EE5907/EE5027 Programming Assignment Report

Student Number: A0229575R   Name: Zhou Ying

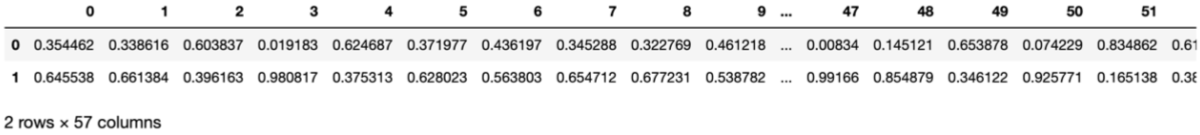
Q1. Beta-binomial Naive Bayes

Seeing the train data as generative model and suppose it follows the Beta-binomial distribution.

The design of classifier contains three parts.
Firstly, for each feature x$_c$, use posterior predictive to calculate its probability,

$$\mathrm{p}(\tilde{\mathrm{x}}|\tilde{\mathrm{y}} = c, \mathrm{D}) = \prod_{j=1}^{D} \mathrm{p}\left(\tilde{\mathrm{x}}_j \middle| \tilde{\mathrm{x}}_{i\in c,j}, \tilde{\mathrm{y}} = c\right)$$

(where $\mathrm{x}_{i\in c,j}$ indicates the j-th feature of all training data points belonging to class c)
So, for each feature j, we compute their probability of x_j given y = 1 and y = 0.
Then, by traversing each dimension j and class y = 1 and y = 0, we can get a probability matrix of each feature (when tentatively setting a=b=2, yclass = 1) showed in figure.1



	0	1	2	3	4	5	6	7	8	9	...	47	48	49	50	51	
0	0.354462	0.338616	0.603837	0.019183	0.624687	0.371977	0.436197	0.345288	0.322769	0.461218	...	0.00834	0.145121	0.653878	0.074229	0.834862	0.61
1	0.645538	0.661384	0.396163	0.980817	0.375313	0.628023	0.563803	0.654712	0.677231	0.538782	...	0.99166	0.854879	0.346122	0.925771	0.165138	0.38

2 rows × 57 columns

Figure 1 probability matrix

The first row is the positive probability of x$_c$, which means the feature add positive value to the outcome, the second row, oppositely, role as negative value to the according outcome.

Secondly, testing new samples and train data. We use the same probability matrix got from train data to predict the probability of the outcome y = 1and y = 0.
According to the value of data, choose the corresponding probability. For example, if the value of data x$_{jc}$ is 0, pick the value from the probability matrix where x$_c$ = 0, y = 1 and x$_c$ = 0, y = 0. Calculate all the probability of j feature, compare them and choose the larger one as the predict value of y.

Thirdly, calculate the error rates by compared the predict value to the true labels of y.

a. Plots of training and test error rates versus *α*
answer:

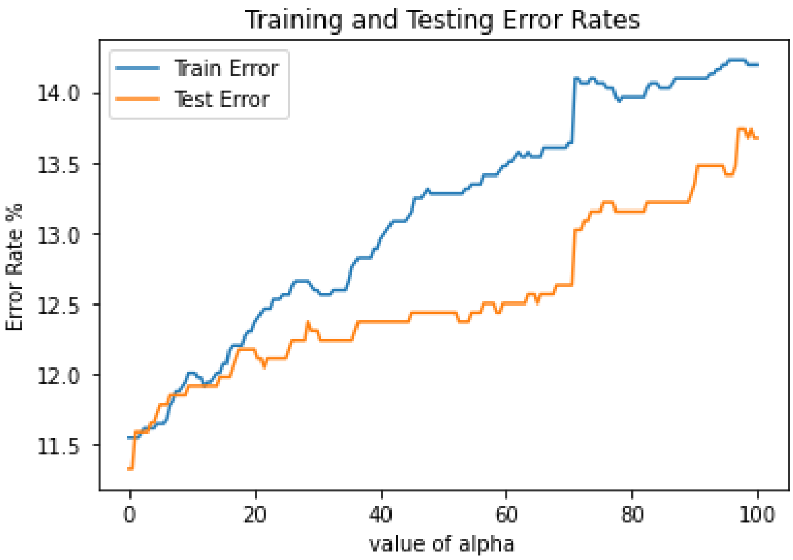


Figure 2 training and test error rates of Beta-binomial naive bayes

b. What do you observe about the training and test errors as α change
answer:

seeing from the above figure, the error rates of both training and testing data increase as the parameter alpha of Beta binomial increases in general, but their difference becomes larger and larger after alpha = 20. Besides, the curve becomes much steeper after alpha adds to larger than 70.

