Machine Learning

Homework 3

A091501 張軼峯

1 Gaussian Process for Regression

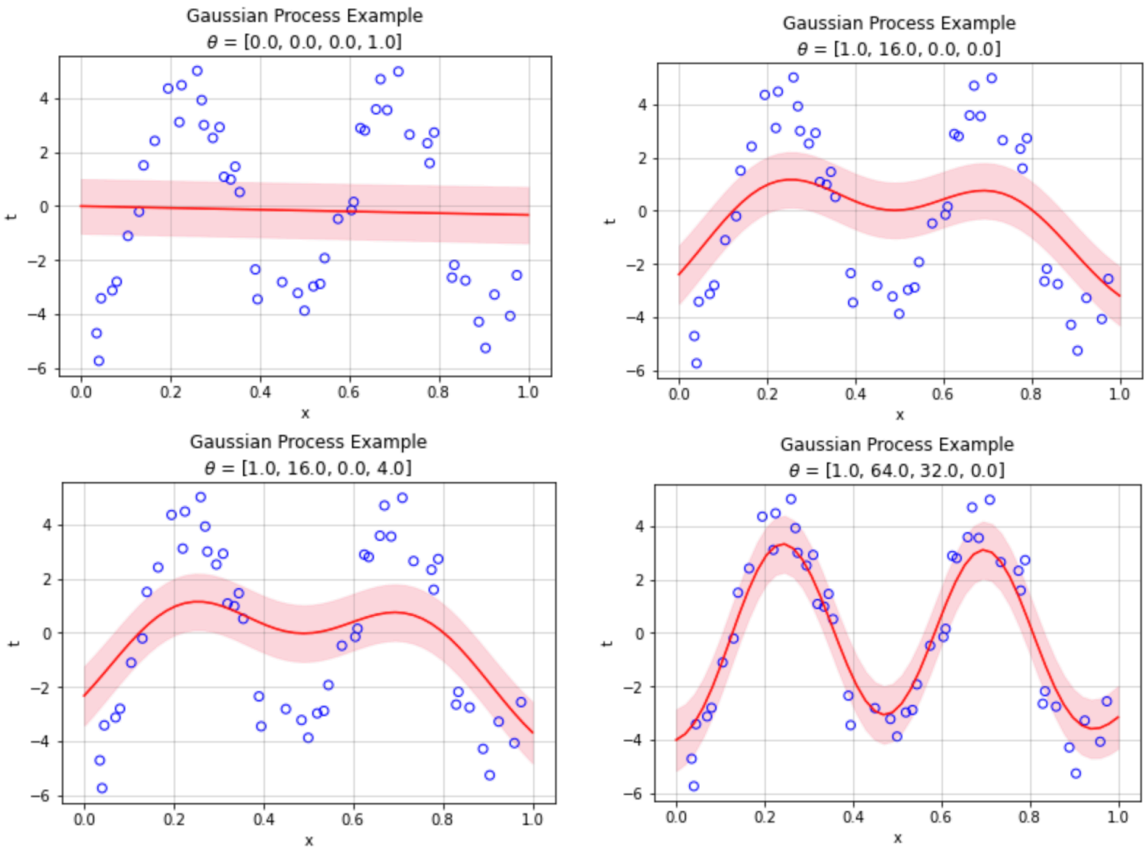
1. Implement the GP with exponential-quadratic kernel function given by

$$k\left(x_n, x_m\right) = \theta_0 \exp\left\{-\frac{\theta_1}{2} \parallel x_n - x_m \parallel^2\right\} + \theta_2 + \theta_3 x_n^T x_m$$

with four different combinations of the hyper-parameters $\theta = \{\theta_0, \theta_1, \theta_2, \theta_3\}$

- Linear kernel $\theta = \{0, 0, 0, 1\}$
- Squared exponential kernel $\theta = \{1, 16, 0, 0\}$
- Exponential-quadratic kernel $\theta = \{1, 16, 0, 4\}$
- Exponential-quadratic kernel $\theta = \{1, 64, 32, 0\}$

