# Making AdaBoost Less Prone to Overfitting On Noisy Datasets

Zainab Ghadiri Modarres, Mahmood Shabankhah and Ali Kamandi

February 21, 2020

# Making AdaBoost Less Prone to Overfitting On Noisy Datasets

Zainab Ghadiri Modarres
School of Engineering Science,
College of Engineering,
University of Tehran
Tehran, Iran
zainab.ghadiri@ut.ac.ir

Mahmood Shabankhah
School of Engineering Science,
College of Engineering,
University of Tehran
Tehran, Iran
shabankhah@ut.ac.ir

Ali Kamandi
School of Engineering Science,
College of Engineering,
University of Tehran
Tehran, Iran
kamandi@ut.ac.ir

***Abstract*— AdaBoost is perhaps one of the most well-known ensemble learning algorithms. In simple terms, the idea in AdaBoost is to train a number of weak learners in an increamental fashion where each new learner tries to focus more on those samples that were misclassfied by the preceding classifiers. Consequently, in the presence of noisy data samples, the new leraners will somehow memorize the data, which in turn will lead to an overfitted model. The main objective of this paper is to provide a generalized version of the Adaboost algorithm that avoids overfitting, and performs better when the data samples are corrupted with noise. To this end, we make use of another ensemble learning algorithm called ValidBoost [15], and introduce a mechanism to dynamically determine the thresholds for both the error rate of each classifier and the error rate in each iteration. These threshholds enable us to control the error rate of the algorithm. Experimental simulations has been made on several benchmark datasets to evaluate the performance of our proposed algorithm.**

***Keywords—Ensemble learning algorithms, Boosting, Adaboost, Overfitting, Noise, zero_one_loss.***

## I. Introduction

For decades, ensemble learning has attracted a lot of attention among computer science and machine learning researchers. In fact, ensemble learning has been developed to reduce variance and improve the accuracy of decision-making problems. Ideas based on ensemble learning are also used in our life. When we consult others and decide on the majority's opinion, we actually use the ensemble learning method [1].

For example, in 1785, the French mathematician and philosopher Nicolas de Caritat presented the famous Condorcet's jury. This is a jury opinion that requires a decision with a binary result. This has two main constraints: 1. Voters are independent; 2. there are only two possible outcomes. He states that the combination of information from different sources in decision making is much better than decision-making based on a single source of information [2].

Ensemble learning methods are considered as an advanced solution for many of the machine learning problems. These methods increase the efficiency of the model by training several models and combining their results [2].

In the following we will read:

Section2: Definition of Ensemble Learning, Section3: Related worked, Section4: provides a generalized Adaboost algorithm, Section5: Experimental results, at the end: Conclusions.

## II. Ensemble Learning

In 2017, Omer Sagi and Lior Rokach introduced the concepts and methods of traditional and modern ensemble learning and discussed new challenges in this field, saying that the main hypothesis of ensemble learning is that by combining Several models, the error of one classifier is likely to be covered by another, and as a result, the overall ensemble learning performance is better than one classifier [2].

The classification error rate consists of two controllable components: the bias is the precision of classifier, and the variance is the accuracy of model with different training data. These two components have a trade-off relationship with each other, and the classifier with low bias class tends to have higher variance and vice versa. The purpose of the ensemble learning systems is to create several classifiers with similar bias, which after the combination of their results; the variance can be reduced [1].

In ensemble learning, each classifier creates different errors on the samples, but generally all agree on the result of the classification correctly. As a result, the classification accuracy is increased. Therefore, averaging the results of each classifier reduces the error of ensemble classification. Here are two important points. First, there are several ways to integrate the classifier in ensemble-based systems, averaging is one of them. Second, integrating the results of classifier do not necessarily lead to guarantee a better performance than the best classifier in the ensemble-based system, but it reduces the likelihood of choosing a poorly-efficient classifier. In general, if we knew which class would be the best, it would not be necessary to use ensemble-based system, but because this is unclear, we need an ensemble learning system [1].

Three strategies are needed to build an effective ensemble leaning model:

1. Data Sampling and Selection: Diversity; Different data sampling leads to produce various ensemble learning algorithms. For example, sampling uniform and with replacing[1] the training data lead to Bagging algorithms, while the sampling of a distribution of misclassified training data is the core of the ensemble algorithms.

2. Training Member Classifiers; Ensemble learning algorithms have been developed for training ensemble classifiers, the most common of which are Bagging (similar algorithms arc-x4 and random forests, Boosting (its various types), stack generalization and hierarchical MoE [1].

3. Combining Ensemble Members; the last step in any ensemble-based system, is a mechanism for combining individual classifiers. The strategy used in this step depends on the classifiers type. For example, classifiers such as SVM generate only discrete-valued label outputs, which most commonly used combination rules for these classifiers is majority voting followed at a distant second by the Borda count. Other classifiers, such as multilayer perceptron or (naive) Bayes classifier, provide continuous valued class-specific outputs, which are interpreted as the support given by the classifier to each class. Some categorical ensemble methods include:

- Combining Class Labels (Majority Voting, Weighted Majority Voting, Borda Count)
- Combining Continuous Outputs (Algebraic Combiners like: Mean Rule, Weighted Average, trimmed mean, Minimum/Maximum/Median Rule, Product Rule, Generalized Mean, Decision Template) [1].

