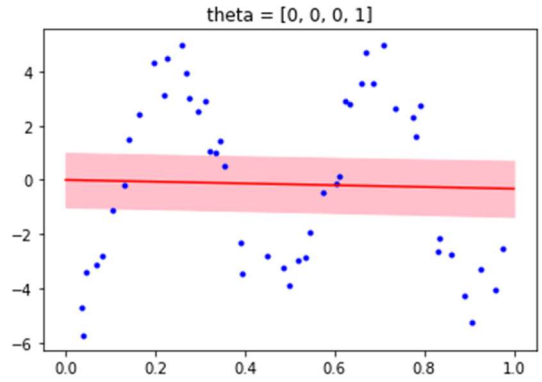
ML HW3

309513121 李偉齊

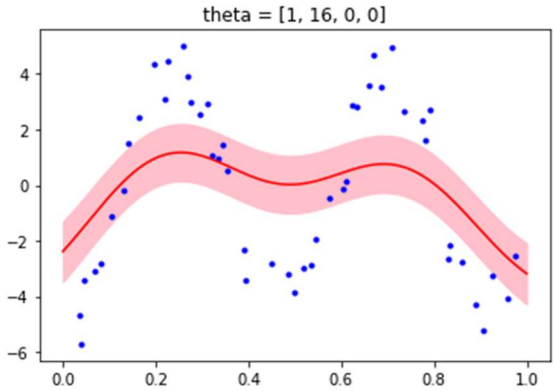
1.Gaussian Process

(1)(2) Blue: Original dataset , Red: mean

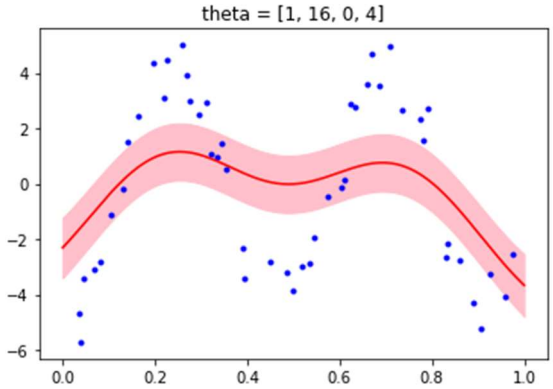
linear kernel θ ={0,0,0,1}



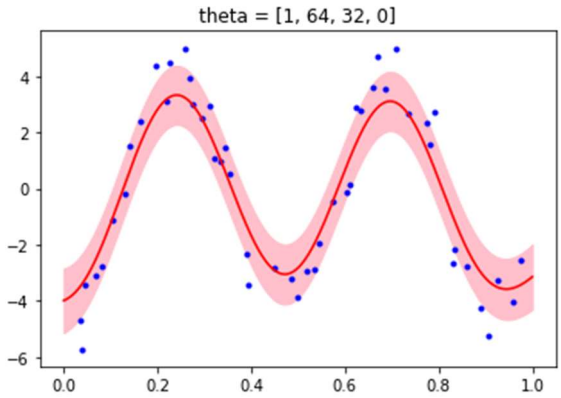
squared exponential kernel θ ={1,16,0,0}



exponential-quadratic kernel θ ={1,16,0,4}



exponential-quadratic kernel θ ={1,64,32,0}

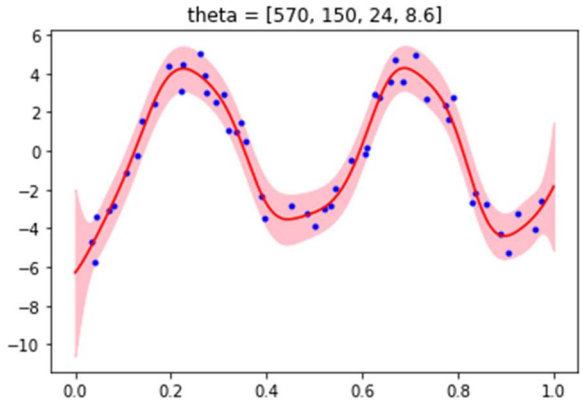


(3)From the formula of “Root Mean Square”, we get the RMS below:

	Train	Test
θ ={0,0,0,1}	3.1292014298222433	3.3443986601861146
θ ={1,16,0,0}	2.4239279278312194	2.6680517502524466
θ ={1,16,0,4}	2.4105764871252062	2.6569980001669165
θ ={1,64,32,0}	1.0428861621832175	1.1627590936118612

(4)

I try the θ again and again, and then finally choose θ ={570,150,24,8.6}



Train data Error : 0.6582610230813177

Test data Error : 1.124881177568828

(5) Consider the predictions we show, the “ linear kernel ” present the worst. The others kernels show some fit result with the original dataset.

2.SVM

(1) I used ‘One-versus-One’ to be my decision strategy.

