**Problem 1**
Please download the MNIST handwritten digit dataset (training set images and training set labels) from http://yann.lecun.com/exdb/mnist/. It contains 28×28-pixel images for the hand-written digits 0,1,. . . ,9 by storing each pixel value ranging between 0 and 255, and their corresponding true labels. Load the data into Python. Preprocess the data by compressing each image to 1/4 of the original size in the following way: Divide each 28×28 image into 2×2 non-overlapping blocks. Calculate the mean pixel value of each 2×2 block, and create a new 14×14 image. This preprocessing step will drastically help your computation. We will be clustering the digits 0,1,2,3,4 in this homework. For visualization purpose, we view each data sample as 14×14 matrix. For using in an algorithm, treat each sample as a vector - you just simply stack each column of the 14×14 matrix into a 196 dimensional vector.

**Solution**
To load in the MNIST dataset, we utilized the `idx2numpy` package, which is a package that is used for converting IDX files to numpy arrays. Specifically, we used the `idx2numpy.convert_from_file` function. We then filtered the data to select only the images which are 0, 1, 2, 3, or 4, as we were only concerned with clustering these five classes for the implementation. To preprocess the images, we began by initializing the new images as $14 \times 14$ matrices of zeros and then performed the calculation as described in the problem by using the `np.add.reduceat` function from `numpy`.

**Problem 2**
Closely following the derivation in class, we now derive the EM algorithm for Gaussian mixture models. More specifically, we will use 2 models, the "mixture of spherical Gaussians" and the "mixture of diagonal Gaussians". By the end of this question, you should have derived two EM algorithms, one for each model. Below, the following denotes the meaning of each symbol:

- each $\mu_j$ is a d-dimensional vector representing the cluster center for cluster j
- each $\Sigma_j$ is a d ×d matrix representing the covariance matrix for cluster j
- $\sum_{j=1}^{k} \pi_j = 1$ and $\forall j, \pi_j \geq 0$ where all the $\pi_j$ are the mixing coefficients.
- $Z_i$ represents the missing variable associated with $x_i$ for $i \in 1, ..., n$. It takes integer values 1,...,k.

The parameters of the model to be estimated are $\theta = (\{\mu_{j,j}, \pi_j\}_{j=1}^{k})$. To be more explicit, we will the notation $p(x_i; \{\mu_{j,j}, \pi_j\}_{j=1}^{k})$ to denote the probability density function $p_\theta(x_i)$.That is, $p(x_i; \{\mu_{j,j}, \pi_j\}_{j=1}^{k}) = p_\theta(x_i)$. The likelihood of the data for $k$ clusters is:

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} p(x_i; \{\mu_{j,j}, \pi_j\}_{j=1}^{k}) \\ &= \prod_{i=1}^{n} p(x_i; \mid Z_i = j; \mu_j; \Sigma_j) P(Z_i = j) \\ &= \prod_{i=1}^{n} \sum_{j=1}^{k} \frac{\pi_j}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp\left( \frac{-1}{2} (x_i - \mu_j)^T (\sigma_j^{-1})(x_i - \mu_j) \right) \end{aligned} \tag{1}$$

In a mixture of diagonal Gaussians model, $\Sigma_j = diag(\sigma_{j1}^2, \ldots, \sigma_{jd}^2)$ is a diagonal matrix for all j = 1,...,k. So, we only have d parameters to estimate for the covariance matrix of each cluster.
In a mixture of spherical Gaussians model, $\Sigma_j = \sigma_j^2 I_d$ for all j = 1,...,k. Here $\sigma_j > 0$ is a scalar and $I_d$ is the $d \times d$ identity matrix. So we only have 1 parameter to estimate for the covariance matrix of each cluster.

(i) First, we do some preliminary work that will be useful later on.

- **Part (a)** Write down the marginal distribution of the $Z_i$ (Hint: What is the probability $p(Z_i = j)$.