The combination of predictive classifiers with uncorrelated errors in an ensemble is the main idea of the ensemble learning algorithms in combining two algorithms. There is a linear relationship between the degree of error reduction and the degree to which patterns of errors made by individual models are uncorrelated. [3]. There are several ways to combine various classifiers:

1- Input manipulation: In this method, each base model is fitted by a different training set and variable input samples.

2- Manipulated learning algorithm: In this method, the use of each basic model varies. For example, one way is to manipulate a basic model in the hypothesis space. This is done by leading the base model to various convergence paths.

3- Partitioning: Diversity can be achieved by dividing the original dataset into smaller subsets and then using each subset to train a different inducer. In horizontal partitioning, we divide the original dataset into several sets that include the entire feature-set so that inducers differed only by their instances. Vertical partitioning works in the opposite way as each inducer uses the same instances but with different features

4- Output manipulation: This approach refers to techniques that combine numerous binary classifiers into a single multiclass classifier. Error-correcting output codes (ECOC) is a successful example of this approach

5- Ensemble hybridization: This approach combines at least two strategies when creating the ensemble methods. The random forest algorithm is the most well-known development of the hybridization approach. RotBoost is an example of a hybrid of the rotation forest and AdaBoost algorithms. In each iteration, a new rotation matrix is generated and used to create a dataset. The AdaBoost ensemble is induced from this dataset.

Ensemble learning methods can be divided into two main categories: independent and dependent.

In the dependent framework, the output of each classifier affects the structure of the next classification. Also, knowledge created in previous repetitions guides learning in subsequent iterations.

In the independent framework, each classifier is independently constructed of other classifiers.

In some combining methods, you see both templates. Let’s introduce each ensemble methods:

TABLE 1: Method categories [2]

| Method name | Fusion method | Dependency | Training approach |
|---|---|---|---|
| *AdaBoost* | ***Weightning*** | ***Dependent*** | ***Input manipulation*** |
| *Bagging* | ***Weightning*** | ***Independent*** | ***Input manipulation*** |
| *Random forest* | ***Weightning*** | ***Independent*** | ***Ensemble hybridization*** |
| *Random subspace methods* | ***Weightning*** | ***Independent*** | ***Ensemble hybridization*** |
| *Gradient boosting machines* | ***Weightning*** | ***Dependent*** | ***Output manipulation*** |
| *Error-correcting output codes* | ***Weightning*** | ***Independent*** | ***Output manipulation*** |
| *Rotation forest* | ***Weightning*** | ***Independent*** | ***Manipulated learning*** |
| *Extremely randomized trees* | ***Weightning*** | ***Independent*** | ***Partitioning*** |
| *Stacking* | ***Meta-learning*** | ***Independent*** | ***Manipulated learning*** |

- AdaBoost [5]: It is the most well-known algorithm to build an ensemble model. Its main idea is focus on examples that have misclassified training data in previous iteration. The focus is based on the weight of each sample in the training dataset. In the first repetition, the weight of all the samples is the same. In each repetition, the weight of misclassified samples increases

[1] Bootstrap

and the weight of the classified samples decreases and all weights have been normalized.

- Bagging [6]: It is an effective and simple method to generate a group of independent models in which each classifier taught using bootstrap samples of the dataset. To ensure the adequacy of the samples in each classifier, each model contains the same number of samples from the original data set. A majority prediction rating is effective in determining the final decision to predict the unseen sample [7].

- Random forest [8]: This algorithm was originally designed for decision trees as base learners, and mainly designed to select the subsets of the properties of the nodes during the branching. Recent research suggests that Random Forest is more resistant to other machine learning algorithms such as SVM, Neural networks, especially with a small training dataset [9].

- Gradient boosting machines [10]: In this algorithm, training of each classifier is dependent on the previously trained classifiers. The main difference between this method and other techniques is that the optimization of this method is used in the function space.

- Rotation forest [11]: A method that causes a variety in decision tree algorithms, with training each classifier in dataset by rotating the specification space.

- Extremely randomized trees [12]: It is another method for producing various collections with randomness training process. It is similar to Random Forest, but there are two differences: 1- Extremely randomized trees don't apply the bagging procedure to construct a set of the training samples for each tree. The same input training set is used to train all trees. 2- Extremely randomized trees pick a node split very extremely, whereas Random Forest finds the best split among random subset of variables [14].

Table1 briefly summarize Ensemble learning methods.

One of the problems in ensemble learning is overfitting. It is said to be a bad phenomenon in statistics, in which the model's degree of freedom is much higher than the real degree of freedom, and thus, although the model yields a very good result on training data, it has a high error on test data. Choosing the right degree of freedom by Cross-validation and Regularization is one of the ways to deal with this phenomenon.

