Comparison of Multinomial Classifier Algorithms for Text classification.

Sri Ram Kannan, Siddharth Nagarajan

Project Report

{sxk138130,Siddharth.Nagarajan}@utdallas.edu

Department of Computer Science, University of Texas at Dallas

Abstract

To implement and compare using experimental results, the ability of classification algorithms in **scaling up** to classify text files of more than 2 class types, using a University files Dataset where each file belongs to one of 4 classes.

Dataset description

This data set contains text content from html pages collected from the Computer Science departments of various universities by the World Wide Knowledge Base (**WebKb**) project of the CMU text learning group. For each class the data set contains pages from the four universities: Cornell, Texas, Washington, Wisconsin and a few other universities. The four classes of the webpage content are:

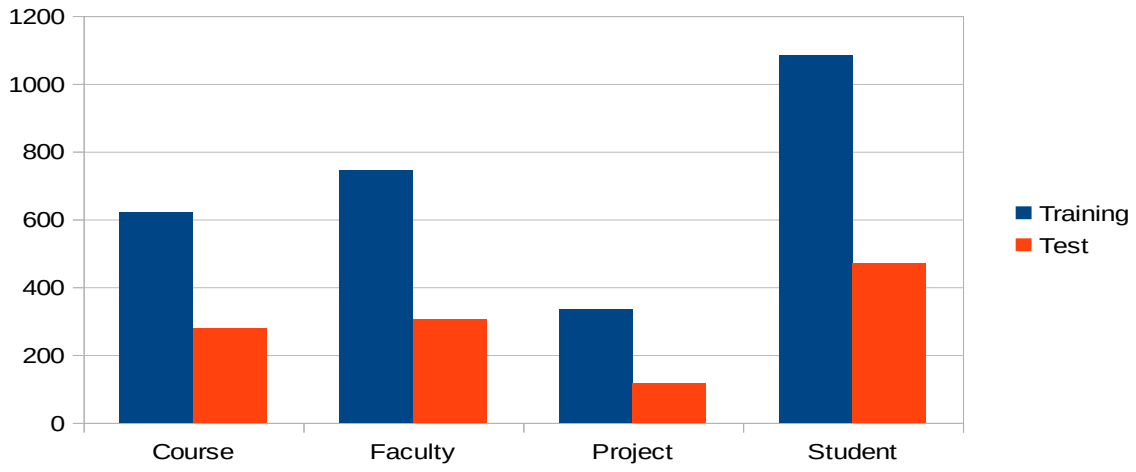
1. Student
2. Faculty
3. Project
4. Course

The files in the dataset are all single labeled across the above mentioned 4 classes and these are obtained by preprocessing the html pages of the university web pages.

Dataset Statistics:

Class-wise Distribution of the dataset.

File Class	Training Set	Test Set
Course	620	279
Faculty	745	305
Project	335	117
Student	1085	471



Classifier Algorithm Implementations:

1. Multiclass Logistic Regression:

In Multiclass logistic regression, a new hypothesis is learnt for each class. During weight vector calculation for a given class, all classes except the particular class are treated as similar.

Example:

Weight vector for class - 'Student': Student = 1,

Course = Faculty = Project = 0

Weight Vector for class - 'Course': Course = 1,

Student = Faculty= Project = 0

On a new input vector from test set to make prediction, four different vector products are computed and the maximum of this is used to predict the the class values.

Formula used: **argmax{ H(x) }, where H(x) = w(x)*f(x)**

Accuracy of Logistic Regression:

67.2636103152 %

'student': 588, 'faculty': 531, 'course': 241, 'project': 36

Representation of evaluation and optimization

Representation	**Logistic Regression/Sigmoid Function**
Evaluation	Square Error and Accuracy
Optimization	Gradient Ascent

Framework for Hypothesis Space:

Size	**Fixed dataset is used to train the hypothesis.**
Randomness	Deterministic
Parameterization	Discrete

Framework for Learning Algorithm:

Search Procedure	Direct Computation
Timing	Eager
Eager Algorithm	Batch

Inference from Logistic Regression:

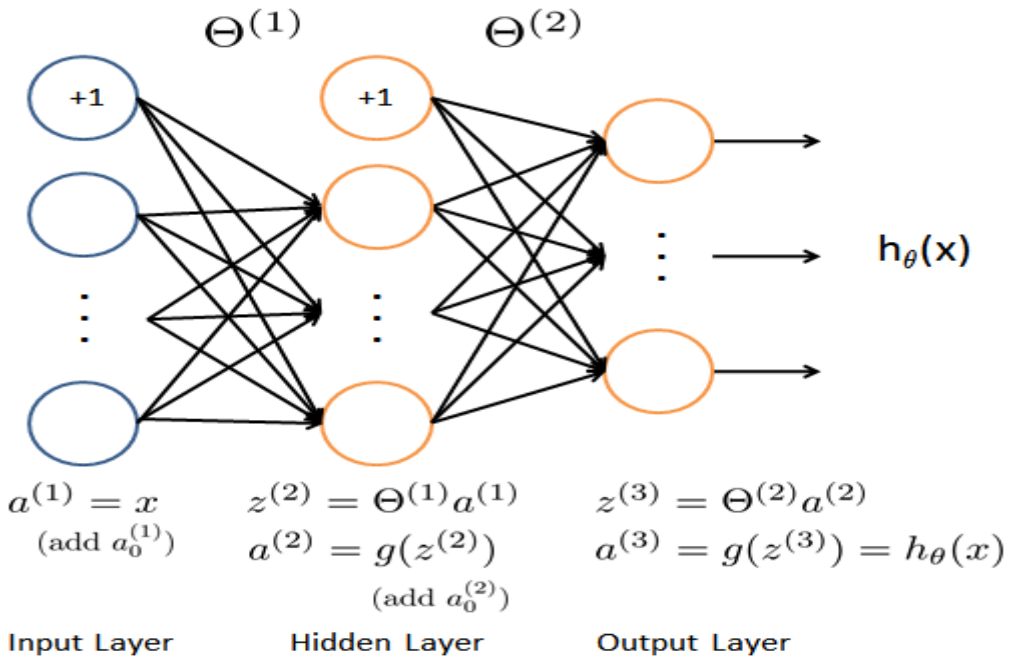
- From the experimental results above, Logistic Regression for multi-class classification yields a reasonable accuracy.
- By changing the values of eta (step-size/learning-rate) and lambda value it is inferred that after a certain number of iterations the accuracy rate remains same.
- When the weight vector reaches global maxima (global maximum point) the accuracy becomes constant irrespective of number of iterations.
- Since the training dataset doesn't contain equal number of records for all the classes, the class with more number of records dominated in test set prediction.

References

- Machine Learning Class Lecture CS229 (Andrew Ng), Multi Class Logistic Regression One - Vs-All:

2. Multilayer Perceptron (Neural Networks):

A Multilayer Perceptron model is implemented to classify the Web Kb dataset.



The neural network has totally three layers:

1. Input Layer: Input layer Units : 6484

2. Hidden Layer: Hidden Layer Units : 350

3. Output Layer: Output Layer Units : 4

Size of Input layer: 2785 * 6484 (with bias)

Size of Hidden layer: 2785 * 351 (with bias)

Size of Output Layer: 2785 * 4

Neural Network model is implemented using Octave for faster calculations of parameter values.

The dataset is inputted in matrix format to the neural network to calculate the cost function during forward propagation and delta values are calculated during backward propagation.

During the runs until convergence, the inbuilt **fmincg** function is used to minimize the cost function obtained from neural network. Then the weight vector after convergence is used to predict the test set records and the accuracy obtained is:

Iteration 10: 39.045553%

Representation	**Neural Network**
Evaluation	Square Error and Accuracy
Optimization	Gradient descent

Framework for Hypothesis Space:

Size	**Fixed dataset is used to train the hypothesis.**
Randomness	Deterministic
Parameterization	Discrete

Framework for Learning Algorithm:

Search Procedure	**Direct Computation**
Timing	Eager
Eager Algorithm	Batch

Inference:

- Since only one hidden layer is used between the input layer and output layer, the accuracy to predict the class remains low. More the hidden layers and its units, the hypothesis will be trained more.
- The number of hidden units in hidden layer is 350 and it is randomly guessed by considering a 400(input units):25(hidden units) proportion.
- The time taken to execute this neural network implementation crossed more than one hour for 10 hard iterations. Hence, neural network exhibits more time in learning and optimizing the hypothesis.