2. Plot the prediction result for training set with one standard deviation in pink region.

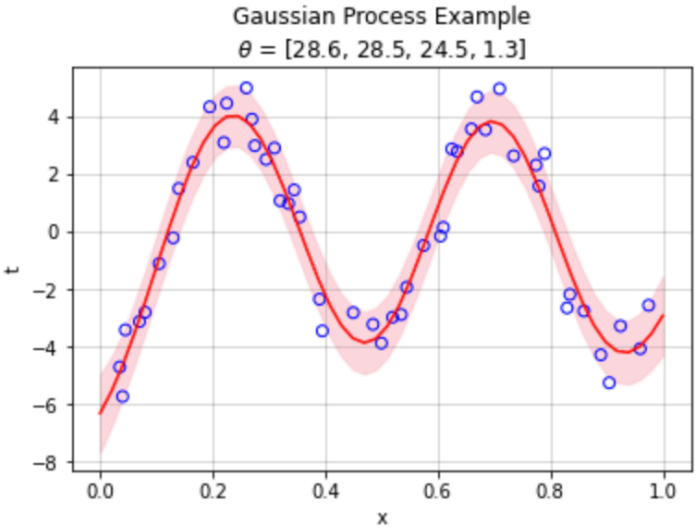


3. Show the corresponding root-mean-square errors for both training and test sets with respect to the four kernels.

Theta	Training RMSE	Test RMSE
[0, 0, 0, 1]	3.1292	3.32018
[1, 16, 0, 0]	2.42393	2.46569
[1, 16, 0, 4]	2.41058	2.45577
[1, 64, 32, 0]	1.04289	1.09253

4. Tune the hyper-parameters by automatic relevance determination (ARD).

ARD_theta = [28.617 28.452 24.467 1.265]
ARD RMSE = [0.773]



5. Discussion

可以觀察到只使用線性 function 時，training RMSE 與 testing RMSE 皆很大，無法 fit 好這組data；第二、第三組實驗的差異在於第三組多了一個 linear function，然而影響不顯著，第二、第三組的結果差異不大。第四組實驗加強了 Gaussian kernel 的能力，模型已能大致符合 data 趨勢。且各組 test RMSE 都僅稍大於 training RMSE，皆尚未overfitting。

Automatic relevance determination 藉由最大化 likelihood，來調整每個 input 對應的 hyper-parameter 的重要性。

Log likelihood:

$$\ln p\left(t\,|\,\theta\right) = -\frac{1}{2}\ln\left|C_N\right| - \frac{1}{2}t^T C_N^{-1} t - \frac{N}{2}\ln\left(2\pi\right)$$

對 θ_i 微分:

$$\frac{\partial \ln p\left(t\,|\,\theta\right)}{\partial \theta_i} = -\frac{1}{2}Tr\left(C_N^{-1}\frac{\partial C_N}{\partial \theta_i}\right) + \frac{1}{2}t^T C_N^{-1}\frac{\partial C_N}{\partial \theta_i}C_N^{-1}t$$

並用 Gradient descent 來迭代更新:

$$\theta_{new} = \theta_{old} + \eta \frac{\partial}{\partial \theta} \ln p(t \mid \theta)$$

作業中實作使用 learning rate: 0.01, epochs = 1000，並得到

ARD_theta = [28.617 28.452 24.467 1.265], ARD RMSE = [0.773]

另外，課本中提到，可將 ARD 推廣至此 kernel：

$$k(x, x') = \theta_0 \exp\left\{ -\frac{1}{2} \sum_{2}^{i=1} \eta_i \left(x_i - x_i'\right)^2 \right\}$$

2 Support Vector Machine

Implement SVM based on two kinds of kernel functions

$$y(x) = \sum_{n=1}^{N} \alpha_n t_n k\left(x, x_n\right) = \mathbf{w}^T x + b \quad , \quad \mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \phi(x_n)$$

- Linear kernel:

  $$k\left(x_i, x_j\right) = \phi\left(x_i\right)^T \phi\left(x_j\right) = x_i^T x_j$$

- Polynomial (homogeneous) kernel of degree 2:

  $$k\left(x_i, x_j\right) = \phi\left(x_i\right)^T \phi\left(x_j\right) = \left(x_i^T x_j\right)^2$$

  $$\phi(x) = \left[x_1^2, \sqrt{2} x_1 x_2, x_2^2\right]$$

  $$x = \left[x_1, x_2\right]$$

1. Analysis the difference between two decision approaches (one-versus-the-rest and one-versus-one). Decide which one you want to use and explain why you choose this approach.
2. Use the dataset to build a SVM with linear kernel to do multi-class classification. Then plot the corresponding decision boundary and support vectors.
3. Repeat (2) with polynomial kernel (degree = 2)