The likelihood of overfitting is that the criterion of fitting the model is not the same as the standard of evaluate it. To measure the effectiveness of the model, not only it should be measured its efficiency on training samples, but also measured the ability of the model on unseen samples.

Overfitting occurs when the model begins to "memorize" the data instead of "learning" in training step. Also, in ensemble learning algorithms such as Adaboost, in each iteration, it focuses on misclassified samples, it has a problem in noisy dataset because it memorized the noisy data and as a result it produces noise in training process and it leads to overfitting in learning step. It is a main problem in machine learning.

Big data is characterized by properties such as speed, diversity, accuracy, variability, scalability, and value. For example, speed is important in real-time decision-making systems, and variability refers to the dynamic nature of data that may lead to produce drift. In recent years, researchers' focus is on scalability of data mining algorithms [13].

## III. Related works

This section focuses on some ways and solutions that have been presented and deal with overfitting.

AdaBoost algorithm learns by repeated calculations, and it classifies by focusing on misclassified data [15]. That is why it has tended to overfit to deal with noisy data. Overfitting occurs when the model begins to "memorize" data instead of "learning".

In ensemble learning algorithms like Adaboost (Algorithm1), in each iteration, their focusing is on the classification of samples that are classified incorrectly in previous iteration, as a result it creates noise during the training process, memorize the model and noisy data learning process, causes overfitting in the algorithm. Moreover, this is a major problem in machine learning. In thesis [15], it is stated that by using a validation set that obtained from training data can be prevented overfitting. To deal with overfitting, two algorithms are presented, which are updated versions of Adaboost. The proposed algorithms are the validboost algorithm (Algorithm 2) and the cross-validated algorithm Adaboost (Algorithm 3).

```
Input: Dataset x, consisting of $N$ objects $\langle X_{1,\dots,}X_N\rangle \in X$ with labels
$\langle y_{1,\dots,}y_N\rangle \in Y = \{1,\cdots,c\}$
Input: Weak learning algorithm WeakLearn
Input: Number of iterations $T$
Initialize: Weight vector $w_i^1 = \frac{1}{N}$ for $i = 1,\cdots,N$
for $t$ = 1 to $T$ do
    Call Weaklearn, providing it with weights $w_i^t$
    Get back hypothesis $h_t: X \to Y$
    Compute $\epsilon = \sum_{i=1}^{N} w_i^t\,[h_t(x_i) \neq y_i]$
    if $\epsilon > 1 - \frac{1}{c}$ then
        Set $\alpha_t = 0$
        Set $w_i^{t+1} = w_i^t$
    else
        Compute $\alpha_t = \log((1-\epsilon)/\epsilon) + \log(c-1)$
        Set $w_i^{t+1} = w_i^t \exp(\alpha_t[h_t(x_i) \neq y_i])$ for all $i$
        Normalize $w_i^{t+1}$
    end
end
Output: Hypothesis $H(x) = \underset{y\in Y}{\operatorname{argmax}} \sum_{t=1}^{T} \alpha_t[h_t(x) = y]$
```

Algorithm1: AdaBoost

One of the common ways to prevent overfitting is to use a validation set to evaluate model and we have real error instead of training errors. A development instance of this is cross-validation. It has been observed that most changes occur in a few first Adeboost iteration. In t's iteration, the effect of a new classifier on the ensemble algorithm is 1/t. That is why we define:

$$\tau = \frac{\log t}{\log T}\#$$

$$\epsilon = \tau\epsilon_v + (1-\tau)\epsilon_t$$

Where "t" is the current iteration and T is the total number of iteration and ϵ_v is the error weight in the test set and ϵ_t is the error weight in the training set. In the first few iterations, the value of τ will be small. As "t" increases, τ will be closer to "1", and the effect of validation will be high for preventing overfitting.

In both algorithms, the set of data is divided into two categories: training and validation. The size of the validation set is τN/2.

```
Input: Dataset x, consisting of $N$ objects $\langle X_{1,\dots,}X_N\rangle \in X$ with labels $\langle y_{1,\dots,}y_N\rangle \in Y = \{1,\cdots,c\}$
Input: Weak learning algorithm WeakLearn
Input: Number of iterations $T$
Initialize: Weight vector $w_i^1 = \frac{1}{N}$ for $i = 1,\cdots,N$
for $t = 1$ to $T$ do
    Set $\tau = \frac{\log t}{\log T}$
    Split $x$ into $x_v$ of size $^{\tau N}/_2$ and $x_t$
    Call Weaklearn on $x_t$, providing it with weights $w_i^t$ for $\{i: x_i \in x_t\}$
    Get back hypothesis $h_t: X \to Y$
    Compute $\epsilon_v = \sum_{\{i: x_i \in x_v\}} w_i^t [h_t(x_i) \neq y_i]$
    Compute $\epsilon_t = \sum_{\{i: x_i \in x_t\}} w_i^t [h_t(x_i) \neq y_i]$
    Set $\epsilon = \tau\epsilon_v + (1-\tau)\epsilon_t$
    if $\epsilon > 1 - \frac{1}{c}$ then
        Set $\alpha_t = 0$
        Set $w_i^{t+1} = w_i^t$
    else
        Compute $\alpha_t = \log((1-\epsilon)/\epsilon) + \log(c-1)$
        Set $w_i^{t+1} = w_i^t \exp(\alpha_t[h_t(x_i) \neq y_i])$ for all $i$
        Normalize $w_i^{t+1}$
    end
end
Output: Hypothesis $H(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{t=1}^{T} \alpha_t [h_t(x) = y]$
```

