Sims_Kelly_HW3_report

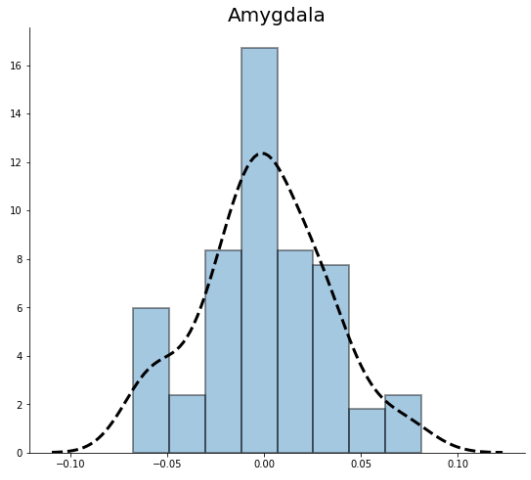
September 29, 2020

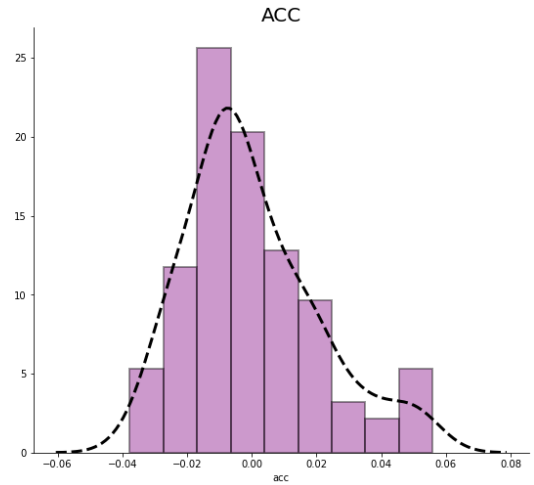
Q1

Density Estimation: Psychological Experiments

We will use this data to study whether or not the two brain regions are likely to be independent of each other and considering different types of political view For this question; you can use the proper package for histogram and KDE; no need to write your own. The data set n90pol.csv contains information on 90 university students who participated in a psychological experiment designed to look for relationships between the size of different regions of the brain and political views. The variables amygdala and acc indicate the volume of two particular brain regions known to be involved in emotions and decision-making, the amygdala and the anterior cingulate cortex; more exactly, these are residuals from the pre- dicted volume, after adjusting for height, sex, and similar body-type variables. The variable orientation gives the students' locations on a five-point scale from 1 (very conservative) to 5 (very liberal). Note that in the dataset, we only have observations for orientation from 2 to 5.

1. Form the 1-dimensional histogram and KDE to estimate the distrbutions of amygdala and acc, respectively. For this question, you can ignore the variable orientation. Answer





2. Form 2-dimensional histogram for the pairs of variables (amygdala, acc) Decide on a suitable number of bins so you can see the shape of the distribution clearly Also

use kernel-density-estimation (KDE) to estimate the 2-dimensional density function of (amygdala, acc). Use a simple multi-dimensional Gaussian kernel, for

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$$

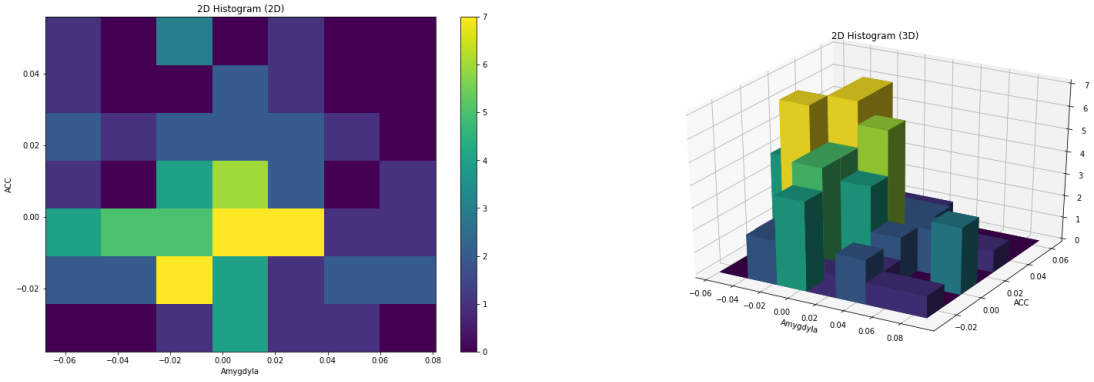
, where x_1 and x_2 are the two dimensions respectively.