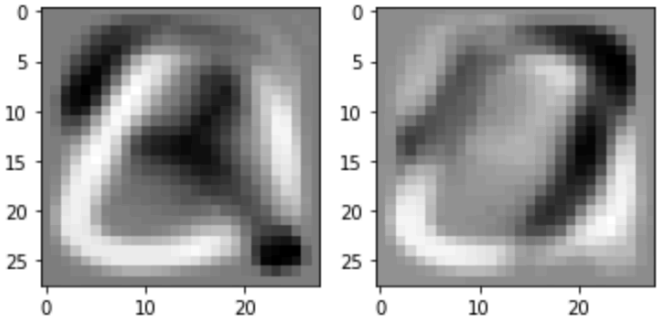
對資料做 normalize 後，利用前次作業的方法先將 data 降至 2 維。Multi-class SVMs 可分為兩種做法: One-Versus-Rest, One-Versus-One。

- One-Versus-Rest：在 K 個分類中，OVR 須建立 K 個 SVM 分類器，dataset 需使用 positive sample: class C_k 和 negative samples: K - 1 classes。此方法的缺點為，不同分類

器視為不同 task，因此不同分類器的 scale 可能不同；另外，training set 被人工處理為 imbalance data，存在 bias，資料原本的 symmetry 將消失。

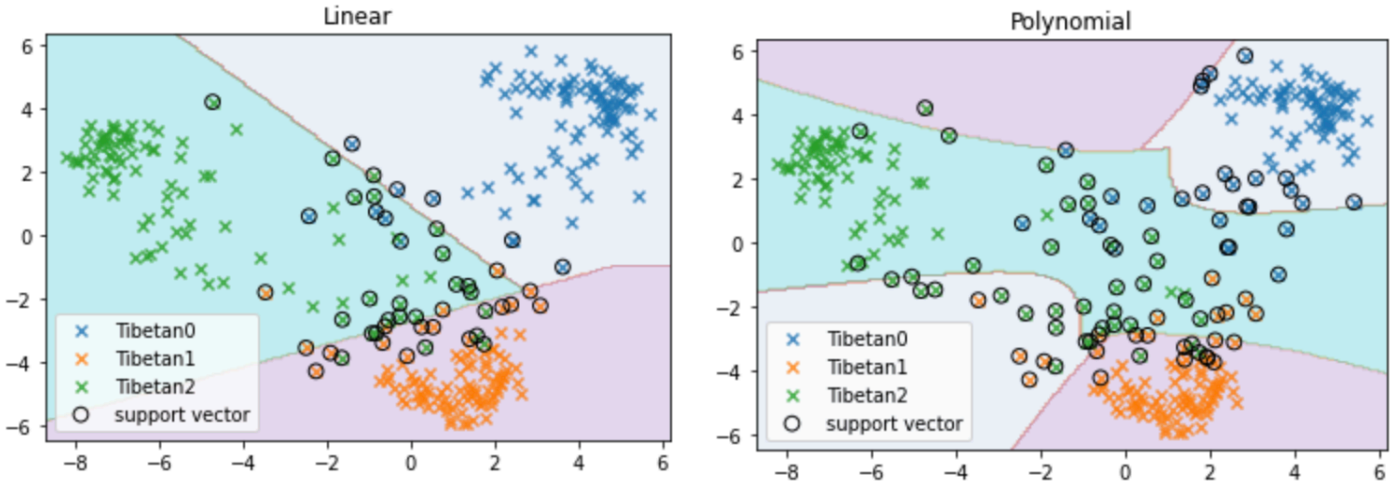
- One-Versus-One (pairwise)：OVO 則是在每兩相異類別間建立 K(K-1)/2 個分類器，再用投票的方式決定該樣本的類別。此方法的缺點為，當 K 較大時，training 與 test 皆較 OVR 耗時，但 OVO 之結果較準確，實用上較常使用，因此本練習也決定採用 OVO。

- 左下圖為 d = 2，對應到前兩個最大 eigenvalue 之 eigenvector 做圖。
- 題目允許使用 sklearn.SVC 之 function 來計算 multipliers (dual_coef_)，因 OVO 皆為 binary classifier， multipliers 之對照如右下圖。



