EE5907 Pattern Recognition - CA1

Student Name: Nie Hongtuo
Matric Number: A0224712N

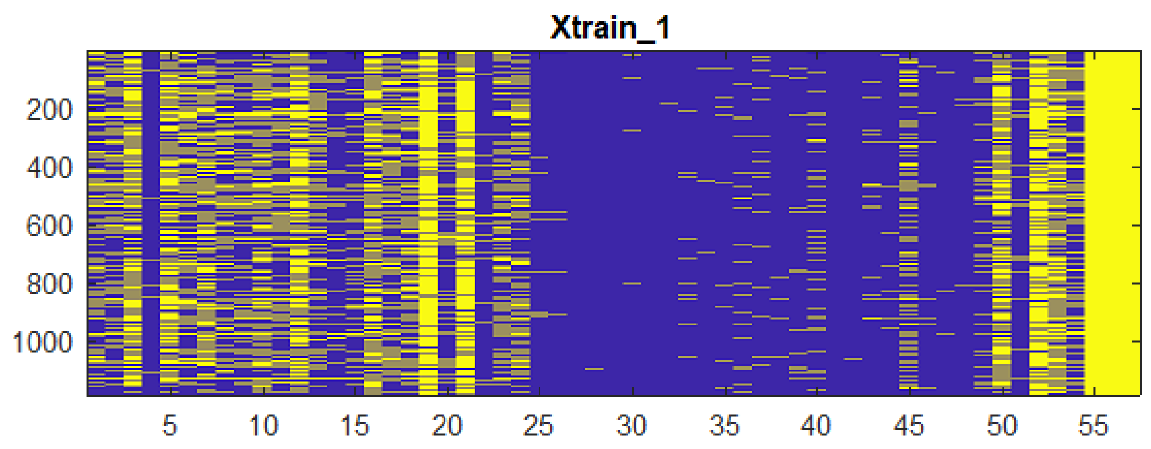
Q1. Beta-binomial Naive Bayes

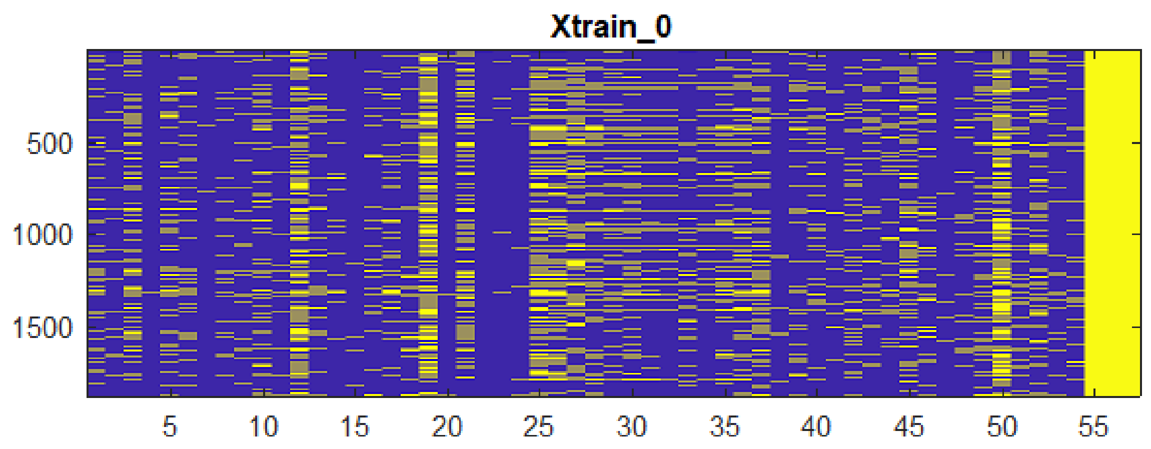
Implementation

In the pre-processing stage, feature binarization is applied to the training and test data, which can be written in:

```
Xtrain = Xtrain > 0;
Xtest = Xtest > 0;
```

We can take a glance at the diversity of the distribution in dataset by visualizing the binary features with different colors (yellow block means $X_{train}(i, j) = 1$):