$$K(x) = \frac{1}{2\pi} e^{-\frac{(x_1)^2+(x_2)^2}{2}}$$

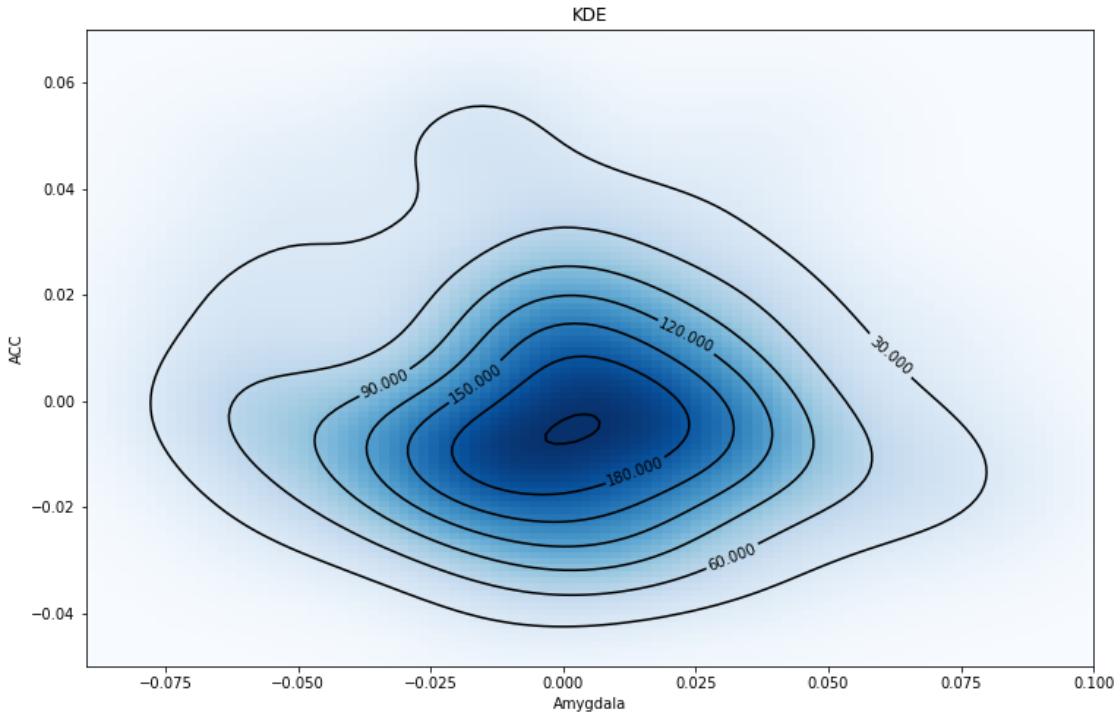
. Recall in this case, the kernel density estimator (KDE) for a density is given by.

$$p(x) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{h} K(\frac{x^i - x}{h})$$

