Received: September 28, 2017

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Impact of Gradient Ascent and Boosting Algorithm in Classification

Syed Muzamil Basha[1] Dharmendra Singh Rajput[2*] Vishnu Vandhan[1]

[1]*School of Computer science and Engineering, Vellore Institute of Technology University, India*
[2]*School of Information Technology and Engineering, Vellore Institute of Technology University, India*
* Corresponding author's Email: dharmendrasingh@vit.ac.in

Abstract: Boosting is the method used to improve the accuracy of any learning algorithm, which often suffers from over fitting problem, because of inappropriate coefficient associated to the data points. The objective of our research is to train the data, such that the weighing error of linear classifier goes to zero and classify the sentiments accurately. In this paper, Gradient ascent approach is used to minimize the weighing error of sentiment classifier by predicting the proper coefficients to the data points in trained dataset. When compared to previous studies on designing a strong classifier, our research is novel in the following areas: Estimation of Maximum Likelihood for logistic regression using Gradient ascent and making use of weights of metric in understanding the behavior of AdaBoost algorithm in classifying the sentiments. In our finding, the first decision stump has training error of 30.44%. After thousand iterations, we observed a smooth transition, where the classification error tends to go down to 8.22% and actually stays at same value. Finally, concluding that Boosting algorithm outperforms Random Forests with lesser Mean squared Test Errors.

Keywords: Gradient ascent, AdaBoost, Machine learning, Classifier.

1. Introduction

The idea of boosting starts from a question, that Kearns and Valiant [1] posed in 1998 "can the weak classifiers combined together to get a stronger classify"? Rob Schapire [2] a year later came up with an algorithm called boosting that really showed a greater impact on machine learning area. Today, it has become a default approach for deploying many computer vision tasks at industry. Even though the week classifier has low bias, it is not strong enough to classify the data points accurately because of inefficient coefficient associated to it. A Linear classifier, takes X as an input in the form of sentences from reviews and feed it through its model, making a prediction y. In which, positive review y cap is plus one, or negative review in which case y cap is minus one. In this process, It associates each and every word with weight (or) coefficient to determine how positively/negatively influential are these words. Initially, we are training a linear classifier by learning the coefficients. Consider, For a Linear classifier with two non-zero coefficients have the shape of the decision boundary as line [3], with three non-zero coefficients the shape is plane and with many non-zero coefficients the preferred shape is hyper plane. From the training data, we have selected some feature extractor that gives H(X) in defining the quality metric which is the likelihood function, used gradient ascent to optimize it to get weights (w). In [4] the author had defined the quality metric for logistic regression. We can interpret these likelihood function to get exact fit training data, and maximize it. We discussed about the gradient ascent algorithm that does it with really simple updates, and we derived a gradient ascent algorithm [5] from the scratch. Gradient ascent is the technique that wins a lot of those machine learning competitions. So there is a company called Kaggle that does a bunch of those competitions. In which, Boosting wins more than half of those competitions. Boosting is an amazing technique used in machine learning, and it is applied to any classifier as it boosts its quality by combining multiple classifiers. This approach has amazing

impact in the machine learning world. Boosting is that we can start to fork out weak classifiers, so these are the things like a simple logistic regression, a shallow decision tree or maybe even a decision stump. And so if we look at the learning curves associated with such models, let's take a logistic regression model. In which, we start from a very simple weak classifier, Which is not good fit to the data, results in high training error. But the training error can be decreased by considering more features. However, the true error decreases, and then increases as you start to over fit the data. And our goal here is to find kind of this optimal trade-off between bias and variance. Now we know the weak classifiers are great because they have low bias but we need something that's a little stronger in order to get good quality, low test error.

