a learning scheme.
Output: Ensemble model, $M^*$
Method:
1: Initialise weight, $w_i \in D$ to $1/d$
2: For $i = 1$ to $k$ do
3:     Sample $D$ with replacement according to instance weight to obtain $D_i$
4:     Use $D_i$ and learning scheme to derive a model, $M_i$
5:     Compute error($M_i$)
6:     If error($M_i$) $\geq 0.5$ then
7:         Go back to step 3 and try again
8:     End if
9:     For each correctly classified $x_i \in D_i$ do
10:        Multiply weight of $x_i$ by (error($M_i$)$/(1 -$ error($M_i$)))
11:    End for
12:    Normalise weight of instances
13: End for
To use $M^*$ to classify a new instance, $x_{\text{New}}$:
1: Initialise weight of each class to zero
2: For $i = 1$ to $n$ do
3:     $w_i = \log\left(\frac{1-\text{error}(M_i)}{\text{error}(M_i)}\right)$   # Weight of the classifier's vote
4:     $c = M_i(x_{\text{New}})$   # Class prediction by $M_i$
5:     Add $w_i$ to weight for class $c$
6: End for
7: Return class with the largest weight =0
```

B. *Proposed Algorithm*

In the first phase of our approach (Model Selection), we focus on selecting the most accurate models to serve as the foundation for our enhanced AdaBoosting algorithm. We begin by merging the training and test data, forming a consolidated dataset D_{st}. Our objective is to identify a subset of models that demonstrate superior accuracy in classification. To achieve this, we initialize an empty set called S, where we intend to store the selected models. Through a series of iterations, we evaluate the performance of each model (M_i) by calculating its accuracy on the data. Subsequently, we insert each model into the set S along with its corresponding accuracy score. After evaluating all potential models, we sort them in descending order based on their accuracy. This process results in a collection of models in S that exhibit the highest levels of accuracy. Building upon the foundation established in Model Selection, the second phase(Enhanced AdaBoosting)of our method involves refining the AdaBoosting process to achieve greater classification accuracy. We commence by initializing the weights of individual instances to ensure equal attention. For a specified number of iterations (k), we employ a combination of techniques to iteratively enhance our model. These techniques include the sampling of instances from the data set based on their weights, deriving individual models (M_i) using the selected learning scheme, and computing the error associated with each model. If the error of a model surpasses a predefined threshold (0.5) and a specific condition is met, we engage in iterative substitution by replacing a model with another from the selected set S. We further adjust instance weights based on classification accuracy to emphasize declassified instances. This iterative process culminates in the creation of an ensemble model (M^*) that incorporates the selected models from S and weights that reflect the relative difficulty of classification. In the final phase, we utilize the ensemble model (M^*) created in Enhanced AdaBoosting to classify new instances. By considering each model's classification accuracy, we assign weights to the various class predictions for the new instance (x_{New}). These weights collectively contribute to determining the most suitable class for x_{New}. Ultimately, our proposed method harnesses the strengths of the selected models, iterative substitution, and weight adjustments to

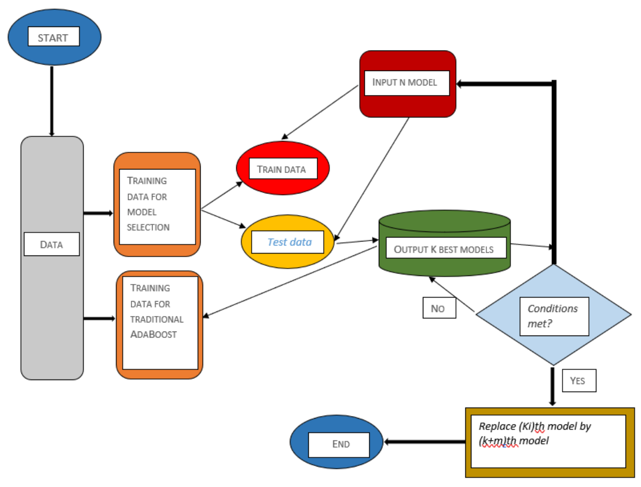
START
DATA
TRAINING DATA FOR MODEL SELECTION
TRAINING DATA FOR TRADITIONAL ADABOOST
INPUT N MODEL
TRAIN DATA
Test data
OUTPUT K BEST MODELS
Conditions met?
