EE5709R Programming Assignment Report

CA1

Name: SHAO GUOLIANG

Matric NO.: A0152113Y

1. Task description

The task is to use four different methods to predict whether an email is a spam email or non-spam email based on the training data set. The four methods are 1) Beta-Bernoulli Naïve Bayes; 2) Gaussian Naïve Bayes; 3) Logistic Regression; and 4) K-Nearest Neighbors. The whole process of learning consists of three parts: data pre-processing, learning algorithm, and test prediction. By using different data pre-processing and different training method, we can compare the prediction error rate of each method.

This report is organized as follows. Part2 will present three different data pre-processing methods; Part3 will present the training results of Beta-Bernoulli Naïve Bayes; Part4 will present the training results of Gaussian Naïve Bayes; Part5 will present the training results of Logistic Regression; Part6 will present training results of KNN method. Part7 will present the survey required for this task.

2. Data pre-processing

(1) Data description:

The data set is a “SPAM E-mail Database” created by Jaap Suermondt Hewlett-Packard Labs in 1999. The past usage shows the misclassification error is around 7%. There are 4601 data points including 1813 of spam data, and the number of data features is 57. The data set has training data set and testing data set. In this assignment each data pre-processing method is performed on training data set and testing data set independently.

(2) Binarization:

This method binarize features using $\mathbb{I}(x_{ij} > 0)$. In this task, I choose features that are equals to 0 remain as 0, and set features that are greater than 0 to 1.

(3) Log-transform:

This method use the log scale to transform each feature using $\log(x_{ij} + 1)$.

(4) Z-normalization:

This method standard each column of feature so that they have 0 mean and unit variance.

3. Beta-Bernoulli Naïve Bayes

(1) Requirements:

Use binarization method to pre-process the data; fit a Beta-bernoulli Naïve Bayes classifier after performing the data processing section; the class label priori can be estimated using ML; tuning the hyper parameter alpha in {0,0.5,1,1.5,…,100} and assuming the Beta distribution has two equal parameters.

(2) Training process:

For each feature, use MAP method to fit the Beta-Bernoulli model. So the multi-dimension Beta-Bernoulli model has a vector parameter theta.

The posterior probability is $P(\theta|D) = Beta(\theta|N_1 + \alpha, N_0 + \alpha)$, based on this model we can calculate the MAP estimation of theta

$$\hat{\theta}_{MAP} = \frac{N_1 + \alpha - 1}{N + 2 * \alpha - 2}$$

where denotes the number of data points labelled as spam (y = 1) or non-spam (y = 0) for each feature in training data set, denotes the total number of training data set, denotes the hyper parameter of Beat function.

After acquiring the vector parameter, we can calculate the probability of the data point being labelled as spam (y=1) using

$$P(y_i = 1|\theta) \propto P(y_i = 1|\theta) \prod_{j=1}^{D} P(x_{ij}|\theta, y_i = 1)$$

and data point being labelled as non-spam(y=0) using

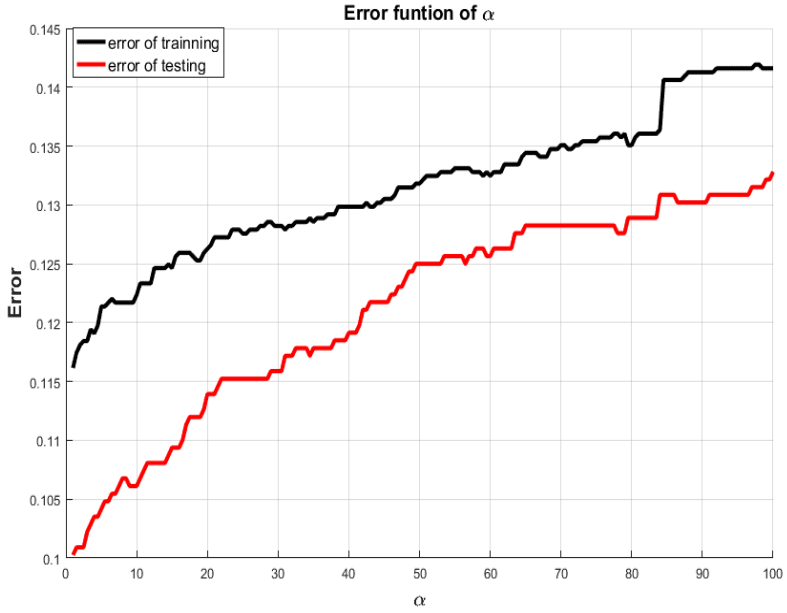
$$P(y_i = 0|\theta) \propto P(y_i = 0|\theta) \prod_{j=1}^{D} P(x_{ij}|\theta, y_i = 0)$$

