Sharif University of Technology
Electrical Engineering Department

Introduction to Machine Learning
Project Phase 1

Amir Hossein Moraveji - Amir Hossein Yari
99104232 - 99102507

April 20, 2023

Contents

Theory Question 1) In your own words, explain how the MM algorithm can deal with nonconvex optimization objective functions by considering simpler convex objective functions.

The MM algorithm deals with nonconvex optimization problems by iteratively solving a convex majorization function that approximates the original nonconvex function, using an anchor point that is dynamically updated. This approach enables the algorithm to solve complex optimization problems that cannot be solved using traditional gradient-based methods.
For example, suppose we have a nonconvex function $f(x)$ that we want to minimize. The MM algorithm can transform this function into a convex function using a surrogate function $g(x, y)$, such that:

$$g(x, y) \leqslant f(x)$$

where y is a parameter that serves as an anchor point for the majorization function. Now, we can solve the convex function $g(x, y)$ by setting its gradient equal to zero and updating y using the current value of x:

$$y_{new} = y + \alpha(x - x_{old})$$

This update moves the anchor point towards the current estimate of x, which encourages the majorization function to better approximate the nonconvex function. Finally, we update x using the maximizer of $g(x, y_new)$:

$$x_{new} = argmax\ g(x, y_{new})$$

This step ensures that the updated value of x satisfies the majorization constraint, which implies that it also satisfies the original nonconvex constraint.
We then iterate these two steps until convergence. The MM algorithm converges to a stationary point of the original nonconvex function, and under certain conditions, it also converges to a global maximum.

Theory Question 2) Briefly explain how the formula for mixture models:

$$p(\mathbf{y};\boldsymbol{\theta}) = \sum_{k=1}^{K} p_Z(z_k;\boldsymbol{\theta}) p_{\mathbf{Y}|\mathbf{Z}}(\mathbf{y}|Z = z_k;\boldsymbol{\theta})$$

is the same as the sum over all possible values of $Z^{(i)}$ in equation (9). Explain why it's easier to optimize $p_{\mathbf{Y},\mathbf{Z}}(\mathbf{y}_n, \mathbf{z}_n; \theta)$ than $p_{\mathbf{Y}}(\mathbf{y}_n; \theta)$ in the context of mixture models.

The formula for mixture models:

$$p(\mathbf{y};\boldsymbol{\theta}) = \sum_{k=1}^{K} p_Z(z_k;\boldsymbol{\theta}) p_{\mathbf{Y}|\mathbf{Z}}(\mathbf{y}|Z = z_k;\boldsymbol{\theta})$$

is equivalent to the sum over all possible values of $Z^{(i)}$ in the expression:

$$\sum_{k=1}^{K} p_{Y,Z}(y^{(i)}, Z^{(i)} = k;\boldsymbol{\theta})$$

In other words, the sum over all possible values of $Z^{(i)}$ in the second expression represents the summation of the joint probability $p_{Y,Z}(y^{(i)}, Z^{(i)} = k;\boldsymbol{\theta})$ over all possible values of k for each data point $\mathbf{y}^{(i)}$.
This is equivalent to the formula for mixture models, where the joint probability $p_{Y,Z}(y, z;\boldsymbol{\theta})$ is decomposed into the product of the latent variable probability $p_Z(z;\boldsymbol{\theta})$ and the conditional probability of $\mathbf{y}$ given $\mathbf{z}$, denoted as $p_{\mathbf{Y}|\mathbf{Z}}(\mathbf{y}|Z = z;\boldsymbol{\theta})$, summed over all possible values of k from 1 to K.

The reason for optimize the joint probability $p_{\mathbf{Y},\mathbf{Z}}(\mathbf{y}_n, \mathbf{z}_n; \theta)$ is easier than the marginal probability $p_{\mathbf{Y}}(\mathbf{y}_n; \theta)$ is that the joint probability $p_{\mathbf{Y},\mathbf{Z}}(\mathbf{y}_n, \mathbf{z}_n; \theta)$ involves both the observed data $\mathbf{y}_n$ and the latent variable $\mathbf{z}_n$, and optimizing it allows the model to account for the dependencies between the observed data and the latent variable.
On the other hand, the marginal probability $p_{\mathbf{Y}}(\mathbf{y}_n; \theta)$ only involves the observed data $\mathbf{y}_n$ and ignores the latent variable $\mathbf{z}_n$. This can lead to a loss of information and potentially inaccurate parameter estimates, as the model may not fully capture the underlying structure and dependencies in the data.
By optimizing the joint probability $p_{\mathbf{Y},\mathbf{Z}}(\mathbf{y}_n, \mathbf{z}_n; \theta)$, the model can take into account

the interactions between the observed data and the latent variable, leading to a more robust and accurate estimation of the model parameters.

