Bernoulli Naive Bayes Classification

Raissa Feuyo
Abdou Niang,
Habib Mbow
Heritiana Andriasolofo

African Institute for Mathematical Sciences - AMMI

April 22, 2022

Contents

Introduction

Binary classification

Binary classification is the task of classifying the elements of a set into two groups on the basis of a classification rule (0 or 1, True or False, Yes or No, etc...)

What is Naive Bayes

- Naive Bayes is a statistical classification technique based on Bayes Theorem, on conditional probability,
- It is one of the simplest machine learning algorithms for classification(Text classification/ Spam Filtering/ Sentiment Analysis) .

Types of Naive Bayes model

There are three types of Naive Bayes model :

- Gaussian: It is used in classification and it assumes that features follow a normal distribution.
- Multinomial: It is used for discrete counts.
- Bernoulli: The binomial model is useful if your feature vectors are binary (i.e. zeros and ones).

The Naive Bayes model are based on conditional probability, which is defined as below.

Conditionnal probability

A conditional probability is the probability of an event, given some other event has already occurred. The formula for conditional probability is

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad \text{where} \quad P(B) \neq 0.$$

Bayes' Theorem

The Bayes's formula follows from the definition of the conditional probability.

Bayes' Theorem

Bayes' Formula is given by :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \text{where} \quad P(B) \quad \neq 0.$$

Chain rule

Another important formula in conditional probability is chain rule.

The chain rule of conditional Probabilities is also called the general product rule. It allows the calculation of any number of the associate distribution of a set of random variables.

It is given by the formula below.

$$\begin{aligned} P(A_n, A_{n-1}, ..., A_1) &= P(A_n|A_{n-1}, ..., A_1)P(A_{n-1}, ..., A_1) \\ &= P(A_n|A_{n-1}, ..., A_1)P(A_{n-1}|A_{n-2}, ..., A_1)P(A_{n-2}, ..., A_1) \\ &= \prod_{k=1}^{n} P(A_k|A_{k-1}, ..., A_1) \end{aligned}$$

Bernoulli Bayes approach for binary classification

Let us consider $X = (x_1, \ldots, x_d)$ a vector random variable of d-dimension , where $x_i \in \{0, 1\}$, and $C = \{0, 1\}$ a binary class.

Bayes appoach

The Bayes optimal classifier is $\hat{C} = \arg \max_{k \in \{0,1\}} P(C = k|X)$.

To describe the probability function $P(C = k|X)$ we have 2^d possibles parameters, each of which corresponds to $P(C = 1|X)$ for a certain value of $X \in \{0, 1\}^d$.

Naive Bayes

In the Naive Bayes approach, we make the (rather naive) generative assumption that given the label, the features are independent of each other. From this assumption and using chain rule, we get

$$P(X|C = k) = \prod_{i=1}^{d} P(xi|C = k)$$

Naive Bayes

But the Bayes optimal classifier is:

$$\begin{aligned}\hat{C} &= \arg\max_{k\in\{0,1\}} P(C=k|X) \\ &= \arg\max_{k\in\{0,1\}} P(C=k)P(X|C=k) \\ &= \arg\max_{k\in\{0,1\}} P(C=k)\prod_{i=1}^{d} P(x_i|C=k))\end{aligned}$$

Now, the number of parameters needed to estimate becomes 2d + 1.

Estimation of $P(x_i = 1|C = k)$

By maximizing the joint likelihood function,

$$L(P(C = k), P(x_i|C = k)) = \prod_{i=1}^{d} P(x_j, k_i),$$

we obtain estimate the parameters given by the formula below:

$$\begin{aligned} P(x_i = 1|C = k) &= \frac{count(x_i = 1 \text{ and } C = k)}{count(C = k)} \\ P(C = k) &= \frac{count(C = k)}{count(C = 0) + count(C = 1)}. \end{aligned}$$

Example

Consider the following data.

Document	Contents	C
doc1	i do like it	1
doc2	i dont like it	0
doc3	you do love it	1
doc4	you dont love	0

Table: Train data

We get the following list of words :

[i,do,dont,like,you,love,it]

Example cont'd

We can then translate our training data to the following using the one hot encoding. Also using the formula above, we can compute the probability parameters.

$\mathbf{X} \backslash \mathbf{x_i}$	**i**	**do**	**dont**	**like**	**you**	**love**	**it**	**C**
doc1	1	1	0	1	0	0	1	1
doc2	1	0	1	1	0	0	1	0
doc3	0	1	0	0	1	1	1	1
doc4	0	0	1	0	1	1	0	0
$P(x_i = 1 \mid C = 0)$	0.5	0	1	0.5	0.5	0.5	0.5	0.5
$P(x_i = 1 \mid C = 1)$	0.5	1	0	0.5	0.5	0.5	1	0.5

Example cont'd

Now let us classify the following texts:

- test1 : i dont appreciate the way you did it
- test2 : i really appreciate the way you did it