3. Multinomial Naïve Bayes Text Classification

Algorithm

The implementation verifies the accuracy of the simple, generative, Probabilistic classifier algorithm, Naïve Bayes which is extended to classify text files of more than 2 class types. The application of Bayes Theorem and the (naïve) independence assumptions is extended to 4 classes instead of the basic 2-class classification strategy.

Variant

The conditional probability computation for calculating the prior probabilities of the different classes and the posteriors is carried out with respect to the 4 different classes and Probability that the set of features (in a file) belong to one of the 4 classes is determined using the Maximum APosteriori decision rule.

$$\text{Classification}(W_1, W_2, \ldots W_n) =$$

$$\text{argMax}(\text{Prob}(\text{Class} = \text{Class}_i) * \text{Product}\{\text{Prob}(W = W_i / \text{Class} = \text{Class}_i\})$$

This computation of maximum posterior probability is extended to the 4 classes, and the decision is based on the maximum value. Log values of the posteriors are added up to avoid underflow.

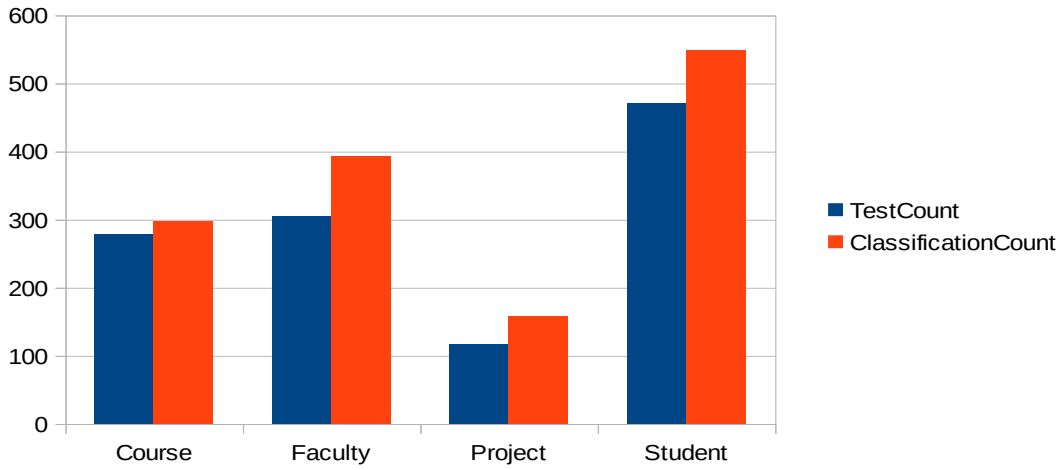
Experimental Results:

Of the 1396 files in the test dataset (with distribution indicated in the graph above) the number of correctly identified files and the error identifications are as follows.

Table for Class Distribution in test set and Classification Result

Class	TestCount	ClassificationCount
Course	279	297
Faculty	305	393
Project	117	158
Student	471	548

