A New Approach of Boosting using Decision Tree Classifier for Classifying Noisy Data

Dewan Md. Farid*, Golam Morshed Maruf†, Chowdhury Mofizur Rahman†
*Computational Intelligence Group, Northumbria University, Newcastle upon Tyne, UK
†Department of Computer Science and Engineering, United International University, Bangladesh
d.farid@northumbria.ac.uk, gmmaruf@cse.uiu.ac.bd, cmr@uiu.ac.bd

***Abstract*—In the last decade, a good number of supervised learning algorithms have been introduced by the intelligent computational researchers in machine learning and data mining. Recently research in classification problems to reduce misclassification rate focuses on aggregation methods like Boosting, which combines many classifiers to generate a single strong classifier. Boosting is also known as AdaBoost algorithm, which uses voting technique to focus on training instances that are hard to classify. In this paper, we introduce a new approach of Boosting using decision tree for classifying noisy data. The proposed approach considers a series of decision tree classifiers and combines the votes of each classifier for classifying known or unknown instances. We update the weights of training instances based on the misclassification error rates that are produced by the training instances in each round of classifier construction. We tested the performance of our proposed algorithm with existing decision tree algorithms by employing benchmark datasets from the UCI machine learning repository. Experimental analysis proved that the proposed approach achieved high classification accuracy for different types of dataset.**

***Index Terms*—Boosting, classification, decision tree, noisy data.**

I. Introduction

Boosting or AdaBoost algorithm was first introduced by Freund and Schapire in 1997 [1]. It is an iterative learning process, which adaptively changes the distribution of training instances so that the base classifiers will focus on training instances that are hard to classify. Boosting approach combines many classifiers to generate a single strong classifier with very low misclassification errors in supervised learning. It classify a data instance by voting the weighted predictions of a set of base classifiers, which are generated in a series of rounds. Boosting have become one of the alternative framework for classifier design, together with the more established classifiers, like naïve Bayesian Classifier (NB), Decision Tree (DT), Neural Network (NN), and Support Vector Machine (SVM) [2], [3], [4]. Boosting assigns a weight to each training instance and adaptively changes the weights at the end of each boosting round. A sample is drawn according to the sampling distribution of the training instances to obtain a new training set. Next, a classifier is induced from the training dataset and used to classify all the training instances in the original dataset. The weights of the training instances are updated at the end of each boosting round. Training instances that are misclassified will have their weights increased, while those that are correctly classified will have their weight decreased. This forces the classifier to focus on training instances that are difficult to classify in subsequent iterations. The major drawback of boosting is overfitting; that is, with many rounds of boosting, the test error increase as the final classifier becomes overly complex [5], [6], [7]. The test error of AdaBoost algorithm often tends to decrease well after the training error is zero, and does not increase even after a very large number of rounds. However, the test error has been observed to increase slightly after an extremely large number of rounds [8].

The decision tree (DT) classifier is most powerful and popular learning algorithm for classification and prediction in supervised learning. It is easy to implement and requires little prior knowledge. DT can be constructed from any size of dataset with many attributes. In DT the successive division of the set of training instances proceeds until all the subsets consists of instances to a single class. A DT has three main components: nodes, leaves, and edges. Each node is labeled with an attribute by which the data is to be partitioned. Each node has a number of edges, which are labeled according to possible values of the attribute. An edge connects either two nodes or a node and a leaf. Leaves are labeled with a decision value for categorization of the data. To make a decision using a DT, start at the root node and follow the tree down the branches until a leaf node representing the class is reached. Each DT represents a rule set, which categorizes data according to the attributes of dataset. In this paper, we present a new approach of boosting using decision tree classifiers, which first initializes the weight of training instances to $\frac{1}{n}$, where n is the total number of training instances in training data. After that it creates a new dataset from original training dataset using selection with replacement technique and builds a decision tree using the new dataset. Then classify the original training instances with this decision tree. The weights of the training instances updated according to how they were classified. If an instance is misclassified then it's weight is increased, or if an instance is correctly classified then it's weight is decreased. Then it creates another new dataset with the misclassification error produced by each training instance from training dataset, and continues the process until all the training instances are correctly classified. To classify a new data instance use all the classifiers (that generated in each round) and consider the class of new data instance with highest classifier's vote. The proposed approach has been successfully tested on a number of benchmark problems from the UCI machine learning repository [18], which achieved high classification accuracy.

