CA1 Report

Name: GUO ZONGYI   ID: A0206639Y

Q1: Beta-binomial Naive Bayes

Main Idea:

In lecture 3, we have learnt that if the feature is subject to Beta(a,b) distribution, the posterior predictive probability should be:

$$p(\tilde{x}=1|D)=\frac{N_1+a}{N+a+b}$$

In the formula, N should be the number of occurrence in particular classes, and N_1 should be the number of 1 in this particular class.

The prior should be:

$$\lambda^{ML}=\frac{\max(N_0,N_1)}{N}$$

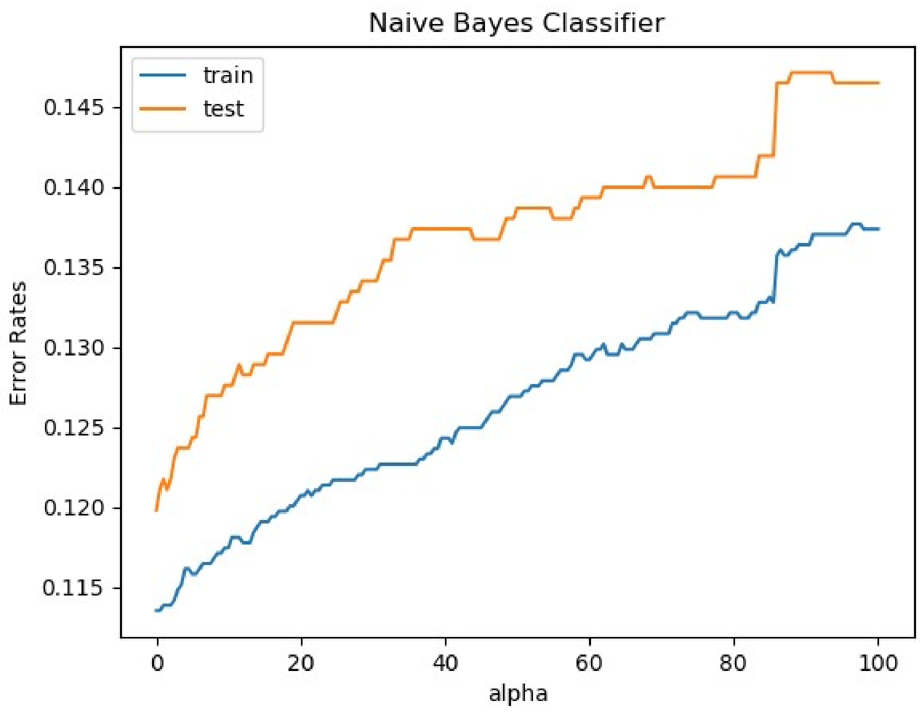
Therefore, in this question, every feature should is subject to $\mathrm{Beta}(\alpha,\alpha)$, so a and b, in above formula, both equal to α.

In training phase, N and N_1 should be figured out, which should be different considering different classes and features.

In test phase, both training and test accuracy should be calculated, by plugging in prior and parameters, obtained from training phase.

(The algorithm is shown in '*2-1 Beta-Binomial NB.py*')

(1)Plots of training and test error rates versus α:



(2) What do you observe about the training and test errors as α change?

When α increases, the error rates also increases, which means the accuracy is decreasing. Therefore, lower α is preferred.

(3) Training and testing error rates for α = 1, 10 and 100

When α = 1 :

Training error is 11.39%, test error is 12.17%.

When α = 10:

Training error is 11.75%, test error is 12.76%.

When α = 100:

Training error is 13.74%, test error is 14.65%.

Q2: Gaussian Naive Bayes

Main Idea:

In lecture 3, we learnt univariate Gaussian Distribution. In this problem, every feature is subject to Gaussian Distribution, but they have different mean and variance. The probability of Gaussian Distribution is:

$$\mathrm{p}(x) = N(x \mid \mu, \sigma^2)$$

The prior should be:

$$\lambda^{ML} = \frac{\max(N_0, N_1)}{N}$$

In training phase, mean and variance of every feature should be calculated in following formulas:

$$\hat{\mu} = \frac{1}{N}\sum_{\mathrm{n}=1}^{N} x_n$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum (x_n - \hat{\mu})^2$$

In test phase, calculate the probability of different classes and choose the bigger one to be the predicted class, using the parameters obtained from training phase.

