Aggressive Sampling for Multi-class to Binary Reduction with Applications to Text Classification

FINAL REPORT

Apoorva Manda
Dept. of Computer Science
Clemson University
Clemson, US
manda@g.clemson.edu

Aradhana Velidimalla
Dept. of Computer Science
Clemson University
Clemson, US
avelidi@g.clemson.edu

I. OVERVIEW

The paper "Aggressive Sampling for Multi-class to Binary Reduction with Applications to Text Classification" [3] addresses the multiclass-classification problem for large number of classes. In this paper an aggressive double sampling strategy is adopted on inter-dependent paired sub samples of text repository DMOZ composed of 100,000 classes, with an aim to reduce the curse of long-tailed distribution problem. This is attained by acquiring the training set as a combination of oversampling of small classes and sub-sampling of large classes. The evaluation parameters considered for the proposed methodology were: utilization of memory, accuracy, training time, prediction time, predictive performance.

The reduction of doubly sampled multi- class to binary was carried out by implementing the following sub procedures:

1) Reduction of Multi-class to binary over pairs of examples
2) Aggressive Double Sampling
3) Prediction with Candidate Selection.

II. ABSTRACT

A brief description on how the experiment was carried out, how the dataset was procured and the implementation of the proposed technique in paper [3] are discussed in this report.

This report also describes the carrying out of the ideas suggested for enhancement of the existing paper.

We tried Principal Component Analysis as a feature selection method, Ensemble Boosting technique (AdaBoost) and Regression techniques (Linear and Logistic Regression) in order to improve the accuracy of the existing model.

III. EXPERIMENTAL SETUP

.

A. DATASET

In this project the WIKIPEDIA dataset [1] is being evaluated using the methodology proposed in the paper [3]. The dataset used is pre-processed and is in .tf format (term frequency file). libSVM format is followed by each datafile. In data file, every line represents a sparse document vector in the format given below:

<label feature: value, feature: value …>

Where,

label represents the category to which the document vector belongs and is an integer value.

The pair feature: value represents a non-zero feature with index = feature and value= Value. Feature is an integer corresponding to a term and value is a double that corresponds to the term frequencies (weight) as shown in Figure1.

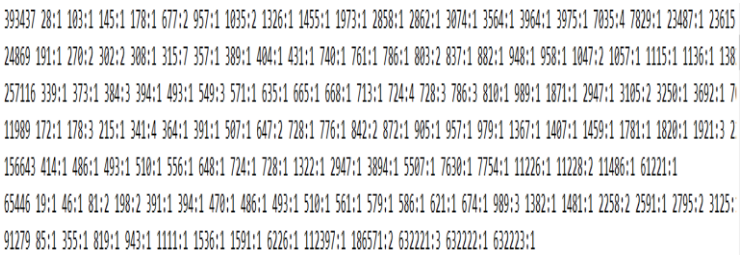


Figure1: Dataset in .tf format

The WIKIPEDIA dataset [1] consists of remapped versions of the data in training and testing .tf files for computation.

.

B. *EVALUATION MEASURES*

The evaluation parameters we used over the test set are accuracy, F1 measure, Logarithmic Loss, Mean Squared Error and Mean Absolute Error.
The time taken for loading, pre-processing, binary reduction, learning of the algorithm, prediction along with the memory usages are also provided.

IV. IMPLEMENTATION

This project is being implemented in mainly 5 phases:

1) Loading of the Data
2) Pre-Processing of the data
3) Reduction
4) Algorithm learning
5) Prediction

The following are passed as argument vector parameters while running the file: Sample number, Sampling rate, number of candidates. Where,
Sample number denotes the average number of examples sampled per class.

Sampling rate is the rate with which the classes are chosen for sampling. A lower bound value for sampling rate is set as (1 / Size) of class. In case if user enters a value lower than this then by default one class will be chosen.

Number of candidates is the Number of candidate classes for prediction.

In the loading phase, the test and train files are loaded and the time for loading the dataset is calculated.