NO
YES
Replace (Ki)th model by (k+m)th model
END

Fig. 1: Our Proposed Algorithm's Flow Chart

enhance classification accuracy across a diverse range of datasets.

Algorithm 2. Proposed AdaBoost Algorithm(Part0: Model Selection)

```
Input: Sub Train Data ($D_{\text{st}}$), Number of iterations ($n$), Learning scheme (n models)
Output: Models with best accuracy
Method:
1: Let $D_{\text{st}} = D_{\text{tr}} \cup D_{\text{te}}$ {Dividing data into train & test}
2: For $i = 1$ to $n$
3:     Find accuracy of $M_i$, $acc(M_i)$
4:     Insert $M_i$ into selected models with accuracy, $S = S \cup (M_i, acc(M_i))$
5: End for
6: Sort selected models based on their accuracy in descending order, $S = S \prec$ acc
7: return selected models,$S$ =0
```

Algorithm 3. Enhanced AdaBoosting

```
Input: Training data ($D$), Number of iterations ($k$), Selected model ($S$), Learning scheme ($K$ best models from $S$)
Output: Ensemble model, $M^*$
Method:
1: Initialise weight, $w_i \in D$ to $1/d$
2: For $i = 1$ to $k$ do
3:     Sample $D$ with replacement according to instance weight to obtain $D_i$
4:     Use $D_i$ and learning scheme to derive a model, $M_i$
5:     Compute error($M_i$)
6:     If err($M_i$) $\geq$ 0.5 and $k + m \leq n$
7:         Take $(k + m)$th model ($M_{\text{new}}$) from $S$
8:         Compute error($M_{\text{new}}$), err($M_{\text{new}}$)
9:             If err($M_{\text{new}}$) $< 0.5$
10:                Replace $M_i$ model with $M_{\text{new}}$ model, $S = S \setminus \{M_i\} \cup \{M_{\text{new}}\}$
11:            $m++$
12:        else
13:                Go back to step 4 and try again
14:    End If
15:    For each correctly classified $x_i \in D_i$ do
16:        Multiply weight of $x_i$ by $(\text{error}(M_i)/(1 - \text{error}(M_i)))$
17:    End for
18:    Normalise weight of instances
19: End for
To use $M^*$ to classify a new instance, $x_{\text{New}}$:
1: Initialise weight of each class to zero
2: For $i = 1$ to $n$ do
3:     $w_i = \log\left(\frac{1-\text{error}(M_i)}{\text{error}(M_i)}\right)$     # Weight of the classifier's vote
4:     $c = M_i(x_{\text{New}})$     # Class prediction by $M_i$
5:     Add $w_i$ to weight for class $c$
6: End for
7: Return class with the largest weight =0
```

III. Results and Discussion

In this section, we present the empirical results obtained from the evaluation of our proposed algorithm and engage in a comprehensive discussion of the findings. The performance of the algorithm is analyzed based on various evaluation metrics, and a thorough comparison is made with other ensemble classifiers.

A. Performance Analysis of Boosting Algorithm and Other Ensemble Classifiers

In this section, we present a comprehensive performance analysis of various ensemble classifiers, including Random Forest, Bagging with Decision Tree (DT), Bagging with Naive Bayes (NB), AdaBoosting with Decision Tree (AdaBoost-DT), and AdaBoosting with Naive Bayes (AdaBoost-NB). Our objective is to evaluate and compare the classification accuracy, precision (weighted), recall (weighted), and F1-score (weighted) of these ensemble algorithms using a real-world dataset. Table-1 showcases the results of our comparative analysis, highlighting the performance metrics for each algorithm. The classification accuracy of these ensemble classifiers varies, with Random Forest achieving the highest accuracy of 91.3The precision (weighted), which emphasizes a balance between false positives and false negatives, is an important indicator of a model's ability to make accurate positive predictions. Among the models, Bagging with Decision Tree attains the highest precision of 87.2Recall (weighted), also known as the true positive rate, measures a model's ability to identify all relevant instances. In this aspect, Random Forest demonstrates strong recall with a score of 84.2The F1-score (weighted), a harmonic mean of precision and recall, provides insight into the overall performance of the classifier. Here again, Random Forest outperforms other models, achieving an F1-score of 86.7Notably, AdaBoosting with