We assume all of the 57 features are following Beta distribution:

$$p(\theta, a, b) = \frac{1}{B(a,b)}\theta^{a-1}(1-\theta)^{b-1}$$

in which $a = b = \alpha$. With the prior $Beta(\alpha, \alpha)$, we can utilize the posterior of θ

$$p(\theta|D) = Beta(\theta|N_1 + a, N_0 + b)$$

to calculate $p(\tilde{x}|D)$, which is actually the mean of $p(\theta|D)$:

$$p(\tilde{x} = 1|D) = E(\theta|D) = \frac{N_1 + \alpha}{N + \alpha + \alpha}$$

$$p(\tilde{x} = 0|D) = 1 - p(\tilde{x} = 1|D)$$

To compute the probability of $p(\tilde{y} = 1|\tilde{x}, D)$, we can implement the following formula:

$$logp(\tilde{y} = c|\tilde{x}, D) \propto logp(\tilde{y} = c|\lambda_{ML}) + \sum_{j=1}^{D} logp(\tilde{x}_j|x_{i\in c,j}, \tilde{y} = c)$$

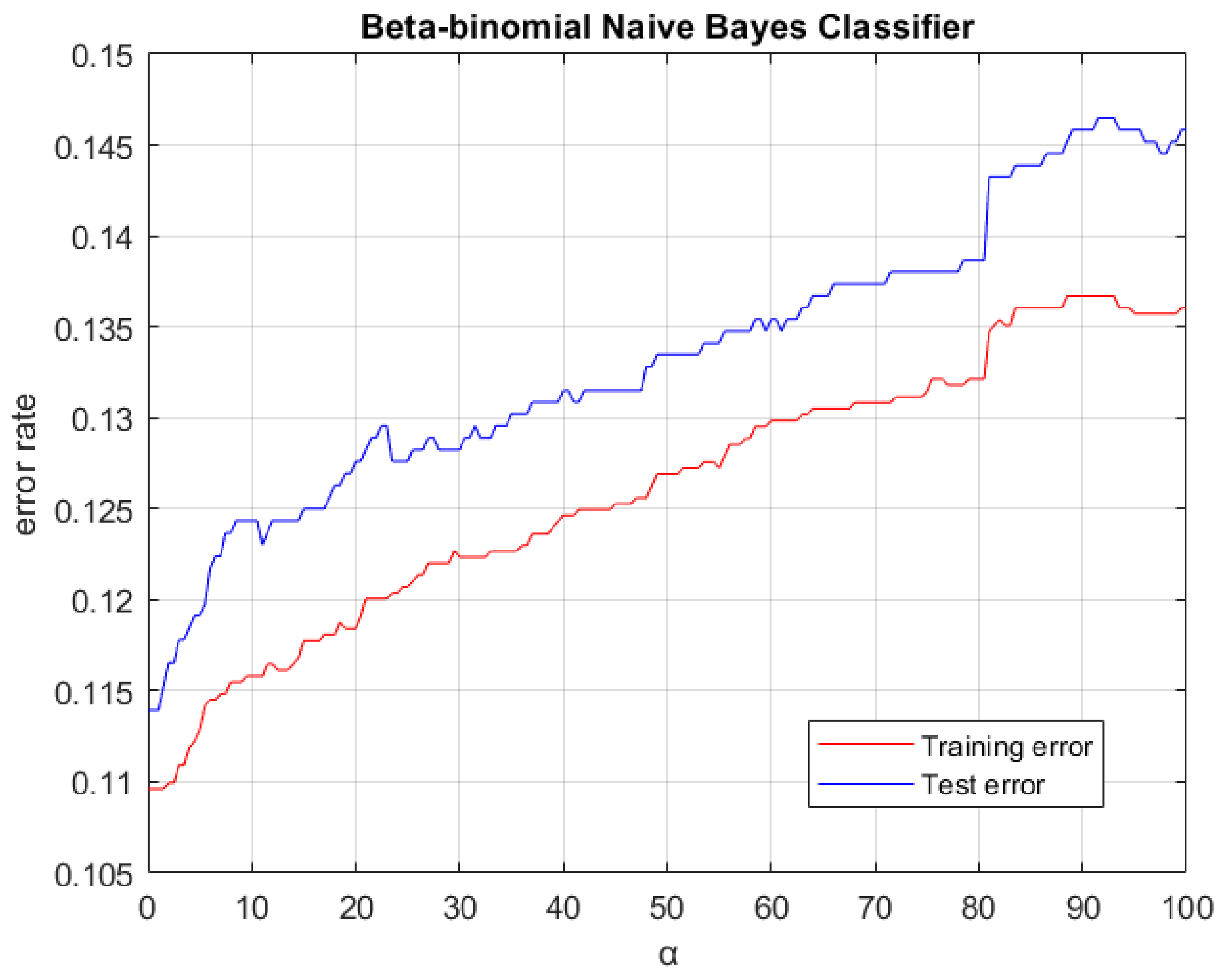
Then compare the probability $p(\tilde{y} = 1|\tilde{x}, D)$ and $p(\tilde{y} = 0|\tilde{x}, D)$ to make the final prediction.

