REPORT

Naïve Bayes and Logistic Regression

BY:

CHANDNI SHANKAR

CXS152830

Table of Contents

1. Naïve Bayes Classifier 3
   1. Implementation Summary3
   2. Code Walkthrough 4
   3. Output 4
2. Logistic Regression Classifier 5
   1. Implementation Summary5
   2. Code Walkthrough 6
   3. Output 6
3. Conclusion6
4. **NAÏVE BAYES CLASSIFIER**
   1. **Implementation Summary**

* Implemented the Multinomial Naïve Bayes Classifier. For the given training set (split into ham and spam files), read all the words of every file and made a vocabulary containing only one instance of a word. While reading the words from the file, registered the actual label of the file and calculated the ham count and spam count for each word occurring in the file i.e the count of each word in all ham documents and count of each word in spam documents. Similarly, we can get the total number of words in all ham documents and spam documents separately.
* Prior for ham and spam is calculated by finding the ratio of training documents belonging to that category (*Nc/N*)
* For every word in the vocab, one Laplace smoothing is performed to make sure that each word in the vocab is seen atleast once.
* The formula used for one Laplace smoothing is:

*Ham probability of vocab word* = count (term) in Ham documents + 1/ (count of all ham terms + no. of vocab words)

*Spam probability of vocab word*= count (term) in spam documents + 1/ (count of all spam terms + no. of vocab words)

* To predict the accuracy of the classifier, we run the classifier against a set of test files given in the test directory (split into ham and spam files).
* For each file, if the word is in the vocabulary, sum of log likelihood of each word in the document is calculated for spam and ham using the probabilities of the vocab word and the priors for ham or spam and output label is given depending on maximum log likelihood.

Formula used for calculating the log likelihood:

*LnP(Y=ham/X)* = sum (ham probability of each word in the document) + prior of ham

*LnP(Y=spam/X)* = sum (spam probability of each word in the document) + prior of spam

If *LnP(Y=ham/X) > LnP(Y=spam/X)*, then predicted label is ham else it is spam.

* If predicted label is same as actual label for the file, then accuracy of classifier is increased
* To predict the accuracy of the classifier excluding the stop words given, I followed the same procedure except while building the vocab. The vocab will not include the stop words that were earlier present in the documents.

Input: Training set, Test set and stop words file.

Output: Accuracy of the built Naïve Bayes classifier for the test set before excluding the stop words and after excluding the stop words.

* 1. **Code Walkthrough**

File: NaiveBayesLR.py

Main(argv) – Retrieves the 3 files sent as arguments and stores the training files as a list of files for each label the document could be classified (Ham and Spam). It also holds the control of the program to call the classifier and predictor.

trainNaiveBayes(classDocs, path, stopWords, classes) – For each file stored in classDocs, the set of words forming the vocabulary, the Naïve Bayes classifier finds the ham probability, spam probability for each word in each of the documents and the total words in ham documents and spam documents. It also returns the list of instances where each instance contains the count of words in the document and the actual label of the document.

naiveBayes1Laplace(vocab, hamProb, spamProb, totalWords): for each word in the vocabulary, it associates a ham probability and spam probability using the one Laplace smoothing.

predictLabel(file, stopWords, wordProb, prior) – predicts the label by using the above mentioned formula for log likelihood for the file given.

calcAccuracy(testDir, stopWords, wordProb, prior) – calculates the accuracy of the classifier for all files in the test directory.

* 1. **Output**

Accuracy of Naive Bayes without Stop Words: 79.0794979079

Accuracy of Naive Bayes with Stop Words: 85.9832635983

Accuracy improves after removing stop words as those words were low information features. The presence of these stop words is found in each document and hence wouldn’t contribute to classifying the document as ham / spam. Thereby, removing the words from feature list helps in including only major contributing words from the documents and increases accuracy.

1. **LOGISTIC REGRESSION CLASSIFIER**
   1. **Implementation Summary**

* Implemented the Logistic Regression Classifier. For the given training set (split into ham and spam files), read all the words of every file and made a vocabulary containing only one instance of a word. While reading the words from the file, registered the actual label of the file and calculated the count of each word.
* Made training instances of each file with an id number, the dictionary of words and its count and the actual label of the file
* To train the logistic regression classifier, the list of vocab words are used to form the feature vector for each instance as length of feature vector is same as the no. of words in the vocabulary.
* Initialized the all values of weight vector containing 1 more than the number of vocabulary words (w0) elements to zero
* Running **100 iterations**, trained the logistic regression classifier using the batch gradient ascent rule.
* For each iteration, first calculated the error in each instance by using the formula (Error = Y – Y^) where Y = 1 when actual label is Ham, Y=0 when actual label is spam and Y^ = exp(sum(wixi)) / (1+ exp(sum(wixi)) for all is where each w is the weight vector and x is the feature vector of that instance
* Then for that iteration, the weight of each word in the vocabulary is updated by first calculating the sum of all error for that weight by multiplying the xi for each of the instances and updating the w for each of the word in the vocabulary using below formula.

TotalErrori = Error \* Xi for all instances where I is the index of weight corresponding to the vocab word being updated

Wi = Wi + η \* totalErrori – λ\*η\*Wi

* To predict the accuracy of the classifier, a set of test files are used. Each test file is made into a feature vector representing the frequency of each word in the vocabulary found in the document and then P(Y/X) is calculated using the found weight vector from training.

*P(Y=1/X) = exp(sum(wixi)) / (1+ exp(sum(wixi))*

*P(Y=0/X) = 1- P(Y=1/X)*

Input: Training set, Test set and stop words file.

Output: Accuracy of the built Logistic Regression classifier for the test set before excluding the stop words and after excluding the stop words

* 1. **Code Walkthrough**

File: NaiveBayesLR.py

Main(argv) – Retrieves the 3 files sent as arguments and stores the training files as a list of files for each label the document could be classified (Ham and Spam). It also holds the control of the program to call the classifier and predictor.

trainLogisticReg(allInstances, vocab) – As mentioned in the implementation summary, to train the classifier, the instance feature vectors are used to learn the weights for each word in the vocabulary.

predictLabelLR(weights, featureVector) – predicts the label by using the above mentioned formula for maximum conditional a-posterior using the weight vector and the featureVector for the file.

calcAccuracyLR(testDir, weights, vocab, stopWords) – calculates the accuracy of the classifier for all files in the test directory.

* 1. **OUTPUT**
     1. η = 0.001

λ = 0.001

Accuracy of Logistic Regression without Stop Words: 93.0962343096

Accuracy of Logistic Regression with Stop Words: 93.0962343096

* + 1. η = 0.001

λ = 0.5

Accuracy of Logistic Regression without Stop Words: 92.8870292887

Accuracy of Logistic Regression with Stop Words: 92.8870292887

* + 1. η = 0.005

λ = 2

Accuracy of Logistic Regression without Stop Words: 94.5606694561

Accuracy of Logistic Regression with Stop Words: 94.5606694561

The accuracy doesn’t seem to have changed when stop words were removed. Though the stop words are low information features, it should have caused slight changes. However, if the weights associated with these stop words didn’t contribute too much to finding the actual label, the accuracy wouldn’t change.

1. **CONCLUSION**

The accuracy of the Naïve Bayes and Logistic Regression classifier to predict the correct class for the data was calculated and was found to predict accurately for approximately 80-86% of the data and approximately 93% using the two classifiers respectively.