Project 1
Multinomial Naive Bayes Algorithm

Yvan Pimi : ypimi@aimsammi.org
John Ndolo : jndolo@aimsammi.org
Aman Wassie : awassie@aimsammi.org
Alioune Diankha : adiankha@aimsammi.org

African Institute for Mathematical Sciences – Senegal (AIMS Senegal)

April 22, 2022

Outline

Objectives

- Implementing and testing of Multinomial Naive Bayes Algorithm from scratch
- Application of Naive Bayes algorithm in Text Classification

Naive Bayes Classifiers

- Naive Bayes classifier is a family of simple "probabilistic classifiers" based on Bayes' theorem to classify data into different classes[1].
- It works on the assumption of independence for all the features
- There are three flavors of naive Bayes classifies which correspond to distribution of features being analyzed:
  1. Bernoulli distribution is binary, did X occur or not?
  2. Multinomial distribution is discrete, often the frequency or count of X.
  3. Gaussian distribution is also known as the normal distribution for continuous variable

Multinomial Naive Bayes

- It is an example of Naive Bayes Classifiers
- It focuses on the count or frequency of occurrence of a feature in a document unlike in binary cases.
- Used for text data analysis and with problems with multiple classes
- Based on Bayes Theorem with an intuition of representing text documents as if were bag of words, i.e unordered set or word, positions ignored and keeping only their frequency in the document[2].
- An example of the intuition in Figure 1 below;

Bag of words assumption

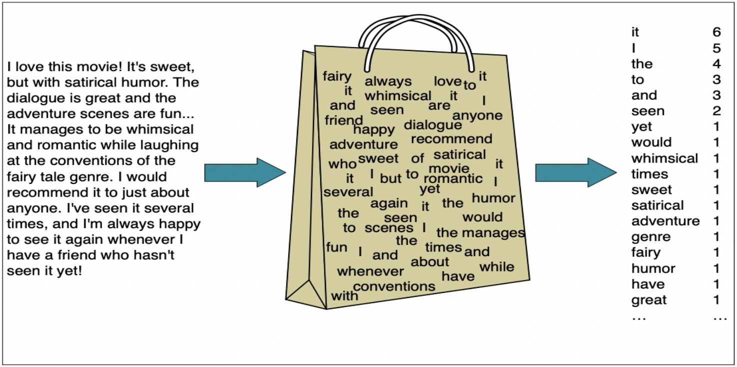

Figure: Intuition of the multinomial naive Bayes classifier applied to a movie review [2]

Bayes theorem

The theorem calculates the probability of an event occurring based on prior knowledge of conditions related to an event. It is based on the following formula:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{1}$$

where:

- $P(A|B)$ is the posterior probability
- $P(B|A)$ is the likelihood probability
- $P(A)$ is the prior probability of A
- $P(B)$ is the marginal or prior probability of B

Bayesian Inference

Naive Bayes as a probabilistic classifiers means we can categorize a document d to a class c with maximum posterior probability given the document.That is;

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \; P(c|d) \tag{2}$$

From Eq.1, the Eq.2 can be expressed into three components as;

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \; P(c|d) = \underset{c \in C}{\operatorname{argmax}} \; \frac{P(d|c) * P(c)}{P(d)} \tag{3}$$

CONT...

We can drop the denominator $P(d)$ since we compute the posterior for each class and the $p(d)$ remains constant for each class. Thus we can choose the maximum posterior probability using the formula;

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(d|c) * P(c) \tag{4}$$

Note: Naive Bayes is called a **generative model** because Eq.4 is viewed as if stating a kind of implicit assumption about how a document is generated:First a class is sampled from $P(c)$, and then words generated by sampling from $P(d|c)$.[1]

CONT...

Let d represent a document and c a class,without loss of generalization, let d be set of features $f_1, f_2, \ldots, f_n$, then Eq.4 can be re-written as;

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(f_1, f_2, \ldots, f_n|c)}^{likelihood} * \overbrace{P(c)}^{prior} \tag{5}$$

Naive Bayes classifiers make two simplifying assumptions for Eq.5

1. Bag of words assumption(Figure 1): assume position doesn't matter and each word has same effect on classification irregardless of its position in the document.
2. Naive Bayes assumption: Independence assumption

CONT...

Consequently, the final equation for the class chosen by a naive Bayes classifier is given as:

$$c_{NB} = \operatorname*{argmax}_{c \in C} P(c) * \prod_{f \in F} P(f|c) \tag{6}$$

To apply the naive Bayes classifier to text, we need to consider word positions, by simply walking an index through every word position in the document:

$$c_{NB} = \operatorname*{argmax}_{c \in C} P(c) * \prod_{i \in positions} P(w_i|c) \tag{7}$$

CONT...

