CS584 Assignment 2: Report

Jingyu Zhu A20311779

1. Problem Statement

In this assignment, I try to classify the test data based on the training data set. To materialize it, I apply the Gaussian discriminant analysis for three different cases, which are 2 classes with 1dimension, 2 classes with n dimension and k classes with n dimension. Given a data set, we can either use the generative learning algorithm, which aims to model each class and then compute the parameters, or use the discriminative learning algorithm, which tends to directly figure out the discriminant function.

2. Proposed Solution

Generative learning algorithm tries to model P(x|y), i.e., the probability of an example x belongs to class y (y can be 0 or 1). As Bayes Rule says, we have:

$$P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)}$$

Based on the equation above and the assumption that the priors are equal, we can derive our membership function:

$$g_i(x) = -\frac{(x-\mu)^2}{2\sigma^2}$$

After we get all the membership functions for all classes, we can put the testing data in the equation to classify it.

In a nutshell, we will go through the following procedures for the generative learning problems:

a. Choose the distribution model.
b. Estimate the distribution parameters.
c. Compute the discriminant functions and classify the examples.

3. Implementation Details

(1) Gaussian Distribution Analysis (GDA)

To begin with the Gaussian Distribution Analysis, first we should make an assumption that all features are continuous numeric data.

Since the data are Gaussian, we can easily estimate the mean, namely μ, and the covariance matrix ∑. Then we have our log-likelihood function defined as follow:

$$L(\theta;\mu,\Sigma) = \log\prod_{i=1}^{m} P(x^i \mid y^i)P(y^i)$$

By applying the Bayes Rule, we can have the following posterior function:

$$P(y = j \mid x) = \frac{P(x \mid y = j)P(y = j)}{\sum_{i=1}^{m} P(x \mid y = j)P(y = j)}$$

Observe the equation above, and we can find the denominators are same for classification problems and the only thing we need to consider is the numerator. Therefore we derive our final membership function from above:

$$g_i(x) \sim \log(P(x \mid y = i)) + \log(P(y = i))$$

To classify the data, we calculate:

$$\hat{y} = \underset{j}{argmax}\, g_j(x)$$

(2) Naïve Bayes

Typically, Naïve Bayes is used to classify data with discrete features. To model P(x|y), we make an assumption that all features are independent. The membership function is as follow:

$$g_i(x) = \prod_{i=1}^{n} P(x_i \mid y = 1)$$

If the features are in Bernoulli distribution and assume the distribution parameters are denoted by α(i), we have following equation:

$$p(x_i|y = 1) = (\alpha_i|y = 1)^{x_i}(1 - \alpha_i|y = 1)^{1-x_i}$$

After we get all α, we can easily classify the testing data by following:

$$g_1(x) = \prod_{i=1}^{n} (\alpha_i \mid y = 1)^{x_i} (1 - \alpha_i \mid y = 1)^{1-x_i} \alpha$$

Furthermore, to avoid the effect of probability 0, we can implement our algorithm with the aid of Laplace smoothing, which is simply as follow:

$$\alpha_{j|y=1} = \frac{\sum_{i=1}^{m} 1\left(y^{(i)} = 1\right) x_j^{(i)} + E}{\sum_{i=1}^{m} 1(y^{(i)} = 1) + 2E}$$

4. Results and discussion

(1) 1D 2-class Gaussian discriminant analysis

The confusion matrix and the performance of classifier are as follow:

```
> resultMatrix
     [,1] [,2]
[1,]   39    5
[2,]   11   45
> precisionOfClass0
[1] 0.8863636
> precisionOfClass1
[1] 0.8035714
> recallOfClass0
[1] 0.78
> recallOfClass1
[1] 0.9
> FMeasureOfClass0
[1] 0.8297872
> FMeasureOfClass1
[1] 0.8490566
> accuracy
[1] 0.84
```

As only one feature is used to model the distribution, we can see that the performance of the algorithm is not quite satisfactory.

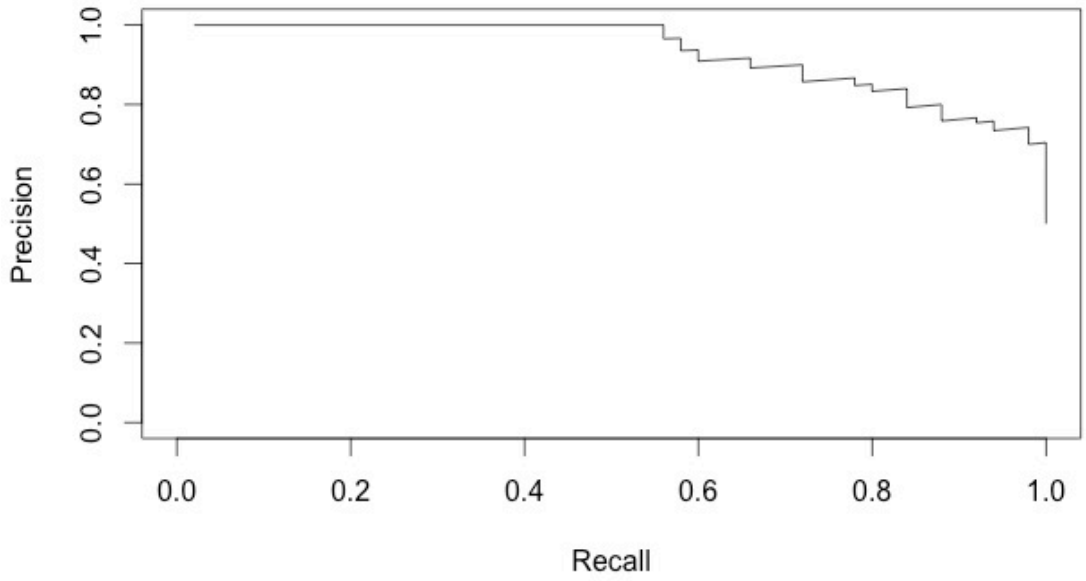
(2) nD 2-class Gaussian discriminant analysis