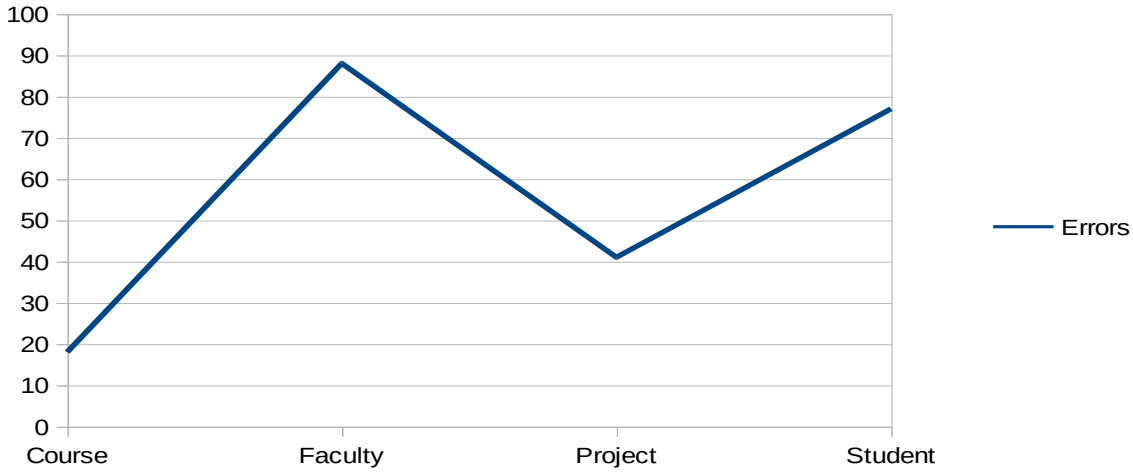
Graph for Count of files of different classes in the test Set and the classification distribution result.



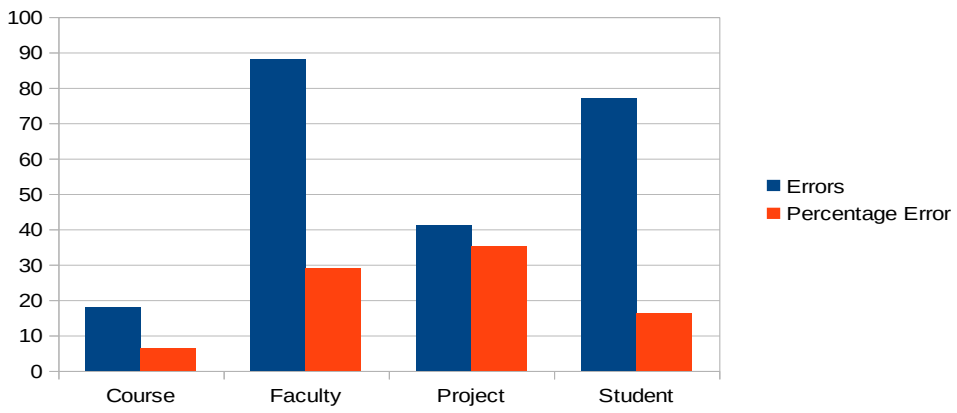
Class-wise Error and Percentage Error Table

Class	Errors	TestCount	Percentage Error
Course	18	279	6.4516129032
Faculty	88	305	28.8524590164
Project	41	117	35.0427350427
Student	77	471	16.3481953291

Class-wise Error Graph



Class-wise Percentage Error Graph



From the error and percentage error graphs, it can be seen that the classification error for each of these classes are low, irrespective of the minimal number of features in the dataset and the coincidence between the vocabularies of text file classes. The overall error percentage is about 16 % which is enough proof to testify the ability of Naïve Bayes Text classification algorithm, even when scaled up to more than the usual Binomial classification. Further the following inferences, both generally and with respect to this dataset can be listed as follows.

Inferences:

- Regardless of the naiveness of the assumptions made (that the presence of one word in the text is independent of the presence/ absence of another) the Multinomial version of the Naïve Bayes performs classically good for classifying text.
- With regard to scaling up the algorithm, the accuracy percentage of **83.9541** % is slightly lower than the accuracy levels that might be expected for classification of 2-class Dataset. However this dip in the accuracy may be attributed to
    a. The nature of the dataset chosen. (The dataset is a University Dataset and the classes are Course/Faculty/Project/Student). The vocabulary across the different classes of this dataset is bound to be more or less similar.
    b. The different classes of the university dataset are still bound to be around Academics Related Vocabulary, more than anything else.

c. The demarcation between the text of the different classes is vague and still the Multinomial Naïve Bayes algorithm has scaled up decently enough to give a classification accuracy of 84 %/

References:

- **Multinomial Naïve Bayes for Text Classification Revisited,** Ashraf.M. Kibriya, Eibe Frank, Bernhard Pfahringer, Geoffrey Holmes. Department of Computer Science, University of Waikato, NZ.
- **http://en.wikipedia.org/wiki/Naive_Bayes_classifier**

4. Multiclass Perceptron Training Algorithm

Algorithm

The implementation verifies the accuracy of the Perceptron Linear Classifier by scaling it up to classify more than 2 classes. The Perceptron Training algorithm is modified by using individual weight vectors for the different classes of text in the Training Data set.