First I classed the class 0 and class 1, labeled class 0 to 1 and class 1 to -1, and then classed class 0 and class2, at last classed class 1 and class as the above

method.

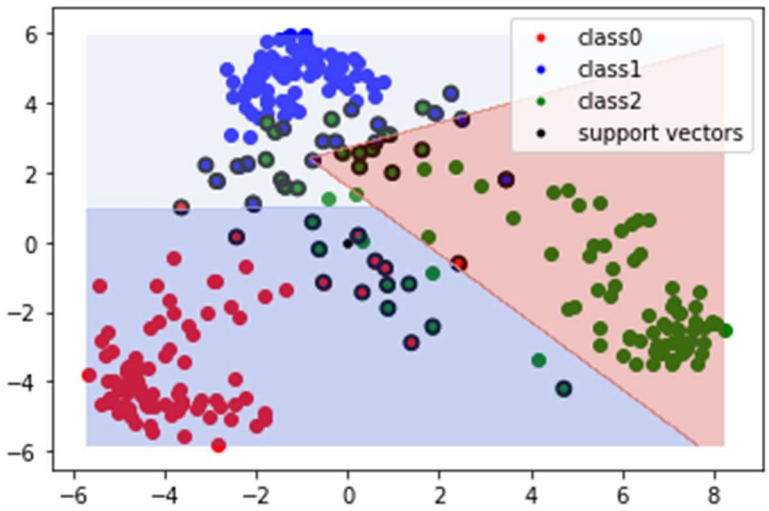
I used “PCA” to “resize” the training data, namely, the data dimensionality reduction to 2.

Then, I generated testing data by consider the max and min values of training data to be testing data’s boundary and get some points in this range evenly.

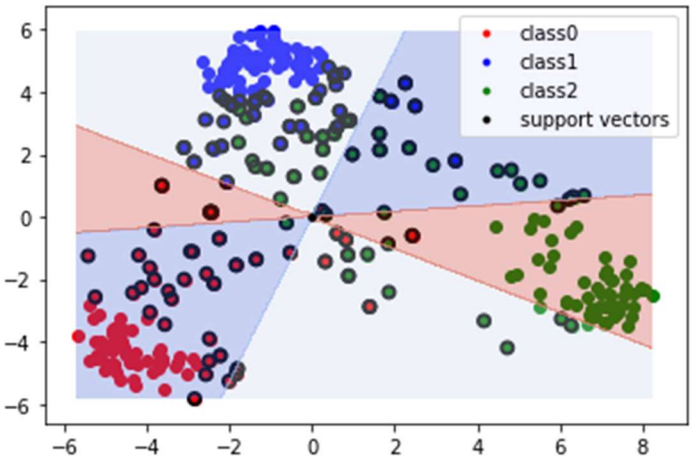
At last, setting the boundary of prediction of y as 0 to classed the test data.

I showed the results in the below pictures.

(2) Linear Kernel:



(3) Polynomial Kernel (degree = 2)



(4) The boundaries of Linear Kernel model are lines straight line, in contrast, the boundaries of Polynomial Kernel model are curves.

The Linear Kernel present the better classification result in this dataset.

3. Gaussian Mixture Model

(1)

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}||x_n - \mu_k||^2$$

K-means model:

Minimizing the above J. The " γ nk" is a binary indictor, If the data below class k, set it to be 1, on the contrary, set it to be 0.

First, initialize the "miu k", randomly choose k points in data points to be "miu k" and fixed it. Then , do "2-norm" to every data and "miu k " and assign the data to the class k which have the smallest "distance".(Show in the below equation)

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

Second, fixed " γ nk" and update it by the below equation:

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}}.$$

(P.S: I set the threshold by do "2-norm" to the "old-miu" and the "updated-miu".)

At last, I show the "miu" in the situations "K = 3, 5, 7, 10" below:

K = 3, "miu" :

[[0.4099986 0.49382641 0.52101836]
 [0.70468336 0.75834093 0.75018212]
 [0.21512237 0.26964824 0.29883648]]

K = 5, "miu" :

[[0.40466475 0.49045617 0.5211064]
 [0.80699428 0.8406455 0.82795446]
 [0.57499162 0.65167418 0.65484112]
 [0.27897552 0.34375156 0.37279131]
 [0.16757428 0.21534333 0.24422427]]

K = 7, "miu":

[[0.30732114 0.38150867 0.40946114]
 [0.22593377 0.27663738 0.30781419]
 [0.12950822 0.17791798 0.20373237]
 [0.40385664 0.48676942 0.51843972]
 [0.86653146 0.88737893 0.87380955]
 [0.51303979 0.60459983 0.62097809]
 [0.67767425 0.73243013 0.72045158]]

K = 10, miu is :