In the pre-processing phase, TfidTransformer is used and the mean vector for every class and a global mean for the entire collection is computed by making use of the csr_matrix and coo_matrix. The centroids of the entire dataset and the class are also being calculated.

In the reduction phase, binary reduction is being carried out and double sampling strategy is implemented.

In the algorithmic learning phase, the binary features and the binary_labels are fit using the Linear Support Vector Classification algorithm. The best fit of the classification is returned by updating the weights using the binary labels and features so obtained.

The prediction is carried out by implementing the nearest neighbour classifier and brute force algorithm for fitting the class centroids in the feature space.

V. ANALYSIS & RESULTS

A. *CHALLENGES IN DATASETS PROCUREMENT*

Since the double sampling strategy [3] mainly addresses the multiclass-classification problem for large number of classes, finding such open source datasets in field of Image/Object recognition, Human Activity recognition etc with more than 10,000 classes was a difficult task.

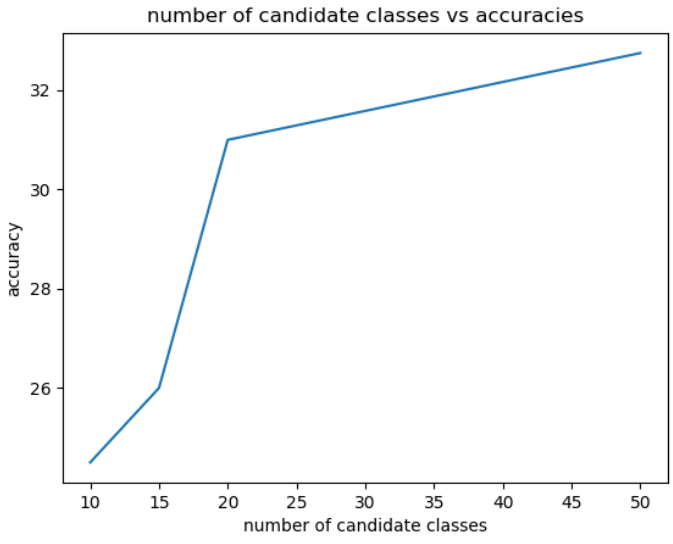
When searched for the same in Kaggle, UCI repositories, the maximum number of classes for the above-mentioned datasets were found to be 51. The challenge posed was that the datasets with such a smaller number of classes would give an extremely low accuracy as the feature selection from 51 classes would produce hardly any effective feature after aggressive double sampling. Also, such datasets with a smaller number of classes were not available in .tf format.

The datasets with large number of classes could only be found in the field of Text processing and could be procured from large data source repositories like Directory Mozilla Firefox, Wikipedia, Yahoo etc and we have chosen the WIKIPEDIA Dataset [1].

The WIKIPEDIA dataset [1] chosen is the one containing testing, training datasets with more than 45k classes in each.

B. *ANALYSIS OF VARIOUS PERFORMACE EVALUATORS*

We have analysed the results (accuracies achieved) obtained by plotting a graph between accuracy and number of candidate classes considered as shown in the Graph 1.



Graph1: Line graph between Number of Candidates and accuracies.

The accuracy values obtained on different parametric values are can also be inferred from the Table 1 below.

No. of samples	Sampling rate	No. of candidates	Accuracy
1	0.001	10	24.5%
3	0.01	15	26%
5	0.1	20	31 %
12	0.1	50	32.75%

Table 1: Variations of the accuracy measures

The other parameters like time taken for loading, pre-processing, binary reduction, learning of the algorithm, accuracy, F1 score, Logarithmic loss, mean absolute error can be inferred from the figures (Figure 2, Figure 3) below.

