ISYE 6740 – HW #3

Jing Ma

2025-02-16

Problem 1:

1.1

As discussed in the lecture ([1]), the advantage of KDE is its flexibility as the fit of the distribution can be tuned via the kernel width. Also KDE produces smooth density curve even when there are few data points. In comparison, if we use histogram, there could be empty gaps between bins due to lack of data points. As a result, histogram cannot provide reasonable density estimation in these cases.

However, KDE has higher memory requirements compared to histogram, as a distribution centered around each data point needs to be created, which may not be efficient in situations where it's not desired to keep all the data points.

1.2

MLE cannot be applied directly to estimate parameters of a Gaussian Mixture Model due to the latent variables z^i. As we have learned in the lecture ([1]), we only know that z consists of k components but we don't know the true labels of each component. If z is known, then we could compute the mean and standard deviation for each component using MLE. However, since we do not have this information, we need to marginalize out z and apply MLE to update parameters per the computed τ.

1.3

From the lecture ([2]), we know that τ_k^i represents the posterior probability:

$$\tau_k^i = p\left(z_k^i | x^i, \theta^t\right)$$

So to derive τ_k^i, we need to derive the posterior using Bayes' rule. We know according to Bayes' rule that the following equation stands:

$$p(x^i, z^i|\theta) = p(x^i|z^i, \theta)p(z^i|\theta) = p(z^i|x^i, \theta)p(x^i|\theta)$$

Therefore

$$\tau_k^i = p(z^i|x^i, \theta) = \frac{p(x^i|z^i, \theta)p(z^i|\theta)}{p(x^i|\theta)} = \frac{\pi_z N(x^i|\mu_z, \Sigma_z, \theta)}{\Sigma_{z'}\pi_{z'}N(x^i|\mu_{z'}, \Sigma_{z'}, \theta)}$$

where

$$\text{Prior: } p(z^i|\theta) = \pi_z$$

$$\text{Likelihood: } p(x^i|z^i, \theta) = N(x^i|\mu_z, \Sigma_z, \theta)$$

Above shows how we derive τ_k^i using Bayes' rule.

1.4

To apply MLE for Gaussian random variable using i.i.d. m observations, we first find the likelihood

$$L(x|\mu, \Sigma) = \prod_{i=1}^{m} N(x^i|\mu, \Sigma)$$

We can then take the log likelihood

$$\log L(x|\mu,\Sigma) = \log \prod_{i=1}^{m} N(x^i|\mu,\Sigma) = \sum_{i=1}^{m} \log N(x^i|\mu,\Sigma)$$

From the lecture ([2]), we know that the Multivariate Normal distribution can be expressed as

$$N(X|\mu,\Sigma) = |\Sigma|^{-\frac{1}{2}}\ (2\pi)^{-\frac{m}{2}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)$$

Therefore, we can rewrite the log likelihood function as following

$$\log L(x|\mu,\Sigma) = \sum_{i=1}^{m}\left[-\frac{1}{2}\log|\Sigma| - \frac{m}{2}\log(2\pi) - \frac{1}{2}(x^i-\mu)^T\Sigma^{-1}(x^i-\mu)\right]$$

Finding MLE for $\hat{\mu}$:

Next, we set the derivative of $\log L(x|\mu,\Sigma)$ with respect to μ to 0

$$\frac{\partial L}{\partial \mu} = -\frac{1}{2}\sum_{i=1}^{m}(x^i-\mu)^T\Sigma^{-1}(x^i-\mu)$$

We know from the Matrix Cookbook ([3]) equation #86 that the derivative of a quadratic function can be applied here. Based on the rule, we can express the partial derivative as following

$$\frac{\partial L}{\partial \mu} = -\frac{1}{2}\sum_{i=1}^{m} -2\Sigma^{-1}(x^i-\mu) = \sum_{i=1}^{m}\Sigma^{-1}(x^i-\mu) = 0$$

We treat Σ^{-1} as constant and eliminate from the expression

$$\sum_{i=1}^{m}(x^i-\mu) = \sum_{i=1}^{m} X - m\mu \Rightarrow \hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^i$$

Finding MLE for $\hat{\Sigma}$:

Follow the similar steps above, this time we set the derivative of $\log L(x|\mu,\Sigma)$ with respect to Σ to 0

$$\frac{\partial L}{\partial \Sigma} = \sum_{i=1}^{m}\left[-\frac{1}{2}\frac{\partial}{\partial\Sigma}\log|\Sigma| - \frac{1}{2}\frac{\partial}{\partial\Sigma}(x^i-\mu)^T\Sigma^{-1}(x^i-\mu)\right]$$

Per the Matrix Cookbook ([3]) equation #57

$$\frac{\partial \ln|\det(X)|}{\partial X} = (X^{-1})^T = (X^T)^{-1}$$

Since Σ is a covariance matrix, which is symmetrical positive semi-definite, $\ln|\det(\Sigma)| = \ln\det(\Sigma)$, so we can rewrite this expression as

$$\frac{\partial \ln|\det(\Sigma)|}{\partial \Sigma} = (\Sigma^{-1})^T = \Sigma^{-1} \tag{1}$$

Further, based on Matrix Cookbook ([3]) equation #11

$$\mathrm{Tr}(A) = \sum_i A_{ii}$$

Since $(x^i-\mu)^T\Sigma^{-1}(x^i-\mu)$ produces a scalar, and the trace of a scalar is the scalar itself, we can use this property to derive the following

$$-\frac{1}{2}\frac{\partial}{\partial\Sigma}(x^i-\mu)^T\Sigma^{-1}(x^i-\mu) = -\frac{1}{2}\frac{\partial}{\partial\Sigma}\operatorname{Tr}\Big((x^i-\mu)^T\Sigma^{-1}(x^i-\mu)\Big) \tag{2}$$

Additionally, according to Matrix Cookbook ([3]) equation #63

$$\frac{\partial}{\partial X}\operatorname{Tr}(AX^{-1}B) = -(X^{-1}BAX^{-1})^T$$

Let $A = (x^i-\mu)^T$ and $B = (x^i-\mu)$, we can rewrite Equation 2 as

$$\begin{aligned}-\frac{1}{2}\frac{\partial}{\partial\Sigma}(x^i-\mu)^T\Sigma^{-1}(x^i-\mu) &= \frac{1}{2}\Big(\Sigma^{-1}(x^i-\mu)(x^i-\mu)^T\Sigma^{-1}\Big)^T \\ &= \frac{1}{2}(\Sigma^{-1})^T\Big((x^i-\mu)^T\Big)^T(x^i-\mu)^T(\Sigma^{-1})^T \\ &= \frac{1}{2}\Sigma^{-1}(x^i-\mu)(x^i-\mu)^T\Sigma^{-1}\end{aligned} \tag{3}$$

