Improvement of Heterogeneous AdaBoost Algorithm by Ant Colony Optimization Algorithm

zahra rahimi far (zahra.rahimifar65@gmail.com)
Payame Noor University

Research Article

Keywords: data boosting in classification, boosting-based methods, overfitting, ant colony meta-heuristic algorithm

Posted Date: May 20th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1540582/v1

Improvement of Heterogeneous AdaBoost Algorithm by Ant Colony Optimization Algorithm

***Zahra Rahimi Far, professor of Payam Noor University**

Zahra.rahimifar65@gmail.com

Abstract

Due to its various uses in different areas, machine learning has become one of the most popular fields of artificial intelligence. It can be certainly said that data classification by trained machine vision algorithms is currently used as a model in most of software types. It is obvious that the variety of methods for modeling the samples leads to different percentages of accurate detection in the data of a shared dataset. The higher is this percentage, the more valid will be the proposed method. The output of classification algorithms may cause a large extent of interference in detection. This interference can be due to the noisy data or the samples that are less frequently observed in data training phase. To boost the output of this method, we use meta-heuristic algorithms that boost the improperly-classified samples by determining the error level in predetermined assumptions. One of these methods is AdaBoost that is characterized by the major deficiency of lack of noise sensitivity and overfitting. Based on the proposed method, we can decrease the noise sensitivity and improve the overfitting problem in less frequent data by using the ant colony optimization algorithm; so that the probability of improper classification of the outliers is significantly reduced.

Keywords: data boosting in classification, boosting-based methods, overfitting, ant colony meta-heuristic algorithm

1. Introduction

Data mining has been used as a new and effective method in the area of artificial intelligence for several years. One of the research areas in data mining is data classification. Classification is a process of finding a model for defining and distinguishing the data concepts and classes to be used for prediction of the classes or the objects with an unknown class label (the objective variable). This model is developed based on the analysis of the training datasets [1]. There is always a set of data considered as imbalanced data. Due to their ambiguous features, such data are usually assigned the least priority by the classification algorithms. Since the policy of most of

classification methods is to classify the data in the major classes, the data classified in the minor classes are considered as the noise and they are classified improperly. The skew distribution of the data cannot delay the learning process by itself. Rather, it can cause other problems such as the low volume of the samples, overlapping, class resolution, and the existence of sub-concepts (the patterns existing among the major models and having less frequency than the major models) which make the problem of imbalanced classes more complicated [2]. To overcome the mentioned problems, there are some algorithms known as boosting algorithms. These algorithms receive noisy classes and boost their power based on their weight or their content in several steps.

2. AdaBoost meta-algorithm

Adaboost meta-algorithm was proposed by Freund Schapire in 1999. At the beginning, he aimed to propose a model to observe the effectiveness of different features in the problem space and to propose some coefficients as the boosting coefficients in order to boost the output of the weak classifications that are considered as the imbalanced datasets [3]. Since AdaBoost can provide better outputs in linear classifications, it has a major problem i.e. lack of noise sensitivity and unwanted boosting of noise; so that overfitting is observed in the training data before testing the final data. This situation occurs when the algorithm tries to reduce the training data error compared to the increased training data error. The model in which overfitting occurs usually loses its predicted efficiency and unexpectedly, it leads to magnification of the dysfunctions. In this situation, instead of trying to learn the training data, the boosting model tries to preserve (memorize) them for classification and detection. Various methods have been proposed for solving this problem. These methods include the binary search tree, regulation etc. So, since the noisy data or the unclassified data are processed by boosting algorithm, it is highly probable that noise is boosted in the training data and an inefficient model is resulted from the boosted noisy classes. So, it is said that boosting algorithms are not sensitive to overfitting.

