Classifying Images of Handwritten Digits Using Density Estimation and Naïve Bayes

Shashi Kiran Chilukuri
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, USA
schiluk6@asu.edu

I. Introduction

This image classification project comes under CSE 575: Statistical Machine Learning course for 2020 Spring B semester. The aim of this project is to develop a computing model that can classify the images, specifically, to classify images of handwritten digits of 1's and 0's. To develop this model, trainsets and testsets for digits 0 and 1 were given[1]. Here is the sample size of each of these datasets:

TABLE I
Sample size of Train and Test Datasets

Dataset	***Sample size***
Training set '0' (train0)	5000
Training set '1' (train1)	5000
Testing set '0' (test0)	980
Testing set '1' (test1)	1135

Here are the images of the first five handwritten digits of train0, train1, test0 and test1 datasets respectively.

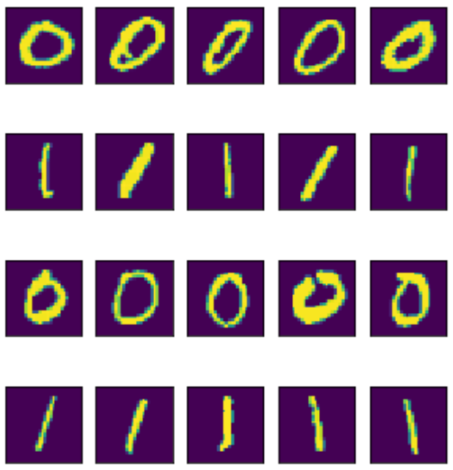


Fig. 1. Images of First five handwritten digits of train0, train1, test0 and test1 datasets respectively

Using this raw data, need to build the Naïve Bayes classifier model that can accurately classify the digit 0 and digit 1. To achieve this, the project is divided into four tasks:

- Task 1: Feature Extraction
- Task 2: Calculate Parameters
- Task 3: Calculate Probability and Feature Prediction
- Task 4: Calculate Accuracy

In the task 1, need to extract two new features from each of the handwritten digit images of both train and test datasets to convert the original image data arrays to 2-dimensional data points. Here, it is given that these two newly extracted features are statistically independent to each other and that each of these images are drawn from a normal distribution.

In the task 2, need to calculate all the parameters needed for the two class Naïve Bayes classifier using the data points generated in Task1.

In the task 3, need to implement the two class Naïve Bayes classifier from scratch that can classify the test dataset accurately. Here, it is given that prior probability of each class is 0.5.

In the task 4, need to check the performance of the model on the test dataset for both digit0 and digit1 separately.

Finally, the model developed from scratch is verified against the standard GaussianNB() model from SkLearn python library and see how accurately it performed.

Before we get into the solution, here is the list of resources used in this project:

TABLE II
Resources

	Resources
Programming language	Python
Integrated Development Environment (IDE)	Jupyter Notebook, Python Environment
Python Libraries	numpy, pandas, matplotlib, ccipy.io, math, geneNewData, statistics, SKlearn
Models/Methods	MLE Density Estimation, Naïve Bayes Classification, GaussianNB()
Dataset	A subset of MNIST dataset [1](data corresponding to digit 0 and digit 1 only)

II. Solution

A. *Task 1: Feature Extraction*

Following two features are extracted from each of the handwritten digit images from both train and test datasets:

- Feature1: The average brightness of each image (Calculate the average of all pixel brightness values within a whole image array)

- Feature2: The standard deviation of the brightness of each image (Calculate the standard deviation of all pixel brightness values within a whole image array)

Now, to calculate the mean and standard deviation for each image, used Python numpy libraries' mean() and std() methods, and are stored as Feature 1 and Feature 2 respectively.

B. *Task 2: Calculate Parameters*

The parameters required for the two-class Naïve Bayes classifier are calculated in this task. The number of parameters that needs to calculated will be eight as there are two classes and two features, and needs to calculate mean and variance for each one of them. To calculate the mean and variance, used the mean() and variance() methods from the 'statistics' python library. Here is the list of parameters and its values obtained after applying appropriate methods:

TABLE III
Parameter List and its values

Parameters	***Values***
(No.1) Mean of feature1 for digit0	44.130842857142859
(No.2) Variance of feature1 for digit0	112.65821501430544
(No.3) Mean of feature2 for digit0	87.363229445467454
(No.4) Variance of feature2 for digit0	100.69963623475806
(No.5) Mean of feature1 for digit1	19.29315892857143
(No.6) Variance of feature1 for digit1	31.346790958404547
(No.7) Mean of feature2 for digit1	61.218260565000172
(No.8) Variance of feature2 for digit1	82.977566631048887