$\alpha_{0,1}^{0}$	$\alpha_{0,2}^{0}$	
$\alpha_{0,1}^{1}$	$\alpha_{0,2}^{1}$	Coefficients for SVs of class 0
$\alpha_{0,1}^{2}$	$\alpha_{0,2}^{2}$	
$\alpha_{1,0}^{0}$	$\alpha_{1,2}^{0}$	Coefficients for SVs of class 1
$\alpha_{1,0}^{1}$	$\alpha_{1,2}^{1}$	
$\alpha_{2,0}^{0}$	$\alpha_{2,1}^{0}$	Coefficients for SVs of class 2
$\alpha_{2,0}^{1}$	$\alpha_{2,1}^{1}$	

接著利用 support_ 提取 support vector 之 index，接著帶入公式即可計算線性分類器之 w 與 b。Linear 與 Polynomial 除使用的 kernel 不同，Polynomial 也須先將 input 代入 $\phi(x) = \left[x_1^2, \sqrt{2}x_1x_2, x_2^2\right]$。Polynomial kernel 將資料投影到高維空間做線性分類，故在二維平面上做圖 decision boundary 會成曲線，可以觀察到 Polynomial kernel 計算得出的 support vector 較多 (Linear: 48 => Polynomial: 85)，且資料點由 Linear kernel 分為三區塊，變為 Polynomial kernel 分為五個區塊。但實際上資料已大致散落在三個角落，Polynomial 分類左下角的部分反而為分類錯誤的情形。



3 Gaussian Mixture Model

Implement a Gaussian mixture model (GMM) and apply it in image segmentation.

1. Please build a K-means model by minimizing

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

   and show the table of estimated $\{\mu_k\}_{k=1}^{K}$.