The AdaBoost output is a linear set of the classifications each of which are boosted separately and added to the data training process. Formula 1 indicates the final output of the AdaBoost algorithm with weak classifications (ht(xi)).

$$\mathrm{Hx} = Sign\left(\sum \ (at * ht(xi)\right) \qquad (1)$$

In the first iteration of the algorithm, after allocating the 1/D error to each of the weak classifiers, a hypothesis is raised based on which, the error of each classifier is calculated (in terms of xi) and its minimum value is returned as the classifier that should be boosted further (formula 2). These weak classifiers can indicate the output of a linear classifying algorithm such as decision tree [5].

$$Zt = \sum Di(t)\exp(-at * yi * ht(xi)) \quad (2)$$

In the following, after choosing "at" as one of the real datasets (R), the weight allocated to all the classifiers is updated (formula 3).

$$Dt + 1(i) = \frac{Di(t)\exp(-at*yi*ht(xi))}{Zt} \quad (3)$$

Updating the weights of the weak classifiers, the algorithm iterations will continue from the first phase.

3. Threshold determination in AdaBoost meta-algorithm

In a general viewpoint, the main goal of AdaBoost is to minimize the error level in each of the weak classifiers in which, a threshold is considered as the linear discriminant. In fact, this discriminant is chosen to determine a centroid in the data of that weak classifier. This centroid is a border between the positive and negative data (in two-class classifiers). In the following, the hypothesis raised for the i^{th} feature is compared to all of these values and the error percentage is determined. It is obvious that the border point that can determine the least error is considered as the final threshold [5].

The first step is distribution of the data by presenting them on the total problem space: In this step, all the samples allocated to each of the existing classes are distributed on the problem space.

The second step is selection of all the dataset features in a periodical manner and formation of a structure for studying the weak classifiers.

The third step is determination of the threshold on the selected feature: For this purpose, first all the dataset samples are studied one by one in a linear manner (on the xi field), and all the data that are equal to, larger, and smaller than the current data are analyzed as the main indexes. In the following, a hypothesis is raised based on which, all the samples belonging to the main indexes are assigned the value of 1 and the remaining data (the minor indexes) are assigned the value of -1.

$$y_{hypothesis}\begin{cases} 1\ if\ x \in index \\ -1\ \text{if x } \neq \text{ index} \end{cases} \quad (4)$$

At the end of every iteration, the number of final outputs corresponding to the raised hypothesis (the value of the objective field i.e. y that is classified by a linear classifier such as decision tree) is determined (formula 5). It is obvious that the higher is this value, the higher it is likely to be

determined as the threshold level in the weak classifier (at this point, the classifier error is less than the others because of the consistence of more samples with the raised hypothesis).

$$\text{y\_error} = \sum \left(y_{predict} \neq y_{hypothesis}\right) \quad (5)$$

After determining the threshold in the upper and lower bounds of each of the weak classifiers, these points are compared to each other (the upper bound refers to all the data larger than the threshold value and vice versa). In the following, the bound resulting in a lower error is determined as the threshold level.

The fourth step is repetition of the second and third steps until reaching the end of the feature vector.

Ultimately, the fifth step is boosting each classifier in a linear manner by creating the maximum correspondence between the raised hypothesis and the samples existing in the imbalanced classes [7].

$$\text{Hx} = Sign\left(\sum \left(at * ht(xi)\right)\right) \quad (6)$$

One of the deficiencies of this method is linear separation of the data that are consistent and inconsistent with the raised hypothesis by AdaBoost meta-algorithm. Selecting the threshold level out of all samples existing in a weak classifier can to some extent guarantee the accuracy of the boosted data. However, this algorithm has the following deficiencies:

- A high time order when selecting the threshold level
- Lack of focus on selection of a more accurate point with less error in searching the threshold level
- Lack of noise sensitivity with the probability of increase in the value ranges after separation
- Unwanted boosting of data in imbalanced classes which leads to overfitting in the dataset