Algorithm2: ValidBoost

For each classifier in the ValidBoost algorithm, the error rate is:

$$\epsilon = \tau\epsilon_v + (1-\tau)\epsilon_t \#$$

And for each classifier in the Cross-Validated Adaboost algorithm, the error rate is:

$$\epsilon = (\epsilon_v + \epsilon_t)/2\#$$

If the error rate of classifier is greater than $1 - \frac{1}{c}\#$(c is the number of classes), then that classifier will be deleted from the process. Otherwise, weights are updated.
Updated in the ValidBoost algorithm is:

$$\alpha_t = \log((1-\epsilon)/\epsilon)\#$$
$$w_i^{t+1} = w_i^t \exp(\alpha_t[h_t(x_i) \neq y_i])$$

Updated in the Cross-Vallidated Adabbost algorithm is:

$$\alpha_t = \log\left(\frac{(1-\epsilon)}{\epsilon}\right) + (\log(c-1))$$
$$w_i^{t+1} = w_i^t \exp(\alpha_t[h_t(x_i) \neq y_i])$$

The performance of validboost algorithm is similar to Adaboost in encounter to non-noisy data, but in the face of noisy data, it's performance is higher than Adaboost algorithm, and it does not suffer overfitting like Adaboost algorithm.

In paper [16] the authors point out the applications of deep learning, especially artificial neural networks, they state that the high number of parameters in CNN allow neural network to learn complex features. And this leads to overfitting in training data. they also state that, despite the proposed methods in this issue, overfitting is still a problem at CNN. Among the many factors that lead to overfitting, the number of FCLs[2] parameters should be considered. The authors of this article suggests the SparseConnect method. SparseConnect is a simple idea to reduce its overfitting by reducing the connections of CFLs. Experimental results in the three benchmarks MNIST, CIFAR10 and ImageNet indicate that SparseConnect provides better output than other methods.

```
Input: Dataset x, consisting of $N$ objects $\langle X_{1,\dots,}X_N\rangle \in X$ with labels $\langle y_{1,\dots,}y_N\rangle \in Y = \{1,\cdots,c\}$
Input: Weak learning algorithm WeakLearn
Input: Number of iterations $T$
Input: Number of folds $k$
Initialize: Weight vector $w_i^1 = \frac{1}{N}$ for $i = 1,\cdots,N$
for $t = 1$ to $T$ do
    Split $x$ into $k$ sets of equal size and with equal prior probabilities, $X_1,\cdots,X_k$
    for $f$=1 to $k$ do
        Set the validation set $x_v$ to $x_f$
        Set the training set $x_t$ to $\bigcup_{i \neq f} x_i$
        Call Weaklearn on $x_t$, providing it with weights $w_i^t$ for $\{i: x_i \in x_t\}$
        Get back hypothesis $h_{t,f}: X \to Y$
        Compute $\epsilon_v = \sum_{\{i: x_i \in x_v\}} w_i^t [h_t(x_i) \neq y_i]$
        Compute $\epsilon_t = \sum_{\{i: x_i \in x_t\}} w_i^t [h_t(x_i) \neq y_i]$
        Set $\epsilon = (\epsilon_v + \epsilon_t)/2$
        if $\epsilon > 1 - \frac{1}{c}$ then
            Set $\alpha_{t,f} = 0$
            Set $w_i^{t+1} = w_i^t$ for all $i$
        else
            Compute $\alpha_{t,f} = \log((1-\epsilon)/\epsilon) + \log(c-1)$
            Set $w_i^{t+1} = w_i^t \exp(\alpha_{t,f}[h_{t,f}(x_i) \neq y_i])$ for all $i$
            Normalize $w_i^{t+1}$
        end
    end
end
Output: Hypothesis $H(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{f=1}^{k} \sum_{t=1}^{T} \alpha_{t,f} [h_{t,f}(x) = y]$
```

Algorithm3: Cross-Validated AdaBoost

The authors of [17] said that classic AdaBoost algorithm is a collection of weak learners and can be used to construct a strong classifier. This algorithm has limitations such as sensitivity to noisy data. In this paper, a selective boosting method, sBoost, is proposed to solve this problem. The focus of this method is based on classifying efficiency instead of misclassified samples. This methodology has been developed to efficiently classify patterns with a noise level of less than 10%. The effectiveness of the developed sBoost technique has been analyzed by a series of simulation tests. The results of this analysis show that the developed sBoost technique can improve the classification accuracy and prevent overfitting.