(The algorithm is shown in '*2-2 Gausian NB.py*')

(1) Training and testing error rates for the log-transformed data.

Training error rate is 16.67%.

Test error rate is 16.47% .

Q3. Logistic regression

Main Idea:

In lecture 4, we have learnt Logistic Regression and some methods to optimize Logistic Regression parameters.

In Logistic Regression, we should minimize Negative Log Likelihood to obtain optimized weight. Negative Log Likelihood should be:

$$NLL(w) = -\sum_{i=1}^{N} [y_i \log \mu_i + (1-y_i)\log(1-\mu_i)]$$

$$\mu_\mathrm{i} = \frac{1}{1+\exp(-w^T x_i)}$$

Newton's Method should be used to optimize the w, in the formula.

$$\mathrm{w}_{k+1} = w_k - H_k^{-1} g_k$$

$$H = X^T S X$$

$$g = X^T(\mu - y)$$

In order to prevent overfitting, l_2 regularization should be applied. The regularized loss function, g and H are shown below:

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

$$\mathrm{g}_{reg}(w) = g(w) + \lambda w$$

$$H_\mathrm{reg}(w) = H(w) + \lambda I$$

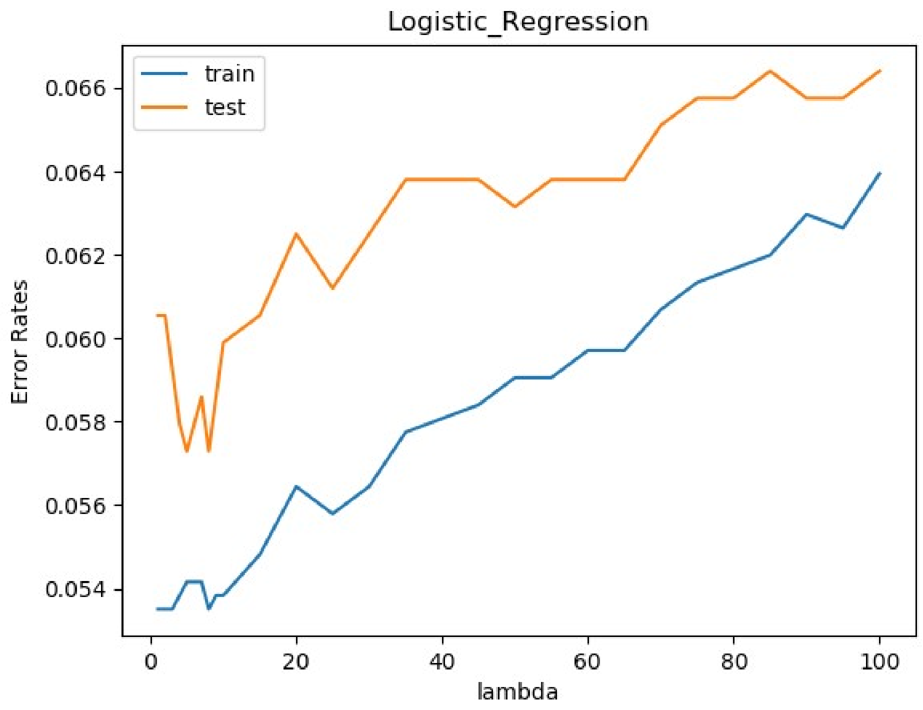
Where, *I* is identity matrix.

Note that bias term should not be regularized.

Suppose that when $NLL_k - 10^{-6} < NLL_\mathrm{k+1} < NLL_k$, training process ends.

(The algorithm is shown in '*2-3 Logistic Regression.py*')

(1) Plots of training and test error rates versus λ.



(2) What do you observe about the training and test errors as λ change?

In general, the error rates of both train and accuracy are increasing, with increasing λ. The training error ranges from 5.35% to 6.39%. The test error ranges from 5.72% to 6.64%.

However, when λ is relatively small, from 1 to 10, the error rates fluctuate. Especially, when λ equals to 8, both training error and test error can be the minimized, which are 5.35% of training error and 5.73% of test error.

Therefore, in this case, the optimized λ is 7, when both training and test phases can achieve minimized error.

(3) Training and testing error rates for λ = 1, 10 and 100.

When lambda = 1, training error rate: 5.35 % test error rate: 5.35 %

When lambda = 10, training error rate: 5.38 % test error rate: 5.99 %

When lambda = 100, training error rate: 6.39 % test error rate: 6.64 %

Q4. K-Nearest Neighbors

Main Idea:

In lecture 4, we have learnt Non-parameter Techniques, which includes K-Nearest Neighbor method.