where x_{ij} is the j-th feature of i-th data point.

The decision boundary is: if $P(y_i = 1|\theta)$ is greater than $P(y_i = 0|\theta)$, the data point is labelled as spam(y = 1), otherwise, labelled as non-spam(y = 0).

(3) Results:

A. The error rate vs. hyper parameter alpha.



B. The error rate for alpha = 1, 10, 100.

Alpha	1	10	100
Training	0.1162	0.1223	0.1416
Testing	0.1009	0.1061	0.1322

(4) Comments:

As the hyper parameter alpha changes, both the training error and the testing error increases. According to the Bate function, when a = 0 and b = 0, the probability of theta follows the uniform distribution. Besides, the training error is always greater than testing error.

4. Gaussian Model Naïve Bayes

(1) Requirement:

Use the log transform and z-normalization method to pre-process the data set; fit a Gaussian naïve Bayes classifier, use ML estimate the class conditional mean and variance for each feature; use some strategies to prevent over-fitting.

(2) Training process:

First use maximum likelihood to estimate the mean and variance of each feature for each class. Then use both Quadratic Discriminant Analysis and Linear Discriminant Analysis to set the decision boundary to classify the test data.

For Quadratic Discriminant Analysis, the probability of being labelled as *c*-th class

$$P(y = c|x, \theta) = \pi_c \frac{1}{(2\pi)^{D/2}|\Sigma_c|^{1/2}} \exp(-\frac{1}{2}(x - \mu_c)^T {\Sigma_c}^{-1}(x - \mu_c))$$

the decision boundary is: if $P(y = 1|x, \theta) > P(y = 0|x, \theta)$, then the data point is labelled as spam(y = 1), other wise, labelled as non-spam(y = 0).

For Linear Discriminant Analysis, based on QDA, we assume the covariance matrix is the same across classes. Then we can derive a decision boundary

$$class\_y = sigm\big(\omega^T(x - x_0)\big)$$

where

$$\omega = \Sigma^{-1}(\mu_1 - \mu_0), \mathrm{x}_0 = \frac{1}{2}(\mu_1 + \mu_0) - (\mu_1 - \mu_0)\frac{\log(\pi_1/\pi_0)}{(\mu_1 - \mu_0)^T\Sigma^{-1}(\mu_1 - \mu_0)}$$

To prevent over-fitting, we can use possible solutions as follows:

A. Use Naïve Bayes assumption, each feature is independent with other feature, so the covariance of each class is diagonal (this is required for the task);
B. Use LDA assumption, assumes that the covariance matrix is the same across classes;
C. Use factorized conjugate priors and use MAP or posterior prediction estimation.

(3) Results:

In this task, I tried both QDA and LDA under the Naïve Bayes assumption. The results are showed in the table below.

Pre-processing	QDA		LDA	
	Training	Testing	Training	Testing
Log	0.651550	0.664063	0.085155	0.074219
Z-normalization	0.397064	0.395182	0.148124	0.131510

(4) Comments:

As shown in the results table, LDA method has a much lower prediction error than QDA method, which means Naïve Bayes assumption cannot fully prevent the over-fitting problem. Besides, when using LDA method training, using log transform data pre-processing shows a lower prediction error rates.

The results in this part show that different training method and different data pre-processing method can have an influence on the accuracy of the model.

