# EE5907/EE5027 Pattern Recognition Programming Assignment CA1 Report

Shuo SUN A0162488U
email shuo.sun@nus.edu.sg

16 September 2021

**Abstract**

This report documents the experiment results and observations from EE5907/EE5027 Pattern Recognition Programming Assignment 1. Four widely-used classification algorithms, including: Beta-Binomial Naïve Bayes Classifier, Gaussian Naïve Bayes Classifier, Logistic Regression Classifier, and K-Nearest Neighbours Classifier, were implemented and evaluate on the same Spam Email Dataset. Several key observations were identified from the experiment results. The original project code has been open-sourced and can be found on GitHub https://github.com/SS47816/email-spam

## Dataset Overview

The dataset used in this programming assignment was [1], which contains 4601 emails including 1813 (39.4%) determined as spams. For each email, the dataset provides 58 attributes, including 1 nominal class label denoting whether this email is a spam (1) or not (0). The rest of the 57 attributes are descriptive features either ranging from 0 to 1 or binary (0 or 1), representing different aspects of the email, such as word/character frequency, capital run length. For the purpose of this project, 3065 emails were randomly selected as the training set and the rest 1536 emails were used as the test set. Throughout the experiment, the same training and test sets were used on all four algorithms for performance-evaluation purpose.

### Data Pre-Processing

Before directly train classifiers on the data, several pre-processing techniques were applied on the data to ensure compatibility and effectiveness. For simplicity, all 57 features from the data were applied to the same technique at one time together, and complex pre-processing techniques were intentionally avoided so that the comparison results among all four algorithms would be less likely affected by the type of pre-processing technique used.

(a) Binarization: A binarization was performed on the data (both training and test sets) to suit the input requirement of the Beta-Binomial Naïve Bayes Classifier. Each feature $x_{i,j}$

was set to 1 if its value was greater than 0, 0 otherwise:

$$x_{i,j}^{binary} = \begin{cases} 1 & \text{if } x_{i,j} > 0 \\ 0 & \text{else} \end{cases} \tag{1}$$

(b) Log-transform: For all three other classifiers, a log-transform was performed on their input data. Each feature was transformed using the following equation:

$$x_{i,j}^{log} = log(x_{i,j} + 0.1) \tag{2}$$

# 1 Beta-Binomial Naïve Bayes Classifier

## 1.1 Implementation

Beta-Binomial Naïve Bayes Classifier is a special type of Naïve Bayes Classifier that assumes its inputs are binary features and uses a Beta-Binomial Model to fit the data. The aim is that for a given training data $D = \{x_1, \ldots, x_N\}$, predict the target class label $\tilde{y}_i$ for each feature vector $\tilde{x}_i$. For generative classifiers, the process can be described as:

$$p(x, y|\boldsymbol{\theta}) = p(y|\boldsymbol{\theta})p(x|y, \boldsymbol{\theta}) \tag{3}$$

By splitting $\boldsymbol{\theta}$ into two symbols: $\boldsymbol{\theta} = \{\lambda, \eta\}$:

$$\begin{aligned} classprior : p(y|\boldsymbol{\theta}) = p(y|\lambda) \\ featurelikelihood : p(x|y, \boldsymbol{\theta}) = p(x|y, \eta) \end{aligned} \tag{4}$$

Naïve Bayes Classifier assumes features are conditionally independent, thus the original equation becomes:

$$p(x|y = c, \eta) = \prod_{j=1}^{D} p(x_j|y = c, \eta) = \prod_{j=1}^{D} p(x_j|\eta_{jc}) \tag{5}$$

where $\eta_{jc}$ is the parameter at $j$-th feature assuming data point from class $c$. In addition, the posterior distribution becomes:

$$p(y|x, \boldsymbol{\theta}) \propto p(y, \lambda)p(x|y, \eta) \tag{6}$$

By applying Maximum Likelihood estimation on $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}_{ML} = argmax_{\boldsymbol{\theta}} p(D|\boldsymbol{\theta}) \tag{7}$$

$$p(\tilde{y}|\tilde{x}, \boldsymbol{\theta}) \propto p(\tilde{y}, \lambda_{ML})p(\tilde{x}|\tilde{y}, \eta_{ML}) \tag{8}$$