Answer 2D Histogram

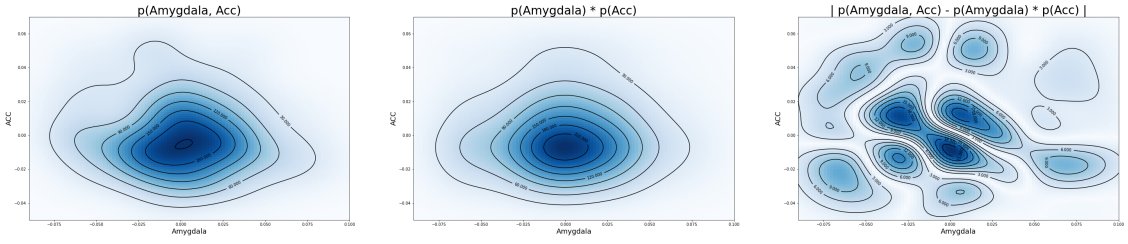


2D KDE



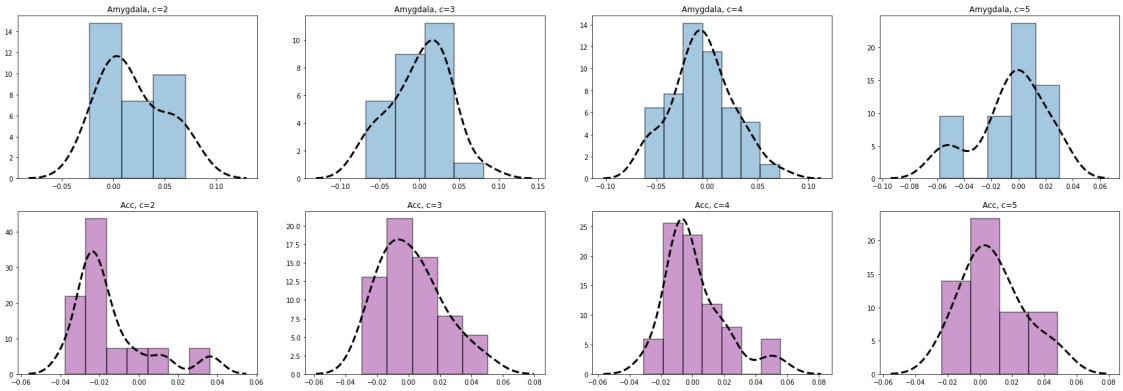
3. Using (a) and (b), using KDE estimators, verify whether or not the variables amygdala and acc are independent? You can tell this by checking do we approximately have p(amygdala, acc) = p(amygdala)p(acc)? To verify this, please show three plots: the map for p(amygdala, acc), the map for p(amygdala)p(acc) and the error map |p(amygdala, acc)−p(amygdala)p(acc)|. Comment on your results and whether this helps us to find out whether the two parts of brains (for emotions and decision-making) functions independently or they are related.

Answer For the two features to be deemed independent, the error between the joint and the product of the marginal distrubtions must be approximately equal to 0. Although the shapes of the density estimates appear very similiar (graphs 1 and 2), we can see on the 3rd graph below that there is somewhat of a significant difference between the two types of density estimates. One could interpret this as the features not being independnet of each other. However, the error isn't of such a great magnitude that it emphatically confirms dependence.



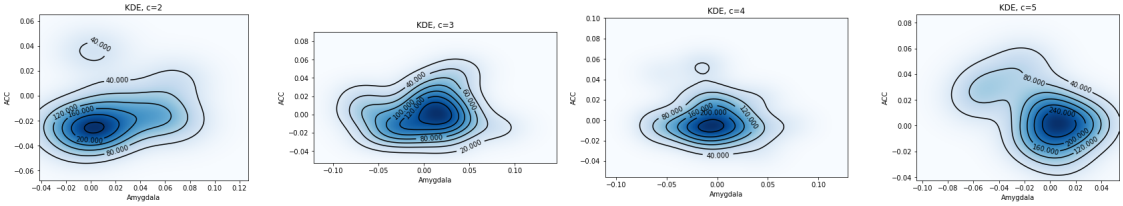
4 Now we will consider the variable orientation. We will estimate the conditional distribution of the volume of the amygdala, conditioning on political orientation:

p(amygdala|orientation = c), c = 2, . . . , 5. Do the same for the volume of the acc: Plot p(acc|orientation = c), c = 2, . . . , 5. You will use KDE to achieve the goal. (Note that the conditional distribution can be understood as fitting a distribution for the data with the same (fixed) orientation. Thus there should be 4 one-dimensional distribution functions to show for this question.)



Answer

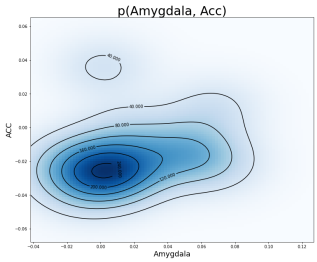
5. Again we will consider the variable orientation. We will estimate the conditional joint distribution of the volume of the amygdala and acc, conditioning on a function of political orientation: p(amygdala, acc|orientation = c), c = 2, . . . , 5. You will use two-dimensional KDE to achieve the goal.

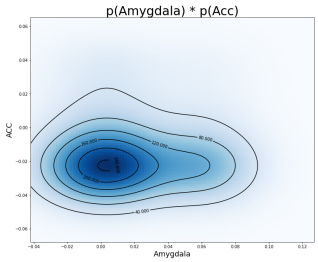


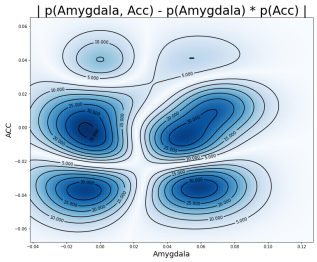
Answer

Using (d) and (e), evaluate whether or not the two variables are likely to be conditionally independent. To verify this, please show three plots: the map for p(amygdala, acc|orientation = c), the map for p(amygdala|orientation = c)p(acc|orientation = c) and the error map |p(amygdala, acc|orientation = c) − p(amygdala|orientation = c)p(acc|orientation = c)|, c = 2, . . . , 5. Comment on your results and whether this helps us to find out whether the two parts of brains (for emotions and decision-making) functions independently or they are related, conditionally on the political orientation (i.e., considering different types of personality).