Now we combine both terms from Equation 1 and Equation 3

$$-\frac{m}{2}\Sigma^{-1} + \frac{1}{2}\sum_{i=1}^{m}\Sigma^{-1}(x^i-\mu)(x^i-\mu)^T\Sigma^{-1} = 0$$

We can factor out Σ^{-1} and constant to simplify the expression

$$\Sigma^{-1}\left(mI - \sum_{i=1}^{m}(x^i-\mu)(x^i-\mu)^T\Sigma^{-1}\right) = 0$$

For this equation to hold true, the term inside the parentheses must be 0

$$mI = \sum_{i=1}^{m}(x^i-\mu)(x^i-\mu)^T\Sigma^{-1}$$

Next, we multiply Σ on both sides

$$m\Sigma = \sum_{i=1}^{m}(x^i-\mu)(x^i-\mu)^T$$

$$\hat{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x^i-\mu)(x^i-\mu)^T$$

Rank of Σ:

When $m < n$, from the discussion provided by Glen_b ([4]), we can see that the covariance matrix is rank deficient. This is because the rank of the sample data, D, is $\min(m, n)$. Since the rank of covariance matrix, Σ, will be no larger than the rank of D, we know that $\operatorname{rank}(\Sigma) < n$.

When $m = 1$, covariance matrix has rank of 0 as there is only one data sample. We can see the reason based on the following expression derived above

$$\hat{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x^i-\mu)(x^i-\mu)^T$$

When there is a single data sample, $\mu = x$, and therefore $\Sigma = 0$.

Problem 2:

2.1

In this question, we are exploring the distributions of two variables, "amygdala" and "acc", via one-dimensional histogram and KDE.

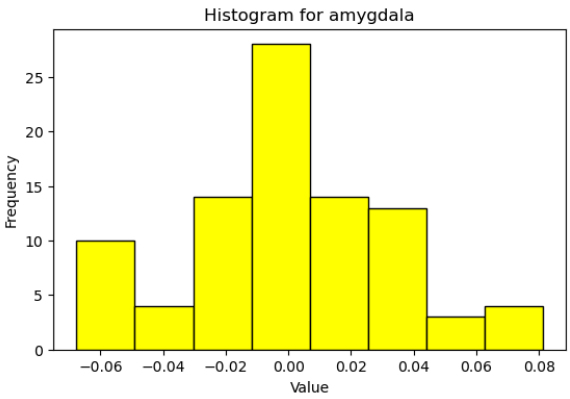
We find the optimal numbers of bins using the Freedman-Diaconis rule ([5]). According to this rule, the bin-width is computed as following

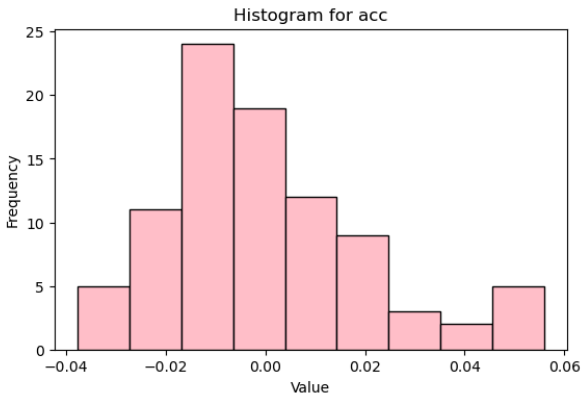
$$\text{Bin width} = 2\frac{\text{IQR}(x)}{\sqrt[3]{n}}$$

We then calculate the number of bins using the following

$$\#\text{ of bins} = \left\lceil \frac{\max(x) - \min(x)}{\text{bin width}} \right\rceil$$

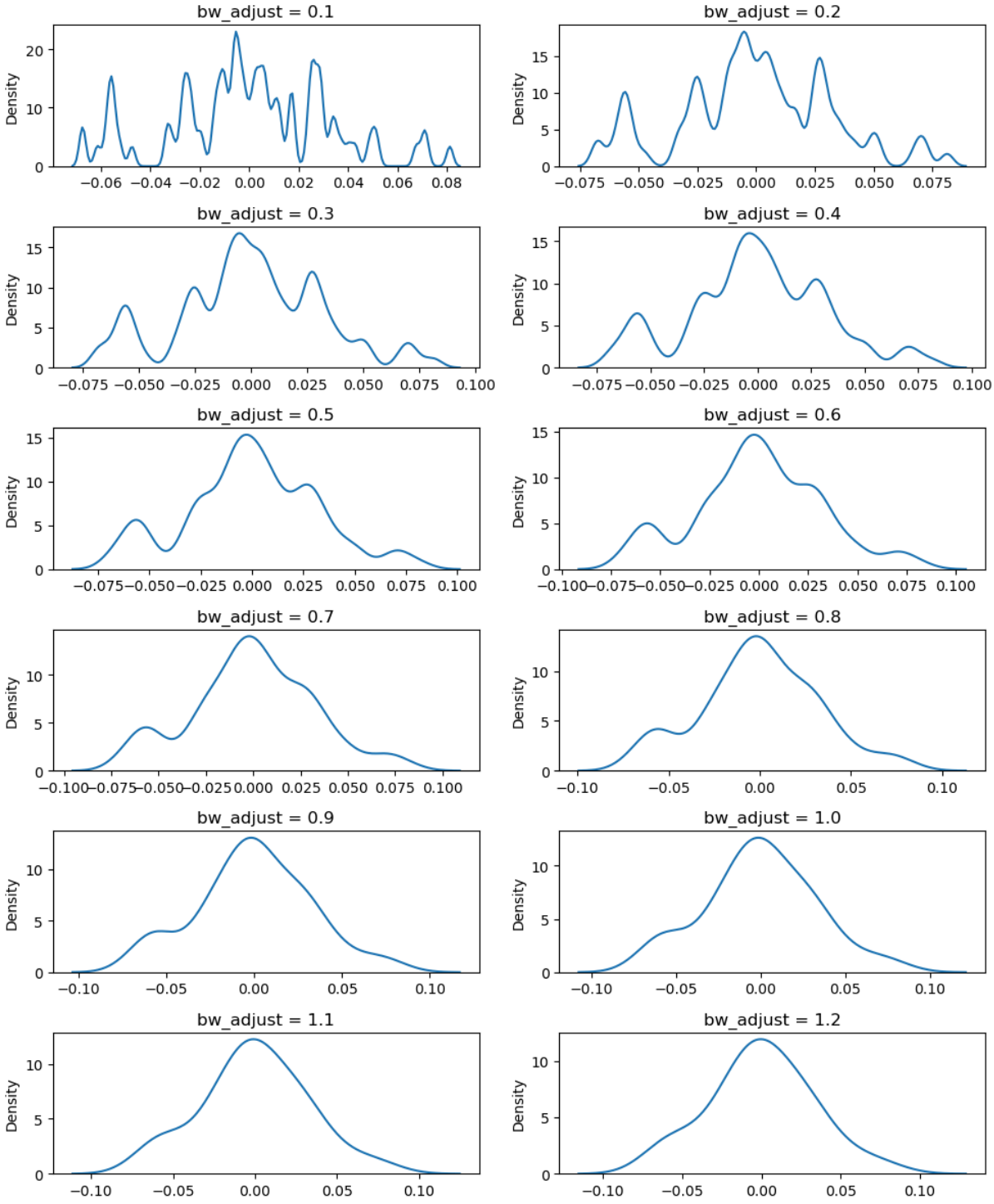
Based on the computation, we obtained the optimal number of bins of 8 and 9 for "amygdala" and "acc", respectively.