In the paper [18], it is stated that there are some methods to overcome the overfitting problem in the Bayesian algorithm, the

[2] fully-connected layers

first method is to introduce a new regularization weight parameter; another method is to stop early, which can be used to stopped repetitive branching process. Also, in this study, cross-validation is described as a simple solution to get better performance. It also suggests that in order to minimize overfitting, the data set should be re-sampled every iteration into the training and validation collections.

As stated in paper [15], the root of overfitting in Adaboost arises from the fact that Adaboost is a repetitive method that wants to reduce the error of classification, and one of reasons of overfitting is that the base learners are not sufficiently weak. In Adaboost, the important key issues are selection and integration of classifiers. The goal of each classifier is to reduce the amount of composition of bias and variance. The ensemble methods do this well, and combine a high-bias classifier with low-bias classifier and a high-variance classifier with a low-variance classifier. In this thesis, the author makes several adjustments on Adaboost and examines the effect of each on the classification efficiency, and its proposed changes improve classifiers efficiency when training data is very noisy.

In paper [19] that outcomes of [15], It is states that Adaboost is an iterative algorithm that quickly rises high performance by focusing on objects that are hardly classified. And because of that, with noisy data set, it is strongly inclined to overfitting. It is also stated that it is possible to avoid overfitting by using validation dataset obtained from the same noisy training dataset. In this paper, the ValidBoost algorithm is introduced, which is an Adaboost algorithm that uses validation dataset. ValidBoost's performance is the same as the performance of Adaboost when dataset has no noise, but It has been improved Adaboost performance with noisy data. The validBoost algorithm is less likely to overfit than Adaboost algorithm.

So far, several methods have been proposed to prevent Adaboost overfitting [20].

The author of paper [4] said one of the major challenges in training deep neural network is prevention of overfitting. Many techniques have been proposed to reduce overfitting without to need large amount of training data, such as enhancing and reinforcement data or new settings like Dropout. In this paper, a new regulation called Decov has been proposed that significantly reduces overfitting. Decov can be used in both one layer or multilayer of neural network. Decov does not need supervision, so it can be added to any activation set. In this paper, Decov is used for completely connected layers and affects all the layered parameters that are used. Decov reduces overfitting by the difference between training accuracy and test accuracy. Decov Acts like a regulator, and performance of algorithm with decov is usually better than without Decov or Dropout.

The purpose of paper [10] is to present the concept drift, and also is a new learning algorithm that used to deal with noisy data and data that changes the samples and their performance (concept drift). The subject matter is that a characteristic of the flow data is their variability, and this causes some instances that enter the system be noisy, or caused deviations in other samples. Therefore, in the proposed algorithm, a factor called confidence level specified that each classifier which its confidence level is less than the threshold is eliminated from the making decision in ensemble algorithm. This threshold is also proven that should not be static and should be dynamic. Therefore, there is an adjustment factor in the algorithm that if the result of classification of the algorithm is the same with one classifier or without that classifier, and the threshold is higher than zero, then the threshold would be lowered to the size of the adjustment factor, and if the results are not the same and the threshold is less than one, then we increase the threshold as much as the decision factor. Algoritm4.

```
Input: ensemble $\widehat{\Psi}$, abstaining threshold $\theta \in [0,1]$, adjustment factor $s \in [0,1]$
$\theta \leftarrow$initialize threshold
L $\leftarrow$size of the ensemble
While end of stream = FALSE do
    Obtain new instance x from the stream
    for $l \leftarrow 1\,; l \leq L; l++$ do
        Obtain classifier support $F_{\Psi_l}(x)$ for each class
        if $\max_{j \in M} F_{\Psi_l}(x,j) < \theta$ then
            $\Psi_1$ abstains from the decision
        else
            $\Psi_1$ participates in voting
    z $\leftarrow$ result of non-abstaining classifiers voting
    Obtain lable y of object x
    if z==y then then
        $\theta \leftarrow \theta - s$   (if $\theta > 0$)
    else
        $\theta \leftarrow \theta - s$   (if $\theta > 0$)
```

Algorthm4: Proposed general framework for dynamic abstaining online ensembles.

In this paper, we introduce a factor called the Equalized Loss of Accuracy (ELA), which helps us to examine whether the performance of the algorithm on noisy data is related to actual robustness of algorithm, or only because of the differences in initial predictive accuracies.

$$ELA_{x\%} = \frac{(100 - ACC_{x\%})}{ACC_{0\%}}$$

## IV. Proposed method

The proposed method is integrated the ValidBoost method with a dynamic threshold. The basic of this method is the ValidBoost algorithm, with the difference that each time the base classifier error rate is compared with a confidence level, if the error exceeds this level, a new holdout sampling of the substituted training data is performed.

It replaces the previous training data, but if the error rate of the underlying class is lower than the confidence level, the confidence level will be replaced by the error rate of the previous step, and go to next iteration.