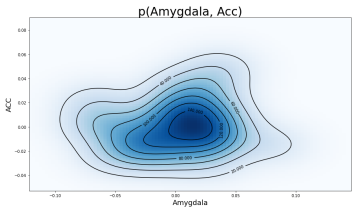
Answer c=2

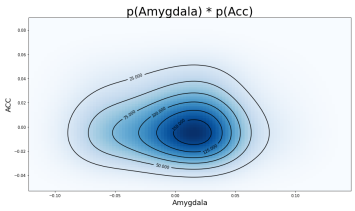


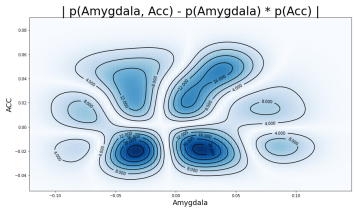




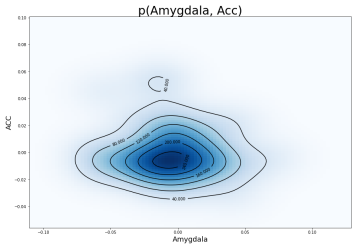
c=3

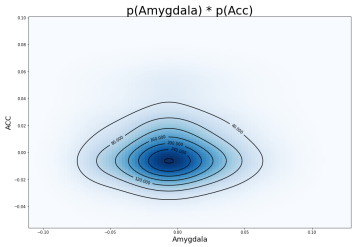


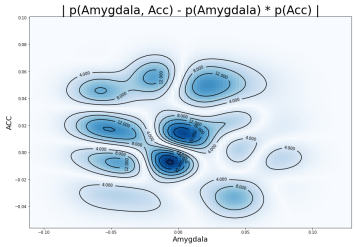




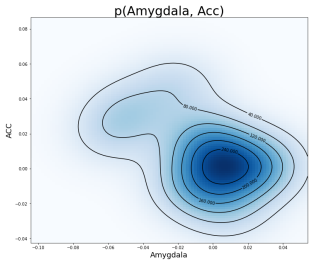
c=4

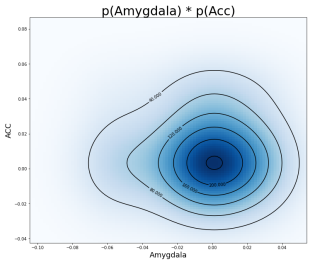


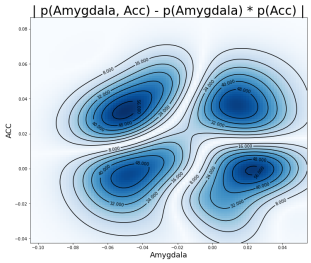




c=5







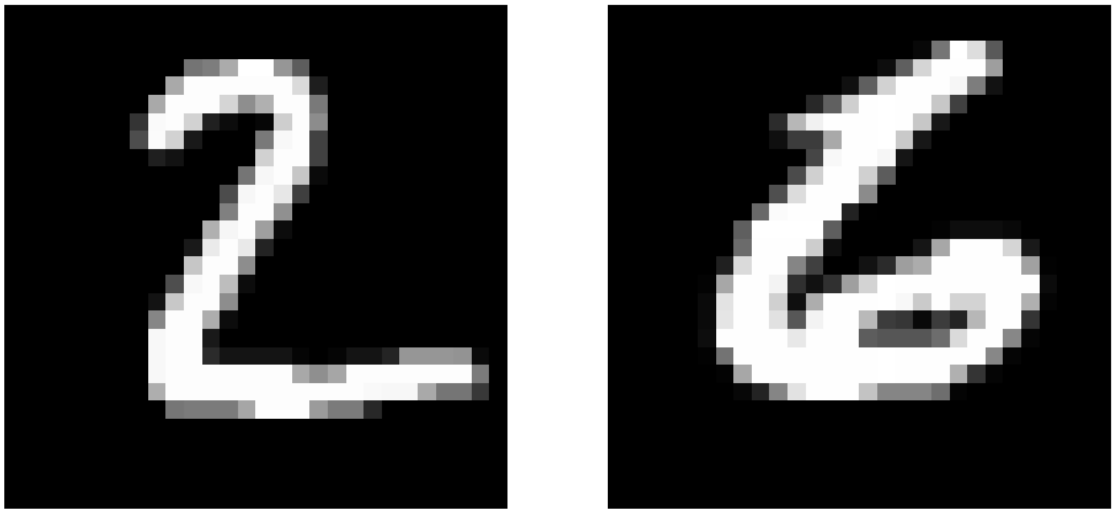
Overall, there is significnat enough error amongst all conditional density estimtates, that it strengthens the argument that the two parts of the brain (amygdala, acc) are act in a dependent fashion of one another.