To choose a weak classifier having lower error, can be done in two approaches. One approach is to add more features. So for example, Instead of using polynomial features in logistic regression, we can add second order polynomials, third order polynomials, fourth order polynomials, and so on, to avoid over fitting. So let's suppose that we have a particular set of weights and we have multiple decision stumps, so classifiers that have provided their vote. The other is to improve the weights of the data points using Gradient ascent method. In our research, we would like to adopt the second approach. When comparing to previous studies on this approach, our research is novel in the following areas: To Model a Linear Binary/Multi class classifier which takes the sentences from the product review dataset and predict the sentiment Yi, Used a Document Term Matrix to encode a categorical input, Estimation of Maximum Likelihood for logistic regression using Gradient ascent, Discussed the effect of step-size on Likelihood function. We find that the coefficient assigned to data points are associate to weighted error. And then we update the weights to reduce classification error. We normalize the weights by dividing each value by this total sum of the weights. We opted a smooth transition where the classification error tends to go down to zero and actually stays at zero. Over fitting behavior of Ada-Boost algorithm can be predicted by the cost functions derived in [6].Our work in this paper is organized as follows: In Related work we aims to understand the research carried out in prediction using AdaBoost algorithm, In Methodology, we elaborate the way in which the gradient ascent helps to AdaBoost algorithm. In result, we discussed our findings along with conclusion and future scope.

2. Related work

The AdaBoost has proved to be a very efficient ensemble learning algorithm, which iteratively generates a set of diverse weak learners and combines their outputs using the weighted majority voting rule as the final decision. In [7] the author proposed a robust multi-class AdaBoost algorithm (Rob_MulAda) whose key ingredients consist in a noise-detection based multi-class loss function and a new weight updating scheme. Adaboost algorithm can also be used for feature extraction. In [8] Nassim et al. 2017 proposed a new speech feature extraction method called Mel Modified Group Delay coefficients (MMGDCs), In which adaboost algorithm is used to build strategy to make the fusion between MMGDCs and MFCCs is better, under noisy environments. whereas, In [9] the author has modelled capabilities of the AdaBoost-DT, for the application of interest are evaluated using statistical parameters and showed that the presented AdaBoost-DT models provides high performance in prediction. In [10] the author made a study to develop an automated system to minimize the manual inference and diagnose breast cancer with good precision. Compared the performance of a Neural Network classifier with Adaboost for tested images and showed high level of overall accuracy (98.68%) and sensitivity (80.15%). Whereas, In [11] the author has proposed Dynamic financial distress prediction (DFDP) approaches, In which Adaboost support vector machine (SVM) ensemble based on time weighting, other is Adaboost SVM internally integrated with time weighting (ADASVM-TW), based on error-time-based sample weight updating function in the Ada-boost iteration. A boosting-based method of learning a feed-forward artificial neural network (ANN) with a single layer of hidden neurons and a single output neuron is presented in [12], Where, an algorithm called Boostron is described that learns a single-layer perceptron using Ada-Boost and decision stumps. The proposed method uses series representation to approximate non-linearity of activation functions to learn the coefficients of nonlinear terms with Ada-Boost.

To address class imbalance in data, In [13] the author proposed a new weight adjustment factor applied to a weighted support vector machine (SVM) as a weak learner of the AdaBoost algorithm useful for the class-imbalance problem by addressing well-known issues: overlap, small disjunct, and data shift. Boosting allows achieving a highly accurate, robust and fast classification by combining many relatively simple rules. In [14] the author make use of Adaboost algorithm to classify

Thomson Scattering images of the TJ-II fusion device. Adaboost is utilized in training process to establish the color mapping model,In [15] the author proposed a mind evolutionary computation (MEC)-back propagation (BP)-adaboost algorithm (Adaboost) neural network-based color correction algorithm for color image collecting equipment.

To solve the classification problem of the status box in Stock trend prediction a special features construction approach is presented in [16]. Which is, a new ensemble method integrated with the AdaBoost algorithm, probabilistic support vector machine (PSVM), and genetic algorithm (GA) is constructed to perform the status boxes classification.