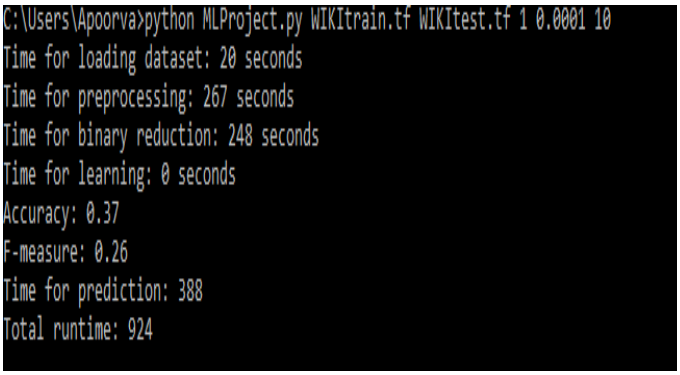


Figure 2: Performance Evaluators with No of samples=1, Sampling rate=0.0001 and No. of Candidates=10

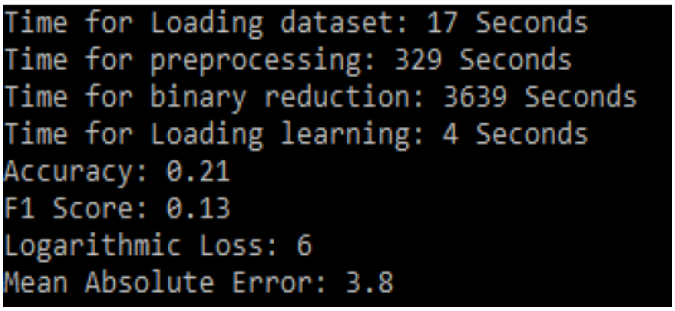


Figure 3: Performance Evaluators with No of samples=3, Sampling rate=0.01 and No. of Candidates=15

C. ANALYSIS OF VARIOUS CLASSIFIERS

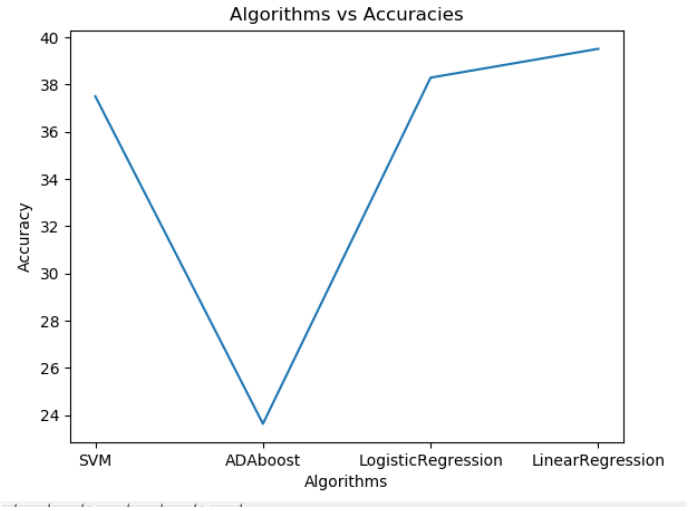
In the proposed paper [3], Support Vector Machine was used as a classifier. The accuracy obtained from the same was around 37.4%.

Our main idea for the enhancing the paper was to improve the results (accuracy and F-measure) by implementing various classifiers. The existing model was trained with three other classifiers namely- Logistic Regression, Linear regression and Ada-Boost.

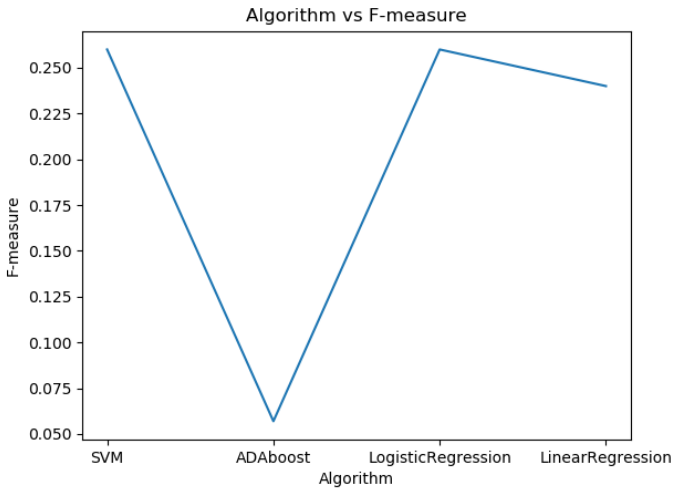
From the Table 2 shown below it can be deduced that the accuracy obtained is maximum in case of Linear Regression (39.51%) which was 2% higher than the proposed methodology in paper (with SVM).
The least accuracy was observed in case of Ada Boost 23.64%.