Q2

Implementing EM for MNIST dataset, with PCA for dimensionality reduction.

Implement the EM algorithm for fitting a Gaussian mixture model for the MNIST dataset. We reduce the dataset to be only two cases, of digits "2" and "6" only. Thus, you will fit GMM with C = 2. Use the data file data.mat or data.dat. True label of the data are also provided in label.mat and label.dat The matrix images is of size 784-by-1990, i.e., there are totally 1990 images, and each column of the matrix corresponds to one image of size 28-by-28 pixels (the image is vectorized; the original image can be recovered by map the vector into a matrix). First use PCA to reduce the dimensionality of the data before applying to EM. We will put all "6" and "2" digits together, to project the original data into 5-dimensional vectors. Now implement EM algorithm for the projected data (with 5-dimensions).

1. Select from data one raw image of "2" and "6" and visualize them, respectively. Answer



Write down detailed expression of the E-step and M-step in the EM algorithm (hint: when computing τ_k^i , you can drop the (2) n/2 factor from the numerator and denominator expression, since it will be canceled out; this can help avoid some numerical issues in computation).

Answer The data **X** in which the GMM is being fit on, contains 1990 observations, and is decomposed to its first 5 principal components. Therefore:

$$X \in \mathbb{R}^{1990x5}$$

We are attmepting to fit 2 clusters, representative of the two images classes {2, 6}. Therefore:

Number of Clusters K = 2

A single cluster **k**, which is a multivariate guassian $\mathcal{N}_{\|}(\mu_k, \Sigma_k)$, is parameterized by mean μ_k and covariance matrix Σ_k. Given the dimension of input data **X**:

$$\mathcal{N}_{\|}(\mu_k, \Sigma_k) = \frac{exp[\frac{-1}{2}(X-\mu_k)^T\Sigma^{-1}(X-\mu_k)]}{\sqrt{(2\pi)^n|\Sigma|}} \tag{1}$$

Where

$$\mu_k \in \mathbb{R}^{5x1}$$

$$\Sigma_k \in \mathbb{R}^{5x5}$$

E-Step (Expectation)

The E-Step of the EM algorithm, aims to update the posterior probabilities for each observation with respect to each cluster in **K**. The posterior probability $\tau_k^{(i)}$ is described by:

$$\tau_k^{(i)} = \frac{\pi_k \mathcal{N}_{\|}(X^{(i)}|\mu_k, \Sigma_k)}{\sum_{k=1}^{K=2} \pi_k \mathcal{N}_{\|}(X^{(i)}|\mu_k, \Sigma_k)} \tag{2}$$

Where π_k is the mixing coefficient for clusteer **k**, $X^{(i)}$ is the i^{th} observation in **X** and $\tau_k^{(i)}$ is the posterior probability that $X^{(i)}$ comes from cluster **k**. Plugging (1) into (2) yields:

$$\tau_k^{(i)} = \frac{\pi_k \frac{exp[\frac{-1}{2}(X^{(i)}-\mu_k)^T\Sigma_k^{-1}(X^{(i)}-\mu_k)]}{\sqrt{(2\pi)^n|\Sigma_k|}}}{\sum_{k=1}^{K=2} \pi_k \frac{exp[\frac{-1}{2}(X^{(i)}-\mu_k)^T\Sigma_k^{-1}(X^{(i)}-\mu_k)]}{\sqrt{(2\pi)^n|\Sigma_k|}}} \tag{3}$$