[[0.3310863 0.42721716 0.46749413]
 [0.20348592 0.25184215 0.28302052]
 [0.27623788 0.33532845 0.36264399]
 [0.61700561 0.62543523 0.59368861]
 [0.74032606 0.76554857 0.74175384]
 [0.11174259 0.16082479 0.18578149]
 [0.40641189 0.53768524 0.6011816]
 [0.88869064 0.90935916 0.89875181]
 [0.49639254 0.70270762 0.7579645]
 [0.48590401 0.4941177 0.46691936]]

(2)

According to the “miu k” and “ γ nk” we knew, they could be the initial parameters of “GMM model”, and we could get the covariance matrix by below equation:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

EM Algorithm:

E-step(expectation):

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

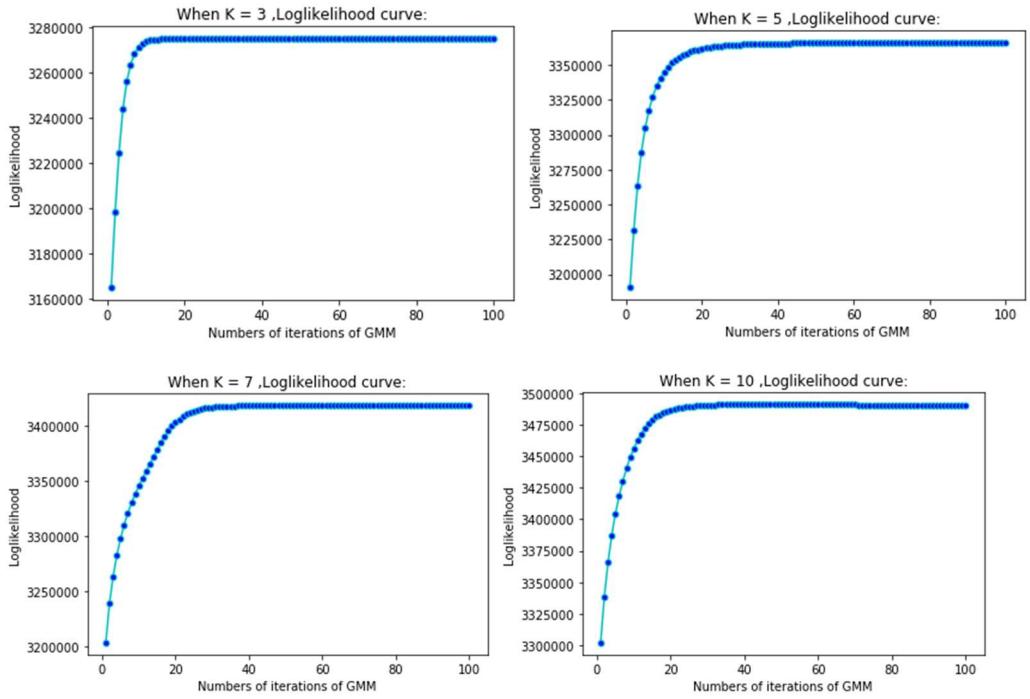
M-step(maximization):

$$\begin{aligned}
\boldsymbol{\mu}_k^{\mathrm{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \\
\boldsymbol{\Sigma}_k^{\mathrm{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\mathrm{new}}\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\mathrm{new}}\right)^{\mathrm{T}} \\
\pi_k^{\mathrm{new}} &= \frac{N_k}{N}
\end{aligned}$$

(PS: $N_k = \sum_{n=1}^{N}\gamma(z_{nk}).$

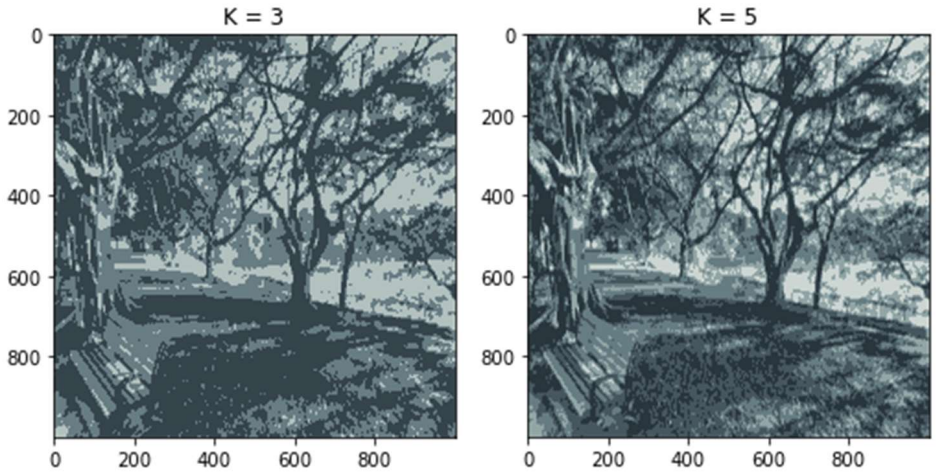
Then calculate the loglikelihood by:

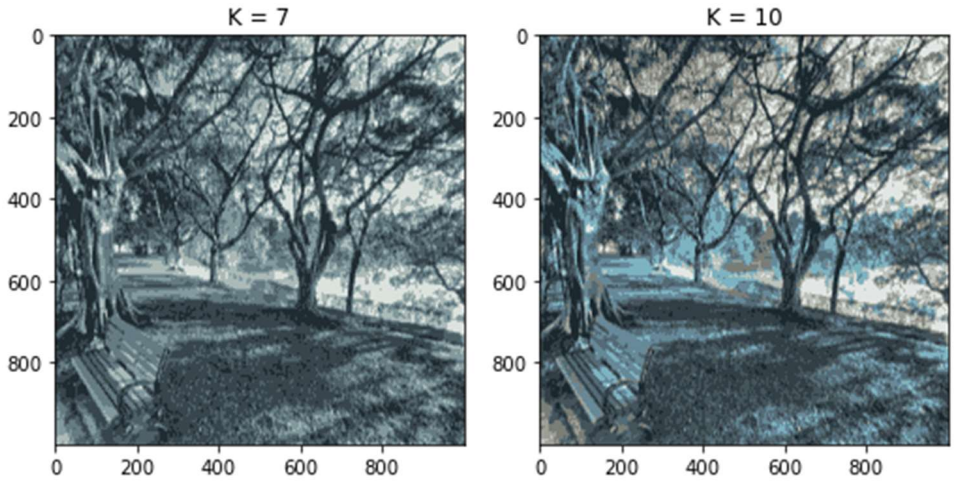
$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right\}$$



(3)

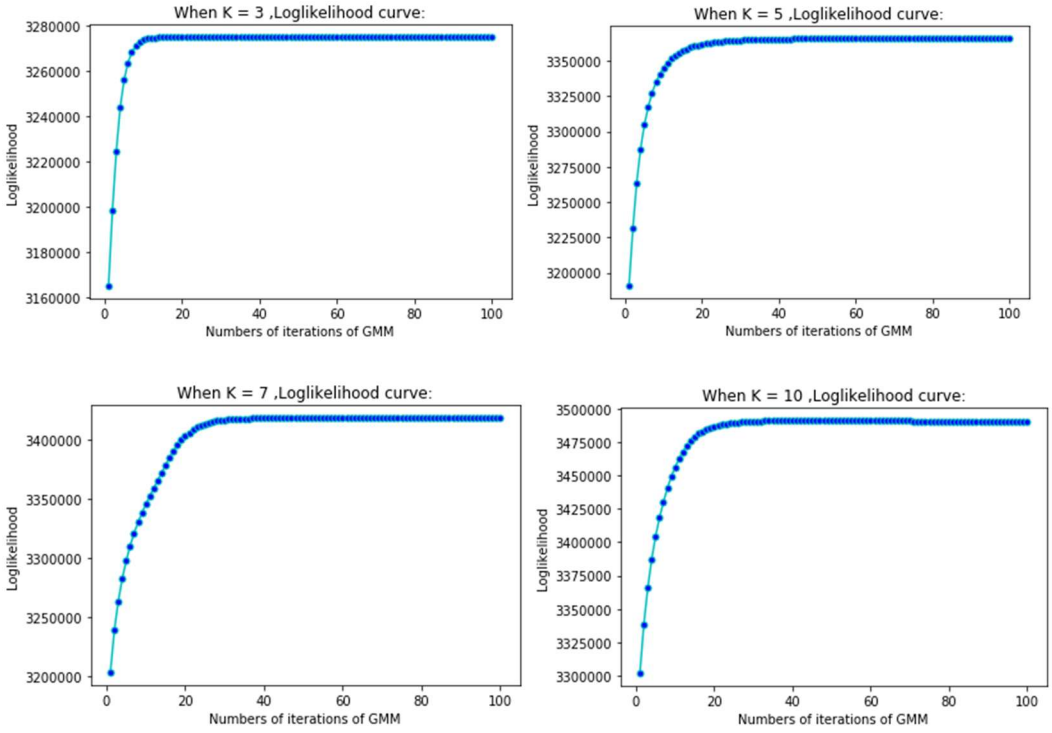
Use μ_k^{new} and $\gamma(z_{nk})$, we got in the EM Algorithm to re-draw the images:





(4)(The same answer with Question 2)

The loglikelihood curves in the situations “K =3, 5, 7, 10” are showing below:



(5)

As the result, we can see that when the K value increase, the image generated clearer and more colorful.

I guess that the higher K we used, the higher loglikelihood achieved, and the re-plot picture will be more completed.