In [17] the author addressed, Accurate and timely traffic flow forecasting application, which is critical for the successful deployment of intelligent transportation systems. Developed a training samples replication strategy to train a series of stacked auto-encoders and an adaptive boosting scheme is proposed to ensemble the trained stacked auto-encoders to improve the accuracy of traffic flow forecasting. In [18] the author aims at the problem of traffic accidents, an Adaboost and Contour Circle (ACC) algorithm was developed based on a traditional Adaboost method and the proposed contour circle (CC) for recognizing whether eyes are in open state or closed state. In which, Adaboost method is used to detect human faces and eye regions, the pixels of the pupil region are removed by the given grid method, the least squares method is utilized to fit the CC of the upper eyelid, the center and radius of the CC are extracted as the feature vector, and the eyes state is recognized according to the defined threshold. In [19] the author used Support Vector Machines (SVM), Genetic Algorithms and Particle Swarm Optimization, and sliding window approach for parameter selection. Applied Discriminant analysis (ADA) for evaluation of financial instances and dynamic formation of bankruptcy classes. Applied correlation-based feature subset evaluator different possible feature selection application are researched. Demonstrated a possibility to develop and apply an intelligent classifier based on original discriminant analysis (ODA) method evaluation and shows that it might perform bankruptcy identification better than original model. In [20] the author aims to solve service discovery problem, Bayesian classifier brings in to web service discovery framework, which can improve service querying speed. Used EM algorithm to estimate prior probability and likelihood functions. Concludes that the EM algorithm and Bayesian classifier supported method outperforms other methods in time complexity.

In [21] the author proposed a system to integrate two different classifiers namely SVM and Gaussian process classifier (GPC) and two different descriptors like multi local quinary (MLQ) patterns and multi local phase quantization (LPQ) with ternary coding for texture classification, In which for each descriptor they have trained a different classifier, the set of scores of each classifier by normalizing mean to zero and standard deviation to one, then all the score sets are combined by the sum rule. Building a high performance ensemble that works on different datasets without parameters tuning. The author objective in [22] is to set up an optimize solution for the intricate algorithmic complexity imposed on learning the structure of Bayesian classifiers using sophisticated algorithms.

In [23] author presented an ear based verification system using a new entropy function (NEF) to display different characteristics of a Linear classifier. Considered features like Effective Gaussian Information source value (EGISV) and Effective Exponential Information source value (EEISV) functions which are derived using the entropy function. Entropy features are classified using refined scores (RS) method in which scores are generated using the Euclidean distance. In [24] the author presented a model that can provide blockage likelihood level and verification using unseen data, based on previous decision tree models. The model was developed using the geographical grouping of sewers and the application of ensemble techniques.

In [25] the author presented a possible enhancement of entropy-based classifiers, addressed problem caused by the class imbalance in the original dataset and proposed a method to test on synthetic data to analyse the robustness with different class proportions in controlled environment. In [26] author derives a linear classifier, the Gaussian Linear Discriminant (GLD), that directly minimizes the Bayes error for binary classification and proposed a local neighbourhood search (LNS) algorithm to obtain a more robust classifier if the data is known to have a non-normal distribution, Evaluated the proposed classifiers on two artificial and ten real-world datasets, and then compared the proposed algorithm with LDA approaches and other linear classifiers. The GLD outperforms the original LDA procedure in terms of the classification accuracy. In [27] the author, proposed an semi-supervised approach that extracts and classifies opinion words from one domain called source domain and predicts opinion words of another

domain called target domain, combined modified maximum entropy and bipartite graph clustering. Made a comparison of opinion classification on reviews of four different product domains. And achieved classification accuracy of 88.4%. In [28] the author have used Fuzzy Logic to classify the sentiments form Tweets, Where as in [29] the author made a comparative study on predictive models. The research work carried out by the author in [24] had achieved 11.37 Mean Squared Test error by weighing the feature using Z-Value. Whereas, in our research work aims to reduce the Mean Squared Test error about 3% using proposed boosting algorithm.

3. Methodology

A linear classifier model is going to build a hyperplane, that separate the positives from the negative samples. And the hyperplane is associated with the score function. Which is weighted combination of the coefficients w_0 multiplied by the features that we have as shown in the Eq. (2). In our model, Let us consider the Input as collection of sentences from reviews as $X=\{X[1], X[1],...,X[d]\}$ where d is number of reviews and Predicted output as Y containing possible values as $\{-1,+1\}$. $X_{[j]}$ is j^{th} input of X, $h_j(X)$ is j^{th} feature belongs to X.

$$\hat{Y}_i = sigmoid(Score(X_i)) \tag{1}$$

$$\begin{aligned} Score(X_i) &= w_0 h_o(X_i) + ... + w_D h_D(X_i) \\ &= \sum_{j=0}^{D} w_j h_j(X_i) = w^T h(X_i) \end{aligned} \tag{2}$$

$$P(Y=+1|X_i,w) = sigmoid(Score(X_i)) \tag{3}$$

$$sigmoid(Score(X_i)) = \frac{1}{1+e^{-w^T h(X)}} \tag{4}$$

$$\begin{aligned} &\hat{P}(y=C:\{1,2,...,n\}|X_{i=1}^{n}) \\ &= \underset{1}{\hat{p}}\left(y=\{+1,-1\}|\, X_{i=1}^{n}, w_{i=1}^{n}\right) \end{aligned} \tag{5}$$

We should maximize the quality metric i.e, Likelihood over all possible weights that assigned to all dimensions in the dataset. For multi class classification the Eq. (5).

