Implementation of AdaBoost.M1 Algorithm

Aawesh Man Shrestha and Zhen Ni
Electrical Engineering and Computer Science Department
South Dakota State University, Brookings, SD, United States
{aawesh.shrestha, zhen.ni}@sdstate.edu

***Abstract*—In this paper, we present the implementation of AdaBoost.M1 algorithm which, theoretically, can be used to significantly reduce the error of any learning algorithm that consistently generates classifiers whose performance is a little better than random guessing. We apply the algorithm in one of the popular data-set, Titanic data-set to classify whether the passenger survives or not based on the different given features. We select the weak learning classifier as a decision stump which is a decision tree of level one. We also show the the training and testing performance for several iteration of the algorithm. The main conclusion of our implementation is that boosting performs significantly better as we iterate through.**

***Index Terms*—Boosting, WeakLearn, machine learning, classification**

I. Introduction

Boosting is a method for improving the performance of any weak learning algorithm. In theory, boosting can be used to significantly reduce the error of any weak learning algorithm that consistently generates classifiers which are slightly better than random guessing [1]. Boosting works by repeatedly running a given weak learning algorithm on various distributions over the training data, and then combining the classifiers produced by the weak learner into a single composite classifier. A 'weak' learner (classifer, predictor, etc) is just one which performs relatively poorly–its accuracy is above chance, but just barely. There is often, but not always, the added implication that it is computationally simple [2]. In this paper, we present the implementation of AdaBoost.M1 [3] algorithm in one of the popular dataset, Titanic dataset. Boosting works by repeatedly running a given weak learning algorithm on various distributions over the training data, and then combining the classifiers produced by the weak learner into a single composite classifier. Significant improvements in classification has been observed at the end of the project. There are two reasons for the improvement in performance that is achieved by boosting.

1) It generates a hypothesis whose error on the training set is small by combining many hypotheses whose error may be large (but better than random guessing). the boosing algorithm tends to generate weights that focuses on the harder samples, thus challenging the weak learning algorithm to perform well on these harder parts of the sample space.
2) It reduces the variance in data–random variability of the combined hypothesis, which were trained on different samples, taken out of the same training set.

II. Algorithm description

In this section, we describe how the boosting algorithm works. There are two version of AdaBoost algorithm but here we focus only on the AdaBoost.M1 algorithm [3] dealing with the binary classification.

Algorithm AdaBoost.M1
Input: sequence of m examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations
Initialize $D_1(i) = 1/m$ for all i.
Do for $t = 1, 2, \ldots, T$

1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \to Y$.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$. If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update distribution D_t: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$.

Fig. 1. The algorithm AdaBoost.M1

This algorithm takes an input a training set of m training examples $S = < (x_1, y_1), ..., (x_m, y_m) >$ where x_i is an instance drawn from space X and represented in some manner usually, a vector of feature set and $y_i \in Y$ is the class label associated with x_i. In this paper, we assume that the set of possible labels Y is of finite cardinality k. In addition, the boosting algorithm has access to another unspecified learning algorithm (Decision stump in this case), called a weak learning algorithm, which is denoted generically as WeakLearn. Decision stump is a machine learning model consisting of a one-level decision tree. That is, it is a decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its leaves). A decision stump makes a prediction based on the value of just a single input feature. Sometimes they are also called 1-rules [4]. The boosting algorithm calls WeakLearn repeatedly in a series of rounds. On round t , the booster provides WeakLearn with a distribution D_t over the training set S. In response, WeakLearn computes a classifier or hypothesis $h_1 : X \to Y$ which should misclassify a non trivial fraction of the training examples, relative to D_t. That is, weak learner's goal is to find a hypothesis h_t which minimizes the training error e_t. Note that this error is measured with respect to the distribution D_t that was provided to the weak learner. This process continues for T rounds, and at last the booster combines the weak hypotheses $h1, ..., h_T$ into a single final hypothesis h_{fin}. In fact, the distribution

D_t is the weight specified to each training samples. Initially, those weights are initialized as uniform distrubution $1/m$. As we iterate through, the weights are updated according to the training error. Weights are increased for those training examples which were classified incorrectly and unchanged for those which were classified correctly. To compute the new distribution D_{t+1} from D_t and the last weak hypothesis h_t, we multiply the weight of example i by some manner $\beta_t \in [0, 1)$ if h_t classifies x_i corectly, and otherwise the weight is left unchanged. The weights are then renormalized by dividing the normalization constant Z_t. By doing this, AdaBoost algorithm focuses on the training examples with more weight which seem to be hardest for the weakLearn.The number β_t is computed as shown in the figure as a function of e_t. The final hypothesis h_{fin} is a weighted vote (i.e. a weighted linear threshold) of the weak hypotheses. That is, for a given instance x, h_{fin}outputs the label y that maximizes the sum of the weights of the weak hypotheses predicting that label. The weight of hypothesis h_t is defined to be $(1/\beta_t)$ so that greater weight is given to hypotheses with lower error.