Simulation

The table shows the training and test error when $\alpha = 1, 10, 100$:

α	Training error	Test error
1	10.96%	11.39%
10	11.58%	12.44%
100	13.61%	14.58%

The figure of training and test error vs hyperparameter α is shown below. Generally speaking, as α is increasing, both training and test error are rising.



Q2. Gaussian Naive Bayes

Implementation

In pre-processing part, all the training and test data is transformed into logarithm:

$$X = log(X + 0.1)$$

After dividing the training data into 2 classes, we can calculate the Maximum Likelihood estimation of mean μ and variance σ^2 over each features.

Then the probability of each feature can be calculated by:

$$p(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp[-\frac{(x-\mu)^2}{2\sigma^2}]$$

Finally we implement the following formula to calculate the probability $p(\tilde{y} = 1|\tilde{x}, D)$ and $p(\tilde{y} = 0|\tilde{x}, D)$ and make the prediction:

$$logp(\tilde{y} = c|\tilde{x}, D) \propto logp(\tilde{y} = c|\lambda_{ML}) + \sum_{j=1}^{D} logp(\tilde{x}_j|x_{i\in c,j}, \tilde{y} = c)$$

Simulation

The training error is 16.67% and the test error is 18.36%.

Q3. Logistic regression

Implementation

In pre-processing part, like the last implementation, all the training and test data is transformed into logarithm. Then we need to construct the negative log likelihood function with sigmoid function $\mu = 1/(1 + exp(-w^T x))$:

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

The first derivative can be computed by:

$$g = \frac{d}{dw} NLL(w) = X^T(\mu - y)$$

Here X is the training data X_{train} with bias terms added ahead.
The second derivative is evaluated by:

$$H = \frac{d}{dw} g(w)^T = X^T S X$$

which is a positive definite matrix, so that cost function can be decreasing. The diagonal matrix S is constructed by putting $\mu_i(1 - \mu_i)$ on the i-th diagonal.

With l_2 regularization implemented, the new NLL function, gradient and hessian becomes NLL_{reg}, g_{reg} and H_{reg}:

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

$$g_{reg} = g(w) + \lambda \cdot [0_{1\times 1} \quad w_{1\times D}]^T$$

$$H_{reg} = H(w) + \lambda \cdot diag(0 \quad I_{1\times D})$$

The number of iterations is set to 20. The stepsize is set to 1 by default. By calculating the sigmoid function with samples X_{train}, X_{test} and the trained weight vector w, we can directly get the probability of samples being positive.