Logistic regression is a specific case of that, where we use logistic function *sigmoid* to squeeze minus infinity to plus infinity into the interval $\{0,1\}$ so we can predict probabilities for every class. Estimation of Maximum likelihood for logistic regression:

Algorithm 1: To find Max of $\ell(w_n)$

1: Start
2: **While** not converged
3: $w^{(t+1)} \leftarrow w^{(t)} + \eta \frac{d\ell}{dw}\Big|_{w^{(t)}}$
4: End **while**
5: Stop

$$\ell(w) = \prod_{i=1}^{N} P(Y_i|X_i,w)\,, \tag{6}$$

where N=Number of data points. The above function can give larger value, with possibly good value of w. Finding the best linear classifier with gradient ascent $\ell(w_n)$ with n variables.

$$\ell(w_n) = \underset{w_n}{Max}\left(\prod_{i=1}^{N} P(Y_i|X_i,w)\right) \tag{7}$$

From the Algorithm 1 the Likelihood function reach the optimum, when partial derivative of weights is equals to zero. while, the algorithm 1 is repeated with step size η untill partial derivative of an attribute with respect to individual weights assigned is less than θ, where θ is assumed as tolerance value. Derivative of first term with respect to the first parameter having weight (w_0). The partial relative of first term with respect to the second parameter having weight (w_1) all the way to the derivative last term the partial derivative with respect to the last parameter having weight (w_D) as shown in Eq. (8) for $d+1$ dimension vector. Now, the derivative of the likelihood is going to be equal to the sum over the data points. Therfore, consider that each data point has a contribution to the derivative, In first case the derivative is considered as big, in next case we can consider the derivative as smaller. But, We are going to sum over the data points of the difference between termed as indicator function, that a data point is plus 1, so indicator of whether this data point is positive as in Eq. (10). Gradient ascent algorithm as a kind of hill climbing algorithm. As per the algorithm, with one parameter w, you can imagine starting at some point, let's say $w^{(t)}$ with t iteration, and then moving little bit uphill to the next parameter, $w^{(t+1)}$.

$$\nabla \ell(w) = \begin{bmatrix} \frac{d\ell}{dw_0} \\ \frac{d\ell}{dw_1} \\ \cdot \\ \frac{d\ell}{dw_d} \end{bmatrix} \quad (8)$$

$$\frac{\partial \ell(w)}{\partial w_j} = \sum_{i=1}^{N} h_j(X_i)(\Uparrow [y_i + 1] - P(Y = +1 | X_i, w)) \quad (9)$$

where $\Uparrow [y_i + 1]$ is indicator function defined as in Equation 10.

$$\Uparrow [y_i + 1] = \begin{cases} 1 if\ y_i\ is +1 \\ 0 if\ y_i\ is -1 \end{cases} \quad (10)$$

Algorithm 2: Gradient ascent

1: Start
2: initialize $w^{(1)}=0$ at $t=1$.
3: **while** $\left\|\nabla \ell(w^{(t)})\right\| \geq \theta$
4: **for** $j=0,1,...,d$
5: $\frac{\partial \ell(w)}{\partial w_j}$
6: $w^{(t+1)} \longleftarrow w^{(t)} + \eta \frac{d\ell}{dw}\Big|_{w^{(t)}}$
7: $t \leftarrow t+1$
8: $\eta \leftarrow \frac{\eta}{t}$
9: End **for**
10: End **while**
11: Stop

where t is number of iterations. In which, we have started from some point say, w_0 and we're just going to follow the gradient here until we get to the optimal value and stopped, when the value of the gradient is sufficiently small with respect to tolerance parameter. After every iteration made, we travers by feature or by coefficient to compute the partial derivative, which is back to coefficient j with new stepsize. Boosting takes this weak classifier and makes it as a stronger classifier. So let's suppose that we have a particular set of weights and we have multiple decision stumps, so classifiers that have provided their vote as shown in Eq. (11).

