Statistical Pattern Recognition

Programming Assignment 1

Adhokshaja V Madhwaraj	160010032
Abhishek R	160020036
Rohan Sheelvant	160020005
Siddharth Sagar	160030034

Problem Statement

Design and Implementation of Bayes Classifier on multiple datasets - namely 20 Newsgroup Dataset and the Fashion MNIST dataset. Appropriate Class Conditional Densities and parametric estimation models were to be chosen for finding the unknown parameters of the class conditionals.

Datasets + Preprocessing

20 Newsgroups Dataset

The 20 Newsgroups dataset contains nearly 19,000 discussions, snippets, emails, articles, etc. on 20 different topics ranging from religion to sports to technology. Each file is a binary file encoded by a **latin1** encoding. The dataset is almost equally divided among the 20 classes.

Preprocessing

1. Generating a Vocabulary
    - All the files in all the folders in the dataset was iterated through
    - In each file, the stop-words (basic words which have no particular meaning given a context in NLP) were removed, and the headers and footers (including signatures were removed)
    - Punctuation marks, numeric values were cleaned, and single letter words were removed.
    - Once the file was cleaned, the words were extracted into a dictionary with a counter updated every time the word is seen
    - The top 100 words of each subclass was collated in a set (to remove any duplicates) and written to a vocabulary file
2. Iterating through all the files and folders

- Using the created vocabulary, a dictionary was created with keys as words, and counter was set to 0
- Iterating through every file, the class name (folder name) and the count of each word was stored (bag of words model). Each time the entire set was saved to a Numpy array and appended to the X_data input for the system

3. Exporting Numpy Array as file and Classes as a .txt file
    - The entire Numpy array was exported to a Numpy array and saved as X_data.npy and classes as classes.txt
4.