Naive Bayes calculations are done in log space, to avoid underflow and increase speed. Thus Eq.7 is expressed as

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i|c) \tag{8}$$

By considering features in log space, Eq.8 computes the predicted class as a linear function of input features. Classifiers that use a linear combination of the inputs to make a classification decision —like naive Bayes and also logistic regression— are called **linear classifiers**[2]

Training the Naive Bayes Classifier

How can we learn the probabilities $p(c)$ **?**

- We simply use the frequencies in the data.
- For the class prior $P(c)$ we ask what percentage of the documents in our training set are in each class c.
- Let N_c be the number of documents in our training data with class c and N_{doc} be the total number of documents. Then:

$$\hat{P}(c) = \frac{N_c}{N_{doc}} \tag{9}$$

How can we learn the probabilities $P(f_i|c)$**?**

- We assume features is just the words in the documents bag of words
- Thus, we compute $P(w_i|c)$, which is the fraction of the times a word w_i appears among all words in all documents of class c.

CONT...

- By concatenating all documents in category c into one big 'category c' text, we use the frequency of w_i in the concatenated document to get the maximum likelihood estimate of the probability as:

$$\hat{P}(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w, c)} \tag{10}$$

Vocabulary(V) consists of the union of all the word types in all classes, not just the words in one class c. In case where the maximum likelihood estimate of the probability is zero, we term the phenomenon as **Zero probabilities**[2]. To solve the issue we use the **Laplace smoothing** given in the general formula:

CONT...

Laplace smoothing

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + \alpha}{(\sum_{w \in V} count(w, c)) + \alpha * |V|} \tag{11}$$

where:

- **V** is the vocabulary
- α is the smoothing

Points to note:

- We ignore the words that occur in test data but are not in our vocabulary since they did not occur in the training document.
- **Stop words:** This is a group of frequent words in a document such as 'the','at,'to' e.t.c. we normally ignore them.

CONT...

Training Algorithm

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return $logprior$, $loglikelihood$, V

function TEST NAIVE BAYES($testdoc$, $logprior$, $loglikelihood$, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in $testdoc$
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c \ sum[c]$

Figure: The naive Bayes algorithm
[2]

Worked example

Let consider this example;

Execution	Categories	Documents
Training	-	Just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Testing	?	predictable with no fun

The prior should be equal to $P(-) = \frac{3}{5}$ and $P(+) = \frac{2}{5}$

CONT..

The probabilities of each word regarding of the category is

$$P(predictable|-) = \frac{1+1}{14+20} \qquad P(predictable|+) = \frac{0+1}{9+20}$$

$$P(no|-) = \frac{1+1}{14+20} \qquad P(no|+) = \frac{0+1}{14+20}$$

$$P(fun|-) = \frac{0+1}{14+20} \qquad P(fun|+) = \frac{1+1}{9+20}$$

so for S = predictable with no fun. we have

$$P(-)P(-|S) = \frac{3}{5}(\frac{2*2*1}{34^3}) = 6.1*10^{-5} \tag{12}$$

$$P(+)P(+|S) = \frac{2}{5}(\frac{1*1*2}{29^3}) = 3.2*10^{-5} \tag{13}$$

thus this sentences is in the negative class.

Model Implementation

Train Module

Summary implementation steps:

- **clean text:** this function contain function such as remove punctuation, split document into word, lemmatize, Stemmatize.
- **prior probabilities:** here we compute the probabilities of each category.
- **word probabilities:** here we compute the probabilities of each word of the training data given a certain class.
- **Posterior Probabilities:** prior probabilities$\times$ word probabilities
- **predict:** after getting word probabilities, we have just to get the argmax of category prediction.

CONT...
Test Module

To test new data we apply the trained prior probabilities and world probabilities. The testing process summary in our code is as follows;

- split our data in train and test sub dataset
- train with the training dataset and test with the test dataset
- compute the accuracy of the model

Pros and Cons of Multinomial Naive Bayes algorithm

1. Pros
   - It can be used to solve multi-class prediction problems
   - It is easy to implement as you only have to calculate probability
   - It can be used on both continuous and discrete data
   - It is highly scalable and can easily handle large datasets
   - It is fast and efficient, you can use it to make real-time predictions
2. Cons
   - Low prediction accuracy compared to other algorithms
   - It is not suitable for regression
   - Assumption of independence doesn't hold always
   - Zero probability phenomenon (When test data not found in any category)

Applications

1. Weather prediction
2. Spam detection
3. Language identification
4. Sentimental analysis
5. Authorship identification
6. News classifications
7. Face recognition
8. Medical diagnosis

References

[1] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[2] Dan Jurafsky and Martin James H. *Speech and language processing*. Pearson Education India, 2020.


Thank
You