The propability of an email being a spam ($\tilde{y} = 1$) or not ($\tilde{y} = 0$) can be expressed as:

$$\log p(\tilde{y} = c|\tilde{x}, D) \propto \log p(\tilde{y} = c|y_{1:N}) + \sum_{j=1}^{D} \log p(\tilde{x}_j|x_{i\in c,j}, \tilde{y} = c) \tag{9}$$

Using a typical Beta-Binomial model on the feature likelihood gives the following probability distribution:

$$p(\tilde{x}_j|x_{i\in c,j}, \tilde{y} = c) = Beta(a, b) = \frac{N_1 + a}{N + a + b} \tag{10}$$

where $N$ the total number of emails whose $j$-th feature satisfies $x_j = \tilde{x}_j$, and $N_1$ is the number of emails within $N$, that satisfies $y = c$. For simplicity, in this assignment, it is assumed that: $a = b = \alpha = \{0, 0.5, 1.0, 1.5, 2.0 \ldots, 100\}$. The error rate is simply computed as the ratio of emails classified wrongly:

$$error\ rate = \frac{N_{FP} + N_{FN}}{N} \tag{11}$$

## 1.2 Experiment Result

The plot of error rates versus $\alpha$ on the training and test sets is shown in Figure 1

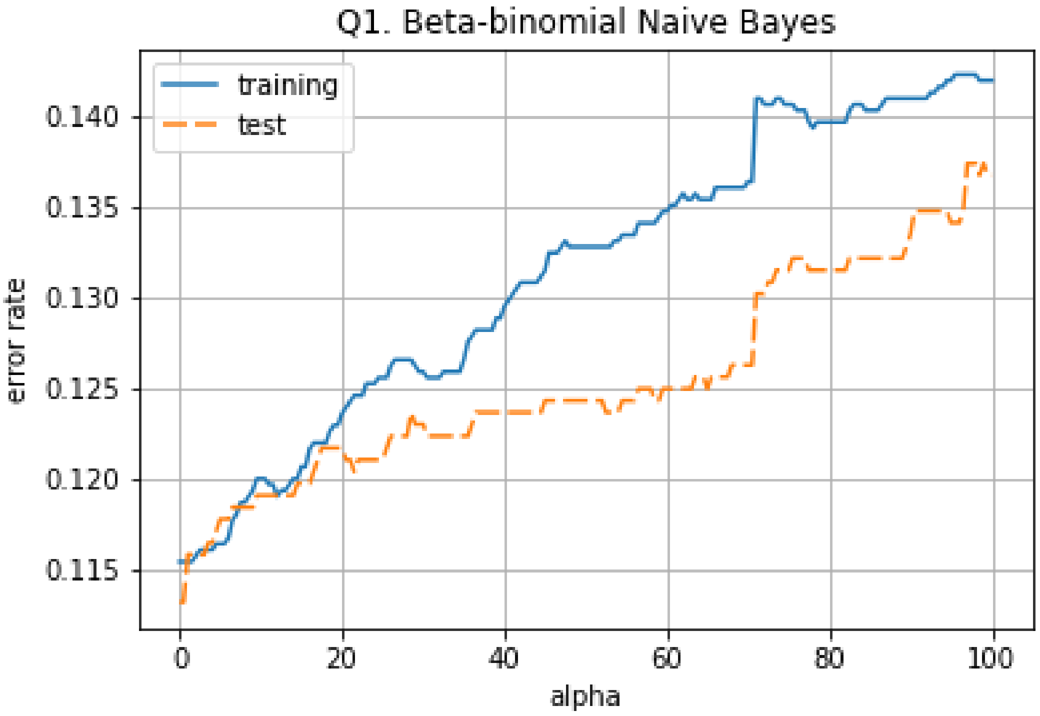

Figure 1: Beta-Binomial Naïve Bayes training and test error rate versus alpha

From Figure 1, we can observe that as the value of $\alpha$ increases from 0.0 to 100, the error rates on both training and test sets increase gradually. In this scenario, the most optimal alpha occurs at point $\alpha = 0.0$.