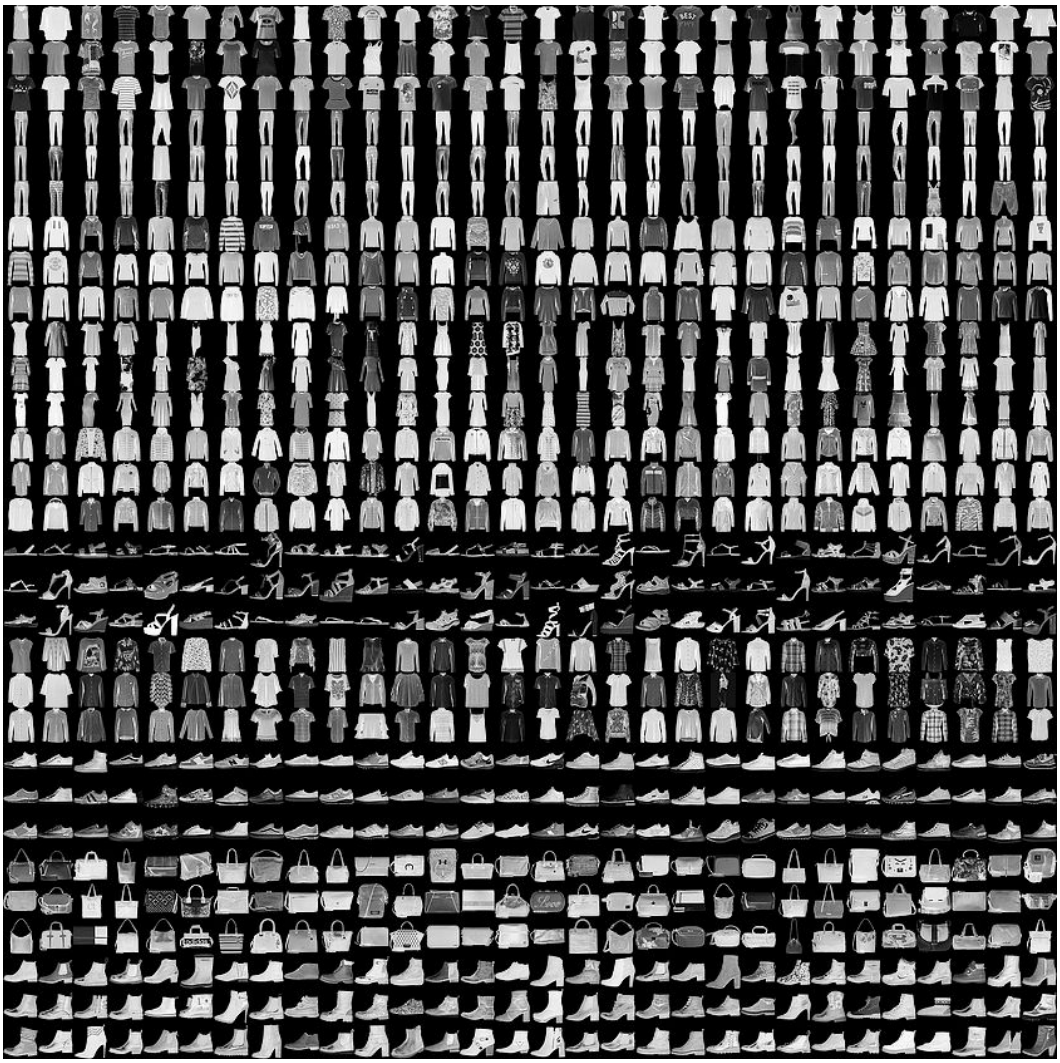
Fashion MNIST dataset

The Fashion MNIST dataset is the fashion equivalent of the MNIST dataset for handwritten numbers. Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

0	T-shirt	1	Trousers	2	Pullover
3	Dress	4	Coat	5	Sandals
6	Shirt	7	Sneaker	8	Bag
9	Ankle Boots				

Preprocessing

The preprocessing in the case of Fashion MNIST was simpler. CSV files were available for the training and testing data. Each CSV file contained the label as the first column and the 784 grayscale pixel values in the subsequent columns (each picture is a 28X28 grayscale image with values ranging from 0-255). Again the data was combined into a Numpy array and the class labels were written into a file, to ensure easy running of the code as before (the code for MNB and Dirichlet was written modularly to ensure to work easily).



Probabilistic Classification

Bayes Classification is a conditional probability model which, given a feature vector **x** = (x_1, x_2, ... ,x_n), it assigns to the probability: **P**(C_k / x_1, x_2, ... ,x_n). Using Bayes theorem, the conditional probability can be decomposed by the following statement

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \qquad \text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

The evidence is a constant, which we are going to ignore in the calculation of the posterior.

Calculation of the Prior

During the preprocessing of the data, we iterated through the folders corresponding to each of the classes, using a counter. Post this, we calculated Prior based on the simple formula

(Number of relevant files / Total number of Files)

Multinomial Naive Bayes Classifier

Naive Bayes Classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong independence assumptions between the features. Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector **x** = (x_1, x_2,.....,x_n), representing some n features (independent variables), it assigns to this instance probabilities **P**(C_k / x_1, x_2,.....,x_n) for each 'k' possible classes.

The multinomial distribution is a generalization of the binomial distribution. In 20 newsgroup datasets, the frequency count of words was used as the feature vector. As the features represent count values and the outcome of each trial has a categorical distribution, we can model the features to follow Multinomial Distribution. In Fashion MNIST dataset, each pixel is used as a feature. As each pixel can take values between 0 and 255, the features are modeled Multinomial Distribution.

In the Multinomial Naive Bayes model, each **P**(x_i/ C_k) follows multinomial distribution, rather than some other distribution.

$$\begin{aligned} p(C_k \mid x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &= p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k) \cdots \\ &= p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k), \end{aligned}$$

The Multinomial Naive Bayes classifier calculates the posterior for each class C_k and then predicts the output class. The predicted class is the class that maximizes the posterior probability.

Estimation Class Conditional Densities

Each feature is modeled as Multinomial Distribution. The PMF of multinomial distribution is proportional $p_1^{x1} * p_2^{x2} ..... p_n^{xn}$, where p_i represents the probability of feature(i) and x_i represents the count of the i^{th} word. For implementing the Multinomial Bayes Classifier, we have estimated the probability of features given class C_y .

Estimate of (p_i/C_y) = (x_i/C_y) / Σ (x_i/C_y) , which is equal to: i^{th} word count / total word count, in-class C_y

Estimation using Laplace Smoothing of (p_i/C_y) = ((x_i/C_y) + alpha) / Σ ((x_i/C_y) + (alpha*total_no_of_words))

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where N is the frequency of words

Implementation of Multinomial Naive Bayes

- From the whole data, a train and test split are achieved with an 80-20 split, with a catch that each class is split on this basis. This change allowed us to get better performance on the dataset than the inbuilt Scikit-Learn MNB function.
- Priors are then obtained.
- From the training data we estimate the probability vector
- Posterior probabilities' calculation is simplified by the use of log over the multiplication. This multiplication was leading to a very small number which was converging to zero, thus on conversion to log and adding, this problem was solved. Consistently, the Prior probability was also converted to log scale, and added for the final probability.
    - Since the probability is a positive, and converting to log scale doesn't affect taking the maximum value