  **Solution**

We know that $Z_i \overset{iid}{\sim} \text{Multinomial}(\pi_1, \ldots, \pi_j)$, which therefore implies

$$\mathbb{P}(Z_i = j) = \pi_j \tag{2}$$

- **Part (b)** Calculate $p(Z_i = j \mid x_i)$. (Hint: Bayes Rule)

  **Solution**
  Applying Bayes rule,

$$\begin{aligned} p(Z_i = j|x_i) &= \frac{p(x_i|Z_i = j)p(Z_i = j)}{p(x_i)} \\ &= \frac{p(x_i|Z_i = j)p(Z_i = j)}{\sum_{j=1}^{k} p(Z_i = j)p(x_i|Z_i = j)} \\ &= \frac{N(x_i; \mu_j, \Sigma_j)\pi_j}{\sum_{j=1}^{k} \pi_j N(x_i; \mu_j, \Sigma_j)} \end{aligned}$$

(ii) We now derive the E- and M- steps. We denote $\theta = (\{\mu_j, \Sigma_j, \pi_j\}_{j=1}^{k})$. From Equation 1, we can write down the log-likelihood and derive a lower bound:

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{n} \log p_\theta(x_i) \\ &= \sum_{i=1}^{n} \log \left[ \sum_{j=1}^{k} p_\theta(x_i; Z_i = j) \right] \\ &= \sum_{i=1}^{n} \log \left[ \sum_{j=1}^{k} F_{ij} \frac{p_\theta(x_i; Z_i = j)}{F_{ij}} \right] \end{aligned}$$

where $F_{ij} > 0$ for all i,j and satisfies $\sum_{j=1}^{k} F_{ij} = 1$ for $i = 1 \ldots n$
Note that if $i = 1, F_{1j} \in \mathbb{R}^k$ corresponds to the the distributions that we pick in the notes, denoted as $q_1$. Similarly for all i from 1 to n.

- **Part (c)** Prove the following lower bound of the log-likelihood function.

$$\sum_{i=1}^{n} \log \left[ \sum_{j=1}^{k} F_{ij} \frac{p_\theta(x_i; Z_i = j)}{F_{ij}} \right] \geq \sum_{i=1}^{n} \left[ \sum_{j=1}^{k} F_{ij} \log \frac{p_\theta(x_i; Z_i = j)}{F_{ij}} \right] \tag{3}$$

  (Hint: you can use the Jensen's inequality $\log \mathbb{E}X \geq \mathbb{E} \log X$ (You are not required to prove the Jensen's inequality)).

  **Solution**

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^{n} \log \left( \sum_{j=1}^{k} \mathrm{F}_{ij} \frac{p_\theta(x_i, Z_i = j)}{F_{ij}} \right) \\ &= \sum_{i=1}^{n} \log \mathbb{E} \left[ \frac{p_\theta(x_i, Z_i = j)}{F_{ij}} \right] \end{aligned}$$

Jensen's inequality states that $\log(\mathbb{E}[X]) \geq \mathbb{E}[\log(X)]$, so applying it we obtain

$$\begin{aligned}\sum_{i=1}^{n} \log \mathbb{E}\left[\frac{p_\theta(x_i, Z_i = j)}{F_{ij}}\right] &\geq \sum_{i=1}^{n} \mathbb{E}\left[\log\left(\frac{p_\theta(x_i, Z_i = j)}{F_{ij}}\right)\right] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \log\left(\frac{p_\theta(x_i, Z_i = j)}{F_{ij}}\right)\end{aligned}$$

Therefore, we have shown that

$$\sum_{i=1}^{n} \log\left(\sum_{j=1}^{k} \mathrm{F}_{ij} \frac{p_\theta(x_i, Z_i = j)}{F_{ij}}\right) \geq \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \log\left(\frac{p_\theta(x_i, Z_i = j)}{F_{ij}}\right)$$

- **Part (d)** (E-step) We define

$$Q(F, \theta) := \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \log\left[\frac{p_\theta(x_i; Z_i = j)}{F_{ij}}\right] \tag{4}$$

to be the lower bound function of $l(\theta)$. Let $\theta'$ be a fixed value of $\theta$ (e.g., $\theta$ could be the parameter value of the current iteration). Recall from Part c that we designed $Q(F, \theta)$ to be a lower bound for $l(\theta)$. We want to make this lower bound as tight as possible. Prove that $l(\theta') = Q(F, \theta')$ when

$$F_{ij} = p_{\theta'}(Z_i = j \mid x_i) \tag{5}$$

**Solution**

From above, we define $Q(F, \theta)$ as

$$\sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \log\left(\frac{p_{\theta'}(x_i, Z_i = j)}{F_{ij}}\right)$$