Algorithm 3: Boosting (Greedy learning ensembles from data)

Step 1: start
Step 2: Consider the Training data
Step 3: Learn classifier $f_1(X)$
Step 4: Prediction $\hat{Y} = sign(f_t(X))$
Step 5: Learn classifier and weights assigned each of the feature $\hat{W}, f_1(X)$
Step 5.1: same weight for all points: $\alpha_i = \frac{1}{N}$
Step 5.2: **for** each $t = \{1,...,T\}$
Step 5.2.1: Learn $f_t(X)$ with data weight α_i
Step 5.2.2: Compute coefficient $\hat{W}_t$
Step 5.2.3: Recomputed weight α_i
Step 5.2.4: Normalize weight α_i
Step 5.2.5: End **for**
Step6: Perform the prediction

$$\hat{Y} = sign\left(\sum_{i=1}^{T} \hat{w}_t f_t(X)\right)$$

Step 7: Stop

$$F(X_i) = sign(W_1 f_1(X_i) + ... + W_n f_n(X_i)) \quad (11)$$

Where X is a data point, f is classifier and W is weight of each classifier assigned based on the important of the feature on which the classifier do prediction. The prediction may be either positive (+1) (or) negative (-1) represented by *y cap* evaluated as in Eq. (12). So think about a learning problem where we take some data, we learn a classifier which gives us some output, $f(x)$, and we use it to predict on some data.

$$\hat{Y} = sign\left(\sum_{i=1}^{T} \hat{w}_t f_t(X)\right) \quad (12)$$

We say that *Y cap* of $f(x)$. Now, this idea of learning from weighted data is not just about decision stumps. It's the result that most machine learning algorithms accept weighted data. In Eq. (13), we describe the way in which the coefficients are computed. The exact weights can be obtained using gradient ascent method for logistic regression.

$$W_j^{(t+1)} \leftarrow W_j^{(t)} + \eta \sum_{i=1}^{N} \alpha_i(X_i)(1[yi = +1] - P(y = +1 \mid x_i, W^{(t)})) \quad (13)$$

where α is weight of each data point. So that's the data, is the weights start with 1 over N but they get different over time. Then we compute the coefficient what *t* for this new classify $f_{(t)}$ that we learned. And then we should recompute the weights α_i. Finally, we say that the prediction y hat is the sign of the weighted combination of *f1, f2, f3, f4* weighted by these coefficients that we learn from later. Measuring error and weighted data is very similar to measuring error in regular data. So, we want to measure the weighted total of the correct examples and the weighted total of the mistakes. So we take our learned classifier f_t, and we feed that review. So keep adding the weight of the mistakes versus the weight of the correct classifications. And use that to measure the error. Weighted classification error can be computed as in Eq. (14).

$$Weighted\_error = \frac{Total\,weight\,of\,mistakes}{Total\,weight\,of\,all\,data\,points} \tag{14}$$

We are computing the coefficient $\hat{W}_t$ of classifier $f_t(X)$.

$$W_E = \frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)}$$

$$\hat{W}_t = \frac{1}{2}\ln(W_E) \tag{15}$$

Based on number of levels in sentiments (2, 3, and 5) the coefficient value of classifier changes as listed in Table 2.

4. Results

The algorithm to estimated to Maximum the Likelihood for logistic regression using Gradient ascent is implemented and compared with five different values ranges from 10^{-4} (Too Big) to 10^{-6} (Too Small). The observation made from the results obtained are, the difference between these values are really small. So, if a classifier is just random, it's not doing anything meaningful. It true that α_i gets an update depending on whether on f_t gets the data point right because this is correct or whether f_t makes a mistake. We are going to increase the weight of data points where we made mistakes and we are going to decrease the weight of data points as in Eqs. (16) and (17) using Ada Boost algorithm.

$$\alpha_i = \{\alpha_i e^{\hat{W}_t}, if\ f_t(X_i) = Y_i \tag{16}$$

Table 1. Recompute weight α_i.