$\mathbf{X \backslash x_i}$	**i**	**do**	**dont**	**like**	**you**	**love**	**it**	**C**
test1	1	0	1	0	1	0	1	?
test2	1	0	0	0	1	0	1	?
$P(x_i = 1 \mid C = 0)$	0.5	0	1	0.5	0.5	0.5	0.5	0.5
$P(x_i = 1 \mid C = 1)$	0.5	1	0	0.5	0.5	0.5	1	0.5

Example : classification of test1

test1

i dont appreciate the way you did it

$\mathbf{X}\backslash\mathbf{x_i}$	**i**	**do**	**dont**	**like**	**you**	**love**	**it**	**C**
test1	1	0	1	0	1	0	1	?
$P(x_i = 1 \mid C = 0)$	0.5	0	1	0.5	0.5	0.5	0.5	0.5
$P(x_i = 1 \mid C = 1)$	0.5	1	0	0.5	0.5	0.5	1	0.5

- $P(C = 0|\text{test1}) = \underbrace{0.5}_{P(C=0)} \times 0.5 \times (1-0) \times 1 \times (1-0.5) \times 0.5 \times \cdots = \frac{1}{2^6}$.
- $P(C = 1|\text{test1}) = \underbrace{0.5}_{P(C=1)} \times \underbrace{(1-1)}_{=0} \times \cdots = 0$.

So we classify test1 to be negative i.e. $C = 0$.

Example : classification of test2

test2

i really appreciate the way you did it

$\mathbf{X}\backslash\mathbf{x_i}$	**i**	**do**	**dont**	**like**	**you**	**love**	**it**	**C**
test2	1	0	0	0	1	0	1	?
$P(x_i = 1 \mid C = 0)$	0.5	0	1	0.5	0.5	0.5	0.5	0.5
$P(x_i = 1 \mid C = 1)$	0.5	1	0	0.5	0.5	0.5	1	0.5

- $P(C = 0|\text{test2}) = \underbrace{0.5}_{P(C=0)} \times 0.5 \times (1-0) \times \underbrace{(1-1)}_{=0} \times \cdots = 0.$
- $P(C = 1|\text{test2}) = \underbrace{0.5}_{P(C=1)} \times 0.5 \times \underbrace{(1-1)}_{=0} \times (1-0) \times \cdots = 0.$

We do not know which class we should give test2.

Laplace smoothing

Why?

The problem from previews slide arises because decided to assign a probability of the appearance of a word in a given class equal to zero if it does not appear in any of the training data of that class.

To solve it, we will make a slight change for the formula of the estimators.

Laplace smoothing

$$P(x_i = 1|C = k) = \frac{1 + count(x_i = 1 \text{ and } C = k)}{2 + count(C = k)}$$

$$P(C = k) = \frac{1 + count(C = k)}{2 + count(C = 0) + count(C = 1)}.$$

Example with Laplace

Let us now use the formula for estimation of the probabilities using Laplacian smoothing.

$\mathbf{X \backslash x_i}$	**i**	**do**	**dont**	**like**	**you**	**love**	**it**	**C**
doc1	1	1	0	1	0	0	1	1
doc2	1	0	1	1	0	0	1	0
doc3	0	1	0	0	1	1	1	1
doc4	0	0	1	0	1	1	0	0
$P(x_i = 1 \mid C = 0)$	0.5	0.25	0.75	0.5	0.5	0.5	0.5	0.5
$P(x_i = 1 \mid C = 1)$	0.5	0.75	0.25	0.5	0.5	0.5	0.75	0.5
test2	1	0	0	0	1	0	1	?

After computing the two probabilities $P(C = 0|\text{test2})$ & $P(C = 1|\text{test2})$, we classify test2 to be positive i.e. $C = 1$.

Implementation

We implemented two module from scratch :

- the `DataProcessing` module and
- the class `Bernoulli_Naive_Bayes_Classifier` module.

Implementation of DataProcessing module

`DataProcessing` module which contains the following functions

- `clean_text` : which removes all non alphabetical symbols and all stopwords from the input string.
- `clean_data` : which applies the `clean_text` function on a panda dataframe.
- `get_unique_words` : which will return a list of all words in a given dataframe.
- one_hot_encoding : which will encode the input dataframe to its corresponding 0/1 vectors using the given list of words in the parameter.

Implementation of the class `Bernoulli_Naive_Bayes_Classifier` module

The class `Bernoulli_Naive_Bayes_Classifier` contains the following methods :

- `fit` : which will train the model
- `predict` : which will make the prediction
- `confusion_matrix` : which will return the confusion matrix

Experience & Result

During our experience, we got the following

- Train = 500
- Test = 50
- Accuracy = 32/50 = 64%
- FN = 15 , FN = 3 , TN =26 , TP = 6

Interpretation of our Results

Observations 1

- It performs well when the input values are categorical rather than numeric. In the case of numerics, a normal distribution is assumed to compute the probabilities (a bell curve, which is a strong assumption).

Observations 2

- In case of small amounts of data or small documents(for example in text classification), Bernoulli Naive Bayes gives more accurate and precise results as compared to other models(It is fast and are able to make to make real-time predictions)

Thank you, ...