CS680, Spring 2020, Assignment 8

Zhijie Wang, [redacted]

July 11, 2020

Exercise 1

1. Since S_k is a diagonal matrix, we can derive the equation in Line 3 and Line 12 as:

$$\begin{aligned} r_{ik} \quad & \frac{\pi_k}{\sqrt{\prod_{j=1}^{d} s_{kj}}} \exp\left[\sum_{j=1}^{d} -\frac{1}{2}\frac{(x_{ij}-\mu_{kj})^2}{s_{kj}}\right] \\ s_{kj} \quad & \frac{\sum_{i=1}^{n} r_{ik} x_{ij}^2}{\sum_{i=1}^{n} r_{ik}} - \mu_j^2 \end{aligned} \tag{1}$$

where s_{kj} is the jth component of $\text{diag}(S_k)$.

The main computation is happening in Line 3, which is calculating r_{ik}, from (1) we can see that calculating r_{ik} needs $O(d^2)$, and there are n data points and K component of GMM, so the final time complexity is $O(nKd^2)$ per iteration.

For the space complexity, π needs $O(K)$, μ needs $O(dK)$, S needs $O(dK)$, r needs $O(nK)$, therefore the space complexity is $O(K + dK + dK + nK)$

Here we test ex1_1.py on Iris dataset, as we can see from the figure below, the negative loss-likelihood was decreasing monotonically. And to show it's performance, we plot the clustering result based on first two dimensions of Iris features.

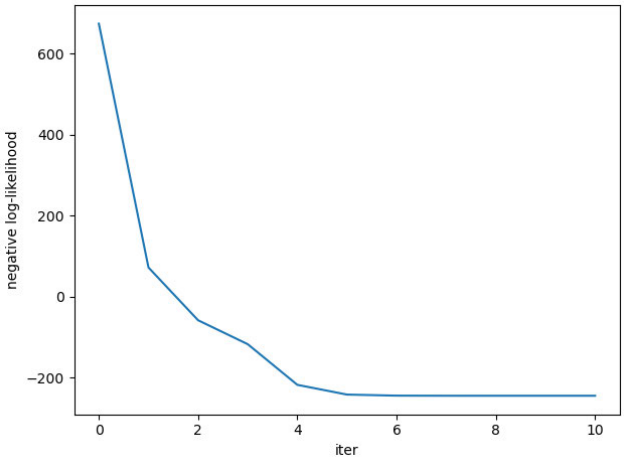

Figure 1: Negative Log-likelihood

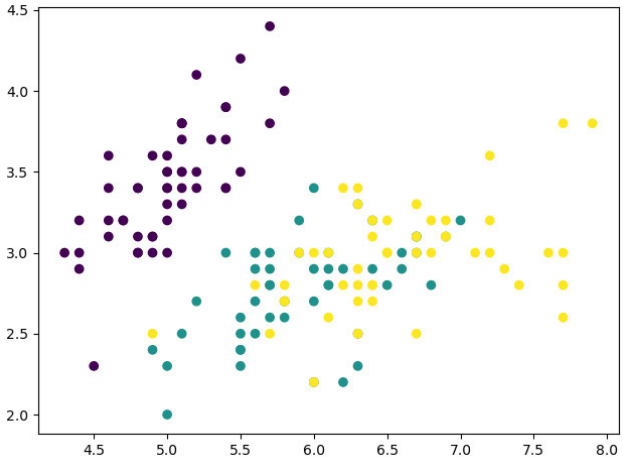

Figure 2: Groundtruth

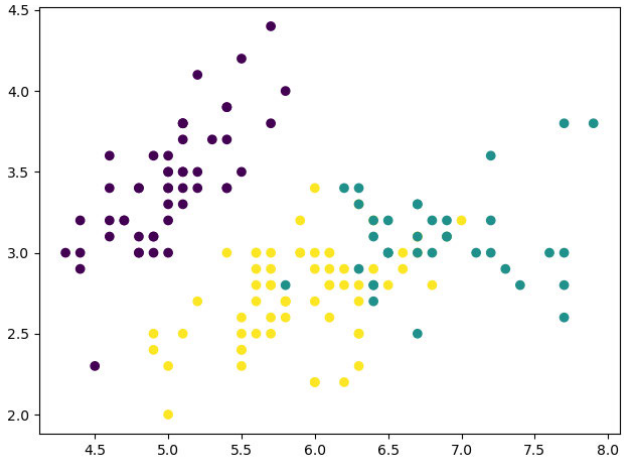

Figure 3: GMM

2. Since S_k is a Identity matrix multiplied by a scalar, we can derive the equation in Line 3 and Line 12 as:

$$\begin{aligned} r_{ik} \quad & \frac{\pi_k}{\sqrt{{s_k}^d}} \exp\left[-\frac{1}{2s_k}(\mathbf{x}_i - \mu_k)^T(\mathbf{x}_i - \mu_k)\right] \\ s_k \quad & \frac{1}{d}\sum_{j=1}^{d} \frac{\sum_{i=1}^{n} r_{ik} x_{ij}^2}{\sum_{i=1}^{n} r_{ik}} - \mu_j^2 \end{aligned} \tag{2}$$

where s_k is the covariance term of $S_k \quad s_k I$.