Three pairs of key points in Figure 1 where $\alpha = \{1.0, 10, 100\}$ were identified and shown in Table 1

Table 1: Beta-Binomial training and test error rates versus $\alpha$

| $\alpha$ | 1 | 10 | 100 |
|---|---|---|---|
| training | 0.1155 | 0.1201 | 0.1419 |
| test | 0.1159 | 0.1191 | 0.1357 |

# 2 Gaussian Naïve Bayes Classifier

## 2.1 Implementation

Similar to Section 1, Gaussian Naïve Bayes Classifier is another type of Naïve Bayes Classifier that uses a Gaussian Model to fit the data:

$$Gaussian(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{12}$$

where $\mu$ and $\sigma^2$ can be estimated using Maximum Likelihood:

$$(\mu, \sigma^2) = argmax_{\mu,\sigma^2} \sum_{n=1}^{N} \Big[ -\frac{(x-\mu)^2}{2\sigma^2} - \log\sqrt{2\pi\sigma^2} \Big] \tag{13}$$

thus:

$$\begin{aligned} \tilde{\mu} &= \frac{1}{N}\sum_{n=1}^{N} x_n \\ \tilde{\sigma}^2 &= \frac{1}{N}\sum_{n=1}^{N} (x_n - \tilde{\mu})^2 \end{aligned} \tag{14}$$

By plugging in the derived parameters $\tilde{\mu}$ and $\tilde{\sigma}^2$ into Equation (9) as the feature likelihood, it can be showed that:

$$\log p(\tilde{y} = c|\tilde{x}, D) \propto \log p(\tilde{y}|\lambda_{ML}) + \sum_{j=1}^{D} \log p(\tilde{x}_j|\tilde{\mu}_c, \tilde{\sigma}_c) \tag{15}$$

The definition of error rate is the same as described in Equation (11).

## 2.2 Experiment Result

The error rates computed on the training and test sets are show in Table 2.

Table 2: Gaussian Naïve Bayes training and test error rates

| | training | test |
|---|---|---|
| error rate | 0.1657 | 0.1602 |

# 3 Logistic Regression

## 3.1 Implementation

Logistic Regression Classifier is one type of widely used discriminate classifier. In a binary case, Bernoulli distribution can be employed to model the probability distribution:

$$Ber(y|\mu(x,w)) = Ber(y|sigm(w^T x)) \tag{16}$$

$$sigm(x) = \frac{1}{1+e^{-x}} \tag{17}$$

Thus:

$$\begin{aligned} p(y=1|x) &= \frac{1}{1+e^{-w^T x}} \\ p(y=0|x) &= 1 - \frac{1}{1+e^{-w^T x}} \end{aligned} \tag{18}$$

To derive the optimal value of parameters $w$, the model was be trained on the training set to satisfy:

$$\tilde{w} = argmin_w - \sum_{i=1}^{N} \log p(y_i|x_i,w) = argmin_w NLL(w) \tag{19}$$

Newton's Method can be employed for the optimization process. The first and second derivatives of the function $f(\theta)$ at each iteration can be expressed as:

$$\begin{aligned} g(w) &= \frac{d}{dw} NLL(w) &&= X^T(\mu - y) \\ H(w) &= \frac{d}{dw} g(w)^T &&= X^T S X \end{aligned} \tag{20}$$

By adding a bias and an $l_2$ regularization to the model, the above equations become:

$$\begin{aligned} g_{reg}(\boldsymbol{w}) &= g(\boldsymbol{w}) + \lambda \boldsymbol{w} \\ H_{reg}(\boldsymbol{w}) &= H(\boldsymbol{w}) + \lambda I \end{aligned} \tag{21}$$

where $\boldsymbol{w}$ is a new vector of the shape $(D+1) \times 1$, formed by concatenate one bias term to the front of $w$, and $I$ is a $(D+1) \times (D+1)$ identity matrix. For every step in the Newton's Method, $w$ is updated by:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta_k d_k \tag{22}$$

$$\eta_k = -H_k^T g_k \tag{23}$$

where $\eta_k$ used in this assignment is a static hyper-parameter learning rate.