III. Dataset description

In this project, we use one of the popular dataset, Titanic data-set. The Titanic dataset describes the survival status of individual passengers on the Titanic [5]. The Titanic dataset does not contain information from the crew, but it does contain actual ages of half of the passengers. The principal source for data about Titanic passengers is the Encyclopedia Titanic. The datasets used here were begun by a variety of researchers. One of the original sources is Eaton & Haas (1994) Titanic: Triumph and Tragedy, Patrick Stephens Ltd, which includes a passenger list created by many researchers and edited by Michael A. Findlay. The variables on the dataset are PassengerId, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked, Survived. The data are described in more details below.

Features	**Description**
Pclass	Passenger Class (1, 2, 3)
Name	Name(String)
Sex	Gender(male, female)
Age	Age(Integer)
SibSp	No. of Siblings/Spouses Aboard(Integer)
Ticket	Ticket Number
Fare	Passenger Fare (British pound)
Cabin	Cabin
Embarked	Port of Embarkation (C, Q, S)
Survived	Survival (0 = No; 1 = Yes)

Table 1. Feature description

IV. Data preparation and implementation

Given the ten features, we use only 8 features namely, "Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked","Survived". These are the best features that are required to determine the survival of the passenger of Titanic. [6]. Once we decide the important features, we then format the data to convert non-numeric to numeric mapping. We use 0 for male and 1 for a female in Sex feature and 0, 1 and 2 for Embarked feature: S, C, and Q. We then fill all the null values present in the data with 0 values followed by dropping the duplicate rows. Then we divide training and testing data into 80% and 20% respectively. Thanks to all the pre-built python library such as pandas and sci-kit learn through which we can achieve these tasks in a very few lines of code efficiently.

The AdaBoost.M1 algorithm has been implemented successfully in python. We run the algorithm for 1000 times in an interval of 10. On each iteration of the main AdaBoost.M1 algorithm, we run the weakLearn for the number of current iteration value of the algorithm. On each iteration, the weights are modified based on the feedback, the new hypothesis and hypothesis weights are saved in a separate list. Prediction error of training data as well as testing has been made and saved in a separate list for each iteration. We then plot the results using the matplotlib library of python. Finally, the model has been tested to new test data that has never been supplied to the algorithm before and predicts the correct label as expected.

V. Results

A very impressive result has been seen in both the training and the testing dataset. The result of our implementation is show in figure 2. A graph has been plotted to visualize the training and testing error with respect to the number of iterations. As we see on the graph, training and testing error decreased from 0.25 and 0.21 to 0.2 and 0.14 respectively as visualized which is the expected output of the AdaBoost.M1 algorithm.

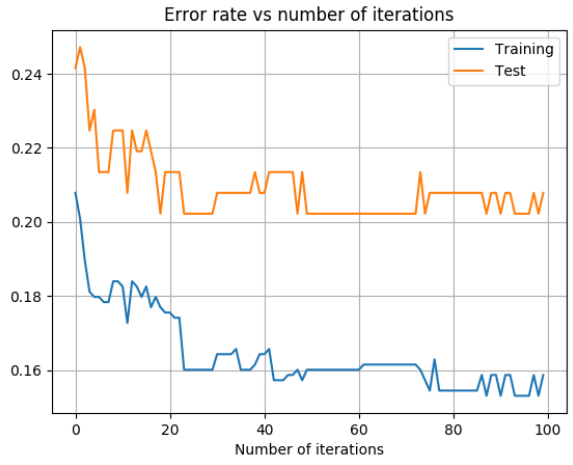

Fig. 2. Error vs number of iteration

VI. Conclusion

In this paper, we presented the implementation of one of the popular boosting algorithm AdaBoost.M1 in detail. We used one of the real and frequently used data-set, Titanic data-set

to predict whether the passenger of the Titanic Ship tragedy survived or not. In fact, we solved a binary classification problem in this paper. We learned that AdaBoost.M1 algorithm can be used to combine weighted weak learning algorithm like decision stump to significantly improve the prediction accuracy while the result of similar work in conjunction to Neural Network as a weak learn which already has the weight for the input nodes and hidden nodes remain beyond the scope of our project. This work does not guarantee that the Adaboost.M1 algorithm work for other algorithms as well unless we perform some real simulations on different data-set. We can set these task as a future work.

REFERENCES

[1] Adaboost-wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/AdaBoost

[2] https://stats.stackexchange.com/questions/82049/what-is-meant-by-weak-learner. [Online]. Available: https://stats.stackexchange.com/questions/82049/what-is-meant-by-weak-learner

[3] R. E. S. Yoav Freund, "Experiments with a new boosting algorithm," 1996.

[4] Titanic dataset description. [Online]. Available: https://en.wikipedia.org/wiki/Decision_stump#cite_note-IL92-1

[5] Titanic dataset description. [Online]. Available: http://campus.lakeforest.edu/frank/FILES/MLFfiles/Bio150/Titanic/TitanicMETA.pdf

[6] Finding important factors to survive titanic. [Online]. Available: https://www.kaggle.com/jatturat/finding-important-factors-to-survive-titanic