Theory Question 3) Read about variational inference (or variational bayesian methods) and compare it with the procedure we used for the EM algorithm (You might want to check Wikipedia for this!).

Variational Bayesian methods (VB) and Expectation-Maximization (EM) algorithm are both iterative optimization techniques used in statistical inference. While the EM algorithm is a general maximum likelihood approach, VB is a Bayesian model.
EM algorithm is an iterative algorithm used to estimate model parameters in situations where the data is incomplete or missing. It involves two steps, the E-step, and the M-step. In the E-step, the algorithm computes the expected values of the missing data given the parameters. Then, in the M-step, the algorithm maximizes the likelihood function given the expected values computed in the E-step.
VB is a Bayesian model that places a prior distribution on the parameters of the model, and it is a method to approximate the posterior distribution over the parameters using a variational distribution. VB has become very popular with the increased use of Bayesian methods in machine learning.
VB methods and EM algorithm share several similarities:

- Both methods are used to estimate the parameters of probabilistic models.
- Both methods involve maximizing an objective function that involves a data likelihood and a prior distribution over the parameters.
- Both methods require an iterative approach to optimize the objective function.
- Both methods operate by alternating between updating an estimate of the latent variables (in EM) or the posterior distribution of the parameters (in VB) and updating the estimate of the parameters (in EM) or the approximate posterior distribution (in VB).
- Both methods rely on a trade-off between model complexity and accuracy of the estimates.
- Both methods can handle missing data.
- Both methods can be used for model selection and hypothesis testing.

Also, there are several key differences between the two methods:

- The EM algorithm aims to find the maximum likelihood estimate of the parameters of a probabilistic model, while Variational Bayesian methods seek to approximate the posterior distribution over the model parameters.
- The EM algorithm alternates between computing the expectation of the likelihood function with respect to the current estimate of the parameter values and maximizing this expected likelihood to obtain updated estimates. Variational Bayesian methods compute a lower bound on the marginal likelihood of the data, which is optimized with respect to a distribution that approximates the posterior over the model parameters.
- The EM algorithm is generally faster and easier to implement than Variational Bayesian methods, as the latter require solving a more complex optimization problem. However, Variational Bayesian methods are more accurate and can handle a wider range of models.
- The EM algorithm guarantees convergence to a local maximum of the likelihood function, but it may get stuck in local optima. Variational Bayesian methods are more robust to local optima, as they optimize a lower bound on the marginal likelihood.
- Variational Bayesian methods are inherently Bayesian, as they compute the posterior distribution over the model parameters. The EM algorithm does not have a Bayesian interpretation, as it only finds the maximum likelihood estimate of the parameters.

Theory Question 4) Compute estimate of parameters for Gaussian Mixture Models for N observed data $\{x_i\}_{i=1}^N$.

1. **Determine model parameters and initialize them.**
2. **Compute complete dataset likelihood.**
3. **Find closed-form solution for parameters using EM algorithm.**

1.

- Determine the number of Gaussian components (K) in the mixture model.
- Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$, and mixture weights π_k for each Gaussian component. This can be done randomly or using some prior knowledge or heuristics.

2. Compute the likelihood of the complete dataset, which involves both observed data and latent variables. The complete dataset likelihood is given by:

$$p(\{\mathbf{x}^{(n)}, z_n\}_{n=1}^N | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)} | \mu_k, \Sigma_k)$$

3.