C. *Task 3: Calculate Probability and Feature Prediction*

In this section, need to find the class that best fits the feature data. And, since it was already given that the data is draw from the normal distribution, we can use the Gaussian Probability Distribution Function to estimate the probability (likelihood) of a given data point. Following is the mathematical expression of Gaussian (Normal) Probability Distribution Function:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Fig. 2. Probability Density Function Formula [2]

This formula is developed in python as a function using python 'math' library. Here, variance is the variance of the feature (x) and mean is the mean of the feature (x). When the feature values (data points), mean and variance (these parameters are calculated in task 2) values are inserted in the above expression, it will calculate probability values. The above expression expressed in python as follows:

```
p(x) =  ((1/(sqrt(2*pi*variance)))*exp(-((x-mean)**2/(2*variance))))
```

Fig. 3. PDF Formula in Python

Now, to calculate the probability that a set of data points belongs to particular class, we use Bayes theorem. Bayes's theorem is stated mathematically as the following equation:

P(class | data) = P(class) * P(data | class) / P(class)
∝ P(class) * P(data | class)
Note: Denominator P(class) is removed to simplify the calculation

Fig. 4. Bayes Rule[2]

From the Bayers rule, we can clearly see that the probability of a class for a given data is not just the probability of data belonging to a class. We need to multiply it by the prior probability of the class.

Since we have two classes (digit 0 and 1), we need to calculate the class probabilities separately for each class i.e. we calculate the probability of digit '0'(class 0) with relevant parameters, then calculate the probability of digit '1' (class 1). And also, since we have two features (Feature1 and Feature2), the probability that a certain feature set belongs to certain class is calculated as below:

p(class=0| F1, F2) = p(F1|class=0) * p(F2|class=0) * p(class=0)
p(class=1| F1, F2) = p(F1|class=1) * p(F2|class=1) * p(class=1)
Note: F1, F2 are features

Fig. 5. Probability of a class for a given feature set[2]

Here is the algorithm used to calculate the class probabilities for a given dataset expressed above:

```
1. For each class (digit 0 and 1) and its parameters obtained from task 2
   a. Probabilities[class] = 0.5
   b. For each feature parameter
         i. Probabilities[class] = Probabilities[class] * Posterior probability
```

Fig. 6. Algorithm to calculate the probability of a class for a given data

It is given that prior probability P(class=0) = P(class=1) = 0.5. By inserting this along with the parameters of both the classes calculated in task2 in the above algorithm, we can calculate the class probability for each of the classes (digit 0, digit 1). Now, by comparing these probabilities, class with best probability is picked as an outcome for given set of data.

Once we train the model with the training data, it is tested with the test data to predict the class. Then, this predicted class is compared against the actual test class data to see how will the model performed.

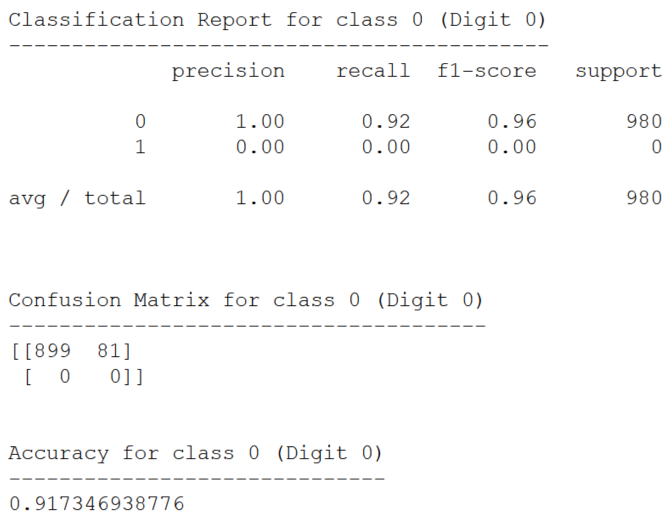
D. *Task 4: Calculate Accuracy*

In this phase, accuracy of the model is calculated for both the classes by comparing the predicted class label to the actual test class label. Here are the calculated accuracies for test dataset digit 0 and digit1:

Accuracy of test set for digit 0	0.917346938776
Accuracy of test set for digit 1	0.923348017621

