Machine Learning Project 2 report

Multinomial Naive Bayes classifier.

Naive Byes classifiers are a family of simple probabilistic classifiers that follow the bayes theorm with strong conditional independence assumption among the features.
T he Naive Byes classifier has many applications in
1. Natural language processing
2. Text classification.
3 . Spam filtering , etc.

In this project the Naive bayes algorithm was used to classify 10,000 documents into the following categories

0. alt.atheism – 1000 docs
1. comp.graphics - 1000 docs
2. comp.os.ms-windows.misc - 1000 docs
3. comp.sys.ibm.pc.hardware - 1000 docs
4. comp.sys.mac.hardware - 1000 docs
5. comp.windows.x - 1000 docs
6. misc.forsale - 1000 docs
7. rec.autos - 1000 docs
8. rec.motorcycles - 1000 docs
9. rec.sport.baseball - 1000 docs
10. rec.sport.hockey - 1000 docs
11. sci.crypt - 1000 docs
12 . sci.electronics - 1000 docs
13 . sci.med - 1000 docs
14 . sci.space - 1000 docs
15 . talk.politics.guns - 1000 docs
16 . talk.politics.mideast - 1000 docs
17 . talk.religion.misc - 1000 docs
18 . talk.politics.misc - 1000 docs
19 . soc.religion.christian – 997 docs

Train and Test data

In this project 50 % of data was used for training the classifier and remaining was used for testing. Hence 500 docs form each category were used for testing and 500 docs from each category were used for training.

TestY or y_true

It is a list of 9997 values containing numbers 0-19 each 500 times (one for each document) and (0 – 19) represents categories in the above given order.

Mega Train document

Documents from each category in the training documents were merged into one mega document in linux using

```
cat file_name > all.
```

Hence each training document in each category is a mega document each consisting of 500 files.
This was done because it is easier for preprocessing the text.

Text Preprocessing

This is one of the most import part of any text mining or classification task.
Since each document consists of gibberish, punctuations it is important to preprocess all these before feeding data into the classifier.
The **testPreprocessor and trainPreprocessor** methods accomplish the following tasks,
1 . Reads each file in binary mode
2. converts each file into a singe string
3. Filter out alphabets using regex
4. Remove excess spaces and converts each word into a list item
5. Filter out stop words such as wouldn't ,therefore, nevertheless, however, whatever, become..
6 .Filter out non english words / gibberish like nbeodf, aerinoq, uhrfewo

Hence after preprocessing , each test document (testX) is a list of words.
Each of these lists are converted to a set (to filter out unique words) and appended to a two dimensional list. Hence size of testX is 9997 ~ 10000. Each list in testX has variable number of unique words depending on the document.

The Naive Bayes classifier algorithm:

Simple (“naïve”) classifica1on method based on Bayes rule
Relies on very simple representa1on of document
It follows the Bag of words model.
For a Document d in class C
P(C|d) = [P(d|C) * P(C)]/[P(d)]
Multinomial naive bayes assumptions
Bag of words model assumption: The position of the words does not matter
Conditional independence: Assume that feature probability P(xi|Cj) are independent given class C

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

In the above formula P(Cj) = **0.05** for all categories because each category has 500 documents and there are 10000 training docs in total 500/10000 = 0.05

In the above equation what if the training model sees a words that is not present in the training data set? In that case the probability becomes zero (0).
This makes the probability for that entire category 0 and this could lead to incorrect prediction. Hence we apply a method called **laplace smoothing** to condition 0 probability

Laplace Smoothing

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

The above formula conditions zero probability by adding ‘1’ (smoothing factor) to the numerator.
The |V| is the total number of unique words in the training set.
In the code the **nbClassifier** method calculates the probabilities of each document for all twenty categories, and np.argmax is used to predict the correct category (0-19) which has the maximum probability.

Log Probability

Since probability of finding a words in a training set may be exponentially very less and sometimes it may cause floating pointing under flow and may become 0. This could again lead to incorrect prediction.
Hence a log of probability is taken

$$\log(P(\text{class}_i \mid \textbf{data})) \propto \log(P(\text{class}_i)) + \sum_j \log(P(\text{data}_j \mid \text{class}_i))$$

This also speeds up the probability calculation process.

Accuracy:

The implementation of the above naive bayes model on 10,000 testing data yields 74% accuracy.
Hence the naive bayes algorithm is not so naive!!!!