$f_t(X_i)=y_i$	$\hat{W}_t$	$\alpha_i e^{-\hat{w}_t}$	**Implication**
Correct	2.3	$e^{-2.3} = 0.1$	Decrease importance of (x_i,y_i)
Correct	0	$e^{-0} = 1$	Keep important the same
Mistake	2.3	$e^{2.3} = 9.98$	Increasing important of (x_i,y_i)
Mistake	0	$e^{0} = 1$	Keep important the same

$$\alpha_i = \{\alpha_i e^{\hat{W}_t}, if\ f_t(X_i) \neq Y_i \tag{17}$$

Finally, we normalized the weights of data points start to 1 over *n*, when we had uniform weights. Which is they should be normalizing weights of the data points throughout the iterations as in Eq. (18).

$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j} \tag{18}$$

In classifying sentiments the first decision stump has training error of 20.94%. So, not good at all. After thirty iterations, we observed a smooth transition, where the classification error tends to go down to 8.67% and actually stays at same value. And that is a key insight of the boosting theorem. So famous AdaBoost Theorem which underlines all the choices made in the algorithm and really has had a lot of impact on machine learning. From Table 2, one can interpret that the coefficient W_t *cap* of an classifier becomes zero, when the *weighted_error(*f_t*)* reaches to 0.5.

Finally, our aim is to generate a Gradient Boosting model implemented in R using gbm package. The result obtained can be interpreted as follows: In Fig. 2 the red line indicates the least Test error from the training data considered in our experiment. The same dataset are considered with same parameters and came to a conclusion that proposed Boosting algorithm outperforms Random Forests with lesser Mean squared Test Errors starting with 30.44 Test Error obtained using 100 trees. The experiment is repeated by constructing trees up to 1000 and obtained 8.22 Test Error at last iteration.

Table 2. Computing W_t *cap*

f_t	0.5ln(W_E)	0.33ln(W_E)	0.2ln(W_E)
0.01	2.297	1.516	0.919
0.05	1.472	0.971	0.588
0.1	1.098	0.725	0.439
0.15	0.867	0.572	0.346
0.2	0.693	0.457	0.277
0.25	0.549	0.362	0.219
0.3	0.423	0.279	0.169
0.35	0.309	0.204	0.123
0.4	0.202	0.133	0.081
0.45	0.100	0.066	0.040
0.5	0	0	0
0.55	-0.100	-0.066	-0.040
0.6	-0.202	-0.133	-0.081
0.65	-0.309	-0.204	-0.123
0.7	-0.423	-0.279	-0.169
0.75	-0.549	-0.362	-0.219
0.8	-0.693	-0.457	-0.277
0.85	-0.867	-0.572	-0.346
0.9	-1.098	-0.725	-0.439
0.95	-1.472	-0.971	-0.588
0.99	-2.297	-1.516	-0.919

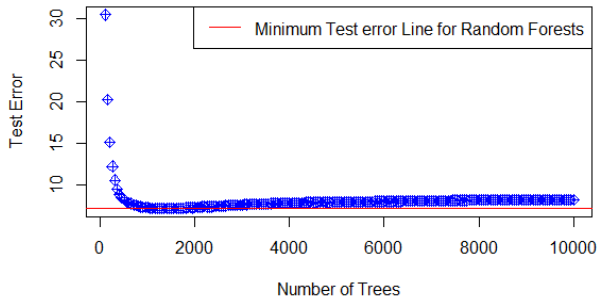

Figure 1. Performance of Boosting Algorithm.

The proposed Boosting algorithm is better in assigning best weights for each of the feature in the dataset using Gradient Ascent method. This weights helps in achieving lesser classification error in classifying the polarities of the sentiments. In our approach, we started with equal weights for all features extracted from the reviews, learnt a classify f_t. We find its coefficient depending on how good it is in terms of weighted error. And then updated the weights to weigh mistakes, mostly the weights are assigned exactly as in Table 1. Finally, normalize the weights by dividing each value by this total sum of the weights. In construct, the traditional Random Forest technique error in weighting the features i.e,

Table 3. Mean squared test error

Number of Trees	Test Error
100	30.44
500	7.96
1000	7.23
2000	7.27
4000	7.80
6000	7.98
8000	8.13
10000	8.22

