Naive Bayes Classifier Report

Arnav Kamat
Aditya Bawangade

November 2022

1 Introduction

Given a large labeled set of data of images of the handwritten English Alphabet; we aim to build a classifier to identify the given image of an alphabet. To address this problem we make use of the Naive Bayes Classifier model.

A Naive Bayes classifier makes use of the available labeled data and calculates the conditional probabilities using the properties of the Bayes Theorem in Probability. This is an effective method to understand the influence of probability in real life and to learn whether the model can be extrapolated to real-world problems.

The accuracy of the given model will be useful in determining so

2 Problem Definition

2.1 Task Definition

The input is a large labeled dataset of 28*28 resolution images of the handwritten English Alphabet. Each pixel is enumerated with the corresponding pixel scale in BW in the range of 0-255. Also, every image is labeled with indexes varying from 0-25 for the precise identification of Alphabets. Note that we initially split the data in the ratio 9:1 for training and testing and call them training and testing datas respectively.

The input data is a 2-dimensional array. The first column of the array identifies the label and the next 784 columns house the color scale values respectively.

2.2 Algorithm Definition

The above problem can be addressed efficiently using the Multinomial Naive Bayes Algorithm.

Baye's Theorem

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

We now distinguish between the label and pixels by treating pixels as 784 independent features. Note that these pixels will be given for the test image. We will now identify the label y_k by using Baye's theorem as follows:

let Data = $x_1 = x_1^t, x_2 = x_2^t, x_3 = x_3^t ..., x_{784} = x_{784}^t$

$$P(y^t = k|\text{Data}) = \frac{P(y^t = k, \text{Data})}{\text{Data}} \tag{1}$$
$$= \frac{P(y^t = k, \text{data})}{\Sigma_{k=0}^{25} P(y^t = k, \text{Data})} \tag{2}$$
$$= \frac{P(y^t = k).P(\text{Data}|y^t = k)}{\Sigma_{k=0}^{25} P(y^t = k).P(\text{Data}|y^t = k)} \tag{3}$$
$$= \frac{P(y^t = k).P(x_1 = x_1^t, x_2 = x_2^t, x_3 = x_3^t, ...x_{784} = x784^t|y^t = k)}{\Sigma_{k=0}^{25} P(y^t = k).P(x_1 = x_1^t, x_2 = x_2^t, x_3 = x_3^t, ...x_{784} = x784^t|y^t = k)} \tag{4}$$
$$= \frac{P(y^t = k).\Pi_{i=1}^{784} P(x_i = x_i^t|y^t = k)}{\Sigma_{k=0}^{25} P(y^t = k)\Pi_{i=1}^{784} P(x_i = x_i^t|y^t = k)} \tag{5}$$

Having calculated the probability of the image being labeled k given the data set of the image; we repeat the process for all of the 25 labels i.e. we calculate the conditional probabilities for all the labels and return the maximum of those values.

After obtaining the label we compare it with the actual label of the test image and thus, determine the accuracy of our model by taking 10% of the data set for testing and comparing the experimental label with the actual label of the image.

3 Experimental Evaluation

3.1 Methodology

In the training stage of the model, we have to obtain two probabilities for each pixel namely Prior and conditional probability.

$$P(y^t = k) = \frac{\text{\#number of samples labeled k}}{\text{\#samples of training Data}}$$

$$P(y^t = k) = \frac{}{\text{\#samples modeled k}}$$

Upon obtaining these probabilities for all the pixels, we now calculate the Probabilities of the data set being any of the labels and return the argmax of the probabilities, compare the probabilities and return the accuracy.

Training Stage:

1. Division into training and testing
2. Fixing the bin size to scale down the color scale values from 256 to 16
3. Smoothing the conditional probability to avoid invalid mathematical expressions due to the total probability law
4. Calculating the logarithm of the conditional probabilities for every pixel to avoid mishandling of the extremely small probabilities

Test Stage: Iterate through the test data and compute the accuracy using the above algorithm

3.2 Results & Conclusions

After repeating the simulation for several number of times we have obtained an accuracy of around 70%. It is pretty clear from the accuracy that Naive Bayes Classifier is suitable as this algorithm works quickly and can save a lot of time however it assumes that the features are independent which isn't the case when it comes to real-life classification problems. Therefore, we've learnt to apply the Bayes theorem to real-life problems like image recognition using the Naive Bayes Classifier approach and understood the influence of several theorems of the Probability theory in such classifications and also the limitations of the same due to the total probability assumptions and the assumption of independence as well.