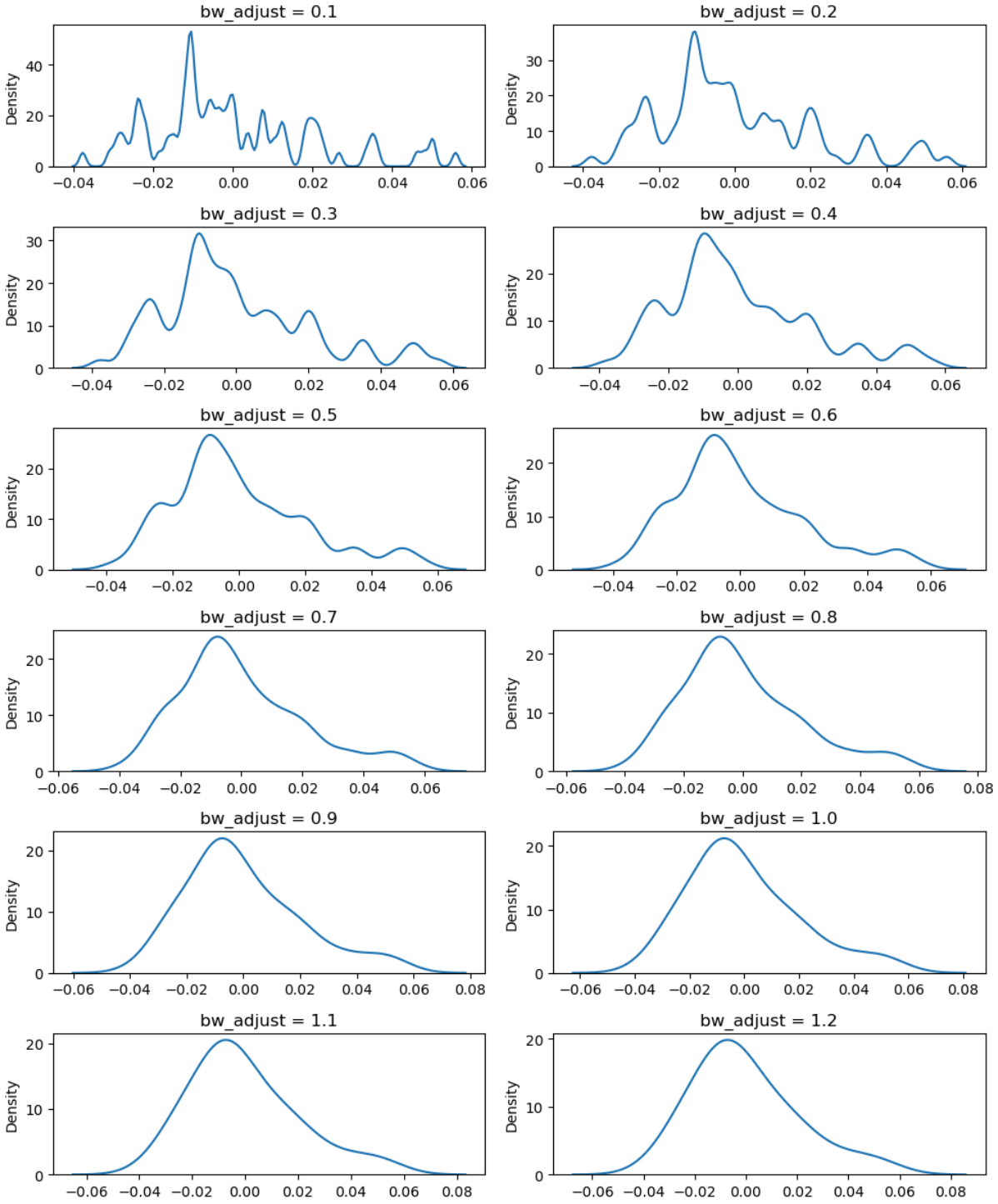


For plotting 1D KDE plot, we experimented with varying kernel bandwidth by tuning the "bw_adjust" parameter of the Seaborn kdeplot() function. "bw_adjust" does not change the kernel bandwidth itself, but rather serves as a scaling factor, which still directly influences the outcome of the plot.

The following are the 1D KDE plot for "amygdala" with 12 different "bw_adjust" values:



Similarly, below shows the 1D KDE plot for "acc" with 12 varying "bw_adjust" values:



As we can see, when "bw_adjust" is too small, the curve shows many drastic peaks and troughs. As the scaling factor gradually increases, the curve starts to take a more smooth look. However, to more accurately depict the density of the distribution, we would like to keep the main traits of the distribution without too much variance nor too smooth-looking. We judgmentally decide on "bw_adjust" = 0.9 for "amygdala" and 1.0 for "acc".

Based on the documentation of Seaborn kdeplot() function, we know that the bandwidth is passed to scipy.stats.gaussian_kde. So to retrieve the actual kernel bandwidth, we need to call scipy.stats.gaussian_kde on the data for each variable, obtain the covariance_factor and multiply