The main computation is still happening in Line 3, which is calculating r_{ik}, from (2) we can see that calculating r_{ik} needs $O(d)$, and there are n data points and K component of GMM, so the final time complexity is $O(nKd)$ per iteration.

For the space complexity, π needs $O(K)$, μ needs $O(dK)$, S needs $O(K)$, r needs $O(nK)$, therefore the space complexity is $O(K + dK + K + nK)$

Here we still test ex1_2.py on Iris dataset, as we can see from the figure below, the negative loss-likelihood was decreasing monotonically. And to show it's performance, we plot the clustering result based on first two dimensions of Iris features.

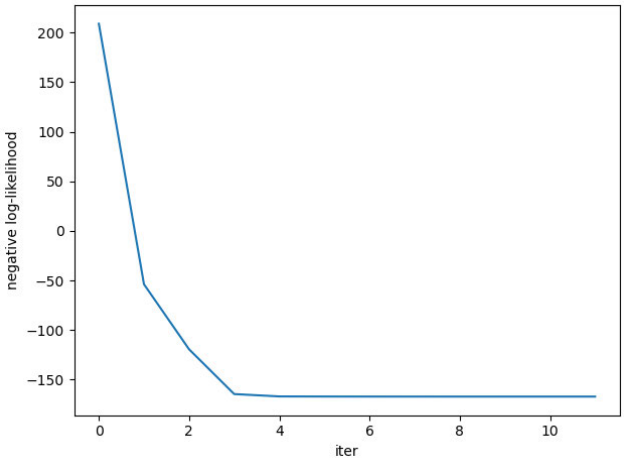

Figure 4: Negative Log-likelihood

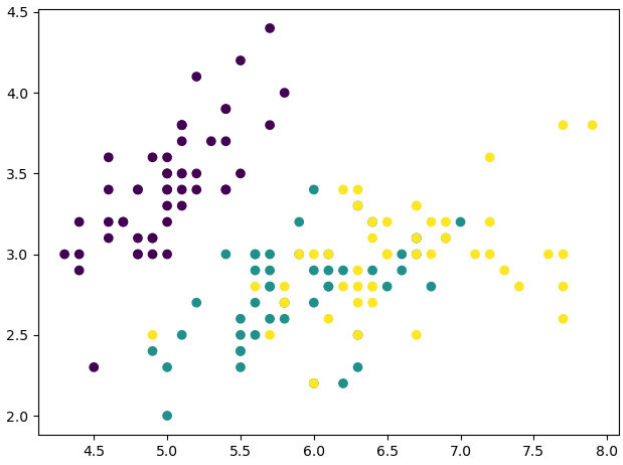

Figure 5: Groundtruth

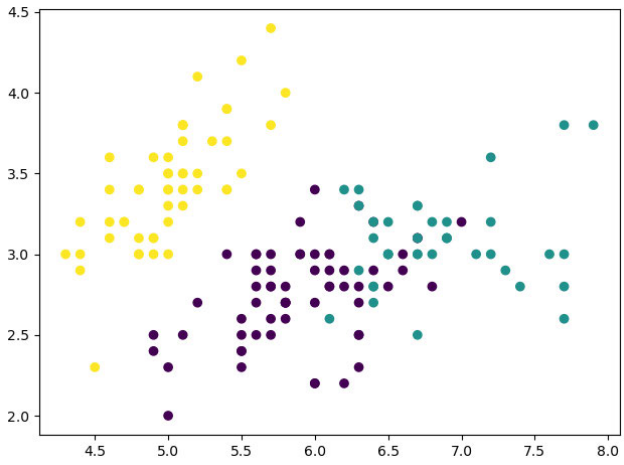

Figure 6: GMM

3. From 1.1 we can know that time complexity is $O(nKd^2)$ for diagonal case, now if we feed d separate GMM, the time complexity will be $O(nK)$ per feature, total time complexity will be $O(nKd)$.

   For the space complexity, 1 GMM will be $O(K + K + K + nK)$ according to 1.1, therefore d separate GMM require $O(d(K + K + K + nK))$ in total.

   Here we still test ex1_3.py on Iris dataset, as we can see from the figure below, the negative loss-likelihood of each GMM was decreasing monotonically. And to show it's performance, we plot the clustering result based on first two dimensions of Iris features. However, the performance is not as good as ex1_1 and ex1_2.