We organize this paper as follows. Section II discusses the types of noise in dataset and decision tree learning methods. The proposed algorithm is presented in Section III, followed by a description of benchmark datasets from the UCI machine learning repository used to evaluate the proposed approach in Section IV. Finally, conclusions and future work are drawn in Section V.

978-1-4799-0400-6/13/$31.00 ©2013 IEEE

II. State of Art

A. Types of Noise in the Training Dataset

The performance of the supervised learning algorithms in machine learning and data mining are depend on the quality of training datasets. Now a days, noise in training dataset is considered to be one of the most challenging issues in classification problems. The followings exhibit some common typical noise being existed in the training dataset.

1) Missing attribute values: Some of the attributes value are unknown or missing. The simplest way to dealing with missing attribute value is to replace the missing attribute value with the most frequent attribute value in the training dataset. Whereas, the most sophisticated way is to calculate the probability for attribute values and assign the probability value rather than the guessed value to each missing attribute value.

2) Contradictory instances: The same training instance appears more than once in the training dataset with different class labels. Contradictory training instances confuse the learning algorithms, so these instances should be avoided or labeled correctly before learning process.

3) Redundant instances: There often exist multiple copies of the same training instance in the training dataset. Redundant instances are not a problem if they do not form contradictions, but this redundancy can change the decision tree produced by ID3 algorithm. For data mining, its better to remove redundancy by keeping only a unique training instance in the training dataset. By doing so, it not only saves the space of storage in training dataset but also speeds significantly up the learning process.

4) Incomplete attribute problem: When the essential attributes of a problem are not used to describe in the training dataset. Suppose to distinguish men from women based on the descriptions of a large group of people in terms of gender, height, weight, qualifications, and so on. The right attribute for men is 'gender = male' and women is 'gender = female'. If we cannot catch the right attribute, then the classification model will be more complex and less accurate.

5) Misclassified instances: The training instances in the training dataset ware labeled with a wrong classification.

B. Decision Tree Learning Methods

The ID3 (Iterative Dichotomiser) method builds decision tree (DT) using information theory [10]. It chooses splitting attributes from a data set with the highest information gain value. The amount of information associated with an attribute value is related to the probability of occurrence. The concept used to quantify information is called entropy, which is used to measure the amount of randomness from a data set. When all data in a set belong to a single class, there is no uncertainty then the entropy is zero. The objective of DT classification is to iteratively partition the given dataset into subsets where all instances in each final subset belong to the single class. The entropy calculation is shown in equation 1. The value for entropy is between 0 and 1 and reaches a maximum when the probabilities are all the same. Given probabilities $p_1, p_2, \cdots, p_s$ where $\sum_{i=1} p_i = 1$,

$$Entropy : H(p_1, p_2, \cdots, p_s) = \sum_{i=1}^{s} (p_i log(\frac{1}{p_i})) \quad (1)$$

Given a dataset, D, $H(D)$ finds the amount of subset of dataset. When that subset is split into s new subsets $S = \{D_1, D_2, \cdots, D_s\}$, we can again look at the entropy of those subsets, A subset of dataset is completely ordered if all instances in it are the same class. ID3 chooses the splitting attribute with the highest gain. The ID3 algorithm calculates the gain by the equation 2.

$$Gain(D, S) = H(D) - \sum_{i=1}^{s} p(D_i) H(D_i) \quad (2)$$

The C4.5 is a successor of ID3 through GainRatio [9]. For splitting purpose, C4.5 uses the largest GainRatio that ensures a larger than average information gain.

$$GainRatio(D, S) = \frac{Gain(D, S)}{H\left(\frac{|D_1|}{|D|}, \cdots, \frac{|D_s|}{|D|}\right)} \quad (3)$$

The C5.0 algorithm improves the performance of building trees using boosting, which does not always help when the training data contains a lot of noise. CART (classification and regression trees) is a process of generating a binary tree for decision making [11]. CART handles missing data and contains a pruning strategy. The SPRINT (Scalable Parallelizable Induction of Decision Trees) algorithm uses an impurity function called gini index to find the best split [12]. Equation 4, defines the gini for a data set, D.

$$gini(D) = 1 - \sum p_j^2 \quad (4)$$

Where, p_j is the frequency of class C_j in D. The goodness of a split of D into subsets D_1 and D_2 is defined by

$$gini_{split}(D) = \frac{n_1}{n(gini(D_1))} + \frac{n_2}{n(gini(D_2))} \quad (5)$$

The split with the best gini value is chosen. A number of research projects for optimal feature selection and classification have been done in the last decade, which adopt hybrid strategy involving evolutionary algorithm and inductive DT learning [13], [14], [15], [16], [17].