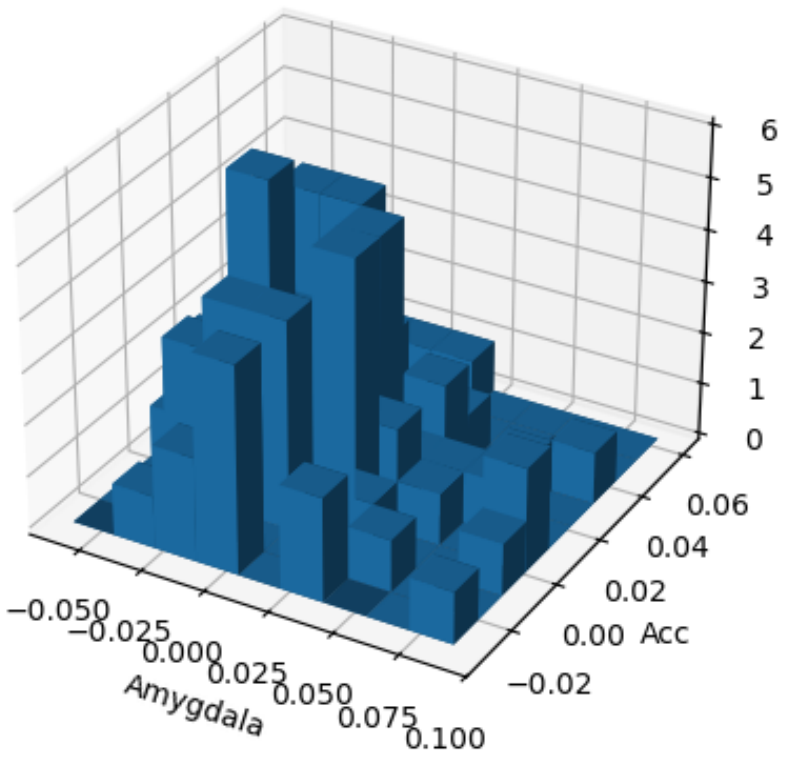
with the standard deviation of the data. Lastly, we scale the bandwidth by the value that we specified in "bw_adjust" to obtain the true bandwidth ([6]).

By following the steps above, we found the kernel bandwidth for "amygdala" and "acc" to be 0.012 and 0.008, respectively.

2.2

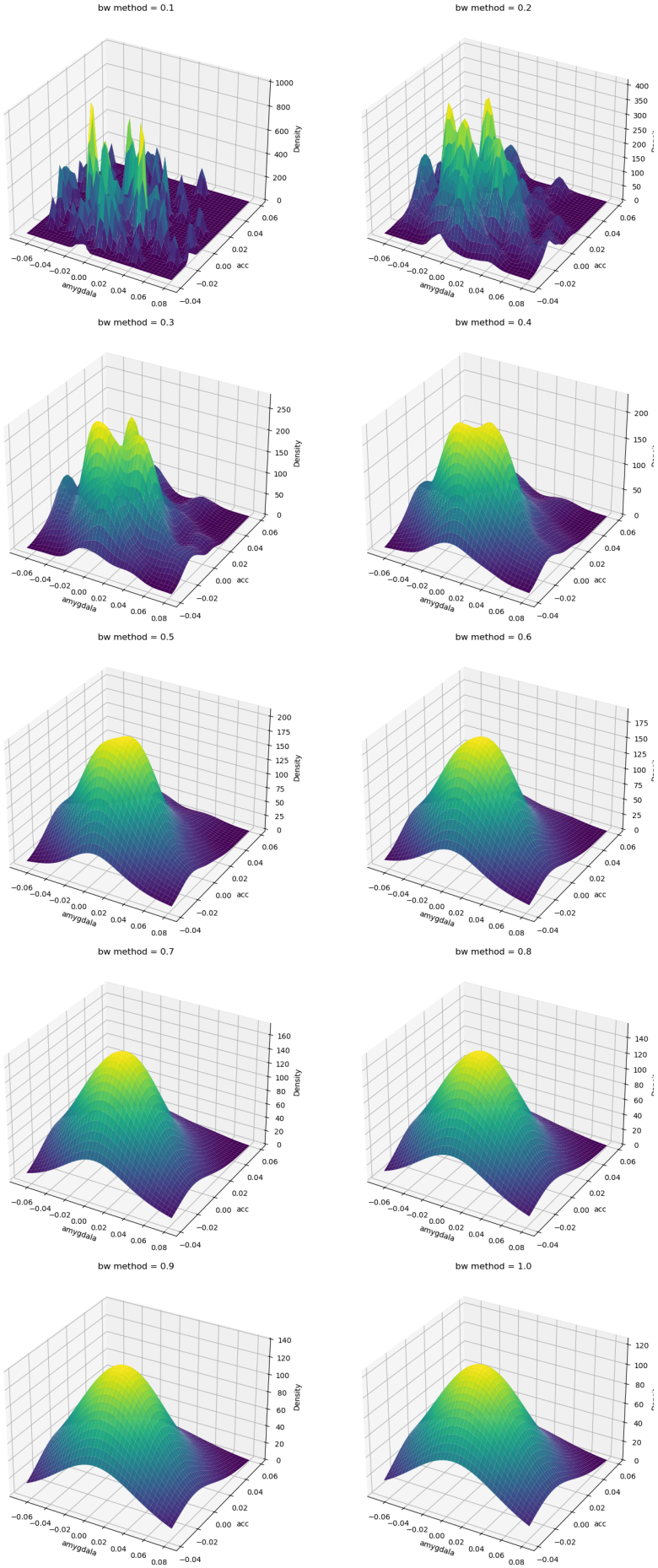
In part 2, we want to plot the 2D histogram for the pairs of variables ("amygdala" and "acc"). Similarly, we also want to determine the suitable number of bins. To achieve this, we again turn to Freedman-Diaconis' Rule. In the 2D space, we pick the larger number of bins between the two variables to ensure that we attain all the information ([7]).

Based on our computation in part 1, we have observed that the number of bins for "acc" is 9, which is larger than that of "amygdala", and therefore we naively choose 9 as our optimal number of bins for plotting the 2D histogram.



2.3

In part 3, we create a 3D KDE plot with both variables and experimented with multiple kernel bandwidth below


bw method = 0.1
bw method = 0.2
bw method = 0.3
bw method = 0.4
bw method = 0.5
bw method = 0.6
bw method = 0.7
bw method = 0.8
bw method = 0.9
bw method = 1.0
amygdala
acc
Density

Similar to the 2D KDE plot, we can observe that as the "bw_method" value increases, the plot becomes increasingly smoother. Again, we judgmentally pick "bw_method" = 1.0 based on visual inspection, though the plots produced using "bw_method" = 0.5 through 1.0 closely resemble each other.

The actual kernel bandwidths are extracted by accessing the covariance matrix

$$\text{kernel bandwidth} = \begin{pmatrix} 1.06311022e-03 & -8.56093196e-05 \\ -8.56093196e-05 & 4.17604800e-04 \end{pmatrix}$$

We can observe from the plot ("bw_method" = 1.0) that the distribution appears unimodal. Additionally, there doesn't seem to exist any obvious outliers. Typically, outliers sit in the regions far away from the central area, and we can indeed see that toward the region where amygdala = 0.08 and acc = 0, the density color appears darker, indicating less probability of occurring. This could suggest possible outliers,but we cannot determine definitively based on visual inspection alone.

Lastly, we want to check the dependence between the two variables. Since there may exist non-linear relationship, if we simply use covariance matrix or Pearson correlation method, it may not return meaningful result. Therefore, we resort to computing the mutual information, which is computed as follows ([8])

$$I(X,Y) = \sum_X \sum_Y p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

where

$$I(X,Y) = \text{MI between variables x and y}$$

$$p(x,y) = \text{The joint probability of the two variables}$$

If two variables are independent, we have

$$p(x,y) = p(x)p(y)$$

Then the log term becomes

$$\log\left(\frac{p(x,y)}{p(x)p(y)}\right) = \log 1 = 0$$

Therefore, if two variables are independent, MI becomes 0.

