BITS F464 : Machine Learning
Assignment - 1
Naive Bayes Classifier

Omkar Pitale	Aditya Chopra	Anuradha Pandey
2019A7PS0083H	2019A7PS0178H	2019A7PS0265H

1 Introduction

Naive Bayes belongs to the generative class of algorithms, and it is a classification technique based on Bayes' Theorem with an assumption that the contributing features are independent. The presence of one particular part will not affect the others. We are supposed to binary classify the email corpus in the dataset given, using Naive Bayes for discrete counts. The tools used for building this classifier from scratch in Python are NumPy, RegEx, and Pandas.

2 Implementation

2.1 Data preprocessing

1. The dataset given is a text file that consists of 1000 emails, and their corresponding sentiment(0 for negative and 1 for positive).

2. We build a dataset using the given file by putting the *string[-1]* position in the sentiment column and *string[0:-1]* in the emails column.

3. Once we get the dataframe, we preprocess the emails by lowering the text, removing additional spaces, removing digits, symbols and tokenizing them. The words corresponding to each string are finally stored in a list.

   For example, consider the following string
   'I love Machine Learning :) '
   After preprocessing, it is converted to
   ['i','love','machine','learning']

2.2 Model

For a given string, after preprocessing, we will have a list, consisting of n words :

$$[x_1, x_2, x_3.....x_n]$$

To predict it's sentiment, we will use the following formula :

$$P(C \mid x) = \frac{P(x \mid C) \times P(C)}{P(x)}$$

where,

$P(C \mid x)$ denotes the posterior probability
$P(x \mid C)$ denotes the class conditional probability
$P(C)$ denotes the class probability

- $P(C)$ is calculated by counting the total number of words in the positive(or negative) class and dividing it by the total number of words present in the corpus.
- To calculate $P(x \mid C)$ where, consider C to be the positive (POS) class, we use the following formula :

$$P(x \mid POS) = \frac{N_{x|POS} + alpha}{N_{POS} + alpha \times N_{vocab}}$$

where,

$N_{x|POS}$ denotes the number of times the word x appears in POS class

N_{POS} denotes the total number of positive words

N_{vocab} denotes the total number of unique words present in the corpus

The $N_{x|POS}$(and likewise for negative) is stored in a dictionary with key as the tuple (word,sentiment) and the corresponding value as the total number of occurences of the word in POS class.

2.3 Testing

$$P(C \mid x) = P(C) \times P(x_1 \mid C) \times P(x_2 \mid C) \times P(x_3 \mid C).....P(x_n \mid C)$$

The sentiment is predicted using the above formula. The dominating probability decides the sentiment.
To measure the performance, accuracy was calculated by counting the total number of correctly classified text. K-fold cross validation was perfomed, where K = 5, and accuracies were rounded off to 2 decimal places.

iter1	0.82
iter2	0.87
iter3	0.8
iter4	0.78
iter5	0.79
iter6	0.82
iter7	0.8
avg	0.81

3 Limitations

This algorithm works quickly and can save a lot of time due to the assumption of independent features.
However, there are 2 major limitations of using this algorithms :

1. The major limitation of naive bayes is the assumption that predictor features are independent. This makes it unfit for usage in most of the real world cases, as it is almost impossible to find features that are completely independent.

2. Another disadvantage of this algorithm is the 'zero frequency problem'. If a categorical variable has a category, which was not present in the training dataset is encountered in the testing dataset, then the model which will assign a zero probability and will be unable to make a prediction. To solve this, we can use laplacian smoothing technique(alpha used in the formula).