Simulation

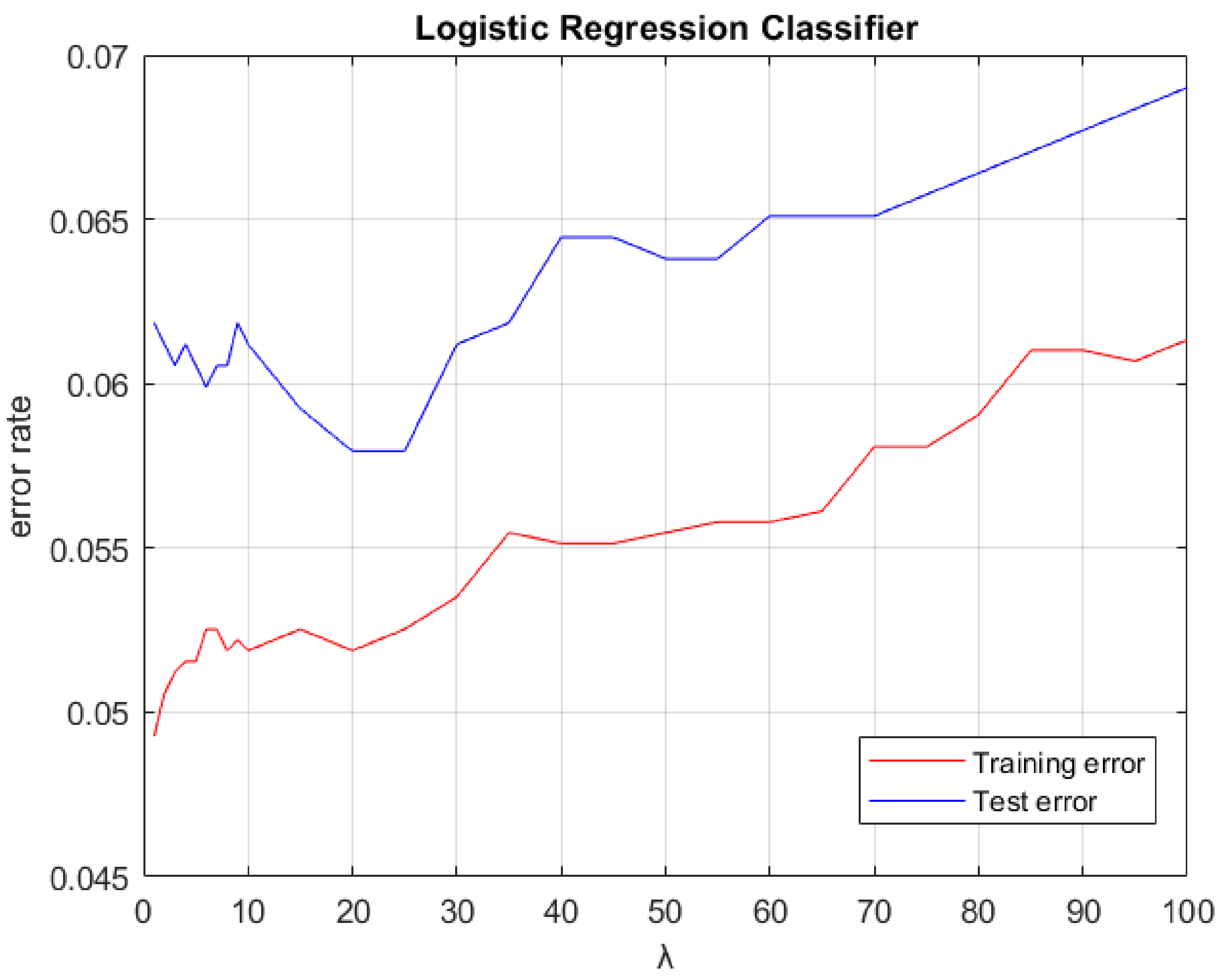
The table shows the training and test error when $\lambda = 1, 10, 100$:

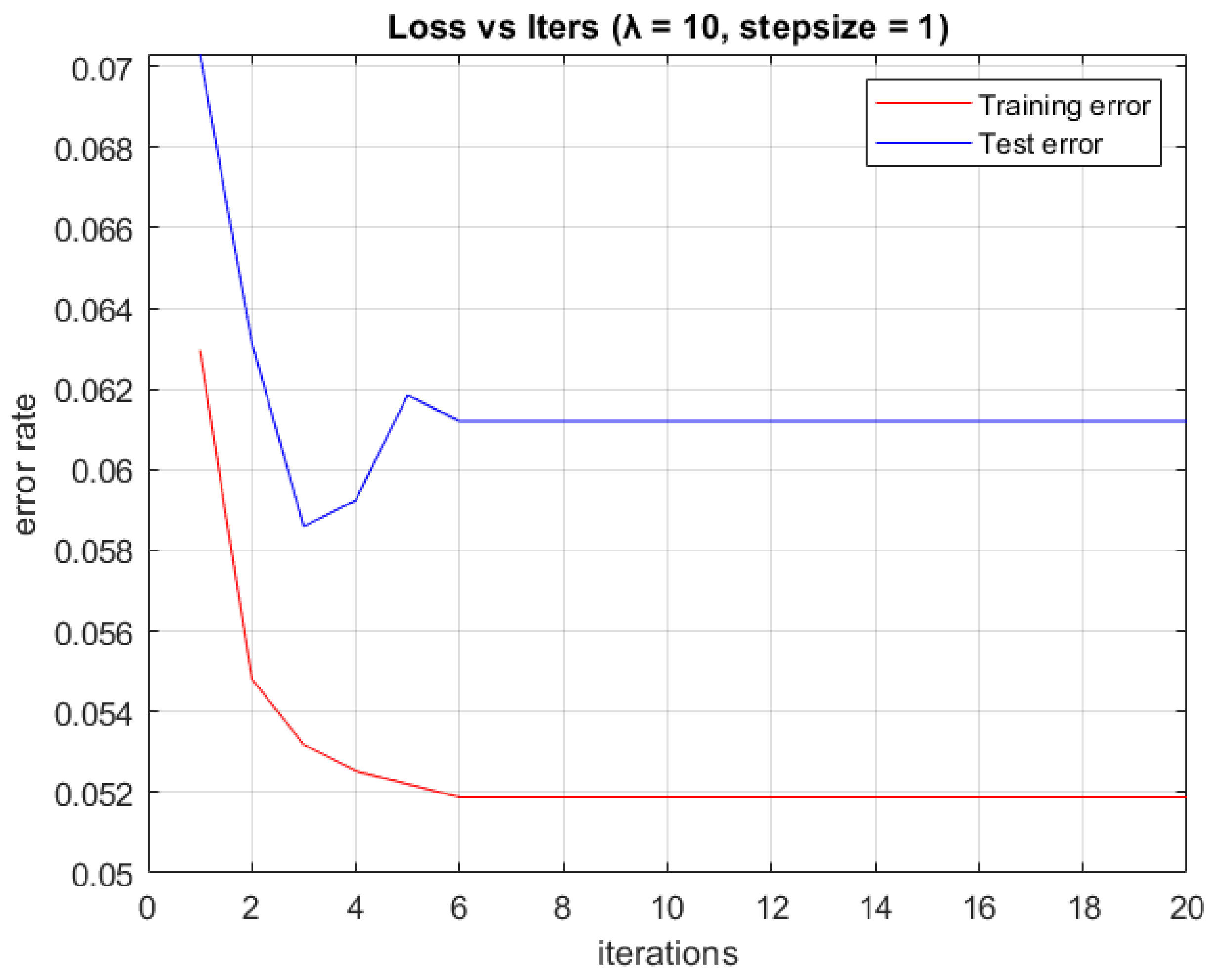
λ	**Training error**	**Test error**
1	4.93%	6.18%
10	5.19%	6.12%
100	6.13%	6.90%

The figure of training and test error vs regularization parameter λ is shown below.

As λ is increasing, training error is rising with little fluctuation, while test error goes down when $\lambda = [0, 20]$, and rises when $\lambda = [20, 100]$. Based on this image of test error, the optimal λ can be set to 20.

Also, we can observe training and test error rate versus iterations to see whether λ converges or not. For example, an image of error vs iters when $\lambda = 10$ is shown in the end.





Q4. K-Nearest Neighbors

Implementation

All the training and test data is transformed into logarithm in the pre-processing stage. Then according to the algorithm of KNN, all we need to do is computing the Euclidean distance from each sample to all the training samples, and analyze the labels of the K nearest neighbours in the feature space.