$$p_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\theta}) = p_{\mathbf{X}}(\mathbf{x}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$p_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p_{\mathbf{X}}(\mathbf{x}^{(n)}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$L(\mathbf{x}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} ln(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))$$

$$\omega_{nk} = p(z = k|\mathbf{x}^{(n)}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

$$\Rightarrow L(\mathbf{x}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} ln(\sum_{k=1}^{K} \omega_{nk} \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\omega_{nk}}) \geqslant \sum_{n=1}^{N} \sum_{k=1}^{K} \omega_{nk} \, ln(\frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\omega_{nk}})$$

$$\Rightarrow Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = \sum_{n=1}^{N}\sum_{k=1}^{K} \omega_{nk}\, ln(\frac{\pi_k\, \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\omega_{nk}})$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} \omega_{nk}\, (ln(\pi_k) - ln(\omega_{nk}) - \frac{d}{2}\, ln(\sqrt{(2\pi)^d}) - \frac{1}{2}\, ln(det(\Sigma_k)) - \frac{1}{2}\, (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}$$
$$(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k))$$

$$\pi_k^{new} = arg\ max \sum_{n=1}^{N}\sum_{k=1}^{K} \omega_{nk}\, ln(\pi_k) \qquad subject\ to \quad \sum_{k=1}^{K} \pi_k = 1$$

$$L(\pi_k, \lambda) = \sum_{n=1}^{N}\sum_{k=1}^{K} \omega_{nk}\, ln(\pi_k) + \lambda(\sum_{k=1}^{K} \pi_k - 1)$$

$$\frac{\partial\, L(\pi_k, \lambda)}{\partial \pi_k} = \frac{1}{\pi_k} \sum_{n=1}^{N} \omega_{nk} + \lambda = 0 \Rightarrow \pi_k = \frac{-1}{\lambda} \sum_{n=1}^{N} \omega_{nk}$$

$$\frac{\partial\, L(\pi_k, \lambda)}{\partial \lambda} = \sum_{k=1}^{K} \pi_k - 1 = 0 \Rightarrow \sum_{k=1}^{K} \pi_k = \frac{-1}{\lambda} \sum_{n=1}^{N}\sum_{k=1}^{K} \omega_{nk} = 1 \Rightarrow \lambda = -\sum_{n=1}^{N}\sum_{k=1}^{K} \omega_{nk}$$

$$\Rightarrow \pi_k^{new} = \frac{-1}{-\sum_{n=1}^{N}\sum_{k=1}^{K} \omega_{nk}} \sum_{n=1}^{N} \omega_{nk} \quad , \quad \sum_{k=1}^{K} \omega_{nk} = 1 \Rightarrow \pi_k^{new} = \frac{1}{N} \sum_{n=1}^{N} \omega_{nk}$$

$$\boldsymbol{\mu}_k^{new} = arg\ max\ Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)$$

$$\frac{\partial Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)}{\partial \boldsymbol{\mu}_k} = 0 \Rightarrow \sum_{n=1}^{N} \omega_{nk} \frac{\partial\, [-\frac{1}{2}\, (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T\, \Sigma_k^{-1}\, (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)]}{\partial\, \boldsymbol{\mu}_k} = 0$$

$$\Rightarrow \sum_{n=1}^{N} \omega_{nk}\, \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k) = 0 \Rightarrow \sum_{n=1}^{N} \omega_{nk}\, \mathbf{x}^{(n)} = \boldsymbol{\mu}_k \sum_{n=1}^{N} \omega_{nk} \Rightarrow \boldsymbol{\mu}_k^{new} = \frac{\sum_{n=1}^{N} \omega_{nk}\, \mathbf{x}^{(n)}}{\sum_{n=1}^{N} \omega_{nk}}$$

$$\boldsymbol{\Sigma}_k^{new} = arg\ max\ Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)$$

$$\frac{\partial\, Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)}{\partial\, \boldsymbol{\Sigma}_k^{-1}} = 0 \Rightarrow \sum_{n=1}^{N} \omega_{nk}\, \frac{\partial\, [-\frac{1}{2}\, ln(det(\boldsymbol{\Sigma}_k)) - \frac{1}{2}\, (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T\, \boldsymbol{\Sigma}_k^{-1}\, (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)]}{\partial\, \boldsymbol{\Sigma}_k^{-1}} = 0$$

$$-\frac{1}{2} \sum_{n=1}^{N} \omega_{nk}\, \frac{\partial\, [ln(det(\boldsymbol{\Sigma}_k)) + (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T\, \boldsymbol{\Sigma}_k^{-1}\, (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)]}{\partial\, \boldsymbol{\Sigma}_k^{-1}} = 0$$

$$\frac{1}{2} \sum_{n=1}^{N} \omega_{nk}\, [\boldsymbol{\Sigma}_k - (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T\, (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)] = 0$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{\sum_{n=1}^{N} \omega_{nk}\, [(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k)^T\, (\mathbf{x}^{(n)} - \boldsymbol{\mu}_k^{9})]}{\sum_{n=1}^{N} \omega_{nk}}$$

Theory Question 5) Compute estimate of parameters for Categorical Mixture Models for N observed data $\{x_i\}_{i=1}^N$.