SNo	CLASSIFIERS	ACCURACY	F -MEASURE
1	ADABOOST	23%	0.057
2	SUPPORT VECTOR MACHINE	37%	0.26
3	LOGISTIC REGRESSION	38%	0.26
4	LINEAR REGRESSION	39%	0.24

Table 2: Comparison of Accuracy and F-score various classifiers



Graph 2: Line Graph for Accuracies of various Algorithms



Graph 3: Line Graph for F-measure of various Algorithms

D. WHY ADABOOST FAILED?

AdaBoost is a one of ensemble learning [2] methods to weight various inputs. It makes use of the exponential loss function which is a convex loss function and grows exponentially negatively. AdaBoost minimizes the exponential loss by iteratively applying the loss function over the transformed new subsampled training set.

But the problem observed was the penalties for misclassification grew exponentially with the predictive output. This ended up having a strong influence on the final model and resulted in a non-convex optimizing problem.
This could probably be the reason why the accuracy observed with AdaBoost was the least when compared to all the other classifiers. This is also supported by the value of the F1 score so obtained as shown in Figure 4

```
C:\Users\Apoorva>python MLProject.py WIKItrain.tf WIKItest.tf 1 0.0001 10
Time for loading dataset: 21 seconds
Time for preprocessing: 273 seconds
Time for binary reduction: 249 seconds
Time for learning: 0 seconds
Accuracy: 0.23
F-measure: 0.057
Time for prediction: 379
Total runtime: 924
```

Figure 4: Results of AdaBoost Classifier

E. PRINCIPLE COMPONENT ANALYSIS

Reducing the dimensionality of the dataset consisting of correlated variables while retaining the variation in dataset is the main idea of Principle component Analysis.

Finding the orthogonal projection of the highest variance in the dataset is the focus of Principle component analysis. This is in order to find the hidden linear correlations between the variables.
This means that it can find directions (vectors) that represents the data.

The probable reason for the poor performance of PCA when applied to WIKIPEDIA dataset [1] as shown in Figure 5 and Figure 6, could be because of data not being linearly correlated.

In the approach described in the paper cosine distances are being evaluated to find the centroid distances, which in turn are used to subsample. This could be an analogy to the probable reasoning of PCA failure as described above.

We carried out PCA with SVM and recorded an alarming response with respect to the evaluation parameters-accuracy and F-measure. The accuracy observed was 0.7% (least) and the F-measure was 0.0791.

```
C:\Users\Apoorva>python MLProject.py WIKItrain.tf WIKItest.tf 1 0.0001 10
Time for loading dataset: 25 seconds
Time for preprocessing: 456 seconds
Time for binary reduction: 291 seconds
Time for learning: 0 seconds
Accuracy: 0.0774
F-measure: 0.0791
Time for prediction: 501
Total runtime: 1277
```

Figure 5: Results of PCA with SVM

We also carried out PCA with Linear Regression and recorded the following results.
The accuracy was 1.38% and the F-measure was 0.1437.

```
C:\Users\Apoorva>python MLProject.py WIKItrain.tf WIKItest.tf 1 0.0001 10
Time for loading dataset: 22 seconds
Time for preprocessing: 290 seconds
Time for binary reduction: 261 seconds
Time for learning: 1 seconds
Accuracy: 0.1385
F-measure: 0.1437
Time for prediction: 398
Total runtime: 975
```

Figure 6: Results of PCA with Linear Regression

F. REGRESSION ANALYSIS

Linear regression is a regression technique which tries to find a relationship between input (x) and output(y) using a linear mathematical function.

When applied to WIKIPEDIA dataset [1], the accuracy obtained was higher in comparison to the other classifiers. The possible reason for this could be: WIKIPEDIA dataset [1] is high dimensional (i.e. it has many features) and high dimensional problems are likely to be linearly separable.