Extracting the constants from the numerator and denominator from (3)

$$\tau_k^{(i)} = \frac{\frac{1}{\sqrt{(2\pi)^n}} \pi_k \frac{exp[\frac{-1}{2}(X^{(i)}-\mu_k)^T\Sigma_k^{-1}(X^{(i)}-\mu_k)]}{\sqrt{|\Sigma_k|}}}{\frac{1}{\sqrt{(2\pi)^n}} \sum_{k=1}^{K=2} \pi_k \frac{exp[\frac{-1}{2}(X^{(i)}-\mu_k)^T\Sigma_k^{-1}(X^{(i)}-\mu_k)]}{\sqrt{|\Sigma_k|}}}$$

Cancelling out the constants yields

$$\boxed{\tau_k^{(i)} = \frac{\pi_k \frac{exp[\frac{-1}{2}(X^{(i)}-\mu_k)^T\Sigma_k^{-1}(X^{(i)}-\mu_k)]}{\sqrt{|\Sigma_k|}}}{\sum_{k=1}^{K=2} \pi_k \frac{exp[\frac{-1}{2}(X^{(i)}-\mu_k)^T\Sigma_k^{-1}(X^{(i)}-\mu_k)]}{\sqrt{|\Sigma_k|}}}} \tag{4}$$

M-Step (Maximization)

After calculating the posteriors for each cluster w.r.t to each obseravation in $\mathbf{X}$, we update π_k, μ_k and Σ_k for each cluster with respect to the new densities corresponding to the posteriors calculated in the E-step. The mean for each cluster is updated by:

$$\mu_k = \frac{\sum_i \tau_k^i X^i}{\sum_i \tau_i^k} \qquad (5)$$

for (i = 1, 2, 3, …. m). The mixing coefficients are updated by:

$$\pi_k = \frac{\sum_i \tau_k^i}{m} \qquad (6)$$

where m=1990 is the total number of observations in $\mathbf{X}$. Finally the covariances for each cluster are updated by:

$$\Sigma_k = \frac{\sum_i \tau_k^i (X^i - \mu_k)(X^i - \mu_k)^T}{\sum_i \tau_k^i} \qquad (7)$$

3. Implement EM algorithm yourself. Use the following initialization

- initialization for mean: random Gaussian vector with zero mean
- initialization for covariance: generate two Gaussian random matrix of size n-by- n: S_1 and S_2 , and initialize the covariance matrix for the two components are $\Sigma_1 = S_1 S_1^T + I_n$, and $\Sigma_2 = S_2 S_2^T + I_n$, where I_n is an identity matrix of size n-by-n.