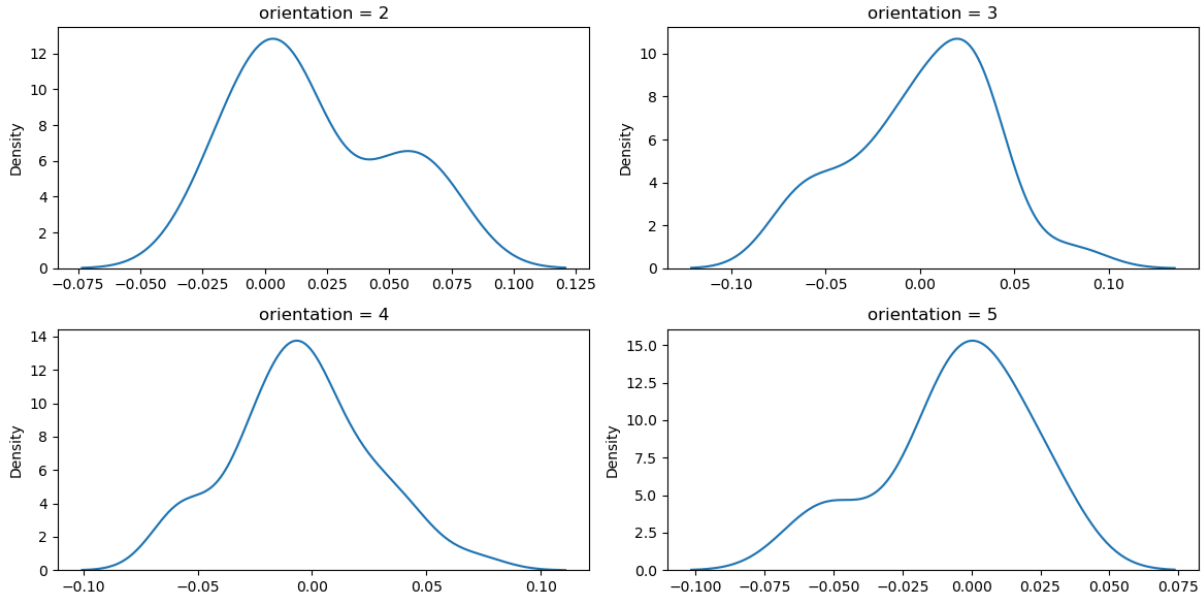
To compute MI between "amygdala" and "acc", we can simply call sklearn.metrics.mutual_info_score. We obtained MI = 4.355 (rounded), which evidences that these two variables are not independent.

2.4

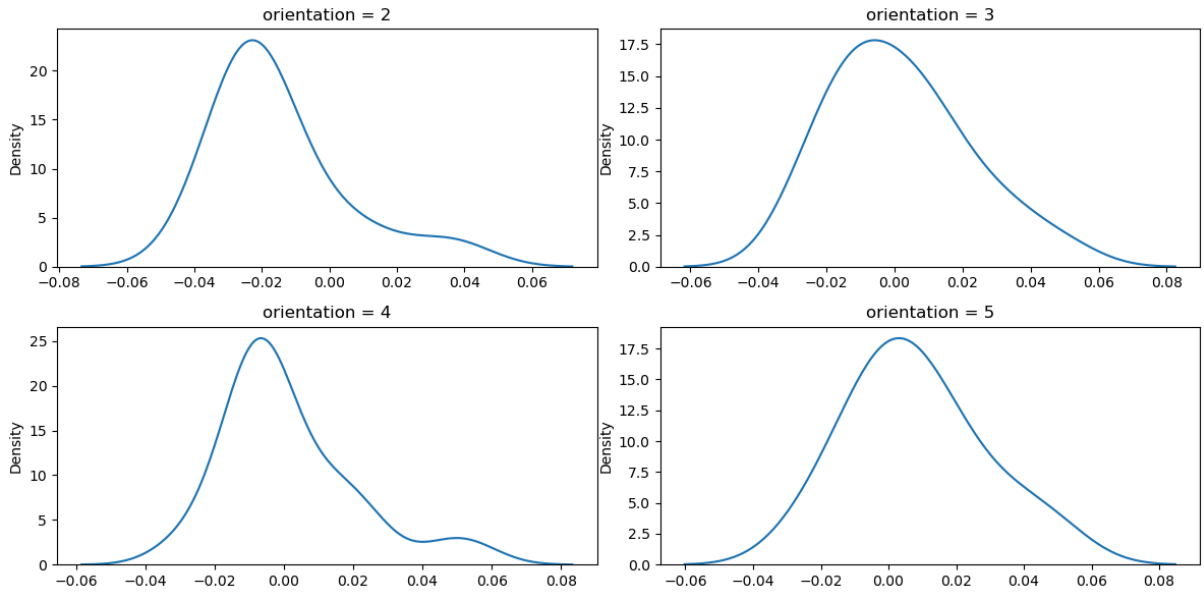
In this section, we want to find out the sample means of the conditional distributions of "amygdala" and "acc", both conditioning on the political orientation column that takes the values between $[2, 5]$.

We approached this problem by retrieving all values of "amygdala" and "acc" based on the "orientation" values, respectively. We then plot KDE using the "bw_adjust" values previously selected in part 1. Recall that we picked "bw_adjust" of 0.9 for "amygdala" and 1.0 for "acc".

The following 2D KDE plots are produced for "amygdala" conditioned on "orientation" values



Similarly, the following plots are produced for "acc"



To compute the sample means of each conditional distributions, we can simply find the means of the sample data associated with the respective "orientation" values. Below is the summary:

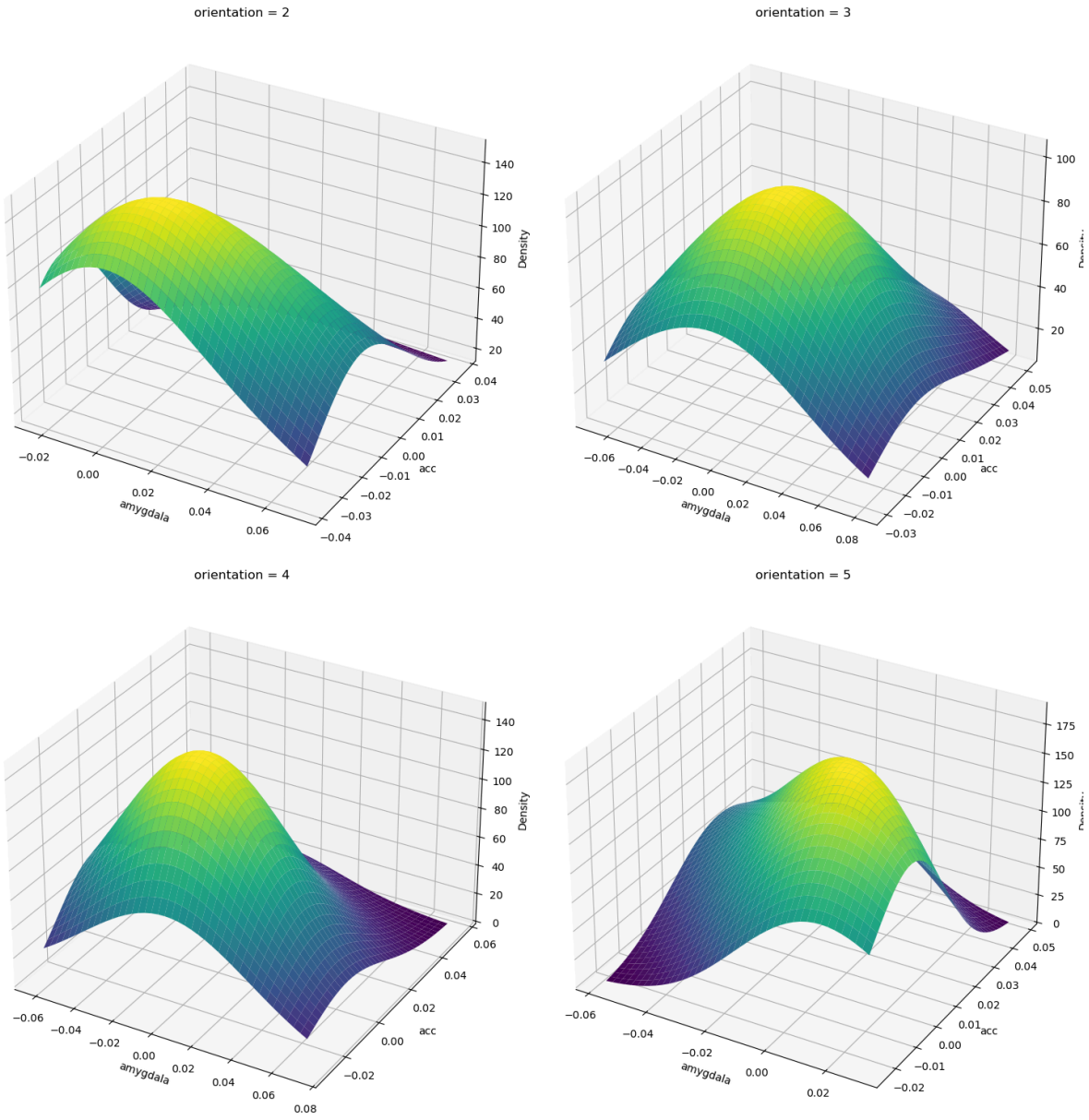
	c = 2	c = 3	c = 4	c = 5
amygdala	0.02	0.	0.	−0.01
acc	−0.01	0.	0.	0.01

Based on visual inspection, we can see that for "amygdala", the KDE plots look slightly different for each "orientation" value, including the shape and shift of the peak. We can observe similar phenomenon for "acc". Further, combined with the sample means of each conditional distribution, we can see that the distributions do differ for each variable and "orientation" values, though the change appears minor.

2.5

In the last part, we want to plot the joint conditional distribution with both variables while conditioned on "orientation". We used "bw_method" = 1.0, which we selected in part 3.

Below are the 3D KDE plots



Via visual inspection, we can clearly see shifts in the modal and shape in each conditional distribution. This shows that there is a difference between brain structure and political view.

Problem 3:

3.1

In this question, we are given two classes of digits, 2 and 6, from the MNIST handwritten digits dataset. To implement EM algorithm, the setup is as follows:

1. Use PCA to reduce the dimensionality to four. Project the dataset to these four dimensions;

2. Initialize prior (π), mean (μ), and covariance matrix (Σ). Given that there are two clusters, we know that both prior and mean are a vector of two elements (scalars), and the covariance matrix is a list of two 4×4 matrices;
3. Compute the posterior (τ) which is $m \times K$ as we have different posterior for each data point and mixture;
4. Loop through each iteration either up to maxIter (100) or until achieving convergence (change $<$ tolerance level);
5. Expectation step: For each mixture, compute the un-normalized posterior. Compute the log likelihood using the following expression:

$$l(\theta; D) = \sum_{i=1}^{m} \log \left(\sum_{z^i=1}^{K} p(x^i, z^i|\theta) \right)$$

6. Normalize the posterior;
7. Maximization step: For each mixture, update the prior, mean, and covariance matrix using the following expressions

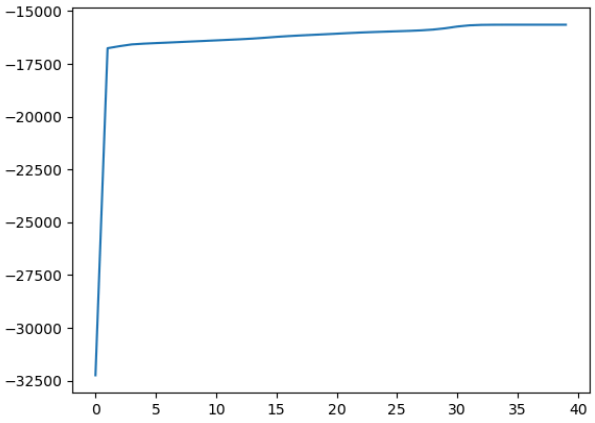
$$\pi_k = \frac{\sum_i \tau_k^i}{m}$$

$$\mu_k = \frac{\sum_i \tau_k^i x^i}{\sum_i \tau_k^i}$$

$$\Sigma_k = \frac{\sum_i \tau_k^i (x^i - \mu_k)(x^i - \mu_k)^T}{\sum_i \tau_k^i}$$