1. **Determine model parameters and initialize them.**
2. **Compute complete dataset likelihood.**
3. **Find closed-form solution for parameters using EM algorithm.**

1.

- Determine the number of Categorical components (K) in the mixture model.
- Let π_k be the mixing coefficient for component k, such that $\sum_{k=1}^{K} \pi_k = 1$.
- Let θ_k be the parameter vector for component k, such that $\boldsymbol{\theta_k} = (\theta_{k1}, \ldots, \theta_{kd})$ where d is the number of possible outcomes for each data point.
- In a categorical model, each θ_{kj} represents the probability of observing outcome j for component k.
- Initialize the parameters π_k and θ_k to random values, subject to the constraint that $\sum_{k=1}^{K} \pi_k = 1$.

2.
Compute the likelihood of the complete dataset, which involves both observed data and latent variables. The complete dataset likelihood is given by:

$$p_{\mathbf{X},\mathbf{Z}}\left(\ \{\mathbf{x}_n\}\ ,\ \{z_n\}\ \mid\ \boldsymbol{\theta}\ \right) = \prod_{n=1}^{N}\sum_{k=1}^{K} p_{\mathbf{X}|\mathbf{Z}}\left(\mathbf{x}_n|z=k\right).p_{\mathbf{Z}}\left(z=k\right) = \prod_{n=1}^{N}\sum_{k=1}^{K} \theta_{x_n}^{(k)}.\pi_k$$

3.

$$q_n^*\left(z_k\right) = p_{\mathbf{Z}|\mathbf{X}}\left(z_k\ \mid\ \mathbf{x}_n\ ,\ \boldsymbol{\theta}\right) = \frac{p_{\mathbf{X},\mathbf{Z}}(z_k\ ,\ \mathbf{x}_n\ \mid\ \boldsymbol{\theta})}{p_{\mathbf{X}}\left(\mathbf{x}_n\right)} = \frac{p_{|\mathbf{Z}}\left(\mathbf{x}_n|z_k\right).p_{\mathbf{Z}}\left(z_k\right)}{\sum_{i=1}^{K} p_{\mathbf{X}|\mathbf{Z}}\left(\mathbf{x}_n|z_i\right).p_{\mathbf{Z}}\left(z_i\right)}$$

$$= \frac{\theta_{\mathbf{x}_n}^{(k)}.\pi_k}{\sum_{i=1}^{K} \theta_{\mathbf{x}_n}^{(i)}.\pi_i}$$

$$Q = \sum_{n=1}^{N}\sum_{k=1}^{K} q_n^*(z_k)\ .\log\left(p_{\mathbf{X},\mathbf{Z}}(\mathbf{x}_n\ ,\ z_k)\right) = \sum_{n=1}^{N}\sum_{k=1}^{K} q_n^*(z_k)\ .\log\left(\pi_k\ .\ \theta_{x_n}^{(k)}\right)$$

$$optimize\ Q\ over\ \pi\ ,\ condition\, C = \sum_{k=1}^{K}\pi_k = 1 \quad \Longrightarrow\ \nabla Q = \lambda.\nabla C$$

$$\frac{\partial Q}{\partial \pi_k} = \frac{1}{\pi_k}\sum_{n=1}^{N} q_n^*(z_k) = \lambda\ .\ \frac{\partial C}{\partial \pi_k} = \lambda \implies \pi_k^* = \frac{1}{\lambda}\sum_{n=1}^{N} q_n^*(z_k)$$

$$\sum_{k=1}^{K}\pi_k^* = 1 = \sum_{k=1}^{K}\frac{1}{\lambda}\sum_{n=1}^{N} q_n^*(z_k) = \frac{1}{\lambda}\sum_{n=1}^{N}\sum_{k=1}^{K} q_n^*(z_k) = \frac{N}{\lambda} \implies \lambda = N$$