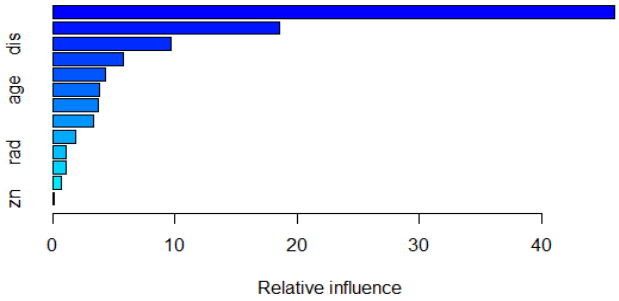

Figure.2 Influence of attribute using random forest

relative influence of each attribute is calculated based on Z-Score (Statistical Parameter) as plotted in Fig. 2 leads in higher Test error as discussed by the author in [24]. Where x-axis representing the relative source of features and y-axis representing the features of the dataset.

5. Conclusion

The model discussed in our research on Linear Binary/Multi class classifier can take a sentences as an input Xi from the product review dataset, encoded a categorical type and gives score to it and predict the sentiment Yi. The exact weights obtained from the Gradient Ascent method helps the proposed Boosting algorithm in building the stronger classifier by combining different weak classifier having their own polarities, Which can outperform Random Forest algorithm with lesser Mean squared Test Errors (8.22) by repeating our experiment. In our Future work, we attempts to implement our proposed Boosting algorithm on Distributed environment (Hadoop) Map-Reduce jobs and compare its performance with traditional Machine Learning algorithms. In which, we select a subpart of that to just pick the magic parameters use cross-validation on the same.

Acknowledgments

We would like to thank our VIT University for providing us the all the research facilities requirement for publishing Scopes Indexed journals. At the same time, we would like to thank all the

reviewers, who help us to improve the quality of our paper.

References

[1] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant, “A general lower bound on the number of examples needed for learning”, *Information and Computation* , Vol.82, No.3, pp.247-261, 1989.

[2] J. Steven, R.P. Anderson, and R.E. Schapire, “Maximum entropy modeling of species geographic distributions”, *Ecological modelling*, Vol.190, No.3, pp.231-259, 2006.

[3] L. Chulhee and D.A. Landgrebe, “Decision boundary feature extraction for nonparametric classification”,*IEEE transactions on systems, man, and cybernetics*, Vol.23, No.2, pp.433-444, 1993.

[4] G. David and M. Klein, “Analysis of matched data using logistic regression”, In: *Proc. of International Conf. on Logistic regression*, New York, pp.389-428. 2010.

[5] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S.J. Glaser, “Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms”, *Journal of Magnetic Resonance*, Vol.172, No.2, pp.296-305, 2005.

[6] L. Mason, J. Baxter, L. Bartlett, and M.R. Frean, “Boosting algorithms as gradient descent”, In: *Proc. of International Conf. on neural information processing systems*, New York, pp.512-518, 2000.

[7] S. Bo, S. Chen, J. Wang, and H. Chen, “A robust multi-class AdaBoost algorithm for mislabeled noisy data”, *Knowledge-Based Systems*, Vol.102, No.1, pp.87-102, 2016.

[8] N. Asbai and A. Amrouche, “Boosting scores fusion approach using Front-End Diversity and adaboost Algorithm, for speaker verification”, *Computers & Electrical Engineering*, Vol.9, No.2, pp.1-12, 2017.

[9] S. Hamidreza and M. Arabloo, "Modeling of CO 2 solubility in MEA, DEA, TEA, and MDEA aqueous solutions using AdaBoost-Decision Tree and Artificial Neural Network”, *International Journal of Greenhouse Gas Control*, Vol.58, No.1, pp.256-265, 2017.

[10] S. Ghada, A. Khadour, and Q. Kanafani, “ANN and Adaboost application for automatic detection of microcalcifications in breast cancer”, *The Egyptian Journal of Radiology and Nuclear Medicine*, Vol.47, No.4, pp.1803-1814, 2016.

[11] S. Jie, H. Fujita, P. Chen, and H. Li, “Dynamic financial distress prediction with concept drift based on time weighting combined with Adaboost support vector machine ensemble” , Knowledge-Based Systems, Vol.120, No.1, pp.4-14, 2017.

[12] B. Mirza, M. Awais, and M. El-Alfy, “AdaBoost-based artificial neural network learning”, *Neurocomputing*, Vol.248, No.1, pp.120-126, 2017.