This algorithm will also be run n times. Each execution time the maximum number of classifier in the algorithm will be n. At each run, if the algorithm error rate exceeds a threshold, the training data is re-selected by bootstrap sampling. And if lower, the threshold is lowered by the adjustment factor. The algorithm is executed in the next loop with the number of new iterations. The proposed algorithm pseudocode is visible in Algorithm 5.

```
Input: Dataset x, consisting of $N$ objects $\langle X_{1,\dots,}X_N\rangle \in X$ with labels
$\langle y_{1,\dots,}y_N\rangle \in Y = \{1,\cdots,c\}$
Input: Weak learning algorithm WeakLearn
Input: Number of iterations $T$
Input: $a$ , $\theta \in [0,1]$, adjustment factor $s \in [0,1]$
Initialize: Weight vector $w_i^1 = \frac{1}{N}$ for $i = 1,\cdots,N$
$\theta$ ←initialize threshold
for $i$ = 1 to $n$ do
T = i
    $a$ ←confidence level
    for $t$ = 1 to $T$ do
        Set $\tau = \frac{\log t}{\log T}$
        Split $x$ into $x_v$ of size $^{\tau N}/_2$ and $x_t$
        Call Weaklearn on $x_t$, providing it with weights $w_i^t$ for
                        $\{i: x_i \in x_t\}$
        Get back hypothesis $h_t: X \to Y$
        Compute $\epsilon_v = \sum_{\{i:\, x_i \in x_v\}} w_i^t [h_t(x_i) \neq y_i]$
        Compute $\epsilon_t = \sum_{\{i:\, x_i \in x_t\}} w_i^t [h_t(x_i) \neq y_i]$
        Set $\epsilon = \tau\epsilon_v + (1-\tau)\epsilon_t$
        if $\epsilon > 1 - \frac{1}{c}$ then
            Set $\alpha_t = 0$
            Set $w_i^{t+1} = w_i^t$
        else
            Compute $\alpha_t = \log((1-\epsilon)/\epsilon) + \log(c-1)$
            Set $w_i^{t+1} = w_i^t \exp(\alpha_t[h_t(x_i) \neq y_i])$ for all $i$
            Normalize $w_i^{t+1}$
        end
        if e < a then then
            a=e
      else
          $bootstrap(x)$

    end
    Output: Hypothesis $H(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{t=1}^{T} \alpha_t [h_t(x) = y]$
Compute Error = zero_one_loss ()
if Error > $\theta$ then then
        $bootstrap(x)$
        $\theta$ = Error
else
        $\theta = \theta$ - s
```

Algorithm5: Proposed Algorithm

## V. Experimental

The datasets used in our experiment selected from the UCI Machine Learning Repository site and the Kaggle site. Information on this dataset given in Table 1.

TABLE 2: DataSet information

| | dataset | #instance | #feathers | #class | balanced | drift-noise |
|---|---|---|---|---|---|---|
| 1 | opticaldigits | 947 | 1024 | 10 | balanced | |
| 2 | abalone | 4177 | 8 | 29 | imbalanced | |
| 3 | electricity | 45312 | 8 | 2 | balanced | yes |
| 4 | default-of-credit-card-clients | 30000 | 24 | 2 | imbalanced | |

The proposed algorithm implemented on four different datasets, multiclass or binary, balanced or imbalanced. For comparison, the zero_one_loss criterion, a common measurement in classification learning. The zero_one_loss is a common measurement in classification learning, which means that the predictor response is incorrect or false. Zero means that the prediction is correct, and one means that a mistake occurs during the classification. It is better that this measurement is closer to zero. Zero_one_loss measurement is similar to the mean squared error regression.

As you can see from the results of the implementation of these three algorithms on the different datasets in Figures one to three, the performance of the proposed method is better than that of Adabost and ValidBust. This dataset does not include noise data.

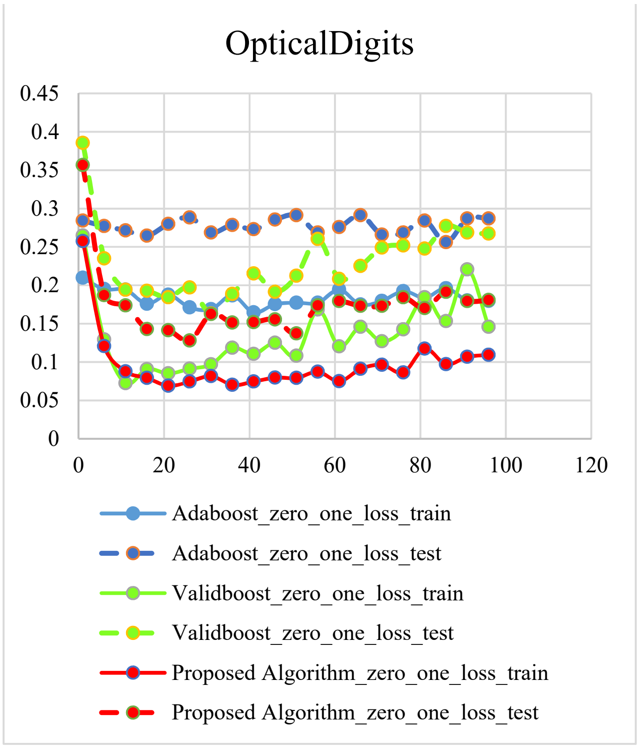

Figure 1: Comparing AdaBoost, ValidBoost and Proposed Algorithm, Dataset: OpticalDigits