$$\Longrightarrow\ \pi_k^* = \frac{1}{N}\sum_{n=1}^{N} q_n^*(z_k)$$

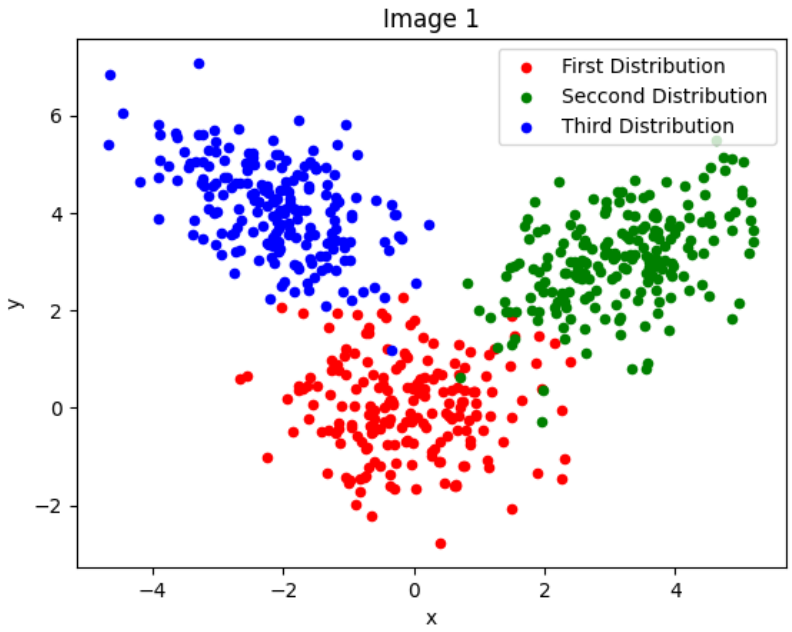
$$optimize\ Q\ over\ \theta^{(k)}\ ,\ condition\, C = \sum_{s=1}^{S}\theta_s^k = 1 \quad \Longrightarrow\ \nabla Q = \lambda.\nabla C$$

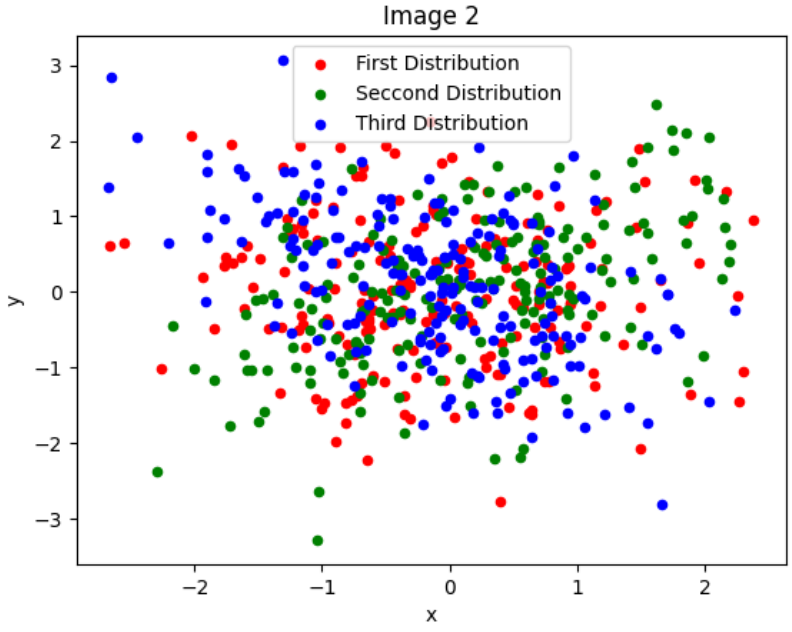
$$\frac{\partial Q}{\partial \theta_s^k} = \frac{1}{\theta_s^k}\sum_{n|\mathbf{x}_n=s} q_n^*(z_k) = \lambda\ .\ \frac{\partial C}{\partial \theta_s^k} = \lambda \implies \theta_s^{k^*} = \frac{1}{\lambda}\sum_{n|x_n=s} q_n^*(z_k)$$

$$\sum_{s=1}^{S}\theta_s^k = 1 = \sum_{s=1}^{S}\frac{1}{\lambda}\sum_{n|\mathbf{x}_n=s} q_n^*(z_k) = \frac{1}{\lambda}\sum_{n=1}^{N} q_n^*(z_k) \implies \lambda = \frac{1}{\sum_{n=1}^{N} q_n^*(z_k)}$$

$$\Longrightarrow \theta_s^{k^*} = \frac{\sum_{n|x_n=s} q_n^*(z_k)}{\sum_{n=1}^{N} q_n^*(z_k)}$$

Simulation Question 1) Each distribution has 200 data points that are concatenated in a two-dimensional array and given to you. Plot the data with three different colors in a graph.





