# DASC-521
# Homework 02: Naive Bayes' Classifier

Ender Erkaya

October 2021

## 1 Introduction

In this homework, we aim to develop Naive Bayes classification of multivariate data for multiple classes, $K > 2$. We assume that classes are gaussian distributed and features are statistically independent. To estimate score functions related to posterior class probabilities, $p(y = c|x)$, we need to estimate class conditional probability densities, $p(x|y)$, and prior probabilites, $p(y)$, for each class, using Bayes Rule given below. Noting that, we derive the decision rule using only the numerator of posterior pdf, because $p(x)$ is same for each class, not dependent of class variable $y$, for each data point $x$.

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

where for statistically independent features:

$$p(x|y) = \prod_{k=1}^{D} p(x_k|y)$$

Since, conditional probabilities are modeled with Gaussian distribution, only mean and covariance can be used to characterize the distribution. Using Naive Bayes approach, we use the assumption that all features are independent, we can characterize the conditional probabilities by each features distribution. Hence, we need to estimate $\mu$ and $\sigma$ for each feature for each class. The class posterior probabilities are characterized by only using features distribution properties($\mu_{c,k}$ and $\sigma_{c,k}$) for each class. Then, we estimate prior class probabilities from the data using $\frac{N_C}{N}$, where $N_C$ is total number of data in class $c$ and $N$ is the total number of data. After estimating parameters for score functions, $g_c(x)$ for each class, we calculate the score functions for a given data $x_i$ and we predict the class $y_i$ using the following decision rule:

$$g_c(x_i) = \sum_{k=1}^{D} log(p(x_{i,k}|y = c) + log(p(y = c)$$

$$\hat{y_i} = \arg\max_c g_c(x_i)$$

## 2 Data for Training and Testing

The given data is inserted into data and label matrices. Feature space size is $D = 784$ and total data size $N = 35000$. The data is separated as training data and test data. We use $N_{train} = 30000$ data for training and $N_{test} = 5000$ data for test. After parameters are estimated using training data, we evaluate them on the test data.

## 3 Parameter Estimation

To estimate conditional proability density for each class, we use the assumption that features are statistically independent and gaussian distributed. Hence, we estimate corresponding Gaussian density parameters, $\{\mu_{c,k}, \sigma_{c,k}\}, k \in 1, .., D$, for each feature within each class. Sample mean estimates for each features within each classes are given in 1 and sample standard deviation estimates for each features in each classes are given in Figure 2:

```
3 print(sample_means)
```

```
[[254.99866667 254.98416667 254.85616667 ... 254.679      254.87816667
  254.95933333]
 [254.99733333 254.99733333 254.9965     ... 254.96883333 254.99216667
  254.98866667]
 [254.99933333 254.99933333 254.99233333 ... 251.52483333 254.4725
  254.97483333]
 [254.99666667 254.98983333 254.91416667 ... 252.39516667 254.44166667
  254.93666667]
 [254.999      254.98433333 254.93783333 ... 250.673      253.23333333
  254.79083333]]
```

Figure 1: Sample Mean Estimates

```
4 print(sample_devs)
```

```
[[ 0.09127736  0.25609108  1.31090756 ...  5.29826629  3.9117332
   1.93959091]
 [ 0.2065419   0.2065419   0.2163818  ...  1.04076669  0.47057267
   0.70062226]
 [ 0.05163547  0.04081939  0.16002465 ... 18.43665868  6.7881694
   1.1061344 ]
 [ 0.18436076  0.21617116  1.81046936 ... 15.67799977  6.34549162
   1.79971911]
 [ 0.04471018  0.64582342  3.03248555 ... 23.62576428 13.9167006
   4.4727787 ]]
```

Figure 2: Sample Standard Deviation Estimates for each features

Additionally, to calculate score functions we estimate class prior probabilities, $p(y = c)$, for each class $c$ (Figure 3).

```
3 print(class_priors)
[0.2 0.2 0.2 0.2 0.2]
```

Figure 3: Class Prior Estimates

# 4 Confusion Matrix for Training Data

We use the below decision rule as Maximum A Posteriori Estimator of $y$ to obtain the predictions for each observed data $x_i$, ie by picking the maximum one of $g_c$s or $softmax(g_c)$s:

$$\hat{y_i} = \arg\max_c g_c(x_i) = \arg\max_c softmax(g_c(x_i))$$

As a result, we obtain the following confusion matrix in Figure 4:

```
4 print(confusion_matrix)
```

| y_truth | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| y_pred | | | | | |
| 1 | 3685 | 49 | 4 | 679 | 6 |
| 2 | 1430 | 5667 | 1140 | 1380 | 532 |
| 3 | 508 | 208 | 4670 | 2948 | 893 |
| 4 | 234 | 60 | 123 | 687 | 180 |
| 5 | 143 | 16 | 63 | 306 | 4389 |

Figure 4: Confusion Matrix for Training Data

# 5 Confusion Matrix for Test Data

We obtain the following confusion matrix for Test Data in Figure 5:

# 6 Conclusion

In this lab, we have larger data set which has $D = 784$ features and $N = 35000$ data. To calculate class posterior probabilities, we use a strong assumption: Features are statistically independent from each other. Hence, we are able to estimate conditional probabilities $p(x|y) = \prod_{k=1}^{D} p(x_k|y)$ using probability

```
9 print(confusion_matrix)
```

| y_truth | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| y_pred | | | | | |
| 1 | 597 | 6 | 0 | 114 | 1 |
| 2 | 237 | 955 | 188 | 267 | 81 |
| 3 | 92 | 25 | 785 | 462 | 167 |
| 4 | 34 | 11 | 16 | 109 | 29 |
| 5 | 40 | 3 | 11 | 48 | 722 |

Figure 5: Confusion Matrix for Test Data

of each features, as multiplication of each feature gaussian probabilities. For the purpose, we first need to estimate parameters for each features in each class. Then, for each variable, conditional probabilities are calculated as sum of logarithm of feature probabilities and prior probabilities are estimated. Score functions are calculated using conditional probability and class prior probability for each class. We use max decision rule to predict the class at the end.