III. Proposed Boosting Algorithm

In this section, we introduce a new boosting algorithm using decision tree classifier, which considers a series of decision trees and combines the votes of each individual decision tree for classifying an unknown or known data instance. This approach builds decision tree in each round and updates the weight of training instances based on the misclassification error rate that produced by the training instances in each round. This proposed algorithm addresses the problem of classifying the large dataset, which improves the classification accuracy.

In a given training dataset $D = \{t_1, \cdots, t_n\}$, where $t_i = \{t_{i1}, \cdots, t_{ih}\}$ and the attributes $\{A_1, A_2, \cdots, A_n\}$. Each attribute A_i contains the following attribute values $\{A_{i1}, A_{i2}, \cdots, A_{ih}\}$. The training data D also contains a set of classes $C = \{C_1, C_2, \cdots, C_m\}$. Each training instance has a particular class C_j. We first initialize the weights of training instances to an equal value of $w_i = \frac{1}{n}$, where n is the total number of training instances in D. Then we generate a new dataset D_i with equal number of instances from original training dataset D using selection with replacement technique and build a decision tree from the new created dataset D_i. Then

we classify all the instances in original training dataset D with this decision tree and the weights of the training instances t_i in training dataset D are updated according to how they were classified. If a data instance was misclassified then it's weight is increased, or if a data instance was correctly classified then it's weight is decreased.

To update the weights of training data D, we compute the misclassification rate, the sum of the weights of each of the training tuple t_i in D that were misclassified. That is,

$$error(M_i) = \sum_{i}^{d} W_i * err(t_i) \tag{6}$$

Where $err(t_i)$ is the misclassification error of instance, t_i. If the instance, t_i was misclassified, then $err(t_i)$ is 1. Otherwise, it is 0. The misclassification rate affects how the weights of the training instances are updated. If a training instance was correctly classified, its weight is multiplied by error $(\frac{M_i}{1-error(M_i)})$. Once the weights of all of the correctly classified instances are updated, the weights for all instances including the misclassified instances are normalized so that their sum remains the same as it was before. To normalize a weight, we multiply the weight by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified instances are increased and the weights of correctly classified instances are decreased. Now we generate another new data set D_i from training data D with maximum weight values and continues the process until all the training instances are correctly classified. To classify a new or unseen instance use all the decision trees of each round (each round is considered as a classifier) and consider the class of new tuple with highest classifier's vote. The main procedure of proposed boosting algorithm is described in Algorithm 1.

Algorithm 1 Boosting using Decision Tree

Input: D, Training data D of labeled instances t_i.
Output: An ensemble classification model.
Procedure:

1: Initialize the weight $w_i = \frac{1}{n}$ of each instance t_i in D, where n is the total number of training instances.
2: Generate a new dataset D_i with equal number of instances from D using selection with replacement technique.
3: Find the best splitting attribute in new dataset D_i with highest information gain value.
4: Create a node and label with splitting attribute. [First node is the root node, T of tree]
5: For each branch of the node, partition the data points and grow sub datasets by applying splitting predicate to dataset D_i.
6: For each sub datasets, if data points are all of the same class, C then a leaf node labeled with C. Else continue steps 3 to 6 until each final subset belong to the same class or leaf node created.
7: When the decision tree construction is complete, classify each training instance t_i in training data D.
8: Updates the weights of each training instances t_i in D, according to how they were classified. If an instance was misclassified then its weight is increased, or if an instance was correctly classified then its weight is decreased. To updates the weights of training instances the misclassification rate is calculated, the sum of the weights of each of the training instance t_i in D that were misclassified: $error(M_i) = \sum_{i}^{d} W_i * err(t_i)$; Where $err(t_i)$ is the misclassification error of instance t_i. If the instance t_i was misclassified, then $err(t_i)$ is 1. Otherwise, it is 0. If a training instance was correctly classified, its weight is multiplied by $(\frac{M_i}{1-error(M_i)})$. Once the weights of all of the correctly classified instances are updated, the weights for all instances including the misclassified instances are normalized so that their sum remains the same as it was before. To normalize a weight, multiply the weight by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified instances are increased and the weights of correctly classified instances are decreased.
9: Repeat steps 2 to 8 until all the training instances t_i in D are correctly classified.
10: To classify a new or unseen instance use all the decision trees (each round is considered as a classifier) and considers the class of new instance with highest classifier's vote.