4. Ant colony algorithm

The deficiencies of linear selection of threshold levels were already stated. To resolve these deficiencies, we have used the ant colony meta-heuristic algorithm. In the proposed model, instead of choosing the threshold level out of the samples existing in a weak classifier, we have chosen a value out of the thousands of existing values, so that it can detect the best location for separating and proving the hypothesis raised by AdaBoost (it is not necessary for the selected value to exist among the samples of the weak classifier). The goal is to choose a threshold level that can reduce the error of each weak classifier. So, we face an optimization problem on a large scale. First, we study ant colony algorithm and then, we present the block diagram of the

proposed model. Optimization problems belong to a branch of artificial intelligence that aims to find the best answer in large-space problems. It is not usually simple to find the minimum and maximum values of a function in an infinite set of values, especially if we intend to check all the different samples in a linear manner. In this situation, the medium-size datasets require an exponential time order. Meta-heuristic algorithms have been proposed to resolve the problem of time complexity and testing all the different combinations of the samples.

Ant colony algorithm is one of these meta-heuristic algorithms. In this algorithm, the ants play the role of intelligent agents that lay a chemical substance called pheromone on the paths leading to the food resources, and in this way, they direct the other members of the colony towards them. First, each ant randomly chooses a path from its colony towards the food resource. In the following, the pheromones remained on the path will direct other members to the track (Figure 1).

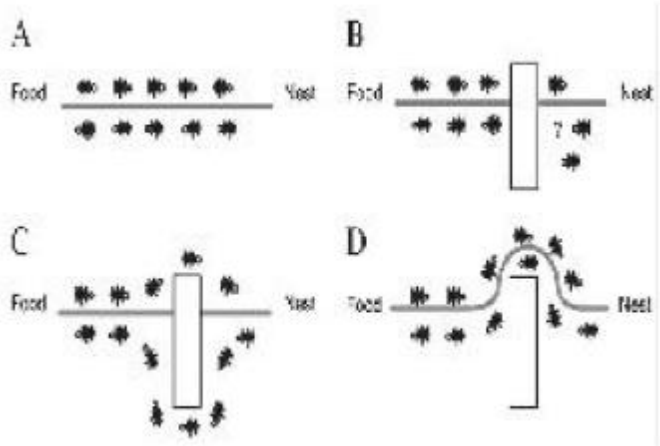

Figure1. A indicates the ants' movement on a smooth path. B indicates the ants' movement towards the food resources passing through the obstacle and moving in one direction. C indicates the ants' movement passing through the obstacle and moving in two directions. D indicates the ants' movement towards the food resources passing through the optimal path.

The ant colony algorithm stages is as follows:

1. Generating primary population of the ants based on the predetermined number
2. Choosing the paths leading to the final answers in a random manner based on the following probability function

$$Pij = \frac{[Tij]a[\eta i,j]B}{\sum_{h}^{s}([Tij]a[\eta i,j]B)} \quad (7)$$

In the above formula, T is equal to the amount of pheromone existing in the path I, j and ŋ indicates the distance between the two nodes I, j.

3. Calculation of the response cost by a function that assigns a score to each of the selected paths.
4. Updating the amount of pheromone existing in the path by formula 8.

$$T_{i,j}(t) = (1-p).\ T_{ij}(t-1) + p.T0 \ldots\ldots \quad (8)$$

5. Finding the best answer obtained in the current path and comparing that to the best answer obtained in the whole paths.

6. Updating the pheromone amount in the optimal path and determining the probability of selecting each of them separately.

$$T_{i,j}(t) = (1-p).\ T_{ij}(t-1) + \frac{P}{L} \ldots. \quad (9)$$

7. Returning to the first step and determining the final answer.

5.Methodology

5.1. The proposed algorithm

The first step: receiving the training dataset and moving in the feature vector starting from the first feature.

The second step: determining the upper and lower bounds in the Xi feature and distributing a specific number of values between these two bounds as the nodes of a graph (in this phase, all the nodes are considered as a connected graph including all the values ranging between the upper and lower bounds. The advantage of this method is selection of the values that do not exist in Xi feature and can be chosen as an optimal threshold level by the ant colony algorithm).

