Abhishek Modi **1001558833**

Bayesian learning for classifying netnews text articles

Data: The dataset contains 20,000 newsgroup messages drawn from the 20 newsgroups. The dataset contains 1000 documents from each of the 20 newsgroups.

Method:

- Half of the dataset was used as training data and the rest as testing data. Each newsgroup had 1000 files. Half of every news group was considered for training data and the other half for testing.
- Each file had raw text data. For naïve bayes algorithms the input should be numbers hence, it was necessary to convert it into numerical vectors.
- To convert into numerical vector two steps are necessary:
  1) Tokenization: Divide the texts into words or smaller sub-texts, which will enable good generalization of relationship between the texts and the labels.
  2) Vectorization: Define a good numerical measure to characterize these texts.
- These two steps can be performed in two different ways: 1) N-gram Vectors and 2) Sequence vectors.
- If ratio of (total number of samples / number of words per sample) < 1500 , then N-gram vectors give better results than sequence vectors and visa-versa.
- In our case this ratio was 51. (calculated shown in the program). Hence N-gram vectors was used to tokenize the text.
- Vectorization is commonly done using: Tf-idf encoding, one-hot encoding, count encoding.
- The problem with one-hot encoding and count encoding is that common words that occur in similar frequency in all documents are not penalized. Hence for this Project TF-IDF encoding is used.
- Select only the top n features from the vector of tokens by discarding tokens that appear fewer than 2 times.
- Perform Multinomial Naïve Bayes.
- Naïve bayes works under the assumption that features are independent.
- Formula for naïve bayes:

$$P(c_i \mid E) \sim P(c_i) P(E \mid c_i)$$

$$P(E \mid c_i) = P(e_1 \wedge e_2 \wedge \cdots \wedge e_m \mid c_i) = \prod_{j=1}^{m} P(e_j \mid c_i)$$

- Let V be the vocabulary of all the words in the documents in D.
  for each category c_i belongs to C:
  - Let Ti be the concatenation of all documents in Di
  - let n_i be the total number of word occurrences in Ti.
  - for each word wj that belongs to V:
    - n_{ij} be the number of occurrences of wj in Ti
    - $$\mathrm{P}(w_i \mid c_i) = (n_{ij} + 1) / (n_i + |V|)$$
    -

Given a test document X
Let n be the number of word occurrences in X
Return the category:

$$\underset{c_i \in C}{\operatorname{argmax}}\ P(c_i)\prod_{i=1}^{n} P(a_i \mid c_i)$$

- where a_i is the word occurring the ith position in X

- Using log to solve this:
  Example: $p_1 * p_2 = e^{\log(p1)+\log(p2)}$

Result: Naïve bayes although often yields good results , it works under a critical assumption that all the features are independent, which might not be always true.

Below are the Results displayed in program:

```
Implementing MultinomialNaiveBayes (made from scratch)...
Predictions:
 [15.  0.  0. ...  0. 19. 19.]

accuracy_dict: {'t': 7702, 'f': 2296}

 Accuracy : 77.03540708141628 %

Implementing sklearn MultinomialNaiveBayes...
Predictions:
 ['0' '0' '0' ... '0' '19' '0']

accuracy_dict: {'t': 8997, 'f': 1001}

Accuracy : 89.9879975995199 %
```

References:

https://developers.google.com/machine-learning/guides/text-classification/

https://courses.cs.washington.edu/courses/csep573/11wi/lectures/20-textcat.pdf