K is a hyper-parameter, from 1 to 100 in this case. Different K means different different number of nearest samples should be picked. Then how many number relate to every class should be figured out. The predicted class of a particular subject is the class which has the largest sample size nearby.

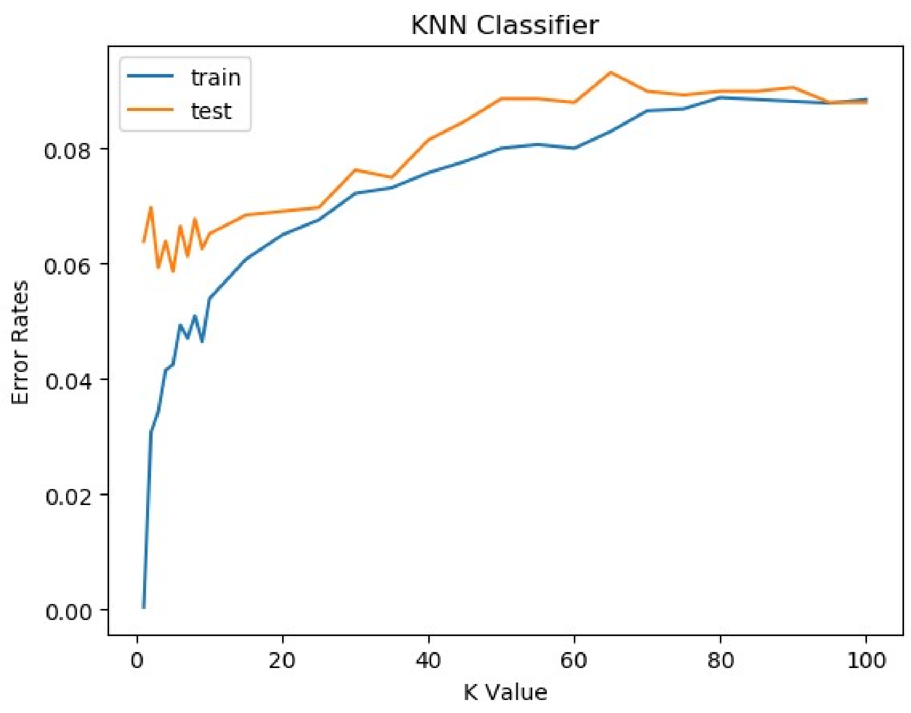
In this case, Euclidean distance should be applied.

$$Euclidean\_distance = \left(\sum_{i=1}^{D} | a_i - b_i |^2 \right)^{\frac{1}{2}}$$

In the algorithm, I import heaqp class, and use nsmallest function to extract the K smallest index from the Euclidean distance.

(The algorithm is shown in ‘*2-4 KNN.py*’)

(1) Plots of training and test error rates versus K



(2) What do you observe about the training and test errors as K change?

In general, with the increasing of K values, the error rates of both train and test are also increasing.

However, when K is relatively small, between 1 and 10, the error rates fluctuate heavily. Especially, when K equals to 5, the test error achieves the lowest, which is 5.86 %.

Besides, on training set, when K equals to one, the training error is very close to zero. That is because training set is used both in training and testing. When K equals to one, we actually use the point itself as the nearest point. Thus, the accuracy is close to 100%.

(3) Training and testing error rates for K = 1, 10 and 100.
When K = 1, training error rate: 0.03 % test error rate: 6.38 %
When K = 10, training error rate: 5.38 % test error rate: 6.51 %
When K = 100, training error rate: 8.84 % test error rate: 8.79 %

Q5. Survey

The time I spent, finishing all the coding part, is approximately 16 hours. Besides, writing the report cost me approximately 2 hours.

Suggestions: (1) Cover some relative code samples in lecture or reading material. Some of my friends, who have little programming background, find the tasks really difficult. Therefore, some samples will help.

(2) When you cover some difficult theories in lectures, it is better to use simple examples and more examples to help students understand. Given the result of answering questions in the class(I mean questions on Poll Everywhere), I guess sometimes most of students have not comprehend the concepts well, because sometimes, the majority of students chosed the wrong answer.

The last but not least, I would like to express my very great appreciation to professor Thomas, for the great lectures and valuable suggestions.