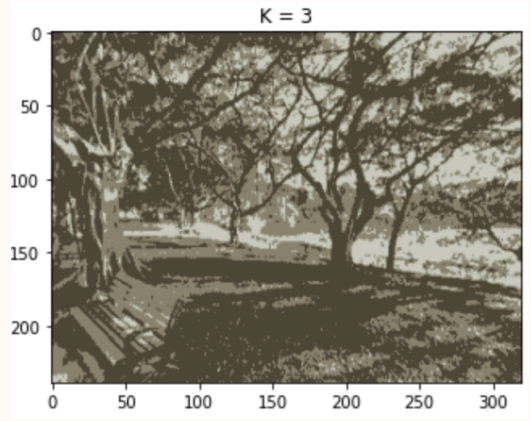
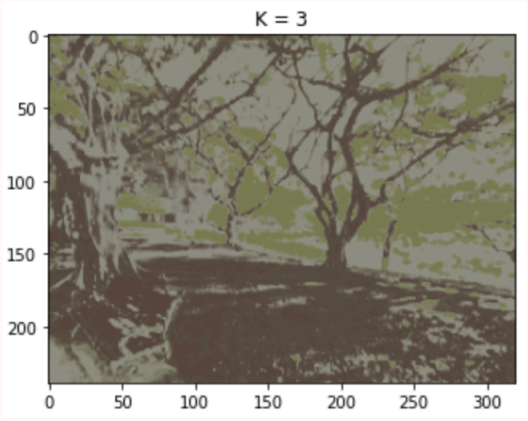
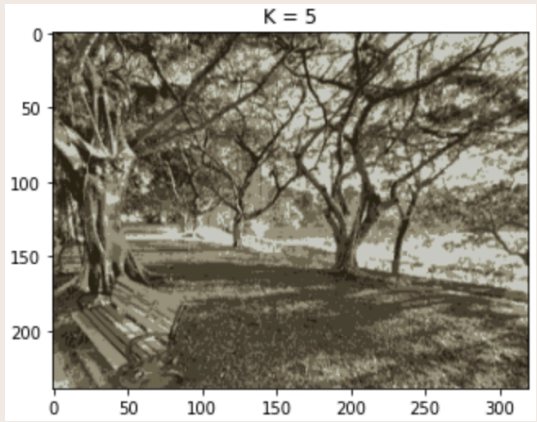
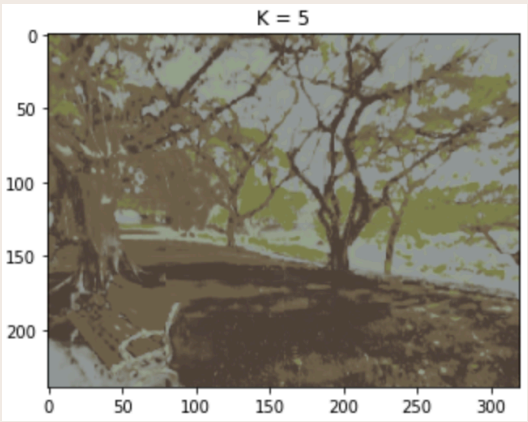
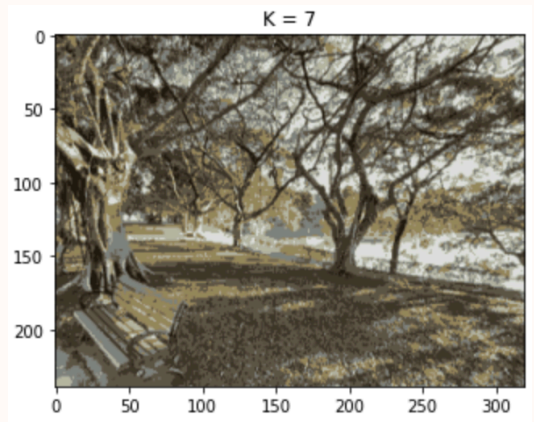
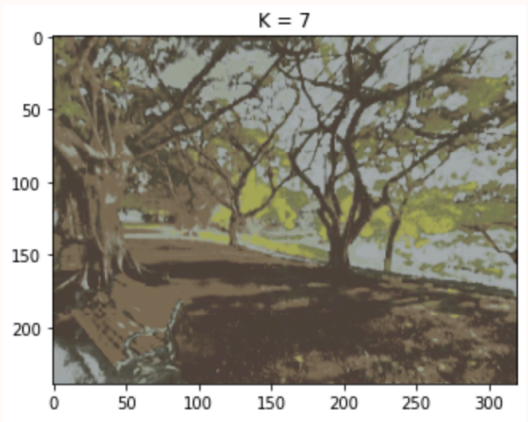
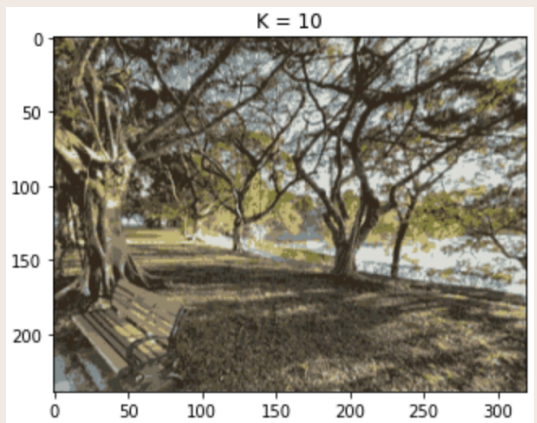
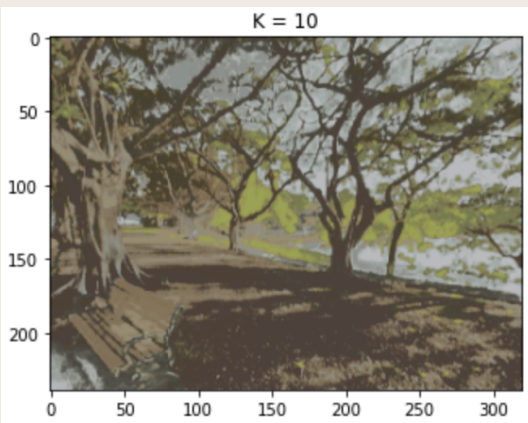
2. Use $\{\mu_k\}_{k=1}^{K}$ calculated by the K-means model as means, and calculate the corresponding variance σ_k^2 and mixing coefficient π_k for initialization of GMM $p(x) = \sum_{K}^{k=1} \pi_k \mathcal{N}\left(x \mid \mu_k, \sigma_k^2\right)$. Optimize the model by maximizing the log likelihood function $\log p\left(x \mid \pi, \mu, \sigma^2\right)$ through EM algorithm. Plot the log likelihood curve of GMM. (Iteration = 100)

3. Repeat step (1) and (2) for K = 3, 5, 7, and 10.

Discussion:

K-means 將各 pixel 的 RGB 三原色視為空間座標中的 x, y, z，利用此座標來迭代計算距離 (抽象的想像為顏色間的距離)，將 K 個 centroid (μ_k) 相近的 pixel 集合為 K 個 cluster，迭代時同時更新 centroid 的值為該 cluster 的平均(中心)，並依此結果畫圖，最後步驟相當於對所有 pixel assign 其顏色座標對應之 centroid 的顏色，故可觀察到 K-Means 的圖輪廓較清晰，但相對於 GMM 缺乏綠色的顏色。