[13] L. Wonji, C.H. Jun, and J.S. Lee, “Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification”, *Information Sciences*, Vol.381, No.1, pp.92-103, 2017.

[14] F. Gonzalo, S. Dormido-Canto, J. Vega, I. Martínez, L. Alfaro, and F. Martínez, “Adaboost classification of TJ-II Thomson Scattering images”, *Fusion Engineering and Design*, Vol.5, No.42, pp.1-5, 2017.

[15] Z. Jing, Y. Yang, and J. Zhang, “A MEC-BP-Adaboost neural network-based color correction algorithm for color image acquisition equipments”, *Optik-International Journal for Light and Electron Optics*, Vol.127, No.2, pp.776-780, 2016.

[16] Z. Xiao-dan, A. Li, and R. Pan, “Stock trend prediction based on a new status box method and AdaBoost probabilistic support vector machine”, *Applied Soft Computing*, Vol.49, No.1, pp.385-398, 2016.

[17] Z. Teng, G. Han, X. Xu, Z. Lin, C. Han, Y. Huang, and J. Qin, “δ-agree AdaBoost stacked autoencoder for short-term traffic flow forecasting”, *Neurocomputing*, Vol.247, No.1, pp.31-38, 2017.

[18] W. Mei, L. Guo, and W. Chen, “Blink detection using Adaboost and contour circle for fatigue recognition”, *Computers & Electrical Engineering*, Vol.58, No.1, pp.502-512, 2017.

[19] M. Paci, L. Nanni, and S. Severi, “An ensemble of classifiers based on different texture descriptors for texture classification”, *Journal of King Saud University - Science*, Vol.25, No.7 pp.235-244, 2013.

[20] Y. Peng, “Service Discovery Framework Supported by EM Algorithm and Bayesian Classifier”, *Physics Procedia*, Vol.33, No.7, pp.206-211, 2012.

[21] M. Paci, L. Nanni, and S. Severi, “An ensemble of classifiers based on different texture descriptors for texture classification”, *Journal of King Saud University - Science*, Vol.25, No.7 pp.235-244, 2013.

[22] H. Bouhamed, A. Masmoudi, and A. Rebai, "Bayesian Classifier Structure-learning Using Several General Algorithms", *Procedia Computer Science*, Vol.46, No.3, pp.476-482, 2015.
[23] M. Bansal and M.Hanmandlu, "A new entropy function for feature extraction with the refined scores as a classifier for the unconstrained ear verification", *Journal of Electrical Systems and Information Technology*, Vol.12, No.10, pp.74-81, 2016.
[24] J. Bailey, E. Harris, E. Keedwell, S. Djordjevic, and Z. Kapelan, "Developing Decision Tree Models to Create a Predictive Blockage Likelihood Model for Real-World Wastewater Networks", *Procedia Engineering*, Vol.154, No.11, pp.1209-1216, 2016.
[25] A. Kirshners, S. Parshutin, and H. Gorskis, "Entropy-Based Classifier Enhancement to Handle Imbalanced Class Problem", *Procedia Computer Science*, Vol.104, No.1, pp.586-591, 2017.
[26] KS. Gyamfi, J. Brusey, A. Hunt, and E. Gaura, "Linear classifier design under heteroscedasticity in Linear Discriminant Analysis", *Expert Systems with Applications*, Vol.79, No.3, pp.44-52, 2017.
[27] S.J. Deshmukh and A.K. Tripathy, "Entropy based classifier for cross-domain opinion mining", *Applied Computing and Informatics*, Vol.53**,** No.2, pp.211-220, 2017.
[28] S.M. Basha, Y. Zhenning, D.S. Rajput, N.Ch.S.N. Iyengar, and D.R. Caytiles, "Weighted Fuzzy Rule Based Sentiment Prediction Analysis on Tweets", *International Journal of Grid and Distributed Computing*, Vol.10, No.6, pp.41-54, 2017.
[29] S.M. Basha, Y. Zhenning, D.S. Rajput, R.D. Caytiles, and N. Ch. S.N Iyengar, "Comparative Study on Performance Analysis of Time Series Predictive Models", *International Journal of Grid and Distributed Computing*, Vol.10, No.8, pp.37-48, 2017.