8. Break the for loop once the algorithm has converged.

The plot below shows that the algorithm continuously improves till convergence



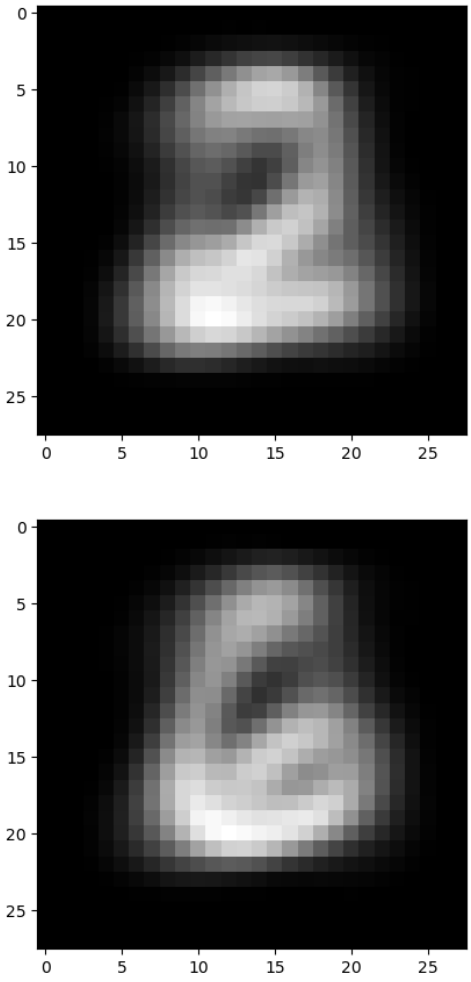
As we can observe from the plot, the log-likelihood improves significantly in less than 5 iterations and then slowly steps toward convergence (stops at iteration #40).

3.2

From part 1, we obtained the following weights for each component

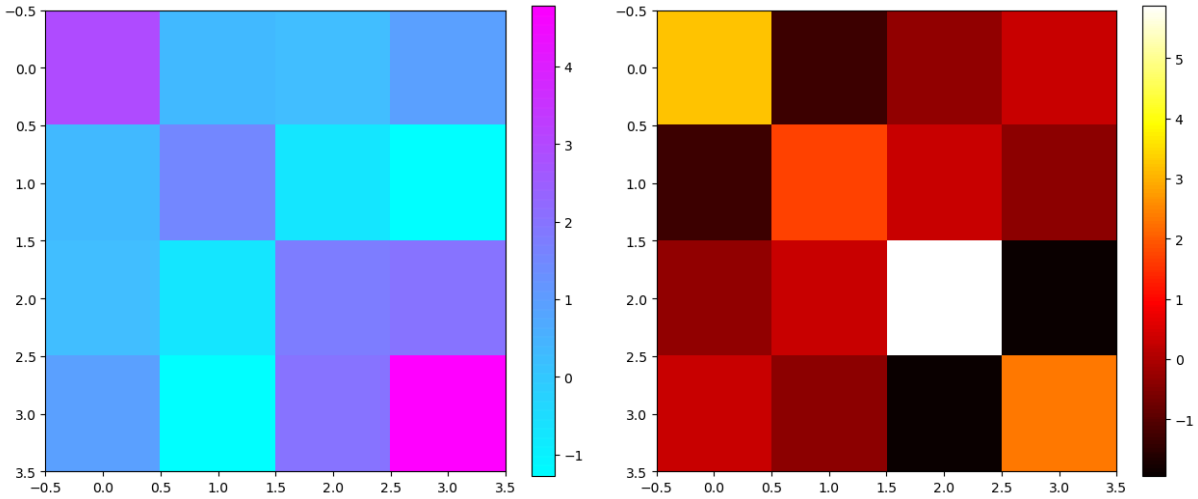
$$\pi_{k=1} = 0.487, \pi_{k=2} = 0.513$$

We also mapped each component back to the original space and obtained the images below



As we can see that the images are bit blurry. This could be due to various reasons. First, Gaussian distribution models areas using probability density, which smoothes the sharp edges. Second, Bishop ([9]) pointed out that there generally exists multiple local maxima of the log likelihood function, so EM is not guaranteed to find the largest of these maxima.

Lastly, we have the heatmap plots of the covariance matrices for both mixtures below



Depending on the color intensity, each block of color represents the corresponding covariance values.

3.3

In the last part, we want to find out the mis-classification for digits "2" and "6" and compare the results to those of K-means.

GMM Labels	True Labels: 2	True Labels: 6
2	966	9
6	66	949

We can then compute the misclassification rate for each class of digits using the formula

$$\text{Misclassification Rate} = \frac{\text{False Negative} + \text{False Positive}}{\text{Total True Label}}$$

We found that the misclassification rate for digit "2" is 0.073 and for digit "6" is 0.078.

In comparison, we also ran K-means to perform classification and computed the misclassification rates using the same method. Below is the confusion matrix

KMeans Labels	True Labels: 2	True Labels: 6
2	968	76
6	64	882

We found that using K-means, the misclassification rate for digit "2" is 0.136 and for digit "6" is 0.146, almost doubled compared to EM algorithm. Therefore, we can conclude that EM algorithm shows better performance given the higher accuracy.

Problem 4:

4.1

4.1.a

In this question, we are given the following distributions:

$$y^{(p)} \sim N\left(\mu_p, \sigma_p^2\right)$$

$$z^{(r)} \sim N\left(\nu_r, \tau_r^2\right)$$

$$x^{(pr)} | y^{(p)}, z^{(r)} \sim N\left(y^{(p)} + z^{(r)}, \sigma^2\right)$$

Further, we know from the question that the variables $y^{(p)}$ and $z^{(r)}$ are independent. The variance σ^2 for $x^{(pr)}$ should be treated as a fixed known constant.

Additionally, we are given that $x^{(pr)} = y^{(p)} + z^{(r)} + \varepsilon^{(pr)}$, where $\varepsilon^{(pr)} \sim N\left(0, \sigma^2\right)$ is independent Gaussian noise. This suggests that we should treat $\varepsilon^{(pr)}$ as independent with $y^{(p)}$ and $z^{(r)}$.

First, we want to find the mean vector in terms of the parameters $\mu_p, \sigma_p^2, \nu_r, \tau_r^2$ and σ^2.

We know from the distributions of $y^{(p)}$ and $z^{(r)}$ that they have mean of μ_p and ν_r, respectively. Further, we know from the given distribution of $x^{(pr)}$ that its mean can be written as $\mu_p + \nu_r$, so we have the mean vector:

$$\text{mean vector } \mu = \begin{pmatrix} \mu_p \\ \nu_r \\ \mu_p + \nu_r \end{pmatrix}$$

Next, we find the covariance matrix. We know that a covariance matrix consists of variances on the diagonal and covariances off-diagonal. So we have the following

$$\operatorname{cov}\big(y^{(p)}, y^{(p)}\big) = \operatorname{var}\big(y^{(p)}\big) = \sigma_p^2$$

$$\operatorname{cov}\big(z^{(r)}, z^{(r)}\big) = \operatorname{var}\big(z^{(r)}\big) = \tau_r^2$$

$$\begin{aligned} \operatorname{cov}\big(x^{(pr)}, x^{(pr)}\big) &= \operatorname{var}\big(x^{(pr)}\big) \\ &= \operatorname{var}\big(y^{(p)} + z^{(r)} + \sigma^2\big) \\ &= \sigma_p^2 + \tau_r^2 + \sigma^2 \end{aligned}$$