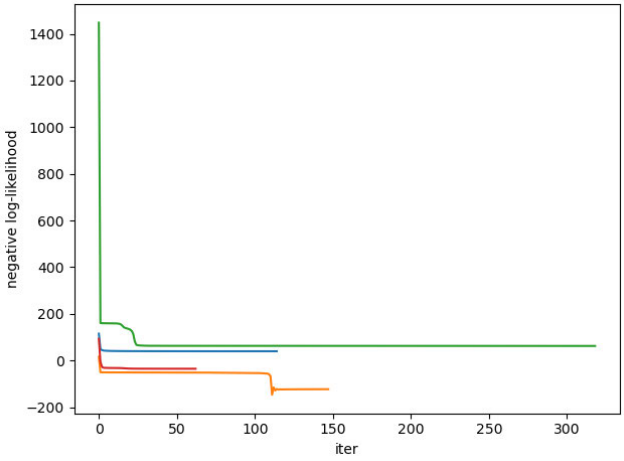

Figure 7: Negative Log-likelihood

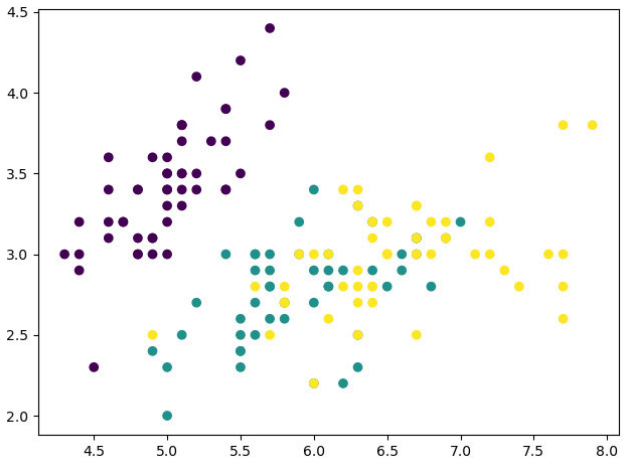

Figure 8: Groundtruth

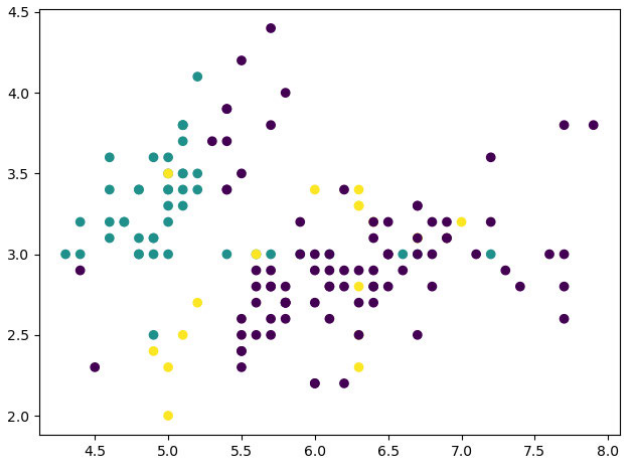

Figure 9: GMM

4. Here we preprocessed the datasets as following: 1). normalized each pixel value to $[0, 1]$, 2). add a Gaussian noise with zero mean and 10^{-3} variance.

   The plot of accuracy w.r.t K is shown below. It looks like K doesn't have much effects in this problem, the accuracy is around 70%.

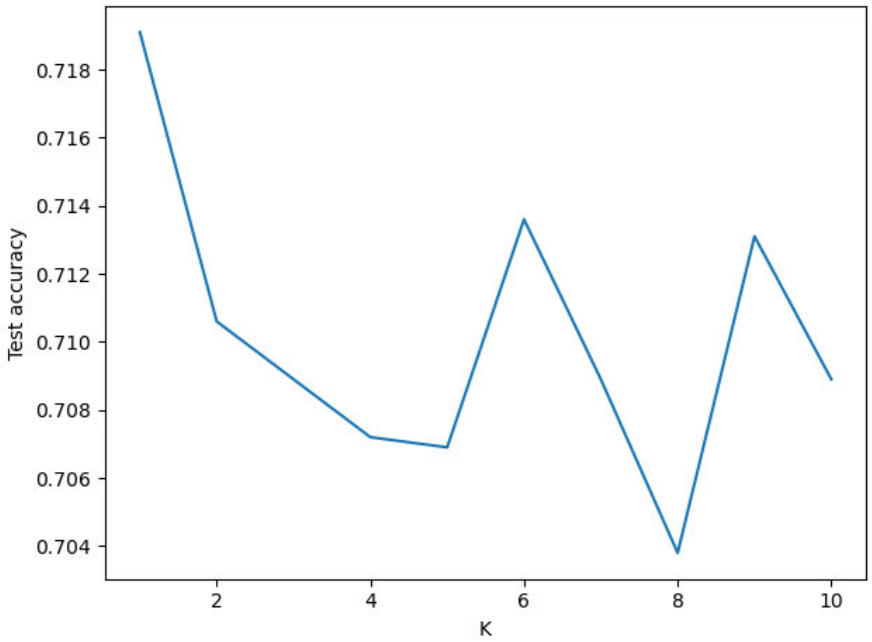

Figure 10: Test accuracy w.r.t K