Simulation Question 2) Write a function that performs the E-step. This means assigning each data to a distribution based on the Euclidean distance. Return as output a 3×600 array specifying which distribution each data belongs to. If the R_{ij} is one, it means that the i–th data is assigned to the j–th distribution. Run this function for one iteration and report the result.

```
def multivariate_normal_prob(vector, mean, covariance):
  vector = np.matrix(vector).T
  mean = np.matrix(mean).T
  covariance = np.matrix(covariance)
  vector = np.asarray(vector)
  mean = np.asarray(mean)
  covariance = np.asarray(covariance)

  # Calculate the Mahalanobis distance
  diff = vector - mean
  inv_covariance = np.linalg.pinv(covariance)
  mahalanobis_distance = np.dot(np.dot(diff.T, inv_covariance), diff)

  # Calculate the probability using the formula for multivariate normal distribution
  d = len(vector)
  coeff = 1 / ((2 * np.pi)**(d/2) * np.linalg.det(covariance)**(1/2))
  prob = coeff * np.exp(-0.5 * mahalanobis_distance)
  return prob


# E step function
def E_Step(N, K, dataset, pi, mu, sigma):
  w = np.zeros([600,3])
  for n in range(N):
    # calculate denominator
    denominator = 0
    for k in range(K):
      denominator = denominator + pi[n,k] * multivariate_normal_prob(dataset.iloc[n], mu[
                                          k,:], sigma[k,:,:])
    for k in range(K):
      w[n,k] = (pi[n,k] * multivariate_normal_prob(dataset.iloc[n], mu[k,:], sigma[k,:,:]
                                          )) / denominator
  return w

# initialize the means, covariances, and mixture weights with random number
pi = np.zeros([600,3])+[0.2,0.3,0.5]
mu = np.matrix([[2.252,1.215], [1.022,2.123], [1.957,2.526]])
sigma = np.zeros([3,2,2])
sigma[0,:,:] = [[5.978,2.887],[1.124,2.884]]
sigma[1,:,:] = [[5.684,2.477],[1.212,2.369]]
sigma[2,:,:] = [[5.258,2.459],[1.964,2.128]]

# apply E step function
R_1 = E_Step(600,3,df1[df1.columns[1:3]], pi, mu, sigma)
R_2 = E_Step(600,3,df2[df2.columns[1:3]], pi, mu, sigma)
```

```
    # save R in .csv file
    arr = np.asarray(R_1)
    pd.DataFrame(arr).to_csv('pi_k(Image 1).csv')
    arr = np.asarray(R_2)
    pd.DataFrame(arr).to_csv('pi_k(Image 2).csv')
```

As you can understand from the code, at first we initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$, and mixture weights π_k for each Gaussian component. After that, we apply the E Step function on the initial values to improve the estimation of π_k parameter. at final save π_k^{new} in pi k.csv and as we expected summation of each row is equal to 1.

Simulation Question 3) Write a function that performs M-step. This means updating the mean and variance of each distribution. Run this function for one iteration and report the new variances and means of each distribution.

