Naive Bayes Classifier

Aditya Jain

1 What modules are used and why

1. **NumPy** : It is a very vast library with lots of functions in it. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. I have used it very often in my project.
2. **Pandas** : Pandas is an open source library in Python. It provides ready to use high-performance data structures and data analysis tools. Pandas module runs on top of NumPy and it is popularly used for data science and data analytics. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. In this project I am using it to read and manipulate data according to my need.
3. **Matplotlib** : Matplotlib is an plotting library in python. It is mostly and widely used for visualization purpose. I have used this module to display some of the images from given dataset.
4. **String** : I have used this module just to make a dictionary of labels as keys and capital letters as my values.

2 Procedure

- Importing necessary modules.
- Reading data from dataset.
- Splitting data into training data set and testing data set.
- Splitting training and testing data sets into image pixels and their corresponding labels.
- Scaling pixel values to '0' and '1' according to threshold value.(here I am using threshold=120)
- Displaying some images for better visualization of given dataset.
- Calculating probabilities of each class. (here classes are referred as labels)

- Calculating Conditional Probability of each pixel on the train data set given image belongs to that class. Here I am calculating the probability that how many times the '1' occur at that pixel in whole train data set given that class . (accordingly P(that pixel=0)=1-P(that pixel=1)).
- Predicting the labels of test data using the above calculated probabilities.
- xi represents ith pixel of that image and Xi checks ith pixel of all the train images given that class and calculate it's probability.

$$P(Y^t = k|X_1 = x_1^t, X_2 = x_2^t, ..., X_{784} = x_{784}^t) = \frac{P(Y^t = k)\prod_{i=1}^{784} P(X_i = x_i^t|Y^t = k)}{\sum P(Y^t = k)\prod_{i=1}^{784} P(X_i = x_i^t|Y^t = k)} \quad (1)$$

- Since here I am scaling it into '0' and '1' so there is no "Inference Issues occur" so I can directly multiply this values.
- Finally printing Model Accuracy and Confusion Matrix.
- Some observation's to check where our model predicts incorrectly.

3 Observations

So the final accuracy of my model comes to be 70.7%. Which is quite good accuracy. I have formed an Confusion matrix to see how actual and predicted labels correlated. So from the confusion matrix I calculated which labels it is predicting more wrongly.

1. ('O','D').(So here the actual label of the image is 'D' but model predicted it 'O' more than 10% of correct labels it predicted).
2. ('N','H').(So here the actual label of the image is 'H' but model predicted it 'N' more than 10% of correct labels it predicted).
3. ('T','J').(So here the actual label of the image is 'J' but model predicted it 'T' more than 10% of correct labels it predicted).
4. ('K','R').(So here the actual label of the image is 'R' but model predicted it 'K' more than 10% of correct labels it predicted).
5. ('A','R').(So here the actual label of the image is 'R' but model predicted it 'A' more than 10% of correct labels it predicted).
6. ('W','U').(So here the actual label of the image is 'U' but model predicted it 'W' more than 10% of correct labels it predicted).
7. ('Y','X').(So here the actual label of the image is 'X' but model predicted it 'Y' more than 10% of correct labels it predicted).

So finally my model does bad at predicting this above labels and very bad in case of predicting 'R'.