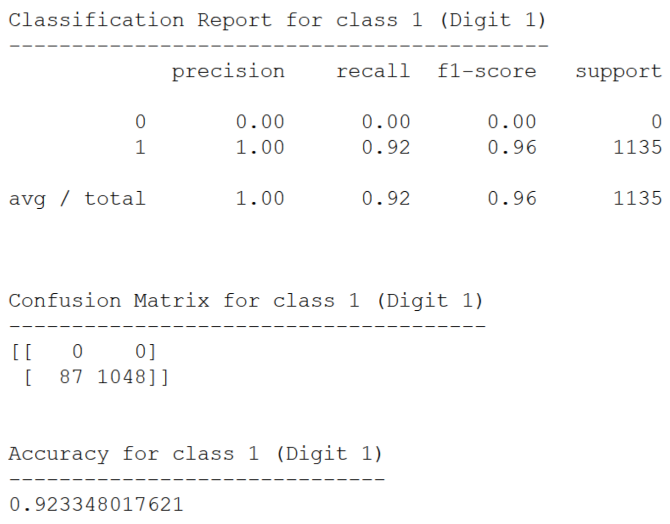
III. Results

Using the concepts of Density Estimation and Naive Bayes[2], successfully developed the classification algorithm from scratch by performing all four tasks i.e. extracting features, calculating parameters, calculating probability by using the Normal (Gaussian) distribution function and predicting the classes and finally, calculating the accuracy of test set for digit 0 and 1. In the end, developed algorithm gave an accuracy score of 0.917346938776 for digit 0 and 0.923348017621 for digit 1 respectively. To validate these results, developed another model using standard GaussianNB() method from SKLearn library with prior probability of 0.5.



```
Classification Report for class 0 (Digit 0)
-------------------------------------------
             precision    recall  f1-score   support

          0       1.00      0.92      0.96       980
          1       0.00      0.00      0.00         0

avg / total       1.00      0.92      0.96       980


Confusion Matrix for class 0 (Digit 0)
--------------------------------------
[[899  81]
 [  0   0]]


Accuracy for class 0 (Digit 0)
------------------------------
0.917346938776
```

Fig. 7. Model performance report for class 0 (Digit 0)



```
Classification Report for class 1 (Digit 1)
-------------------------------------------
             precision    recall  f1-score   support

          0       0.00      0.00      0.00         0
          1       1.00      0.92      0.96      1135

avg / total       1.00      0.92      0.96      1135


Confusion Matrix for class 1 (Digit 1)
--------------------------------------
[[   0    0]
 [  87 1048]]


Accuracy for class 1 (Digit 1)
------------------------------
0.923348017621
```

Fig. 8. Model performance report for class 1 (Digit 1)

This model gave exactly same prediction and accuracy scores as model created from scratch. Above is the performance report of standard python model for digit 0 and digit 1 . This clearly indicates that the model developed from scratch works perfectly fine as the standard method from Sklearn library.

IV. Contributions

This is an individual project. All the tasks, right from understanding the project requirements to preparing the report for project submission are performed individually by me. But here, I would like to give some credit to Prof. Ghayekhlo for providing guidance and support throughout the project lifecycle. Here is the list of activities I performed as part of this project:

- Understand the project requirements and its tasks[3].
- Extract Digit 0 and Digit 1 train and test image datasets.
- Visualize the train and test datasets using python's Matplotlib library.
- Extract two new features from the both train and test images as per the requirement.
- Calculate the parameters required for the Gaussian Probability Distribution Function.
- Develop the two class Naive Bayes classifier from scratch using the concepts from Bayes Theorem in python using basic libraries like numpy, Scipy, math etc.
- Calculate class probabilities of both classes for a given set of features on test data.
- Compare and pick the class label that performed best for each set of features.
- Calculate the accuracy of the test data for Digit 0 and Digit 1 separately.
- Create the standard GaussianNB() model using Python's SKLearn Library.
- Generate performance reports of standard python model for test data.
- Compare the results of model developed from scratch with standard python model.
- Report how well the model developed from scratch performed when compared with the standard model.
- Prepare the project submission and the portfolio reports.

V. Lessons Learned

Here are some of the lessons learnt from this project:

- Understanding the requirements is key for the success of the project.
- Getting to know the importance of feature extraction from the image data for classifying them.
- This project helped me to strengthen my understanding on concepts and mathematical expressions related to Density Estimation and Naïve Bayes machine learning models, as a result, able to develop the model from scratch that performs on par with the standard library model.

References

[1] MNIST dataset. Available: http://yann.lecun.com/exdb/mnist/

[2] CSE 575 Statistical Machine Leaning, Spring B 2020, ASU

[3] Project requirements. Available: https://www.coursera.org/learn/cse575-statistical-machine-learning/ungradedLab/ii9TB/part-1-naive-bayes-classifier