To show $\ell(\theta') = Q(F, \theta')$, we substitute $p_{\theta'}(Z_i = j \mid x_i)$ for $F_{ij}$ and then simplify as follows

$$\begin{aligned}Q(F, \theta') &= \sum_{i=1}^{n}\sum_{j=1}^{k} p_{\theta'}(Z_i = j \mid x_i) \log\left(\frac{p_{\theta'}(x_i, Z_i = j)}{p_{\theta'}(Z_i = j \mid x_i)}\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k} p_{\theta'}(Z_i = j \mid x_i) \log(p_{\theta'}(x_i)) \\ &= \sum_{j=1}^{k} p_{\theta'}(Z_i = j \mid x_i) \sum_{i=1}^{n} \log(p_{\theta'}(x_i)) \\ &= \sum_{j=1}^{k} F_{ij} \sum_{i=1}^{n} \log(p_{\theta'}(x_i)) \\ &= 1 \times \sum_{i=1}^{n} \log(p_{\theta'}(x_i)) \\ &= \ell(\theta')\end{aligned}$$

(iii) Once the E-step is derived, we now derive the M-step. First, we plug

$$F_{ij} = p_{\theta^{(t)}}(Z_i = j \mid x_i)$$

into Equation 3 and define the lower bound function at $\theta^{(t)}$ to be

$$Q(\theta^{(t)}, \theta) := \sum_{i=1}^{n}\sum_{j=1}^{k} p_{\theta^{(t)}}(Z_i = j \mid x_i) \log\left[\frac{p_\theta(x_i, Z_i = j)}{p_{\theta^{(t)}}(Z_i = j \mid x_i)}\right] \tag{6}$$

- **Part (e)** (M-step for mixture of spherical Gaussians) In the M-step, we aim to find $\theta^{(t+1)}$ that maximize the lower bound function $Q(\theta^{(t)}, \theta)$. Under the mixture of spherical Gaussians model, derive the M-step updating equations for $\mu_j^{(t+1)}$ ,$\Sigma_j^{(t+1)}$ and $\pi_j^{(t+1)}$ .

  **Solution**

  We are given that $Q(\theta^{(t)}, \theta) = \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\log(\pi_j) + F_{ij}\log(N(x_i; \mu_j, \Sigma_j))$. Our main objective is to maximize the $Q(\theta^{(t)}, \theta)$, specifically

  $$\theta^{(t+1)} = \underset{\theta}{\text{argmax}}\ Q(\theta^{(t)}, \theta)$$

  where, $\theta = (\mu, \Sigma, \pi)$.

  $\underline{\mu_j^{t+1}}$:

  For us to compute the $\mu_j^{(t+1)}$ and $\Sigma_j^{(t+1)}$, we can ignore the terms that depend purely on $\pi_j$. Likewise, we can ignore the terms that depend on $\mu_j, \Sigma_j$ to compute $\pi_j^{(t+1)}$. Lets compute all the above parameters in $\theta$.

$$\begin{aligned}
\mu_j^{(t+1)} &= \underset{\mu}{\text{argmax}}\ \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\log(N(x_i; \mu_j, \Sigma_j)) \\
&= \underset{\mu}{\text{argmax}}\ \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\log\left(|2\pi\Sigma_j|^{-1/2}\text{exp}(-\frac{1}{2}(x_i - \mu_j)^T\Sigma_j^{-1}(x_i - \mu_j))\right) \\
&= \frac{\partial}{\partial\mu_j}\ \left(-\sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\pi\log\left(|\Sigma_j|\right) - \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\frac{1}{2}(x_i - \mu_j)^T\Sigma_j^{-1}(x_i - \mu_j)\right) \\
&= \sum_{i=1}^{n} F_{ij}\Sigma_j^{-1}(x_i - \mu_j) \quad \text{(On fixing a particular j)} \\
&= \sum_{i=1}^{n} F_{ij}\Sigma_j^{-1}x_i - \sum_{i=1}^{n} F_{ij}\Sigma_j^{-1}\mu_j = 0 \\
&\Rightarrow \sum_{i=1}^{n} F_{ij}\Sigma_j^{-1}x_i = \sum_{i=1}^{n} F_{ij}\Sigma_j^{-1}\mu_j \\
&\Rightarrow \mu_j^{(t+1)} = \frac{\sum_{i=1}^{n} F_{ij}x_i}{\sum_{i=1}^{n} F_{ij}}
\end{aligned} \tag{7}$$

  $\underline{\sigma_j^{2(t+1)}}$:

  Instead of trying to find the entire $\Sigma_j^{(t+1)}$, we try to estimate $\sigma_j^{2(t+1)}$ because $\Sigma_j = \sigma_j^2 I_d$. Using a bit of the above proof we have the following expression

$$
\begin{aligned}
\sigma_j^{2(t+1)} &= \underset{\sigma_j^2}{\text{argmax}} \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \log(N(x_i;\mu_j,\Sigma_j)) \\
&= \underset{\sigma_j^2}{\text{argmax}} \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \log\left(|2\pi\Sigma_j|^{-1/2}\text{exp}(-\frac{1}{2}(x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j))\right) \\
&= \underset{\sigma_j^2}{\text{argmax}} \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \log\left(|2\pi\sigma_j^2 I_d|^{-1/2}\text{exp}(-\frac{1}{2}(x_i-\mu_j)^T(\sigma_j^2 I_d)^{-1}(x_i-\mu_j))\right) \\
&= \underset{\sigma_j^2}{\text{argmax}} \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \frac{-d}{2}\log\left(2\pi\sigma_j^2\right) + F_{ij}(-\frac{1}{2}(x_i-\mu_j)^T(\sigma_j^2 I_d)^{-1}(x_i-\mu_j))\right) \\
&= \frac{\partial}{\partial\sigma_j^2}\left(\sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \frac{-d}{2}\log\left(2\pi\sigma_j^2\right) + F_{ij}(-\frac{1}{2}(x_i-\mu_j)^T(\sigma_j^2 I_d)^{-1}(x_i-\mu_j)))\right) \\
&= \sum_{i=1}^{n} F_{ij}\frac{-d}{2\sigma_j^2} + F_{ij}\frac{1}{2(\sigma_j^2)^2}(x_i-\mu_j)^T(x_i-\mu_j) = 0 \\
&\Rightarrow \sum_{i=1}^{n} F_{ij}\frac{d}{2\sigma_j^2} = \sum_{i=1}^{n} F_{ij}\frac{1}{2(\sigma_j^2)^2}(x_i-\mu_j)^T(x_i-\mu_j) \\
&\Rightarrow \sigma_j^{2(t+1)} = \frac{\sum_{i=1}^{n} F_{ij}(x_i-\mu_j)^T(x_i-\mu_j)}{d\sum_{i=1}^{n} F_{ij}}
\end{aligned} \tag{8}
$$

$\underline{\pi_j^{(t+1)}}$:

By intuition, we know that $\pi$ should follow a multinomial distribution. To show that the result for $\pi_j$ is similar to the MLE of multinomial distribution, we try to maximize $Q(\theta^{(t)},\theta)$ with respect to $\pi$. In $Q(\theta^{(t)},\theta)$, we can ignore the terms in the Normal distribution and only focus on terms that depend on $\pi$

$$
\pi_j^{(t+1)} = \underset{\pi}{\text{argmax}} \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\log\pi_j
$$

We can add the above expression with a constant, using the fact $\sum_{j=1}^{k}\pi_j = 1$. We add a new constant term $\lambda(1-\sum_{j=1}^{k}\pi_j)$ to the above expression to get a new expression as below

$$
\begin{aligned}
\pi_j^{(t+1)} &= \underset{\pi}{\text{argmax}} \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\log\pi_j + \lambda(1-\sum_{j=1}^{k}\pi_j) \\
&= \frac{\partial}{\partial\pi_j}\left(\sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\log\pi_j + \lambda(1-\sum_{j=1}^{k}\pi_j)\right) \\
&= \frac{1}{\pi_j}\sum_{i=1}^{n} F_{ij} - \lambda = 0 \\
&\Rightarrow \pi_j^{(t+1)} = \frac{\sum_{i=1}^{n} F_{ij}}{\lambda}
\end{aligned}
$$

Now, if we solve for $\lambda$, we have the following expression on using the above expression by taking a sum across all j, we know that the constraint on $\pi_j^{(t+1)}$ is such that $\sum_{j=1}^{k} \pi_j^{t+1} = 1$

$$\begin{aligned}\sum_{j=1}^{k} \pi_j^{(t+1)} &= \frac{\sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}}{\lambda} \\ 1 &= \frac{n}{\lambda} \\ \Rightarrow \lambda &= n\end{aligned}$$

Resubstituting the value for $\lambda$, we get

$$\pi_j^{(t+1)} = \frac{\sum_{i=1}^{n} F_{ij}}{n} \tag{9}$$

- **Part (f)** (M-step for mixture of diagonal Gaussians) In the M-step, we aim to find $\theta^{(t+1)}$ that maximize the lower bound function $Q(\theta^{(t)}, \theta)$. Under the mixture of spherical Gaussians model, derive the M-step updating equations for $\theta_j^{(t+1)}$ ,$\Sigma_j^{(t+1)}$ and $\pi_j^{(t+1)}$ .