c. Training and testing error rates for α = 1, 10 and 100
answer:
when i = 1,
The training error = 11.549755301794454%
The testing error = 11.588541666666668%

when i = 10,
The training error = 12.00652528548124%
The testing error = 11.9140625%

when i = 100,
The training error = 14.192495921696574%
The testing error = 13.671875%

Q2. Gaussian Naive Bayes

Seeing the train data as generative model and suppose it follows Gaussian distribution.

Similarly, the program flow is same as Q1, but changing the estimate method of train data to maximum likelihood estimate method to calculate the class conditional mean and variance of each feature

From the lecture side, get the values of mean and variance expression of univariate gaussian distribution as:

$$p(x) = N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-0.5(x-\mu)^2/\sigma^2\right)$$

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

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

Then, by traversing each dimension j and class y = 1 and y = 0, we can also get a probability matrix.

a. Training and testing error rates for the log-transformed data

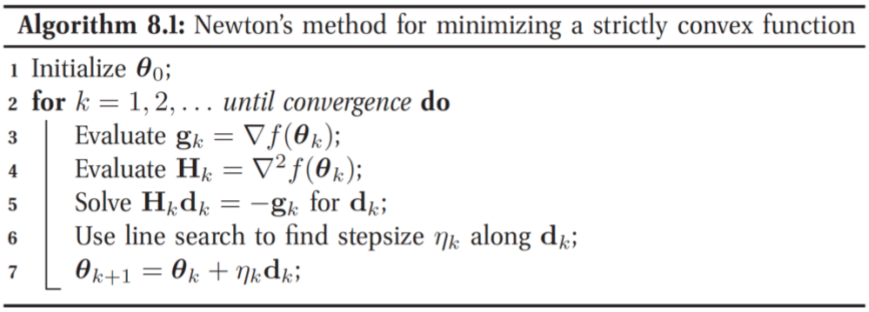
answer:

(16.574225122349105, 16.015625)

Q3. Logistic Regression

For discriminative model q3 and q4, we can get the predict label y from the data and do not have to assume a distribution.

Firstly, estimate parameter w of LR model. To get the minimization of negative likely hood of w, and since NLL is convex, we can use newton's method to get the optimal outcome. The pseudocode (from week lecture notes) is as follows:



Algorithm 8.1: Newton's method for minimizing a strictly convex function

```
Initialize $\boldsymbol{\theta}_0$;
for $k = 1, 2, \dots$ until convergence do
    Evaluate $\mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k)$;
    Evaluate $\mathbf{H}_k = \nabla^2 f(\boldsymbol{\theta}_k)$;
    Solve $\mathbf{H}_k \mathbf{d}_k = -\mathbf{g}_k$ for $\mathbf{d}_k$;
    Use line search to find stepsize $\eta_k$ along $\mathbf{d}_k$;
    $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{d}_k$;
```