Answer

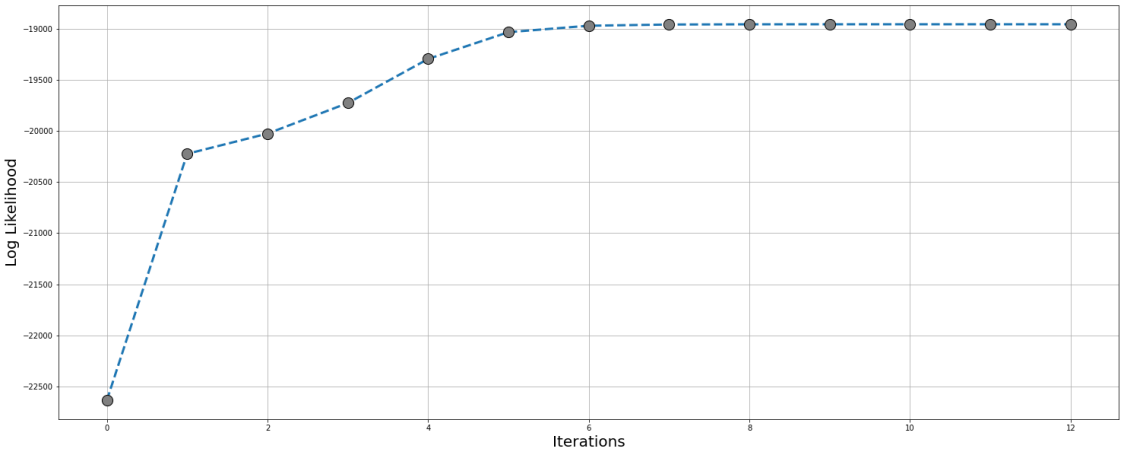
```
class GMM:
    def __init__(self, K: int, maxIter: int, tol=1e-3, seed=42, savefig=None) -> None:
        np.random.seed(seed)

        #Number of gaussian clusters
        self.K = K
        self.maxIter = maxIter
        self.tol = tol
        self.lls = []
        self.save = savefig

    def fit(self, X: np.array):

        assert len(X.shape) >= 2, 'If passing a vector of data, ensure two dimensions by resha
        #initialize clusters
        self._initialize(X=X)

        for i in range(self.maxIter):

            print('.........iteration {}.........'.format(i))
```

```
            #run the E step
            self._expectation(X)

            #run the M step
            self._maximization(X)

            diff = np.linalg.norm(self.mus - self.mus_old)

            if np.linalg.norm(self.mus-self.mus_old) < self.tol:
                print('Converged!')
                break

            self.mus_old = self.mus.copy()

        if i == self.maxIter-1:
            print('Max iterations reached')

        plt.figure(figsize=(25,10))
        plt.plot(self.lls, ls='--', lw=3)
        plt.scatter(range(len(self.lls)), self.lls, marker='o', edgecolor='k', s=200, color='g
        plt.grid()
        plt.xlabel('Iterations', fontsize=20)
        plt.ylabel('Log Likelihood', fontsize=20)
        if self.save:
            plt.savefig(self.save, bbox_layout='tight', bbox_inches='tight')
        plt.show()

    def _initialize(self, X):
        #get shape of the data
        m,n = X.shape

        #initialize K-gaussian clusters with mean randomly sampled from 0 centered Gaussian
        self.mus = np.random.randn(self.K, n)
        self.mus_old = self.mus.copy()

        #initialize K-covariance matrices
        self.covs = []
        for _ in range(self.K):
            S = np.random.randn(n,n)
            S = S@S.T + np.eye(n)
            self.covs.append(S)


        #initialize priors
        self.pis = np.random.random(self.K)
        self.pis = self.pis / np.sum(self.pis)

        #initialize the posterior
```

```
        self.tau = np.full((m,self.K), fill_value=0.)

    def _expectation(self, X: np.array):

        for i in range(self.K):
            self.tau[:, i] = self.pis[i] * multivariate_normal.pdf(X,
                                                                   self.mus[i],
                                                                   self.covs[i])
        #normalize tau
        sum_tau = np.sum(self.tau, axis=1).reshape(-1,1)
        self.tau = np.divide(self.tau, np.tile(sum_tau, (1,self.K)))

        #calculate log-likelihood
        ll = np.sum(np.log(sum_tau))
        self.lls.append(ll)

    def _maximization(self, X: np.array):

        m,n = X.shape

        for i in range(self.K):

            #update priors
            self.pis[i] = np.sum(self.tau[:,i]) / m

            #update cluster mean
            self.mus[i] = X.T @ self.tau[:, i] / np.sum(self.tau[:,i], axis=0)

            #update cluster covariance
            dummy = X - np.tile(self.mus[i], (m,1))
            self.covs[i] = dummy.T @ np.diag(self.tau[:, i]) @ dummy / np.sum(self.tau[:,i], a
```

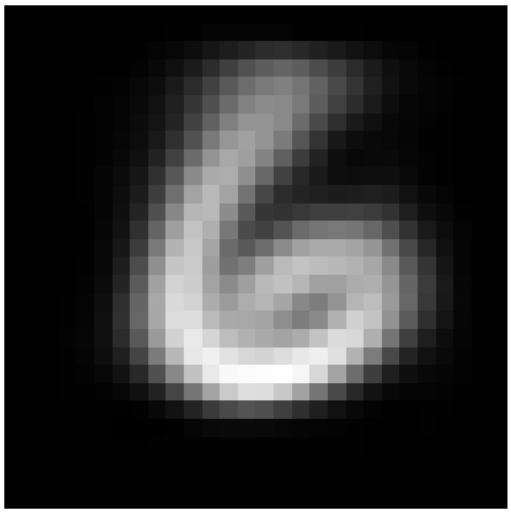
4. Report, the fitted GMM model when EM has terminated in your algorithms as follows. Make sure to report the weights for each component, and the mean of each component (you can reformat the vector to make them into 28-by-28 matrices and show images). Ideally, you should be able to see these means corresponds to "average" images. Report the two 784-by-784 covariance matrices by visualize their intensities.