The formula of posterior is derived based on the joint probability $p(x, y=c) = \frac{k_c/N}{V}$:

$$p(y=c|x) = \frac{p(x, y=c)}{\sum_{c'=1}^{C} p(x, y=c')} = \frac{\frac{k_c/N}{V}}{\sum_{c'=1}^{C} \frac{k_{c'}/N}{V}} = \frac{k_c}{K}$$

Simulation

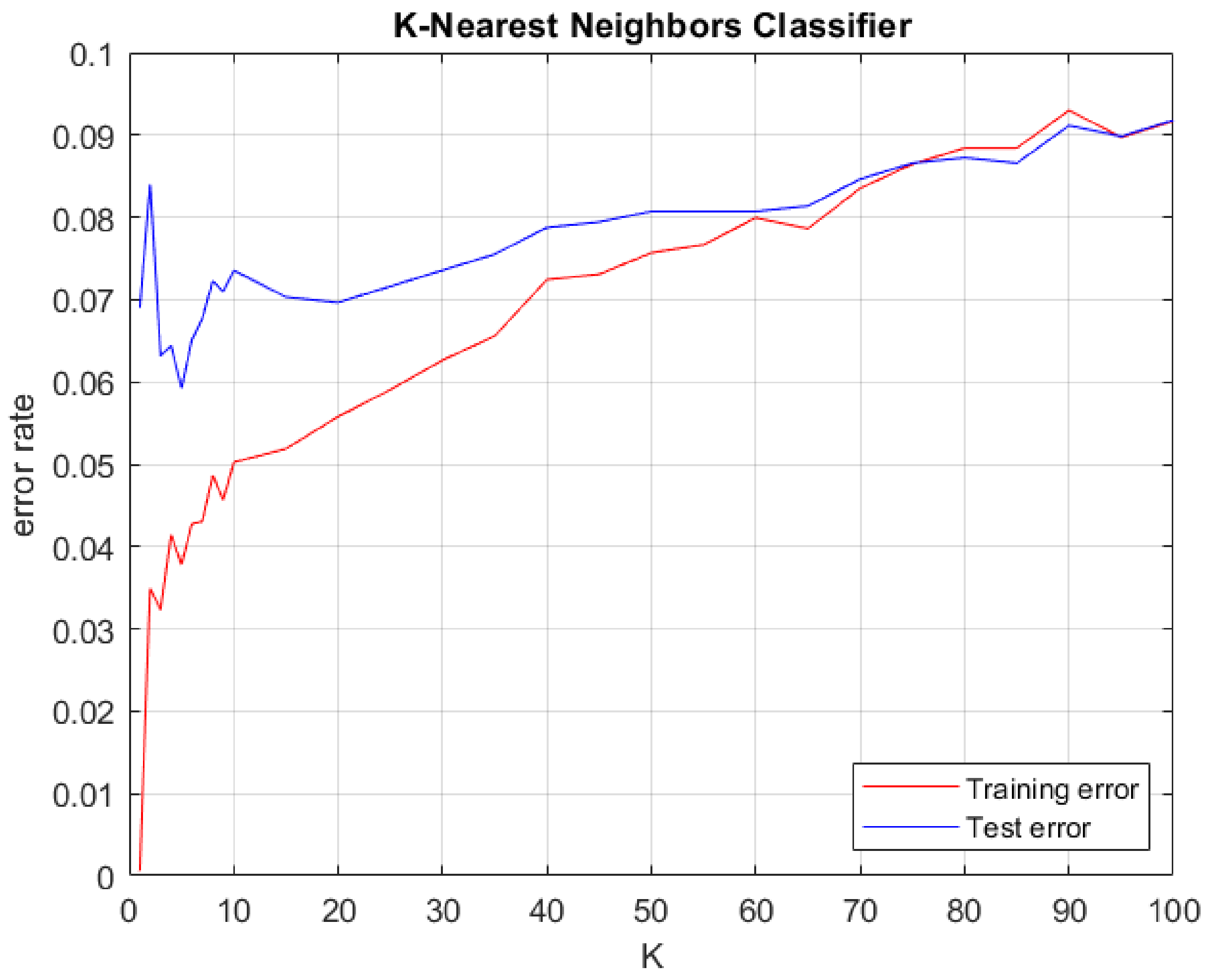
The table shows the training and test error when $K = 1, 10, 100$:

K	Training error	Test error
1	0.07%	6.90%

K	Training error	Test error
10	5.02%	7.36%
100	9.17%	9.18%

The figure of training and test error vs regularization parameter λ is shown below.

As K is increasing, training error increases sharply at first, then continue to rise in a slower pace, while test error fluctuates when K is small, then keeps rising slowly. At the same time, the gap between training and test error is decreasing.



Q5. Survey

The whole task took about 50 hours to accomplish.