5. Logistic Regression

(1) Requirement:

Use three different data pre-processing method; fit a logistic regression model; use regularization, for each regularization parameter value $\lambda = \{1,2,\dots,9,10,15,\dots,100\}$

(2) Training process:

Logistic Regression is a discriminate classifier. Use sigmoid function to express the posterior probability

$$P(y=1|x) = \frac{1}{1+e^{-\omega^T x}}, P(y=0|x) = \frac{1}{1+e^{\omega^T x}}$$

Then use the negative log likelihood to generate the cost function:

$$NLL(\omega) = -\sum_{i=1}^{N} \log\big(p(y_i|x_i,\omega)\big) = -\sum_{i=1}^{N} y_i \log\mu + (1-y_i)\log(1-\mu)$$

In order to use regularization, we add a regularization term at the end of NLL:

$$NLL_{reg}(\omega) = NLL(\omega) + \frac{1}{2}\lambda\omega^T\omega$$

And then use the Newton method to minimize this convex cost function. The main steps of Newton method are shown as follows:

A. Initialize the parameter vector
B. For k = 1,2,… until convergence do
C. Calculate the first order gradient of cost function $g = x^T(\mu - y)$
D. Calculate the second order gradient of cost function , where $H = x^T S x$
E. Solve to find the direction of gradient descent $d_k = -H^{-1}g$

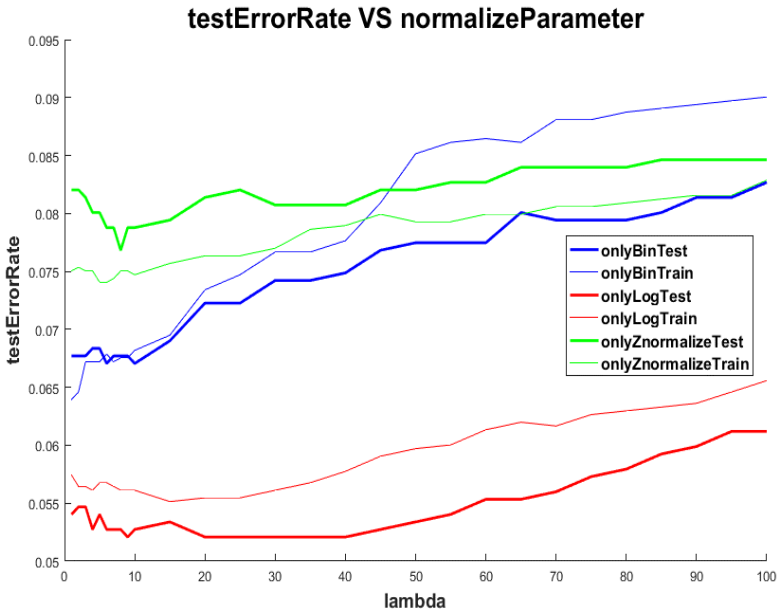
F. Use line search to find the step-size η_k along d_k, in this task, we set the step size as 1

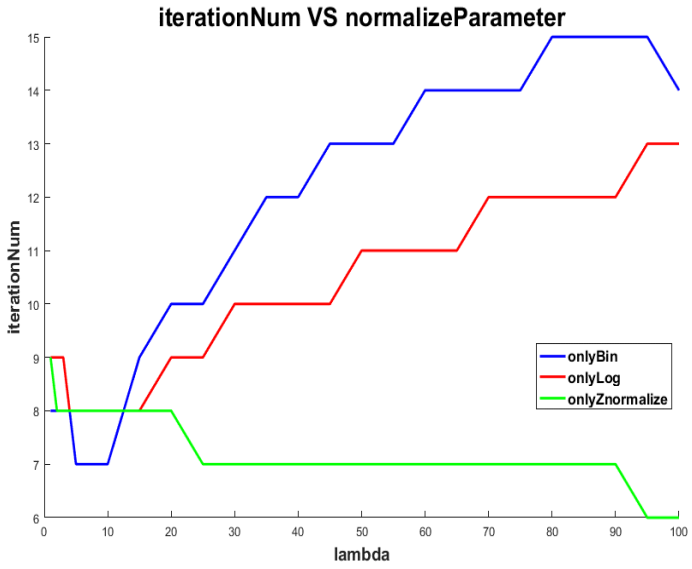
G. Update the parameter using $\omega_{k+1} = \omega_k + \eta_k d_k$

As the convergence error is related to the scale of cost function, in this task I use the $\eta_k d_k / NLL(w_{k+1})$ as the convergence error, when the convergence error reaches to 1e-6, we can believe we find the local optimal parameter.