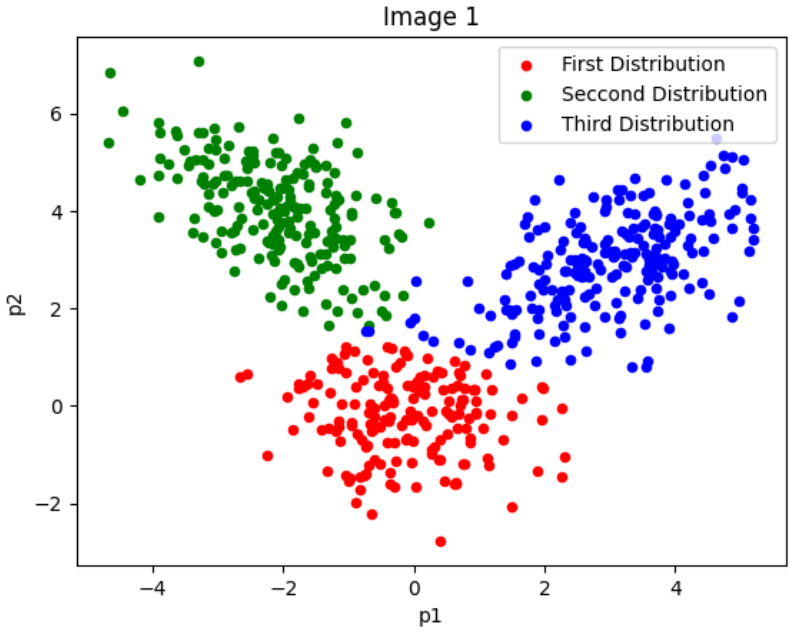
```
# M step function
def M_Step(N, K, dataset, w):
  # update mu
  mu_new = np.zeros([K,2])
  for k in range(K):
    # calculate denominator and numerator
    denominator = 0
    numerator = 0
    for n in range(N):
      denominator = denominator + w[n,k]
      numerator = numerator + w[n,k] * dataset.iloc[n]
    mu_new[k,:] = numerator / denominator

  # update sigma
  sigma_new = np.zeros([3,2,2])
  for k in range(K):
    # calculate numerator
    numerator = np.zeros([2,2])
    for n in range(N):
      numerator = numerator + w[n,k] * ((np.matrix(dataset.iloc[n]) - mu[k,:]).T @ (np.
                                        matrix(dataset.iloc[n]) - mu[k,:]))
    sigma_new[k,:,:] = numerator / denominator

  return mu_new,sigma_new

# apply M step function im image 1
print('OUTPUT OF IMAGE 1')
mu_new, sigma_new = M_Step(600,3,df1[df1.columns[1:3]], R_1)
print('updated mean : ')
print(mu_new)
print('updated sigma : ')
print(sigma_new)

# previous stage mean and sigma
print('previous stage mean : ')
print(mu)
print('previous stage sigma : ')
print(sigma)

# compare with main mean and sigma
print('main mean : ')
print(np.matrix([df1[df1.columns[1:3]].iloc[0:200].mean(),df1[df1.columns[1:3]].iloc[200:
                                        400].mean(),df1[df1.columns[1:3]].iloc[400:600
                                        ].mean()]))
print('main sigma : ')
sigma = np.zeros([3,2,2])
sigma[0,:,:] = np.matrix(df1[df1.columns[1:3]].iloc[0:200].cov())
sigma[1,:,:] = np.matrix(df1[df1.columns[1:3]].iloc[200:400].cov())
sigma[2,:,:] = np.matrix(df1[df1.columns[1:3]].iloc[400:600].cov())
```