The third step: distribution of a specific amount of pheromone on the graph paths and initializing the pheromone table.

The fourth step: random selection of each node by the colony members and determining the effectiveness of it in the problem space by fitness function.

The fifth step: defining the fitness function to score the answer obtained by each ant as an intelligent agent. In this function, all members that are equal to, larger and smaller than the proposed answer are studied and respectively assigned a value of 1 and -1 (the raised hypothesis). In the following, the error of the proposed answer is calculated (returning the average number of the samples that are inconsistent with the raised hypothesis).

The sixth step: selection of the best fitness value as the global optimum value and consequently, updating the pheromone table of the graph paths.

The seventh step: ending the current iteration and rearranging the intelligent agents by returning to the second step.

5.2. The proposed algorithm parameters table

Using the proposed model of ant colony algorithm in determining the threshold points leads to getting alternating answers. So, it is necessary to determine the parameters that can reduce this alternation and make the problem more convergent. It should be noted that table 5 indicates the parameters that have been obtained experimentally by repeated running of the algorithm on mentioned datasets. (These parameters are static and they will not change in every iteration)

Table 5-1. The proposed algorithm parameters

Parameter	Value
The number of iterations of the algorithm	100
The number of iterations of ant colony algorithm in each round	10
The ants population in each round	50
The changes in pheromone amount	0.01

The other parameter that should exist in the above table is the number of dataset folds (k-fold). Since the k value can directly affect the algorithm output, it has been considered as one of the comparative parameters.

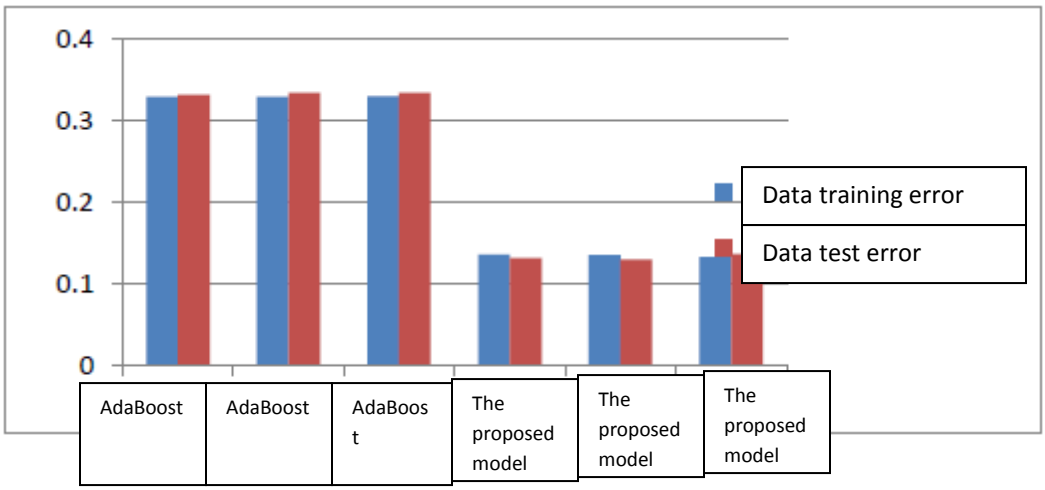

Chart5-1. The error of the decision tree classifier after being boosted by the proposed model and the AdaBoost- Fisher-Iris dataset