Dirichlet model

The Dirichlet distribution is a model of how proportions vary. Let **p** denote a random vector whose elements sum to 1, so that pk represents the proportion of item k. Under the Dirichlet model with parameter vector **α**, the probability density at **p** is

$$p(\mathbf{p}) \sim D(\alpha_1, ...\alpha_k) = \frac{\Gamma(\sum_k \alpha_k)}{\Pi_k \Gamma(\alpha_k)} \Pi_k p_k^{\alpha_k - 1}$$

Where $p_k > 0$ and $\sum_k p_k = 1$

To find alpha values we have tried the following two methods:

- Using inbuilt function Dirichlet in python3 to estimate maximum-likelihood estimate of **α**.

https://github.com/ericsuh/dirichlet.git

- For the other one, we are following a paper written by Thomas P. Minka.

  To find the approximation MLE, we use the following equations which give the relation between alphas and, estimation and variance of random variable **p**.

$$E[p_k] = \frac{\alpha_k}{\sum_k \alpha_k}$$

$$\text{var}(p_k) = \frac{E[p_k](1 - E[p_k])}{1 + \sum_k \alpha_k}$$

$$\log \sum_k \alpha_k = \frac{1}{K-1} \sum_{k=1}^{K-1} \log \left(\frac{E[p_k](1 - E[p_k])}{\text{var}(p_k)} - 1 \right)$$

  So we can get all the alphas for **p** to get the p(**p**), which in our case we use it as a class conditional probability for Bayes Classifier.

Implementation of Dirichlet

- From the whole data, a train and test split are achieved with an 80-20 split, with a catch that each class is split on this basis. This change allowed us to get better performance on the dataset than the inbuilt Scikit-Learn MNB function.
- Priors are then obtained.
- To calculate Dirichlet

Accuracy Values

Dataset 1 - 20 Newsgroups Dataset

Type of Problem	MNB (Designed)	MNB (Scikit-Learn)	Dirichlet (Type 1)	Dirichlet (Type 2)
2 Classes - Similar classes (Atheism + Christianity)	86.94	86.94	55.0	55.0
2 Classes - Different classes (Baseball + Comp.Graphics)	94.92	94.92	51.27	51.27
2 Classes - Different classes (Sci.Med + Rec.Autos)	90.15	90.15	48.99	48.99
2 Classes - Misc + Other (Misc.forsale + Comp.windows.x)	95.14	95.14	54.48	54.48
6 Classes (Chosen as the main groups)	78.01	78.04	20.45	20.45
All 20 Classes	66.14	66.19	5.65	5.65

Dataset 2 - Fashion MNIST

Type of Problem	MNB (Designed)	MNB (Scikit-Learn)	Dirichlet (Type 1)	Dirichlet (Type 2)
2 Classes - Similar classes (T-shirt + Pullover) (0 + 2)	95.21	95.21	Unable to converge	51.29
2 Classes - Different classes (Sneakers + Boots) (7 + 9)	90.38	90.38	Unable to converge	47.92
2 Classes - Different classes (Sandals + Shirt) (5 + 6)	99.17	99.17	Unable to converge	49.75
2 Classes - Different classes (Trousers + Bag) (1 + 8)	97.83	97.83	Unable to converge	48.25
6 Classes (Randomly chosen) (0 + 1 + 5 + 8 + 9 + 3)	84.57	84.57	Unable to converge	16.46

All 10 Classes	65.16	65.16	Unable to converge	10.38

Observations

- The Multinomial Naive Bayes that we have designed has a performance that is highly comparable to that of the inbuilt function from Scikit-Learn. This is validation that for the given dataset, the classifier is performing to its capacity
- In the 2-class case with similar classes (in 20 Newsgroup dataset - Atheism and Christianity, having similar names like Jesus, god, religion, etc., and in Fashion MNIST case, Sandals and Sneakers, Shirt and Pullover, etc.), we can see that the classifier is confused by the similarity of features seen and thus there is a drop in accuracy
    - In the 2-class different classes case, we can see from the high accuracy score that the features are easily distinguishable
- Dirichlet distribution is not highly effective on the bag-of-words model that we are using, and this is seen in the compared values
- We used two methods of Dirichlet - The first employs an iterative method, and the second one uses an approximation of the MLE estimate of Dirichlet distribution (based on **Estimating a Dirichlet distribution, Thomas P. Minka, 2000**). This second method can give results when the first iterative method is incapable of giving results.
- The inbuilt function used to calculate Dirichilet parameter **α** uses iterative method, in case of fashion dataset it didn't converge.