After finding the optimal parameter ω, we use the sigmoid function again to predict the label of each data point using $sigm(\omega^T x)$.

(3) Results:





Error Rate vs. lambda table

parameter	Bin_train	Bin_test	Log_train	Log_test	Znorm_train	Znorm_test
$\lambda = 1$	0.06395	0.06771	0.05742	0.05404	0.07504	0.08203
$\lambda = 10$	0.06819	0.06706	0.05612	0.05273	0.07471	0.07878
$\lambda = 100$	0.09005	0.08268	0.06558	0.0612	0.08287	0.08464

(4) Comments:

Error rate change as lambda change:

As we can see in the graph, when lambda = 10, the test error rate is the lowest under all the three data pre-processing method. Nearly all the training and testing error rate decrease first and begin to increase when lambda is around 10.

Error rate between different preprocessing strategies:

As the graph shows, the log transform pre-processing method has the lowest training and testing error rate. Besides, when lambda is small, binarization method has a lower error rate than Z-normalization method, however, when lambda is large, the error rate of binarization is almost the same with Z-normalization.

Iteration number between different preprocessing strategies:

Apart from the test error, I plot the iteration number of Newton training for different methods. The graph shows when lambda is small, the iteration number is almost the same within the three methods, but when lambda is large, the iteration number of binirization and log transform start to increase while the Z-normalization method starts to decrease.

6. K-Nearest Neighbors

(1) Requirement :

Use three different data pre-processing method; fit a KNN classifier; for each regularization parameter value $K = \{1,2,\dots,9,10,15,\dots,100\}$; for continueous data use the eculidean distance, for binarized data use the hamming distance.

(2) Training process:

In this task, we are not asked to do cross-validation test, so I directly use the test data to do the classification. KNN is a non-parameter method to perform classification.

By calculating the distance between one test data point and all the training data point, we select K number of nearest data point compared to the test data point in the training data set. Thus we can calculate the probability of this test data point to be labelled as spam(y = 1) by the following formula:

$$P(y = c) = \frac{K_c(x)}{K * V_c}$$

where $K_c(x)$ is the number of training data that labelled as c, K is the number of K-nearest neighbors, V_c is the window for capture K data points. In this task, V_c is defined

as the longest distance between the test data point and the captured K training data points.

The decision boundary is: if $P(y = 1) > P(y = 0)$, then label the test data point as spam(y = 1), otherwise label as non-spam(y = 0).

(3) Results:

A. The graph of error rate vs. NO. of K

B. The table of error rate vs. K = 1,10,100

parameter	Bin_train	Bin_test	Log_train	Log_test	Znorm_train	Znorm_test
$K = 1$	0.08333	0.07374	0.06264	0.07227	0.0956	0.09049
$K = 10$	0.08138	0.07667	0.06297	0.05664	0.09494	0.09505
$K = 100$	0.1074	0.1191	0.09331	0.08919	0.1338	0.1322

(4) Comments:

The error rate change as K changes:

Nearly all the error rate of three different pre-processing strategies decrease at first and then begin to increase after K is over 10. Because when calculating the training error rate, we skip that training data point, so when we capture the 1 nearest neighbor of this data point (when K = 1), the prediction label will fully depend on this nearest neighbor. However, the nearest neighbor may not be in the same class as this training data point. As a result, the training error rate is not 0 when K = 1.

The observation about the error rate of different strategies:

As the graph shows, the log transform preprocessing strategy has the lowest training and testing error rate, on the other hand, the Z-normalize strategy has the highest training and testing error rate.

7. Survey:

Task	Time used (hours)
Q1. Naïve Bayes	4
Q2. Gaussian Model	3
Q3. Logistic Regression	6
Q4. KNN	5