**Solution**

$\underline{\mu_j^{(t+1)}}$:

We can use the expression obtained above in the spherical case since the computation of $\mu$ doesn't change from equation (7),

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{n} F_{ij} x_i}{\sum_{i=1}^{n} F_{ij}}$$

$\underline{\Sigma_j^{(t+1)}}$:

Using the expression for $Q(\theta^{(t)}, \theta)$, we have

$$\begin{aligned}\Sigma_j^{(t+1)} &= \underset{\Sigma}{\text{argmax}} \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \log(N(x_i; \mu_j, \Sigma_j)) \\ &= \underset{\Sigma}{\text{argmax}} \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij} \log\left(|2\pi\Sigma_j|^{-1/2} \exp(-\frac{1}{2}(x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j))\right) \\ &= \frac{\partial}{\partial \Sigma_j}\left(-\sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\pi \log\left(|\Sigma_j|\right) - \sum_{i=1}^{n}\sum_{j=1}^{k} F_{ij}\frac{1}{2}(x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)\right) \\ &= \sum_{i=1}^{n}\frac{1}{2}F_{ij}\frac{1}{|\Sigma_j|}\Sigma_j|\Sigma_j| - \frac{1}{2}F_{ij}(x_i-\mu_j)^T(x_i-\mu_j) \quad \text{(On fixing a particular j)} \\ &= \sum_{i=1}^{n} F_{ij}\Sigma_j - \sum_{i=1}^{n} F_{ij}(x_i-\mu_j)^T(x_i-\mu_j) \ = 0 \\ \Rightarrow \sum_{i=1}^{n} F_{ij}\Sigma_j &= \sum_{i=1}^{n} F_{ij}(x_i-\mu_j)^T(x_i-\mu_j) \\ \Rightarrow \Sigma_j^{(t+1)} &= \frac{\sum_{i=1}^{n} F_{ij}(x_i-\mu_j)^T(x_i-\mu_j)}{\sum_{i=1}^{n} F_{ij}}\end{aligned} \tag{10}$$

$\pi_j^{(t+1)}$:

We can use the expression obtained above in the spherical case since the computation of $\pi$ doesn't change from equation (9)

$$\pi_j^{(t+1)} = \frac{\sum_{i=1}^{n} F_{ij}}{n}$$

**Problem 3**

We now implement the EM algorithm from last question to cluster the MNIST data set.

(i) Program the EM algorithm you derived for mixture of spherical Gaussians. Assume 5 clusters. Terminate the algorithm when the fractional change of the log-likelihood goes under 0.0001. (Try 3 random initializations and present the best one in terms of maximizing the likelihood function).

(ii) Program the EM algorithm you derived for mixture of diagonal Gaussians. Assume 5 clusters. Terminate the algorithm when the fractional change in the log-likelihood goes under 0.0001. (Try 3 random initializations and present the best one in terms of maximizing the likelihood function).

Note that to assign a sample $x_i$ to a cluster j, you first calculate $F_{ij}$ using the parameters from the last iteration of EM algorithm you implemented. Next, assign sample $x_i$ to the cluster j for which $F_{ij}$ is maximum, i.e., the probability of sample i belonging to cluster j is maximum. Recall that the dataset has the true labels for each classes. Calculate the error of the algorithm (for the two different model). Here, error is just the number of mis-clustered samples divided by the total number of samples. In your opinion, were mixture models of Gaussian distributions suitable for modeling the MNIST data?

**Solution**

For both the above cases, we implement the same set of functions with the only difference being how the initial data is generated. It is important to observe that spherical gaussian is a special case of the diagonal gaussian, where all the values on the diagonal are equal. Using this information, we first implement a function to generate the data, which we called `generate_data`. The function checks whether it is an update to the distributions or has to generate new distributions and based on that, it returns a stack of multivariate normal distributions. The function also checks if a spherical or diagonal gaussian is to be generated or not. Using a random seed for every generation, we create 3 stacks of spherical and 3 stacks of diagonal gaussian distributions.