Variant

The binary (2-class) Perceptron operates with a single weight vector. The Class of the given instance is determined by finding out the dot product between this **W vector** and the **Feature count vector** for the file, adding them up with the bias value and making the decision based on a Threshold (Activation) value. This process is repeated until convergence and in each step weights being updated according to the classification error made.

The multinomial Perceptron variant operates with multiple weight vectors, one for each class type. The weights for the vocabulary of the different classes are individually defined (Initial Values are Random) for each class's own feature set.

For the above dataset, 4 different weight vectors (size of each, equal to the corresponding vocabulary size), is used. During classification, the product of each **weight vector $\mathbf{W_C}$** and the Feature count vector of the file is computed. This results in 4 different product values which are added up with the corresponding

bias values. The Class that the file belongs to is determined by finding the maximum of this vector Product.

- The Weight Vector for each class is $\mathbf{W_C}$.
- The **vector Product score** for the file for each class is found out by computing ,
  Wc* f(x), where f(x) is the feature count vector of the file.
- This is added with the bias value $\mathbf{W_{C0}}$ of the corresponding class value.
- During classification in the training phase, the weight update rule of this classifier can be written as ,

$$\mathbf{W_{Ci} = W_{Ci}+f(x_i)}$$

$$\mathbf{W_{Cm} =W_{Cm}-f(x_i).}$$

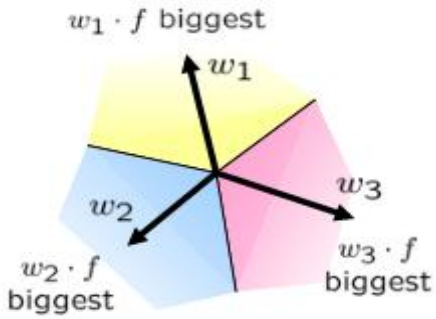
  This is done when there is a wrong classification. The weight values corresponding to the correct class of the file are increased and the weight values that contributed to the misclassification (Weight values of the corresponding wrong class value) are decreased.

- Now, of the 4 different vector products, the maximum value determines the class to which the file belongs.

Class, C = argMax(W_{C0}+Wc* f(x)) for each of the 4 classes.

The visual representation for imagining this decision rule can be illustrated using the following picture.

Representation of Multiclass Perceptron Decision Rule (Reference [2]).



The process of weight update and classification is repeated until convergence and in this implementation, the learning rate is fixed at 0.1. The number of iterations is hard limited to100.

Experimental Results

This implementation of the multi class perceptron, for the universities dataset, yielded a classification accuracy varying from ~79%-84 %, across a number of trial runs.

Of the 1396 files in the test dataset the number of correctly identified files and the erroneous identifications are as follows.

- **Number of Correct classifications:** 1129
- **Number of Wrong classifications:** 267
- **Accuracy in classifying the files :** 80.87392550143267 %

Inferences

- Perceptrons, the classical Linear classifiers are capable of being extended to classify files into multiple class labels.
- With respect to scaling, the Perceptron training algorithm scales reasonably well for this 4-class classification, yielding a maximum accuracy of 84% in some trial runs.
- From the experimental results of this 4-class classification**,** It is evident enough that the simple perceptron, usually regarded as a strictly linear classifier, can be scaled up to perform classification (Pseudo-non linear) for multiclass text.

References

- **Learning and Perception,** Roberto Paredes, Alfons Juan, Enrique Vidal, Universidad Politecnica de Valencia**, February, 2008**
- **Artificial Intelligence Course Lecture, Number 21: Perceptrons,** Pieter Abbeel – UC Berkeley.
- **http://en.wikipedia.org/wiki/Perceptron#Multiclass_perceptron**