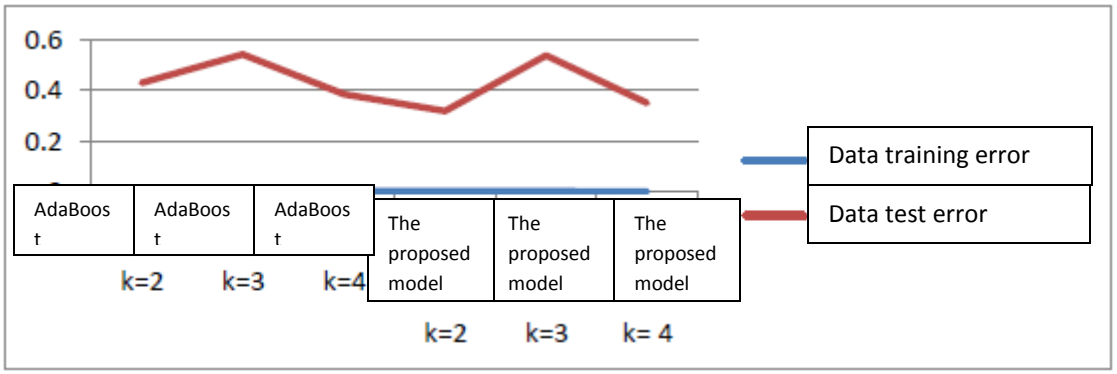

Chart5-2. The error of the decision tree classifier after being boosted by the proposed model and AdaBoost- drink dataset

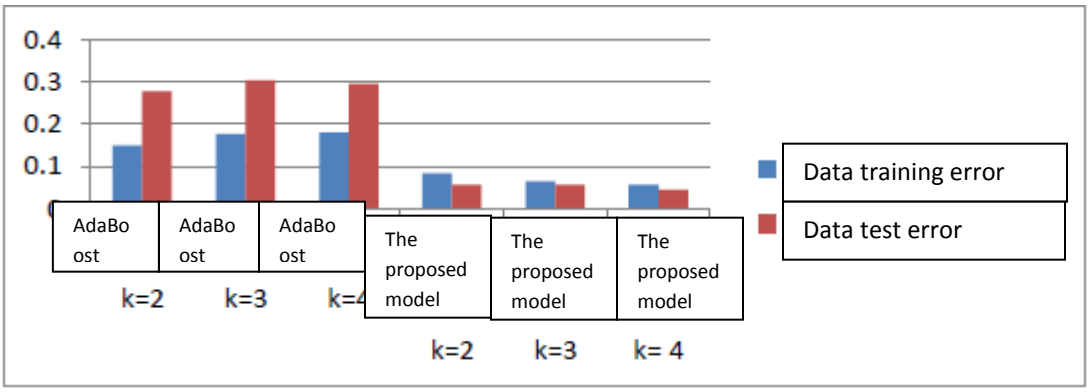

Figure 5-3. The error of the decision tree classifier after being boosted by the proposed model and AdaBoost- BreastTissue dataset

6.Discusion

With Examining of the outputs have generated by the proposed algorithm and Adabost shows the effect of selecting the threshold value on the amount of error generated in the training and testing phases of the data. We can seen the error rate of the proposed model during data training and after it, in both datasets has been significantly reduced due to the logical behavior with the data of unbalanced categories as well as Examples are low replication in the database. The outputs generated on the above datasets show a reduction in error when the threshold value is dynamically selected, because in this case all the samples that are in the upper and lower bounds of the threshold value are correct. Are classified and have the least interference with the hypothesis; Therefore, if we see Perth data in the database, the threshold value will be formed somehow at the point closest to these samples and will categorize them as much as possible. However, the error rate of each weak trainer is also reduced and the result is a correct classification of data that is rarely repeated. These samples can be attributed to the class that has the least distance with them near the threshold values, we will see the elimination of the basic problem of adabost, ie over fitting, because after strengthening the training samples and Experimentally, the proposed model will not attempt to retain the data and will perform the classification based on the maximum similarity and the minimum distance with other categorized samples. It is natural that in this case we will still see the error rate even slightly, because the classification of data that has not been repeated at all, is always accompanied by a percentage of error.