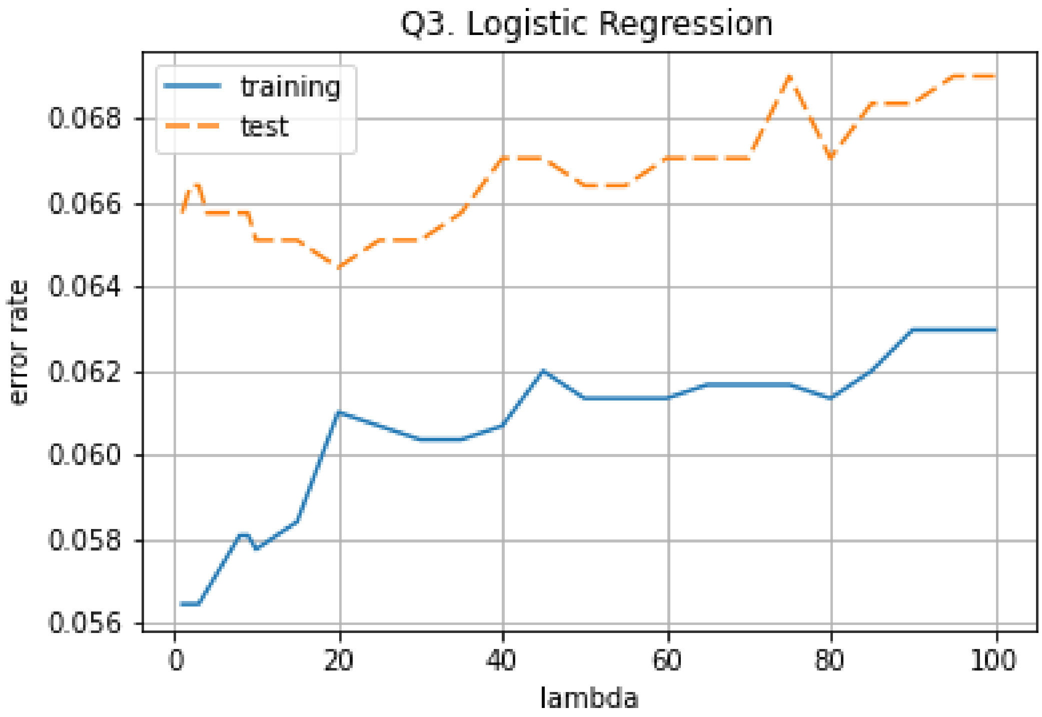

Figure 2: Logistic Regression training and test error rate versus $\lambda$

### 3.2 Experiment Result

The plot of error rates versus $\lambda$ on the training and test sets is shown in Figure 2

From Figure 2, we can observe that as the value of $\lambda$ increases from 0.0 to 20, the error rates on the training set increases while decreases on the test sets. Then when $\lambda$ enters the range from 20 to 100, both curves increases gradually.

Three pairs of key points in Figure 1 where $\lambda = \{1.0, 10, 100\}$ were identified and shown in Table 3

Table 3: Logistic Regression training and test error rates versus $\lambda$

| $\lambda$ | 1 | 10 | 100 |
|---|---|---|---|
| training | 0.0564 | 0.0577 | 0.0630 |
| test | 0.0658 | 0.0651 | 0.0690 |

# 4 K-Nearest Neighbours

## 4.1 Implementation

K-Nearest Neighbours Classifier is a type of non-parametric classifier. In the K-Nearest Neighbours Classifier, $K$ number of nearest neighbours were identified and used to estimate the class label of the target feature vector. In this assignment, euclidean distance was used to compute the distance between different feature vectors:

$$dist = \sqrt{\sum_{j=1}^{D} |a_j - b_j|^2} \tag{24}$$

After the $K$ number of nearest neighbours were found based on the above criterion, the joint probability can be computed as:

$$p(y = c|x) = \frac{k_c}{K} \tag{25}$$

where $k_c$ is the number of neighbours that of class $c$ among $K$ nearest neighbours.