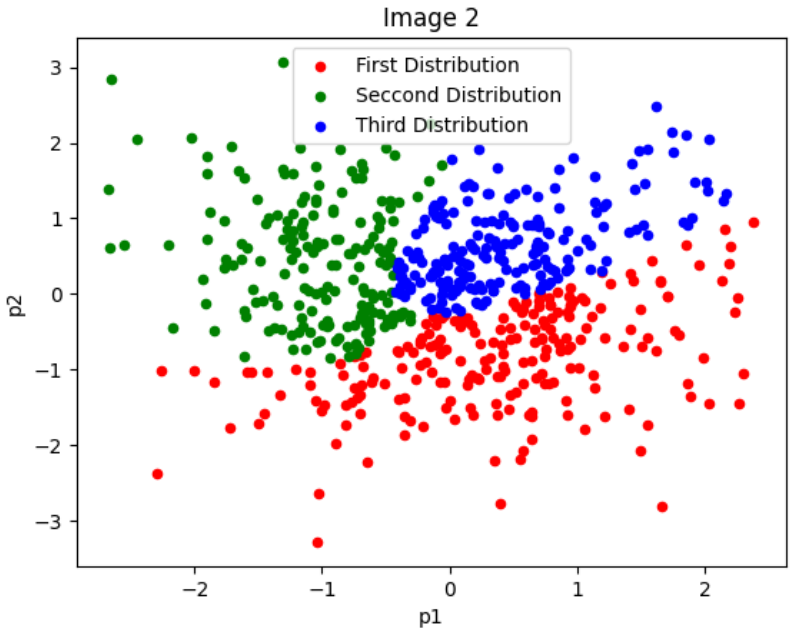
```
print(sigma)

# apply M step function in image 2
print('\n\n OUTPUT OF IMAGE 2')
mu_new, sigma_new = M_Step(600,3,df2[df2.columns[1:3]], R_2)
print('updated mean : ')
print(mu_new)
print('updated sigma : ')
print(sigma_new)

# previous stage mean and sigma
print('previous stage mean : ')
print(mu)
print('previous stage sigma : ')
print(sigma)

# compare with main mean and sigma
print('main mean : ')
print(np.matrix([df2[df2.columns[1:3]].iloc[0:200].mean(),df2[df2.columns[1:3]].iloc[200:
                                        400].mean(),df2[df2.columns[1:3]].iloc[400:600
                                        ].mean()]))
print('main sigma : ')
sigma = np.zeros([3,2,2])
sigma[0,:,:] = np.matrix(df2[df2.columns[1:3]].iloc[0:200].cov())
sigma[1,:,:] = np.matrix(df2[df2.columns[1:3]].iloc[200:400].cov())
sigma[2,:,:] = np.matrix(df2[df2.columns[1:3]].iloc[400:600].cov())
print(sigma)
```

```
OUTPUT OF IMAGE 1
updated mean :
[[ 0.75663775  0.71770831]
[-0.83403292  3.0972402 ]
[ 1.46562268  2.32025204]]
updated sigma :
[[[ 2.7644759   0.82114678]
[ 0.82114678  1.9578197 ]]

[[10.00079165 -4.11654108]
[-4.11654108  5.51346329]]

[[ 4.87623486  1.68875855]
[ 1.68875855  2.48616741]]]
previous stage mean :
[[2.252 1.215]
[1.022 2.123]
[1.957 2.526]]
previous stage sigma :
[[[5.978 2.887]
[1.124 2.884]]

[[5.684 2.477]
[1.212 2.369]]

[[5.258 2.459]
[1.964 2.128]]]
```

```
main mean :
[[-0.1030724   0.00457454]
 [ 3.13796774  3.01442499]
 [-2.11151422  4.11454718]]
main sigma :
[[[ 0.97358273 -0.0472312 ]
  [-0.0472312   0.93918291]]

 [[ 1.02775355  0.4604333 ]
  [ 0.4604333   0.94546824]]

 [[ 0.8731411  -0.44841   ]
  [-0.44841     0.90568068]]]


OUTPUT OF IMAGE 2
updated mean :
[[ 0.20156802 -0.45310689]
 [-0.17299762  0.19692419]
 [-0.12254544  0.4066152 ]]
updated sigma :
[[[5.0865413  3.46324752]
  [3.46324752 3.63330082]]

 [[2.10722456 1.98903585]
  [1.98903585 3.98357614]]

 [[5.26608621 4.5045859 ]
  [4.5045859  5.16360124]]]
previous stage mean :
[[2.252 1.215]
 [1.022 2.123]
 [1.957 2.526]]
previous stage sigma :
[[[ 0.97358273 -0.0472312 ]
  [-0.0472312   0.93918291]]

 [[ 1.02775355  0.4604333 ]
  [ 0.4604333   0.94546824]]

 [[ 0.8731411  -0.44841   ]
  [-0.44841     0.90568068]]]
main mean :
[[-0.1030724   0.00457454]
 [ 0.13796774  0.01442499]
 [-0.11151422  0.11454718]]
main sigma :
[[[ 0.97358273 -0.0472312 ]
  [-0.0472312   0.93918291]]

 [[ 1.02775355  0.4604333 ]
  [ 0.4604333   0.94546824]]

 [[ 0.8731411  -0.44841   ]
  [-0.44841     0.90568068]]]
```

Simulation Question 4) Using the functions you have written, run the EM algorithm until a convergence is reached or the maximum number of steps is passed. Replot the three new distributions and compare with the correct labels.





As you can see from figures clustering is well done.

Simulation Question 5) Compare the parameters obtained from each of the images and explain the reason for their difference.

Once the EM algorithm has converged, it produces a set of parameter estimates that can be used to generate new data points that follow the same statistical distribution as the original dataset. The quality of the parameter estimates depends on several factors, including the complexity of the GMM, the amount and quality of the input data, and the initialization of the algorithm.
It's important to keep in mind that the results of the EM algorithm are just estimates, and they may not perfectly match the original dataset. However, the goal of the EM algorithm is to provide a useful approximation of the underlying statistical distribution, which can be used for a variety of purposes, such as data clustering, anomaly detection, and generative modeling. For the first image, the results are very close to the true parameters. That is because the means of the distribution are very far from each other, and it was easy to distinguish them. Still some misclassifications can be seen on the borders because the probabilities of different distributions are close in those areas.
For the second image, the results are very different from the true image. That is because the parameters of the distributions are not very different and so it is hard to distinguish them.