7.Result

Table 5-2. The Result From the implementation of proposed algorithms and Adaboost on Fisher Iris Dataset

Error rate in data testing	Error rate in data training	Amont of k	Name of algorithm
0,2769	0.1495	2	Adaboost
0,3039	0.1759	3	Adaboost
0,2952	0.1793	4	Adaboost
0,0558	0,0836	2	suggested model
0,0562	0,0641	3	suggested model
0,0445	0,0557	4	suggested model

Table 5-3. The Result From the implementation of proposed algorithms and Adaboost on Drinks Dataset

Error rate in data testing	Error rate in data training	Amont of k	Name of algorithm
0.4302	0.0010	2	Adaboost
0.5443	0.0011	3	Adaboost
0.3865	0.0015	4	Adaboost
0.3194	0.00062	2	suggested model
0.39	0.00083	3	suggested model

0.3511	0.000174	4	suggested model

Table 5-4. The Result From the implementation of proposed algorithms and Adaboost on BreastTissue Dataset

Error rate in data testing	Error rate in data training	Amont of k	Name of algorithm
0,2769	0,1495	2	Adaboost
0,3039	0,1759	3	Adaboost
0,2952	0,1793	4	Adaboost
0,0558	0,0836	2	suggested model
0,0562	0,0641	3	suggested model
0,0445	0,0557	4	suggested model

8. Conclusion

The outputs obtained from the proposed algorithm and AdaBoost indicate the effect of choosing the threshold level on the error occurred in data training phase and data test phases. As seen in the above charts, in the proposed model, the error level has significantly decreased in the data training phase and then, in all 6 datasets. This result can be due to the behavior toward the imbalanced class data and the low frequency of the samples in the dataset. The outputs obtained from the above mentioned datasets indicate the reduction of error at the times when the threshold level has been selected in a dynamic manner. In this situation, all samples belonging to the upper and lower bounds of the threshold level are classified properly and they have the least interference with the raised hypothesis. Therefore, if there are outlier data, the threshold bound will be found in the shortest distance from these samples, and they will be classified as far as possible. Nevertheless, the error of each of the weak trainers is also decreased and it results in proper classification of the data that have been rarely repeated. These samples can be assigned to the class with the shortest distance from them. So, this process leads to elimination of the fundamental problem of AdaBoost i.e. overfitting; because after boosting the training and testing samples, the proposed model will not try to preserve the data and classification will be done based on the maximum similarity and the minimum distance from the other classified samples.

Obviously, there is still a small amount of error in this situation; because classification of the data that have not been repeated so far is always accompanied by an insignificant level of error.

8. Resources

1. Mobin Mohsenzadeh, Artificial Intelligence, Fuzzy Logic, Neural Networks and Machine Learning Methods, July2013

2.Mingyi Wang, Zuozhou Chen, Sylvie Cloutier. A Hybrid Bayesian Network Learning Method for Constructing gene networks,tsugua 31, 2010.

3.Bartlett, Peter L., and Mikhail Traskin. "Adaboost is consistent." Journal of Machine Learning Research , 2347-2368, 2007.

4.Freund, Yoav, Robert Schapire, and Naoki Abe. "A short introduction to boosting." Journal-Japanese Society For Artificial Intelligence 14.771-780 (1999): 1612.

5. Jiri Matas and, Sˇ ochman,Centre for Machine Perception Czech Technical University, 2014.

6. O. Okobiah, S. P. Mohanty, and E. Kougianos, "Ordinary Kriging Metamodel-Assisted Ant Colony Algorithm for Fast Analog Design Optimization Archived March 4, 2016, at the Wayback Machine. «, in Proceedings of the 13th IEEE International Symposium on Quality Electronic Design (ISQED), pp. 458-463, 2012.

7. Sapna Katiyar, Ibraheem, Abdul Quaiyum Ansari,Ant Colony Optimization: A Tutorial Review,Conference Paper · August 2015.

8. Mikel Galar,A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches,2011.

9. *Panda B, Herbach JS, Basu S, Bayardo RJ Planet: massively parallel learning of tree ensembles* with *mapreduce , J ACM (JACM,)(2009) . 45(6):983–1006;*