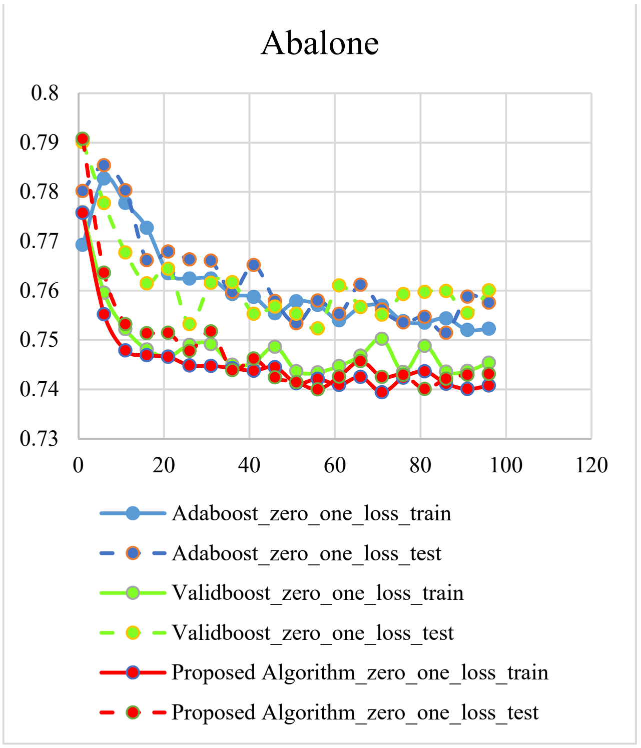

Figure 2: Comparing AdaBoost, ValidBoost and Proposed Algorithm, Dataset: Abalone

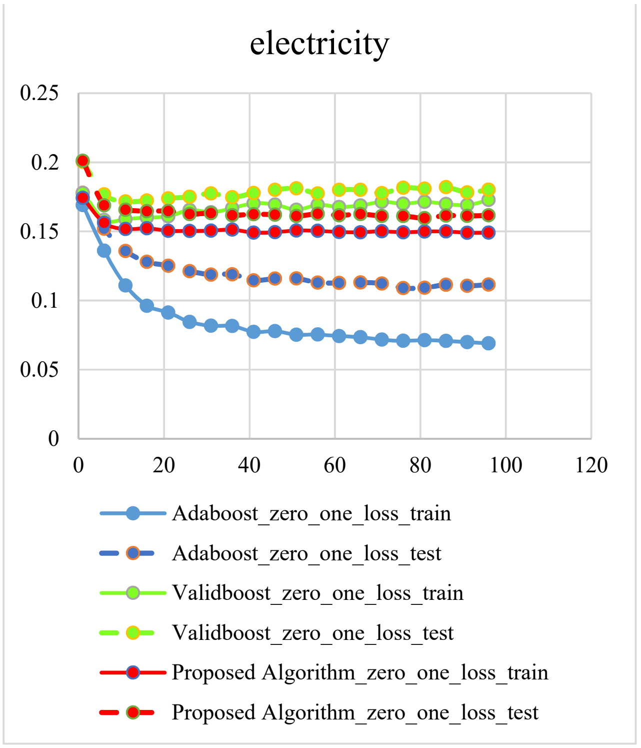

Figure 4: Comparing AdaBoost, ValidBoost and Proposed Algorithm, Dataset: Electricity

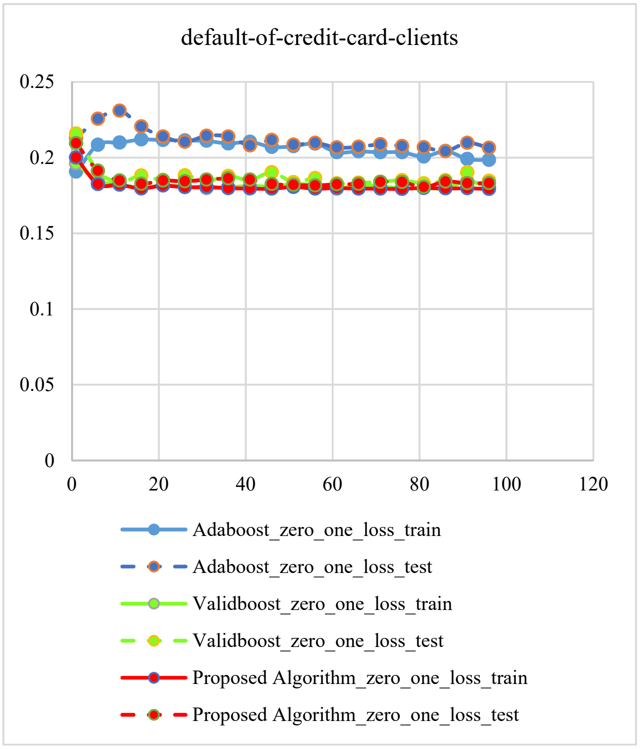

Figure 3: Comparing AdaBoost, ValidBoost and Proposed Algorithm, Dataset: Default-of-credit-card-clients