The confusion matrix and the performance of classifier are as follow:

```
     [,1] [,2]
[1,]   50    0
[2,]    0   50
> precisionOfClass0
[1] 1
> precisionOfClass1
[1] 1
> recallOfClass0
[1] 1
> recallOfClass1
[1] 1
> FMeasureOfClass0
[1] 1
> FMeasureOfClass1
[1] 1
> accuracy
[1] 1
```

We can see from the snapshot above that the algorithm classified all the testing data set correctly because all features had been used for training.

Then I plot the precision-recall curve and the graph is as follow (I added some noise to the graph):



(3) n-D k-class Gaussian discriminant analysis

The confusion matrix and the performance of classifier are as follow:

```
> resultMatrix
     [,1] [,2] [,3]
[1,]   50    0    0
[2,]    0   48    2
[3,]    0    1   49
> precisionOfClass0
[1] 1
> precisionOfClass1
[1] 0.96
> precisionOfClass2
[1] 0.98
> recallOfClass0
[1] 1
> recallOfClass1
[1] 0.9795918
> recallOfClass2
[1] 0.9607843
> FMeasureOfClass0
[1] 1
> FMeasureOfClass1
[1] 0.969697
> FMeasureOfClass2
[1] 0.970297
> accuracy
[1] 0.98
```

We can see from the graph that the algorithm performs pretty well and all the parameters are approximately 100%.

(4) Naive Bayes with Bernoulli features
The confusion matrix and the performance of classifier are as follow:

```
> resultMatrix
     [,1] [,2]
[1,]   50    0
[2,]    0   50
> precisionOfClass0
[1] 1
> precisionOfClass1
[1] 1
> recallOfClass0
[1] 1
> recallOfClass1
[1] 1
> FMeasureOfClass0
[1] 1
> FMeasureOfClass1
[1] 1
> accuracy
[1] 1
```

We can see from the graph that the generative learning algorithm, which is in Naive Bayes with Bernoulli features, classifies the data without any mistake.

(5) Naive Bayes with Binomial features

The confusion matrix and the performance of classifier are as follow:

```
> resultMatrix
     [,1] [,2]
[1,]   49    0
[2,]    1   50
> precisionOfClass0
[1] 1
> precisionOfClass1
[1] 0.9803922
> recallOfClass0
[1] 0.98
> recallOfClass1
[1] 1
> FMeasureOfClass0
[1] 0.989899
> FMeasureOfClass1
[1] 0.990099
> accuracy
[1] 0.99
```

We can see from above that the algorithm worked properly with minor errors.

At last, the process of deriving the parameter estimate equations for Naive Bayes with Binomial features via maximum likelihood is as follow.

First, we have our original probability equation of binomial:

$$P(x_j \mid y=1) = \begin{pmatrix} p(j) \\ x_j \end{pmatrix} (\alpha_{j|y=1})^{x_j} (1-\alpha_{j|y=1})^{p(j)-x_j}$$

Second, we join all the probabilities, compute its logarithm and we can get:

$$l_1(\theta) = \sum_{i=1}^{m}\sum_{j=1}^{n}(\log\begin{pmatrix} p(i) \\ x_j^{(i)} \end{pmatrix} + x_j^{(i)}\log(\alpha_{j|y=1}) + (p(i) - x_j^{(i)})\log(1-\alpha_{j|y=1}))$$

Third, calculate the derivative of the equation above and make the derivative equal to 0:

$$\frac{\partial l_1(\theta)}{\partial \alpha_{p|y=1}} = \sum_{i=1}^{m} x_p^{(i)} \cdot \frac{1}{\alpha_{p|y=1}} - \sum_{i=1}^{m}(p(i) - x_p^{(i)}) \cdot \frac{1}{1-\alpha_{p|y=1}} = 0$$

Finally, as we have:

$$a = \sum_{i=1}^{m} x_p^{(i)} \ , \ b = \sum_{i=1}^{m}(p(i) - x_p^{(i)})$$

Combined it with the foregoing equation, we can have the α as following:

$$\alpha_{p|y=1} = \frac{a}{a+b} = \frac{\sum_{i=1}^{m} x_p^{(i)}}{\sum_{i=1}^{m} p(i)}$$