Next, we try to implement the E step for the EM algorithm. The function `expectation_step` takes the following parameters:

- $X \sim \mathbb{R}^{n \times d}$ := The data matrix consisting of the image data from the MNIST Dataset
- Initial Distributions := A stack of multivariate diagonal or spherical gaussians ie. $\sim N(X; \mu_j, \Sigma_j)$ for j $\in 1 \dots 5$
- $pi \sim Dirchlet$ := consists of random probabilities assigned to every cluster j
- k := Number of clusters to be considered

In this function, we use the `logsumexp` trick to compute every row of the value $F$ which we saw in the previous proofs. In normal instances for us to compute $F_{ij}$, we would use the following:

$$F_{ij} = \frac{\pi_j N(x_i; \mu_j, \Sigma_j)}{\sum_{k'=1}^{5} \pi_{k'} N(x_i; \mu_{k'}, \Sigma_{k'})}$$

However, the logsumexp trick helps in reducing the computation time drastically. Also, in order to get predictions for every $x_i$ belonging to a cluster, we use the `np.argmax()` function on the newly created array $F_i$ to see which cluster does $x_i$ belong to. After computing both the predictions and the F matrix, we return both of these values from the function.

Next, for the maximization step, the following parameters are passed to the function called `maximization_step`:

- $X \sim \mathbb{R}^{n \times d}$ := The data matrix consisting of the image data from the MNIST dataset
- Initial Distributions := A stack of multivariate diagonal or spherical gaussians ie. $\sim N(X; \mu_j, \Sigma_j)$ for j $\in 1 \dots 5$
- k := Number of clusters to be considered

In this function, we try to parse through all the cluster values and generate a new $\theta^{(t+1)}$ consisting of $(\mu_j^{(t+1)}, \Sigma_j^{(t+1)}, \pi^{(t+1)})$. The values inside the vector are generated using the formulas obtained in equations (7), (8), (9) & (10). The function returns back the $\theta^{(t+1)}$ vector.

Finally, we connect all the functions together in a function named `expectation_maximization` which takes in the following inputs

- $X \sim \mathbb{R}^{n \times d}$ := The data matrix consisting of the image data from the MNIST Dataset
- Initial Distributions := A stack of multivariate diagonal or spherical gaussians ie. $\sim N(X; \mu_j, \Sigma_j)$ for j $\in 1 \dots 5$
- $pi \sim Dirchlet$ := consists of random probabilities assigned to every cluster j
- k := Number of clusters to be considered
- threshold := Limiting value of difference in the log likelihoods of the normal distributions at $t^{th}$ and $(t+1)^{th}$
- maxiters := In case the difference of the log likelihood doesn't go below the threshold, the algorithm stops after the specified iterations

In this function, we call the above defined `expectation_step` and `maximization_step` functions and update the normal distributions on every iteration. We also check for the difference of the log likelihoods for all the distributions and check if the value goes below the threshold value decided above. Once the difference goes below the threshold, we return the predictions for every cluster.

In order to clean the predictions, we look at the most common labels in the training dataset for a particular value of the predictions and compute the accuracy for all cases.

The table below summarizes the log likelihoods and the accuracy's of all the initializers

| | Seeds | Accuracy | Log Likelihood |
|---|---|---|---|
| Spherical Gaussians | 1 | 71.03% | 88.70 |
| | 2 | 71.16% | 88.71 |
| | 3 | 71.05% | 88.70 |
| Diagonal Gaussians | 1 | 84.43% | 88.70 |
| | 2 | 71.06% | 88.69 |
| | 3 | 84.39% | 88.61 |

Table 1: Log likelihoods for each initializer.

Based on the experiment, GMM can be used for prediction and clustering of the MNIST dataset, however, it is heavily dependent on how we initialize the initial distributions and probabilities. If we find the optimal initializer, GMM can be used effectively for modeling the MNIST dataset.

**Pledge:**
Please sign below (print full name) after checking (✓) the following. If you can not honestly check each of these responses, please email me at kbala@ucdavis.edu to explain your situation.

- We pledge that we are honest students with academic integrity and we have not cheated on this homework. ✓

- These answers are our own work. ✓
- We did not give any other students assistance on this homework. ✓
- We understand that to submit work that is not our own and pretend that it is our is a violation of the UC Davis code of conduct and will be reported to Student Judicial Affairs. ✓
- We understand that suspected misconduct on this homework will be reported to the Office of Student Support and Judicial Affairs and, if established, will result in disciplinary sanctions up through Dismissal from the University and a grade penalty up to a grade of "F" for the course. ✓

Team Member 1: Jay Bendre Team Member 2: Grant Gambetta