The Electricity dataset is a binary balanced data set with noisy data. The Adaboost algorithm is very sensitive to noise data and suffering from overfitting. As you can see in the Adaboost algorithm, the error rate of the training data significantly reduced but the error rate of test data did not decrease; it means that the algorithm on the training data is much better than the test data. It is the concept of overfitting that occurs because of noisy data, and the algorithm starts to memorize the model of noisy data rather than learning. As you can see in Fig. 5, the proposed algorithm slightly improves the Adaboost overfitting problem and performs better than ValidBoost.

## VI. Conclusion

In this paper, we studied various verions of Adaboost algorithm. Adaboost tries to train a sequence of classfiers where each new classifier is more focused on those instances that have been misclassified by the preceding classifiers. Therefore, if the data samples are corrupted with noise, AdaBoost might lead to overfitted results. To tackle this problem, Dirk introduced ValidBoost [15]. We made some minor modifications in ValidBoost by introducing a dynamic threshold in order to control the reduce even more the error rate of the algorithm. Experimental simulations show that our proposed algorithm performs better than both ValidBoost and AdaBoost in terms of error rates and is less sensitive to noise than AdaBoost.

## References

[1] Zhang, Cha, and Yunqian Ma, eds. “Ensemble machine learning: methods and applications.”, Springer Science & Business Media, 2012.

[2] Sagi, Omer, and Lior Rokach. “Ensemble learning: A survey.” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8.4 (2018): e1249.

[3] Ali, Kamal M., and Michael John Pazzani. “On the link between error correlation and error reduction in decision tree ensembles.” Information and Computer Science, University of California, Irvine, 1995.

[4] Cogswell, Michael, et al. "Reducing overfitting in deep networks by decorrelating representations." Published as a conference paper at ICLR 2016.

[5] Freund, Yoav, and Robert E. Schapire. “A desicion-theoretic generalization of on-line learning and an application to boosting.” European conference on computational learning theory. Springer, Berlin, Heidelberg, 1995.

[6] Breiman, Leo. “Bagging predictors.” Machine learning 24.2 (1996): 123-140.

[7] Kuncheva, Ludmila I. “Combining pattern classifiers: methods and algorithms.” John Wiley & Sons, 2014.

[8] Amit, Yali, and Donald Geman. “Randomized Inquiries About Shape: An Application to Handwritten Digit Recognition.” No. TR-401. CHICAGO UNIV IL DEPT OF STATISTICS, 1994.

[9] Han, Te, et al. “Comparison of random forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery.” Transactions of the Institute of Measurement and Control 40.8 (2018): 2681-2693.

[10] Krawczyk, Bartosz, and Alberto Cano. "Online ensemble learning with abstaining classifiers for drifting and noisy data streams." Applied Soft Computing 68 (2018): 677-692.

[11] Rodriguez, Juan José, Ludmila I. Kuncheva, and Carlos J. Alonso. “Rotation forest: A new classifier ensemble method.” IEEE transactions on pattern analysis and machine intelligence 28.10 (2006): 1619-1630.

[12] Geurts, Pierre, Damien Ernst, and Louis Wehenkel. “Extremely randomized trees.” Machine learning 63.1 (2006): 3-42.

[13] Wu, Xindong, et al. “Data mining with big data.” IEEE transactions on knowledge and data engineering 26.1 (2013): 97-107.

[14] https://docs.opencv.org/2.4/modules/ml/doc/ertrees.html

[15] Dirk W. J. Meijer, “Regularizing AdaBoost to prevent overfitting on label noise”, Master thesis, the Delft University of Technology, 2016 .

[16] Xu, Qi, et al. “Overfitting remedy by sparsifying regularization on fully-connected layers of CNNs.” Neurocomputing 328 (2019): 69-74.

[17] Li, De Z., Wilson Wang, and Fathy Ismail. "A selective boosting technique for pattern classification." Neurocomputing 156 (2015): 186-192.

[18] Julien-Charles Lévesque, “Bayesian Hyperparameter Optimization: Overfitting, Ensembles and Conditional Spaces”, Thèse, 2018.

[19] Meijer, Dirk WJ, and David MJ Tax. "Regularizing AdaBoost with validation sets of increasing size." 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, 2016.

[20] Bylander, Tom, and Lisa Tate. "Using Validation Sets to Avoid Overfitting in AdaBoost." Flairs conference. 2006.