GMM 則是需紀錄 K 個 cluster (distribution) 的 mean, covariance 以計算 probability density，接著利用 Expectation-Maximization 來 maximize log likelihood function，在 EM step 中重複迭代更新 E-step: $\gamma(z_{nk})$, M-step: μ_k, Σ_k, π_k，最後 Evaluate log likelihood 來觀察結果是否收斂(此題設定 termination 條件為 iteration=100)。GMM 明顯需要花較多 iteration 及計算量才能到達收斂，由 Log likelihood 也可觀察到 K 值越高，likelihood 的收斂值越高。GMM 使用 weight π_k 計算 responsibilities $\gamma(z_{nk})$ 來代表該 pixel (x_n) 的 posterior probability，並以 probability 來畫圖，視覺上來說有均勻化的效果，因此可以看到GMM 所畫的圖色調較柔和，而犧牲了明顯的輪廓線。

K Means	GMM
	
	
	
	

K-Means Model **Gaussian Mixture Model**

Estimated $\{\mu_k\}_{k=1}^{K}$ **Estimated** $\{\mu_k\}_{k=1}^{K}$

(為 Normalize 至 0 ~ 1 之間的數值，(1,1,1) 顯示為白色。)

K = 3	R	G	B
1	0.29223634	0.26342911	0.209032
2	0.76538927	0.77237387	0.72110659
3	0.52643993	0.49940102	0.41398951
K = 5	**R**	**G**	**B**
1	0.59294987	0.62836037	0.63002863
2	0.47278965	0.44259626	0.36560883
3	0.28073646	0.25185244	0.1993672
4	0.68552684	0.62908577	0.45101958
5	0.83184541	0.84229543	0.80768308
K = 7	**R**	**G**	**B**
1	0.60997862	0.53832138	0.38177272
2	0.48977999	0.48651413	0.45028134
3	0.18643255	0.15997613	0.11171022
4	0.40688862	0.37588748	0.29622852
5	0.80027954	0.8189378	0.7972235
6	0.29338593	0.26381005	0.215437
7	0.67357431	0.65916948	0.55429711

K = 3	R	G	B
1	0.33815726	0.2838846	0.24448339
2	0.47819889	0.49015821	0.31938616
3	0.5665088	0.55978673	0.51823311
K = 5	**R**	**G**	**B**
1	0.48885459	0.50258601	0.28995021
2	0.59856717	0.62950974	0.56305394
3	0.57573141	0.59580927	0.59791034
4	0.43638596	0.37939646	0.30007114
5	0.29818922	0.2500837	0.2215729
K = 7	**R**	**G**	**B**
1	0.66212931	0.66617441	0.339816
2	0.61300356	0.63851975	0.58439228
3	0.28593501	0.27216358	0.23123562
4	0.46282232	0.47927376	0.32970645
5	0.30783012	0.2554444	0.22751973
6	0.4841378	0.41726272	0.32936525
7	0.6470349	0.66921425	0.6772055

K-Means Model

Estimated $\{\mu_k\}_{k=1}^{K}$

K = 10	R	G	B
1	0.43735223	0.43282015	0.40334028
2	0.13405343	0.10791639	0.06204682
3	0.23319132	0.20525792	0.15683932
4	0.57273159	0.61435474	0.62938375
5	0.42963372	0.38858441	0.27309978
6	0.82692133	0.84162308	0.81614287
7	0.31779488	0.28927537	0.24285825
8	0.57059108	0.51055556	0.41532666
9	0.6550932	0.66290971	0.30157831
10	0.72606796	0.67445614	0.53243067

Gaussian Mixture Model

Estimated $\{\mu_k\}_{k=1}^{K}$

K = 10	R	G	B
1	0.66795464	0.70316781	0.69285507
2	0.44732552	0.46378536	0.32634724
3	0.59951915	0.55530273	0.44898934
4	0.50574036	0.42897175	0.33731125
5	0.73510619	0.74245837	0.75129601
6	0.28770914	0.25526214	0.17690552
7	0.2867403	0.28251965	0.25801363
8	0.6273072	0.63420788	0.33361582
9	0.56596775	0.56957737	0.55329508
10	0.30989157	0.25798646	0.23034523

Log Likelihood of GMM