Now we want to find the covariances among each pair of variables

$$\begin{aligned} \operatorname{cov}\big(x^{(pr)}, y^{(p)}\big) &= \operatorname{cov}\big(y^{(p)} + z^{(r)} + \varepsilon^{(pr)}, y^{(p)}\big) \\ &= \operatorname{cov}\big(y^{(p)}, y^{(p)}\big) + \operatorname{cov}\big(z^{(r)}, y^{(p)}\big) + \operatorname{cov}\big(\varepsilon^{(pr)}, y^{(p)}\big) \end{aligned}$$

Given that $z^{(r)}$ and $\varepsilon^{(pr)}$ are independent of $y^{(p)}$, respectively, the covariances are 0. So we can simplify the above covariance as

$$\operatorname{cov}\big(x^{(pr)}, y^{(p)}\big) = \operatorname{cov}\big(y^{(p)}, y^{(p)}\big) = \mu_p^2$$

Similarly

$$\operatorname{cov}\big(x^{(pr)}, z^{(r)}\big) = \operatorname{cov}\big(z^{(r)}, z^{(r)}\big) = \tau_r^2$$

Therefore, we have the covariance matrix (row from top to bottom and column from left to right are $y^{(p)}$, $z^{(r)}$ and $x^{(pr)}$)

$$\Sigma = \begin{pmatrix} \sigma_p^2 & 0 & \sigma_p^2 \\ 0 & \tau_r^2 & \tau_r^2 \\ \sigma_p^2 & \tau_r^2 & \sigma_p^2 + \tau_r^2 + \sigma^2 \end{pmatrix}$$

4.1.b

In this part, we want to derive an expression for $Q_{pr}(\theta'|\theta) = E\big[\log p\big(y^{(p)}, z^{(r)}, x^{(pr)}\big)|x^{(pr)}, \theta\big]$.

This function can be decomposed as the following

$$Q_{pr}(\theta'|\theta) = E\big[\log p\big(x^{(pr)}|y^{(p)}, z^{(r)}\big)\big] + E\big[\log p\big(y^{(p)}|\theta'\big)\big] + E\big[\log p\big(z^{(r)}|\theta'\big)\big]$$

Since this is a function of θ', let

$$\theta' = \begin{pmatrix} \mu_p' \\ \sigma_p'^2 \\ \nu_r' \\ \tau_r'^2 \end{pmatrix}$$

denote the proposed parameters to be updated, whereas θ represents the current guesses of the parameters.

Given

$$x^{(pr)}|y^{(p)}, z^{(r)} \sim N\big(y^{(p)} + z^{(r)}, \sigma^2\big)$$

We have

$$\begin{aligned} E\big[\log p\big(x^{(pr)}|y^{(p)}, z^{(r)}\big)\big] &= E\left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{\big(x^{(pr)} - y^{(p)} - z^{(r)}\big)^2}{2\sigma^2}\right] \\ &= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{E\Big[\big(x^{(pr)} - y^{(p)} - z^{(r)}\big)^2\Big]}{2\sigma^2} \end{aligned}$$

Similarly, we can express $E\big[\log p\big(y^{(p)}|\theta'\big)\big]$ and $E\big[\log p\big(z^{(r)}|\theta'\big)\big]$ as

$$E\big[\log p\big(y^{(p)}|\theta'\big)\big] = -\frac{1}{2}\log\big(2\pi\sigma_p'^2\big) - \frac{E\Big[\big(y^{(p)} - \mu_p'\big)^2|x^{(pr)}, \theta\Big]}{2\sigma_p'^2}$$

$$E\big[\log p\big(z^{(r)}|\theta'\big)\big] = -\frac{1}{2}\log(2\pi\tau_r'^2) - \frac{E\Big[\big(z^{(r)} - \nu_r'\big)^2|x^{(pr)}, \theta\Big]}{2\tau_r'^2}$$

Combining all the terms above, we obtain

$$\begin{aligned} Q_{pr}(\theta'|\theta) = &-\frac{1}{2}\log(2\pi\sigma^2) - \frac{E\Big[\big(x^{(pr)} - y^{(p)} - z^{(r)}\big)^2\Big]}{2\sigma^2} \\ &-\frac{1}{2}\log\big(2\pi\sigma_p'^2\big) - \frac{E\Big[\big(y^{(p)} - \mu_p'\big)^2|x^{(pr)}, \theta\Big]}{2\sigma_p'^2} \\ &-\frac{1}{2}\log(2\pi\tau_r'^2) - \frac{E\Big[\big(z^{(r)} - \nu_r'\big)^2|x^{(pr)}, \theta\Big]}{2\tau_r'^2} \end{aligned}$$

4.2

To implement the Maximization step, we can set the partial derivative to zero with respect to each parameter that we want to update

$$\frac{\partial Q_{pr}(\theta'|\theta)}{\partial \mu_p'} = -\frac{E\big[y^{(p)} - \mu_p'|x^{(pr)}, \theta\big]}{\sigma_p'^2} = 0 \Rightarrow \mu_p' = E\big[y^{(p)}|x^{(pr)}, \theta\big]$$

$$\frac{\partial Q_{pr}(\theta'|\theta)}{\partial \sigma_p'^2} = -\frac{1}{2\sigma_p'^2} + \frac{E\Big[\big(y^{(p)} - \mu_p'\big)^2|x^{(pr)}, \theta\Big]}{2\sigma_p'^4} = 0 \Rightarrow \sigma_p'^2 = E\Big[\big(y^{(p)} - \mu_p'\big)^2|x^{(pr)}, \theta\Big]$$

Similarly, we can follow the steps for $z^{(r)}$ as we did for $y^{(p)}$ and get the following updated parameters

$$\nu_r' = E\big[z^{(r)}|x^{(pr)}, \theta\big]$$

$$\tau_r'^2 = E\Big[\big(z^{(r)} - \nu_r'\big)^2|x^{(pr)}, \theta\Big]$$

References:

1. Topic 6 Density Estimation lecture slides.
2. Topic 7 GMM and EM lecture slides.

3. Matrix Cookbook.
4. https://stats.stackexchange.com/questions/60622/why-is-a-sample-covariance-matrix-singular-when-sample-size-is-less-than-number
5. https://stats.stackexchange.com/questions/798/calculating-optimal-number-of-bins-in-a-histogram
6. https://stackoverflow.com/questions/23630515/getting-bandwidth-used-by-scipys-gaussian-kde-function#:~:text=The%20bandwidth%20is%20kernel.,thumb%20in%20the%20default%20case
7. https://stats.stackexchange.com/questions/114490/optimal-bin-width-for-two-dimensional-histogram
8. https://www.blog.trainindata.com/mutual-information-with-python/
9. Pattern Recognition and Machine Learning. Christopher Bishop.