Jeffrey Wagner

CIS 678 - Machine Learning

Project 2 - Naive Bayes for Document Tagging

February 19, 2019

**Introduction**

The goal of this project was to create a naive Bayes classifier with the ability to utilize a tagged training set of documents to learn to tag new documents with a relatively high degree of accuracy (~80%). The training set used during development contained 11,293 documents belonging to 20 different newsgroups. The test set contained a further 7,527 documents belonging to the same newsgroups. The documents did not contain any capitalization or punctuation and relative word position was not considered as a factor. The program was written using Python 3.7 along with its included libraries. Outside libraries including Pandas, NumPy, and SciKit-Learn were utilized to generate the metrics used to evaluate the classifier’s effectiveness. The source code for the classifier is available in appendix A.

**Method**

 The classifier uses a maximum a posteriori decision rule (see figure 1) to tag documents with the newsgroup that they most likely belong to based on the words they contain. The algorithm was primarily implemented using lists and dictionaries that allowed tracking of several different metrics for each newsgroup, such as the probabilities for words in the newsgroup, the probabilities of the newsgroups themselves, and the vocabularies of each newsgroup. A future revised version of the algorithm could potentially make use of a newsgroup class that condenses these metrics into attributes of a single object for each newsgroup.

Figure 1

There were several challenges encountered during implementation. The first was that if a word never appeared in a newsgroup’s training documents and then appeared in the test documents, that newsgroup would be assigned a probability of zero for that document. This was solved by calculating a minimum word probability for each newsgroup (1 / |vocabulary|) that would be used for words that were not present during training. The next challenge was arithmetic underflow created by multiplying many very small probabilities together when determining the newsgroup tag for a new document. This was solved by adding the natural logarithms of each probability rather than multiplying the raw probabilities, taking advantage of the relationship log(M\*N) = log(M) + log(N).

The final and greatest challenge was improving the classifier’s performance to reach the goal of approximately 80% accuracy. Initial versions of the classifier only achieved ~20% accuracy. Common approaches for improving performance of similar classifiers include stemming (removing common English word endings) and the removal of short (1-2 character) words. Both techniques were attempted separately and in combination, and greatly improved performance in all cases (~70-80% accuracy). The solution that was ultimately used and produced the best results was a more experimental technique of removing the most frequently occurring words that were present in the training documents from the test documents. All words from all training documents were counted and ranked by frequency, then the top n% frequently occurring words were removed. This technique relies on the assumption that the more frequently a piece of evidence (word) occurs, the less likely it is to be strongly associated with a particular class (newsgroup). After some experimentation, the algorithm was tuned to search for the most optimal percentage of words to exclude in the range of 0.0% to 1.0% (0.1% increments) and apply that percentage for the final tag assignments. In the case of the newsgroup data, the algorithm found that excluding the most frequently occurring 0.2% of training words led to the highest accuracy.

After the algorithm runs, the precision, recall, F1, misclassification rate, and a confusion matrix were calculated and written to text and CSV files along with the results of the classifier (see appendix B and C).

**Results**

The classifier was ultimately able to achieve precision of 0.83, recall of 0.82, F1 of 0.41, and a misclassification rate of 0.17. These results exceeded the original 80% accuracy goal and greatly outperformed random guessing, which would be expected to achieve figures of ~0.05 recall and precision, F1 of ~0.025, and a misclassification rate of ~0.95.

**Discussion**

While the classifier was able to meet the established performance goals, there is still room for improvement in both accuracy and computational efficiency. Other examples of naive Bayes classifiers have achieved precision and recall figures beyond 0.90 for similar data sets.

The misclassifications that occurred with this classifier were typically amongst very similar newsgroups such as the pairs politics / guns, religion / atheism, and windows / graphics. More research may be needed to find optimal techniques for differentiating between such groups.

Although it is computationally expensive, the technique of searching for an optimal percentage of most frequently occurring words to exclude and then applying it to the test data seems to have merit when it comes to increasing accuracy, as it was able to outperform other common techniques for increasing accuracy in the case of this data set. The refinement of this technique and its application to other data sets represents an opportunity for future research.