a. Plots of training and test error rates versus *λ*

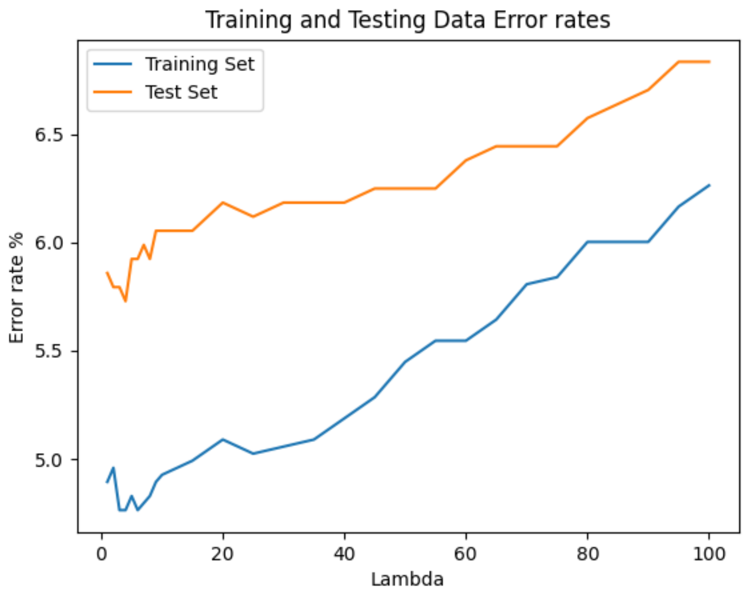


Figure 3 training and testing error rates of LR

b. What do you observe about the training and test errors as *λ* change?

answer:

Seeing from the above figure, the error rates of both training data and testing data increase as lambda increases. But at the beginning, when lambda is less than 7, there is a slight decreasing of the error rates. Besides, when lambda is less than 60 the error rate of test data fluctuates in 6% and when lambda is larger than 60, there seems a steeper increase in testing data error rates, so when lambda is between 5 and 8, the model is much more accurate.

c. Training and testing error rates for *λ* = 1, 10 and 100.

answer:

lambda = 1
training error:0.04894
testing error: 0.058594

lambda = 10
training error: 0.049266
testing error: 0.060547

lambda = 100
training error: 0.062643
testing error: 0.068359

Q4. K-Nearest Neighbors

The programing flow have two main parts.

Firstly, calculate the Euclidean distance between training data and testing data (the testing data is same as training data when calculate training error rate.)

Secondly, sort the data by distance and find those nearest data using function "argsort()". Then, get the return types: idx of data. Sum and compare the corresponding label of ytrain data of different class to get the prediction. Finally, calculate the error rates by compared the predict value to the true labels of y.

a. Plots of training and test error rates versus *K*

answer:

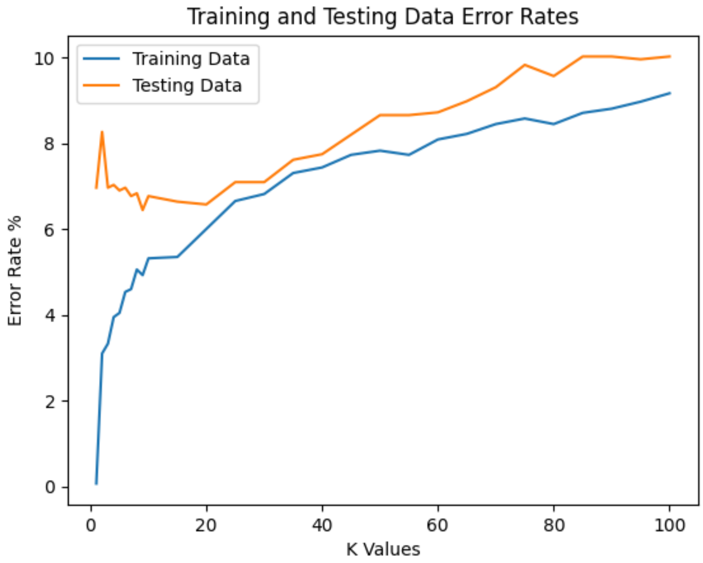


Figure 4 training and testing error rates of KNN

b. What do you observe about the training and test errors as *K* change?

answer:

The training error is been increasing when K increases and it increases vary fast when K is less than 20, and then the speed slows down and the curve seems to converge.

The test data error rate fluctuates between 7-10%. When K is very small, it seems that the distance range does not have much effect on the outcome of classification, but when K increases and lager than 50, the prediction of test data performs not that good, which means it may over-fitting.

c. Training and testing error rates for *K* = 1, 10 and 100.

answer:

K = 1
training error: 0.06525285481239804%
testing error: 6.966145833333333%

K = 10
training error: 5.318107667210441%
testing error: 6.770833333333333%

K = 100
training error: 9.168026101141926%
testing error: 10.026041666666668%

Q5. Survey

It takes approximately 60 hours to review the theory and lectures, try some easy data set in Jupyter first, do the programming and write the report.