## 4.2 Experiment Result

The plot of error rates versus $K$ on the training and test sets is shown in Figure 3.

From Figure 3, we can observe that as the value of $K$ increases from 0.0 to 100, the error rates on the training increase sharply first, then gradually tends to flatten. However on the test set, the error rate tends to be unstable for at first then gradually form a increasing trend after $K = 20$.

Three pairs of key points in Figure 3 where $K = \{1.0, 10, 100\}$ were identified and shown in Table 4.

Table 4: KNN training and test error rates versus $K$

| $K$ | 1 | 10 | 100 |
|---|---|---|---|
| training | 0.0007 | 0.0532 | 0.0917 |
| test | 0.0697 | 0.0677 | 0.1003 |

# 5 Survey

The total time spent on the project is roughly about $30 - 40hrs$. Checking lecture notes and actually coding the algorithms took about $10 + hrs$, while testing and debugging took another $10hrs$ hours. Finally writing the report and organising everything took another $10 + hrs$. In general, this programming assignment is not very difficult since we don't need to spend time adding many tricks to make the results very competitive. However it is still quite time-consuming to test and debug the algorithms, since I am quite worried if I make any careless mistakes somewhere without knowing. Typing equations in the report is another time-killer.

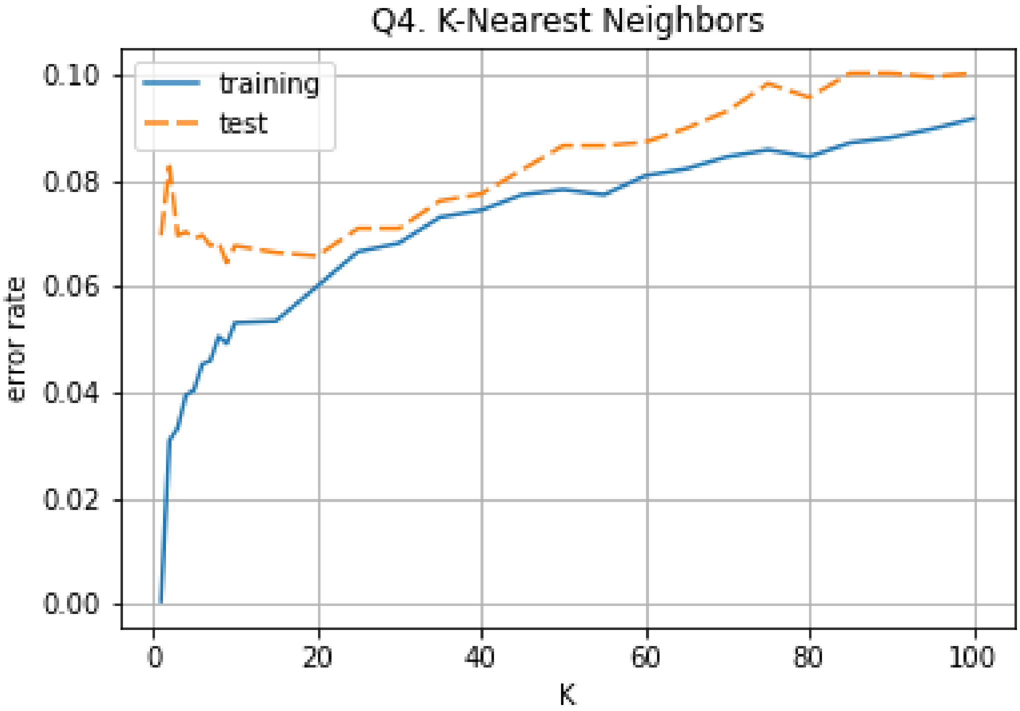

Figure 3: Beta-Binomial Naïve Bayes training and test error rate versus $K$

## References

[1] D. Dua and C. Graff, “UCI machine learning repository,” 2017. [Online]. Available: http://archive.ics.uci.edu/ml