IV. Experiments

In this section, we describe the data sets and experimental results.

A. Data Sets

A set of data items, the dataset, is a very basic concept of data mining and machine learning research. A dataset is roughly equivalent to a two-dimensional spreadsheet or database table. The proposed boosting method was tested on a number of widely used benchmark problems from UCI machine learning repository [18]. The datasets from the UCI repository are relatively small size and often lacking time tag. The main reason of using datasets from the UCI repository is that a number of solutions exist in the literature for classification. Table I and II describe about the datasets from UCI machine learning repository, which are used in experimental analysis.

1) Fitting Contact Lenses Database: It is very small dataset with only 24 data instances, 4 attributes and 3 class attribute (soft, hard, and none). All the attribute values are nominal in this dataset. The instances are complete and noise free and 9 rules cover the training set.

2) Diabetes: This data set prepared for the use of participants for the 1994 AAAI Spring Symposium on Artificial Intelligence in Medicine. Diabetes patient records were obtained from two sources: an automatic electronic recording device and paper records. The automatic device had an internal clock to timestamp events, whereas the paper records only provided "logical time" slots (breakfast, lunch, dinner, bedtime).

3) Iris Plants Database: This is one of the best known dataset in the pattern recognition literature. This dataset contains 3 class values (Iris Setosa, Iris Versicolor, and Iris Virginica), where each class refers to a type of iris plant. There are 150 instances and 4 attributes in this dataset (50 in each of three classes). One class is linearly separable from the other 2 classes.

4) Image Segmentation Data: The goal of this dataset is to provide an empirical basis for research on image segmentation and boundary detection. There are 1500 data instances in this

TABLE I
ATTRIBUTES NUMBERS AND TYPES IN DATASETS

Dataset	No of Attributes	Attributes Type
Contact Lenses	4	Nominal
Diabetes	8	Real
Glass Identification	9	Real
Iris Plants	4	Real
Image Segmentation	19	Real
Large Soybean	35	Nominal
Voting	16	Nominal
Weather	4	Nominal + Real

TABLE II
TOTAL INSTANCES AND CLASS VALUES IN DATASETS

Dataset	No of Instances	Class Attribute Values
Contact Lenses	24	3
Diabetes	768	2
Glass Identification	214	7
Iris Plants	150	3
Image Segmentation	1500	7
Large Soybean	683	19
Voting	435	2
Weather	14	2

dataset with 19 attributes and all the attributes are real. There are 7 class attribute values: brickface, sky, foliage, cement, window, path, and grass.

5) Large Soybean Database: There are 35 attributes in this dataset and all attributes are nominalized. There are 683 data instances and 19 class values in this dataset.

B. Experimental Results

We implement our algorithm in Java. The code for decision tree has been adapted from the Weka machine learning open source repository (http://www.cs.waikato.ac.nz/ ml/weka/). Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. The experiments were run on an Intel Core 2 Duo Processor 2.0 GHz processor (2 MB Cache, 800 MHz FSB) with 1 GB of RAM. The experimental results using 10 fold cross validation in table III and IV illustrate that proposed boosting approach achieved better classification rates than traditional decision tree C4.5 classifier.

V. CONCLUSIONS

In this paper, we introduce a new boosting approach using decision tree (DT) classifier to improve the classification rates with very low misclassification errors. This approach addresses the problem of classifying noisy data, because noisy data are very hard to classify. The DT classifier has several advantages such as it is easy to implement, requires little prior knowledge, abd also can be constructed from large datasets with many attributes. The performance of our proposed algorithm tested on 8 benchmark datasets from the UCI machine learning repository, and the experimental result proves that the proposed approach achieved high classification accuracy with very low misclassification error. The future research issue will be applying this method in real life classification problems.