Answer Log Likelihood / Iterations

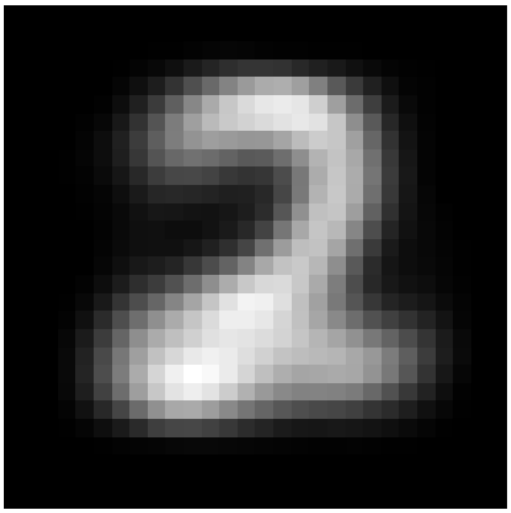


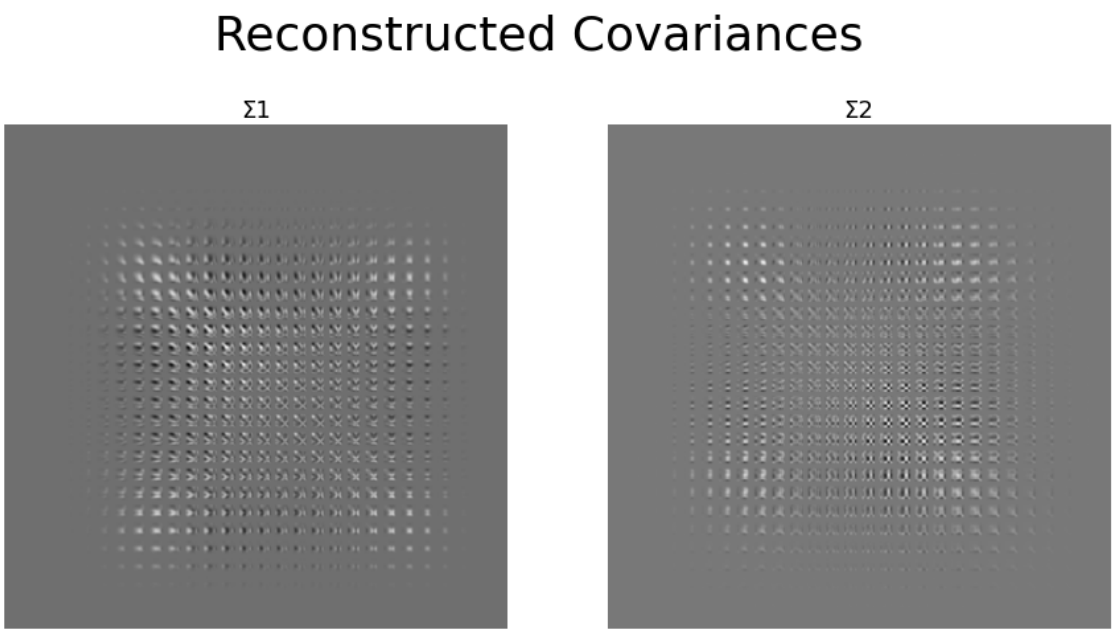
Reconstructed Means

Mean 1



Mean 2





Cluster Weights (π_k)

Cluster (K)	Weights (π_k)
1	0.5068
2	0.4931

5. Use the τ_k^i to infer the labels of the images, and compare with the true labels. Report the mis-classification rate for digits "2" and "6" respectively. Perform K-means clustering with K = 2 (you may call a package or use the code from your previous homework). Find out the mis-classification rate for digits "2" and "6" respectively, and compare with GMM. Which one achieves the better performance?

Answer GMM

Cluster 1: Image of Number 6 Cluster 2: Image of Number 2

Cluster 1	
2's	60 points
6's	951 points
Misclassification rate	5.93%

Cluster 2	
2's	972 points
6's	7 points
Misclassification rate	0.72%

GMM	
Overall Inaccuracy	3.37%
Model Imprecision (2's)	5.81%
Model Imprecision (6's)	0.73%

KMeans

Cluster 1: Image of Number 6 Cluster 2: Image of Number 2

Cluster 1	
2's	79 points
6's	915 points
Misclassification rate	5.93%

Cluster 2	
2's	953 points
6's	43 points
Misclassification rate	4.32%

KMeans	
Overall Inaccuracy	6.13%
Model Imprecision (2)	7.66%
Model Imprecision (6)	4.49%

Overall GMM performs better than KMeans. This is especially True w.r.t to classifying images of the number 2. GMM incorrectly grouped seven images of 6's in the cluster of 2's. Conversely KMeans incorrectly grouped 43 images of 6's in the cluster of 2's. GMM's overall accuracy 96.63% compared to KMeans 93.87%