When regularization parameters (like variable coefficients in linear function, correlation and covariance) are rightly adjusted, they try to control the classifier's complexity. They regularize the feature weights and keep them small hence aid to avoid over fitting. This results in better predictive performance. This is demonstrated by the results obtained as shown in Figure 7.

```
C:\Users\Apoorva>python MLProject.py WIKItrain.tf WIKItest.tf 1 0.0001 10
Time for loading dataset: 19 seconds
Time for preprocessing: 274 seconds
Time for binary reduction: 264 seconds
Time for learning: 0 seconds
Accuracy: 0.39
F-measure: 0.247
Time for prediction: 389
Total runtime: 949
```

Figure 7: Results of Linear Regression Classifier

The accuracy observed was 39% (highest) and the F-measure was 0.2470.

We also recorded results with Logistic Regression which are shown in Figure 8. Logistic regression is a go-to technique for binary classification problems. It predicts the probability of belonging to the default class, which can be separated into 0 or 1 classification.
Logistic regression gives a better performance in terms of accuracy when applied on the newly paired sub-samples which are in form of binary labels and features.

```
C:\Users\Apoorva>python MLProject.py WIKItrain.tf WIKItest.tf 1 0.0001 10
Time for loading dataset: 20 seconds
Time for preprocessing: 273 seconds
Time for binary reduction: 247 seconds
Time for learning: 0 seconds
Accuracy: 0.38
F-measure: 0.26
Time for prediction: 379
Total runtime: 921
```

Figure 8: Results of Logistic Regression

G. WHY VOTING CLASSIFIER FAILED?

Voting classifier is described as a machine learning model that trains on ensemble of several models. It predicts a class(output) based on their highest probability of chosen class as an output.

Existing model calculates feature weights for better classification. Calculation of feature weights (feature importance) is important as different features have different applicability to a learning problem. Some are less important while some are very important. Faster and accurate learning could be possible if algorithms consider feature importance for classification.

The existing model used 'coef_ 'attribute (to return an array of output coefficients as weights) to calculate weights.
In accordance with the central idea of the paper [3], we tried implementing algorithms which had coef_ as an attribute in order to calculate feature weights. The algorithms that best suited this approach were Linear regression, Logistic regression and Ada Boost.

We failed while trying to implement Voting Classifier (as shown in Figure 9). The reason for failure could be that coef_ attribute is not applicable in case Voting Classifier.

```
    weights = eclf.coef_
AttributeError: 'VotingClassifier' object has no attribute 'coef_'
```

Figure 9: Results of Voting Classifier

VI. CONCLUSION

It can be inferred from the Table 1 that as the number of samples, sampling rate, number of candidates are varied the accuracy was altering accordingly.

Among all the classifiers implemented, it can be concluded that for WIKIPEDIA dataset [1], Linear regression showed a marginal increase in the accuracy (40% approximately).

PCA and Ensemble boosting (Ada Boost) have also been implemented. However, the accuracy reduced.

VII. FUTURE SCOPE

Though prediction accuracy of the model improved with Linear Regression technique in comparison with its value in the exiting model, it is still very low.

Further research and experiments can be carried out using deep learning algorithms like Convolution Neural Network and Artificial Neural Network in order to further improve its performance and accuracy.

VIII. REFERENCES

[1] WIKIPEDIA DATASETS https://drive.google.com/open?id=1iAVpPOp9GbAkAM88FGEHTkEe4vYbVY-4

[2] Wenjia Wang, “Some Fundamental Issues in Ensemble Methods”, 2008 InternationalJoint Conference on Neural Networks (IJCNN 2008)

[3] Bikash Joshi et al., on "Aggressive Double Sampling for Multi-Class To Binary Reduction with Applications To Text Classification", Advances in Neural Information Processing Systems, 4159-4168.

[4] Basant Agarwal,and Namita Mittal,“Text Classification Using Machine Learning Methods-A Survey”, In: Babu B. et al. (eds) Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012. Advances in Intelligent Systems and Computing, vol 236. Springer, New Delhi