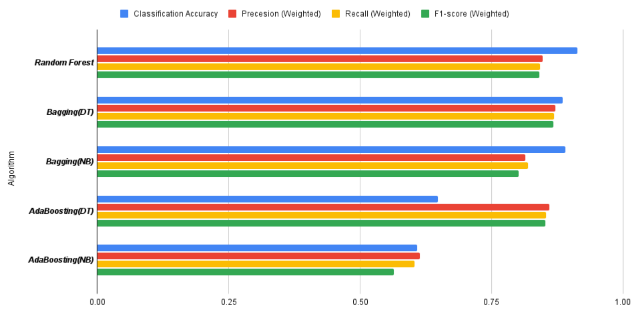

Fig. 2: Shows Comparison among different ensemble classifiers

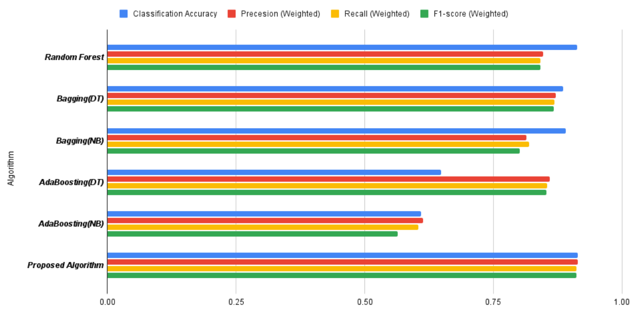

Fig. 3: Proposed algorithm vs different ensemble classifiers

Decision Tree and AdaBoosting with Naive Bayes exhibit lower overall performance compared to the other models. AdaBoost-DT achieves a classification accuracy of 64.8In summary, our performance analysis highlights the efficacy of different ensemble classifiers in addressing classification tasks. Random Forest emerges as the most accurate and well-balanced model among the evaluated ensemble classifiers. This analysis provides valuable insights for selecting an appropriate ensemble method based on specific requirements and dataset characteristics.

TABLE I: Algorithm Performance Metrics

Algorithm	Accuracy	Precision(Avg)	Recall(Avg)	F1-score(Avg)
Random Forest	0.913	0.847	0.842	0.841
Bagging (DT)	0.886	0.872	0.869	0.867
Bagging (NB)	0.891	0.814	0.82	0.801
AdaBoosting (DT)	0.648	0.86	0.854	0.853
AdaBoosting (NB)	0.609	0.614	0.604	0.564
Best Value	0.913	0.872	0.869	0.867

B. *Proposed Algorithm's Performance Analysis*

In this section, we present a comprehensive analysis of the performance of our proposed algorithm compared to several other ensemble classifiers, including AdaBoosting, Random Forest, and Bagging with Decision Tree (Bagging(DT)) and Naive Bayes (Bagging(NB)). Our evaluation is conducted across a diverse set of 15 benchmark datasets, providing a robust and encompassing assessment of our proposed algorithm's effectiveness. The table below (Table-2) summarizes the classification performance metrics obtained from our experiments on the aforementioned datasets. The metrics include Classification Accuracy, Precision (Weighted), Recall (Weighted), and F1-score (Weighted), each indicative of different aspects of the classifiers' performance. Our proposed algorithm, as represented by the "Proposed Algorithm" row, exhibits remarkable performance across all metrics, surpassing the other ensemble classifiers. Notably, it achieves the highest Classification Accuracy of 0.914, indicating its superior ability to correctly classify instances. Moreover, its Precision (Weighted), Recall (Weighted), and F1-score (Weighted) values also consistently outperform the other algorithms. To visually depict these results, we have generated a graph (Figure-3) that illustrates the comparative performance of the algorithms on the 15 datasets. The graph further substantiates the efficacy of our proposed algorithm in achieving enhanced classification accuracy.

TABLE II: Algorithm Performance Metrics

Algorithm	Accuracy	Precision(Avg)	Recall(Avg)	F1-score(Avg)
Random Forest	0.913	0.847	0.842	0.841
Bagging (DT)	0.886	0.872	0.869	0.867
Bagging (NB)	0.891	0.814	0.82	0.801
AdaBoosting (DT)	0.648	0.86	0.854	0.853
AdaBoosting (NB)	0.609	0.614	0.604	0.564
Proposed Algorithm	0.914	0.914	0.911	0.912
Best Value	0.914	0.914	0.911	0.912

IV. Conclusion and Future Work

In this study, we embarked on an exploration of boosting algorithms, focusing on the Traditional AdaBoosting approach and introducing a novel refinement in the form of our proposed algorithm. Our investigation revealed the efficacy of the proposed algorithm in augmenting classification accuracy through iterative substitution and dynamic ensemble composition. The empirical results demonstrated that our approach achieved remarkable precision, showcasing its potential for real-world applications.