TABLE III
PERFORMANCE OF C4.5 CLASSIFIER

Dataset	Classification Rate (%)	Misclassification Rate (%)
Contact Lenses	83.33	16.66
Diabetes	73.82	26.17
Glass Identification	66.82	33.17
Iris Plants	96.00	4.00
Image Segmentation	95.73	4.26
Large Soybean	91.50	8.49
Voting	96.32	3.67
Weather	64.28	35.71

TABLE IV
PERFORMANCE OF PROPOSED BOOSTING METHOD

Dataset	Classification Rate (%)	Misclassification Rate (%)
Contact Lenses	100	0.0
Diabetes	91.33	8.66
Glass Identification	93.48	6.51
Iris Plants	98.00	2.0
Image Segmentation	97.68	2.31
Large Soybean	100	0.0
Voting	98.56	1.43
Weather	100	0.0

ACKNOWLEDGMENT

We appreciate the support for this research received from the European Union (EU) sponsored (Erasmus Mundus) cLINK (Centre of excellence for Learning, Innovation, Networking and Knowledge) project (EU Grant No. 2645).

REFERENCES

[1] Y. Freund, and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, Vol. 55, 1997, pp. 119-139.

[2] R. Jin, and G. Agrawal, "Efficient decision tree construction on streaming data," *In Proc. of ACM SIGKDD*, 2003, pp. 571-576.

[3] T. Hastie, R. Tibshirani, and J. Rriedman, "The element of statistical learning," *Data Mining, Inference and Prediction, Heidelberg, Germany: Springer-Verlag*, 2001.

[4] K. M. A. Chai, H. T. Ng, and H. L. Chieu, "Bayesian online classifiers for text classification and filtering," *In Proc. of SIGIR 2002, Tampere, Finland*, August 11-15, pp. 97-104.

[5] L. Breiman, "Arcing classifier," *The Annals of Statistics*, Vol. 26, 1998, pp. 801-849.

[6] H. Drucker, and C. Cortes, "Boosting decision trees," *Advances in Neural Information Processing Systems*, Vol. 8, 1996, pp. 479-485.

[7] J. R. Quinlan, "Bagging, boosting, and C4.5," *Proc. of the 13 National Conference on Artificial Intelligence*, 1996, pp. 725-730.

[8] A. J. Grove, and D. Schuurmans, "Boosting in the limit: Maximizing the margin of learned ensembles," *Proc. of the 15 National Conference on Artificial Intelligence*, 1998.

[9] J. R. Quinlan, "C4.5: Programs for Machine Learning," *Morgan Kaufmann Publishers*, San Mateo, CA, 1993.

[10] J. R. Quinlan, "Induction of Decision Tree," *Machine Learning*, Vol. 1, 1986, pp. 81-106.

[11] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees," *Statistics probability series*, Wadsworth, Belmont, 1984.

[12] John Shafer, Rakeeh Agrawal, and Manish Mehta, "SPRINT: A scalable parallel classifier for data mining," *Morgan Kaufmann*, 1996, pp. 544-555.

[13] D. Turney, "Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm," *Journal of Artificial Intelligence Research*, 1995, pp. 369-409.

[14] J. Bala, J. Huang, H. Vafaie, K. DeJong, and H. Wechsler, "Hybrid Learning using Genetic Algorithms and Decision Trees for Pattern Classification," *Proc. 14th Int'l Con. on Artificial Intelligence*, Montreal, August 19-25, 1995, pp. 1-6.

[15] C. Guerra Salcedo, S. Chen, D. Whitley, and Stephen Smith, "Fast and Accurate Feature Selection using Hybrid Genetic Strategies," *Proc. Genetic and Evolutionary Computation Con.*, 1999, pp. 1-8.

[16] S. R. Safavian, and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 3, 1991, pp. 660-674.

[17] W. Y. Loh, and X. Shih, "Split selection methods for classification tree," *Statistica Sininca*, Vol. 7, 1997, pp. 815-840.

[18] A. Frank, and A. Asuncion, "UCI Machine Learning Repository," *University of California, Irvine, School of Information and Computer Sciences*, 2010, http://archive.ics.uci.edu/ml