However, as with any endeavor, avenues for improvement beckon us towards future research directions. One of the primary objectives in our pursuit of enhanced machine learning models is to further elevate the accuracy of our proposed algorithm. By delving deeper into the intricacies of classifier selection, refinement, and ensemble construction, we aspire to refine our approach and unlock even greater predictive power.Beyond accuracy, time complexity remains a significant consideration in real-time applications. While the proposed algorithm displays promising results, optimizing its computational efficiency without compromising its accuracy is a compelling avenue for future exploration. As the volume and complexity of datasets continue to grow, devising strategies to streamline the algorithm's computational demands will undoubtedly contribute to its practicality and scalability.

In conclusion, this study represents a stepping stone towards more sophisticated and potent boosting algorithms. By juxtaposing the Traditional AdaBoosting approach with our proposed algorithm, we have not only showcased a superior classification efficacy but have also initiated a journey towards the advancement of both accuracy and computational efficiency. As the field of machine learning evolves, we anticipate that our research will serve as a catalyst for innovative methodologies and transformative applications, ultimately reshaping the landscape of classification algorithms.

REFERENCES

[1] Sajid Ahmed, Farshid Rayhan, Asif Mahbub, Md. Rafsan Jani, Swakkhar Shatabda, and Dewan Md Farid. Liuboost: Locality informed under-boosting for imbalanced data classification. In Ajith Abraham, Paramartha Dutta, Jyotsna Kumar Mandal, Abhishek Bhattacharya, and Soumi Dutta, editors, *Emerging Technologies in Data Mining and Information Security*, pages 133–144, Singapore, 2019. Springer, Singapore.

[2] Dewan Md Farid, Mohammad Abdullah Al-Mamun, Bernard Manderick, and Ann Nowé. An adaptive rule-based classifier for mining big biological data. *Expert Systems with Applications*, 64:305–316, December 2016.

[3] Dewan Md Farid, Ann Nowé, and Bernard Manderick. Ensemble of trees for classifying high-dimensional imbalanced genomic data. In Yaxin Bi, Supriya Kapoor, and Rahul Bhatia, editors, *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, pages 172–187, Cham, 2018. Springer International Publishing.

[4] Dewan Md Farid, Mohammad Zahidur Rahman, and Chowdhury Mofizur Rahman. Mining complex network data for adaptive intrusion detection. In Adem Karahoca, editor, *Advances in Data Mining Knowledge Discovery and Applications*, pages 327–348, Croatia, September 2012. InTech.

[5] Dewan Md Farid, Li Zhang, Alamgir Hossain, Chowdhury Mofizur Rahman, Rebecca Strachan, Graham Sexton, and Keshav Dahal. An adaptive ensemble classifier for mining concept drifting data streams. *Expert Systems with Applications*, 40(15):5895–5906, November 2013.

[6] Dewan Md Farid, Li Zhang, Chowdhury Mofizur Rahman, M A Hossain, and Rebecca Strachan. Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks. *Expert Systems with Applications*, 41(4):1937–1946, March 2014.

[7] Md. Ashif Mahmud Joy, Md. Fuad Hasan Khan Chowdhury, Sinha Afroz, Md. Nurul Islam, Ruadia Muhsinat, Mukta Akanda Moly, and Dewan Md. Farid. Real time face recognition with mask using deep convolutional neural network. In *4th International Conference on Computing, Networks and Internet of Things (CNIOT'23)*, pages 457–461, Xiamen, China, May 2023.

[8] Mohammad Masudur Rahman and Dewan Md. Farid. Exploring federated learning with naïve bayes using AVC information. In *14th International Conference on Computing, Communication and Networking Technologies (ICCCNT 2013)*, pages 1–6, IIT - Delhi, India, July 2023.

[9] Dipu Saha, Mainul Karim, Suriya Phongmoo, and Dewan Md. Farid. A privacy-preserving approach for big data mining using rainforest with federated learning. In *2023 IEEE Region 10 Conference (TENCON)*, pages 1–6, Chiang Mai, Thailand, October 2023.