INDIAN INSTITUTE OF TECHNOLOGY DHARWAD

Lab 3

CS 412: Statistical Pattern Recognition

MEMBERS
GANESH SAMARTH - 170020030
MANDEEP BAWA - 170030038
S V PRAVEEN - 170010025

Link to Code - GitHub Repo mixture-models

Solution 1

Estimation using EM, K-means and Gradient Descent

I. MNIST Dataset

It is a dataset containing handwritten digits {0, 1, ..., 9 } and their labels. Each image is in a 28*28 pixel format. The digits are size-normalized and centered in a fixed-size image.

Referring to Chapter 9, Section 9.3.3 "Pattern Recognition and Machine Learning" by Christopher Bishop

Distribution of classes taken as given in the reference

Label	Samples
2	200
3	200
4	200

II. Preprocessing

Dataset is converted to 0, 1 random variable taking threshold point as 0.5

III. Mixed Bernoulli Distribution

Mixed Multivariate Bernoulli distribution is a mixture of many Multivariate Bernoulli distributions. Each Multivariate Bernoulli Distribution tries to modulate a particular label from mixed samples.

$$f(x) \ = \sum_{j=1}^{k} \lambda_j * f_j(x)$$

Where f(x) is the mixture density formed by mixtures of f_j

We have chosen a mixture of 3 classes in this estimation
Multivariate Bernoulli distribution is a generalization of the one dimensional Bernoulli distribution to higher dimensions.
Here, we built a 784 dimensional bernoulli distribution.

For samples D= (x_1, x_2, ..., x_n), the log likelihood of the data is given by

$$log(p(D/\mu, \lambda)) \ = \sum_{i=1}^{n} ln(\sum_{j=1}^{j=k} \lambda_j * \ p(x_i/\mu_j))$$

$$p(x_i/\mu_j) \ = \ \prod_{t=1}^{786} (\mu_{jt})^{x_{jt}} * \ (1 \ - \ \mu_{jt})^{1-x_{jt}}$$

where µis a (3*786) mean vector, λis a (3,1) priors, k is the number of classes

As summation of log is coming in the log likelihood equation, we will use EM algorithm to get MLE estimates.

E - step

Calculating posterior

$$\gamma_{ij} = \frac{\lambda_j * p(x_i/\mu_j)}{\sum_{j=1}^{k} \lambda_j * p(x_i/\mu_j)}$$

Expectation over data

$$Q(\theta, \theta^0) \ = \ \sum_{i=1}^{n}\sum_{j=1}^{j=k} \gamma_{ij} \ * \ (\ ln(\lambda_j) \ + \ \sum_{t=1}^{t=786} [x_{it} \ * \ ln(\mu_{jt}) \ + \ (1 \ - \ x_{it}) \ * \ (1 \ - \ ln(\mu_{jt})) \ ] \ )$$

M - step

We will maximize $Q(\theta, \theta^{0})$over θ

$$\theta^{1} = argmax\, Q(\theta, \theta^{0})$$

a. Iterative Improvements

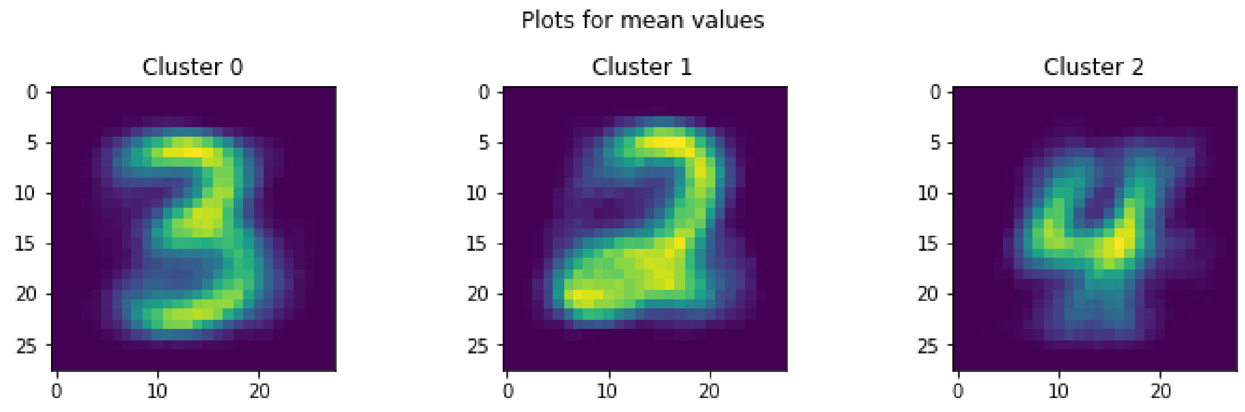
$$N_{k} = \sum_{i=1}^{n} \gamma_{ik}^{0}$$

$$\mu_{k}^{1} = \frac{1}{N_{k}} \sum_{i=1}^{n} \gamma_{ik}^{0} * x_{i}$$

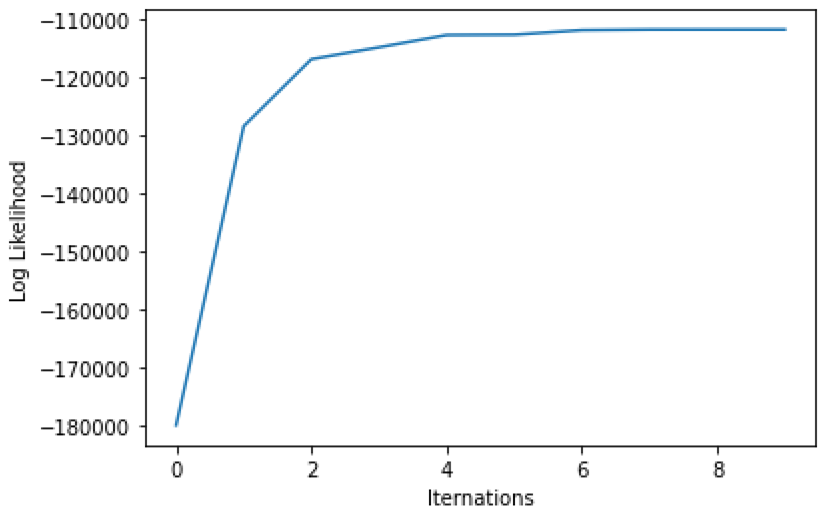
$$\lambda_{k}^{1} = \frac{N_{k}}{N}$$

b. Results

After getting the MLE estimates, we can plot the mean values



Variations in Log Likelihood with iterations



IV. K-means Clustering

Now, we use K-means clustering to separate the classes from mixture densities.
It is a very simple and efficient algorithm, which finds the best estimates using iterations.

a. Algorithm

Step 1

Initialize μ with random values from 0 to 1

Step 2

Assign the points to the clusters based on the closest μ

$$z_i = argmin_k(||x_i - \mu_k||_2)$$

One Hot encode z_i to a vector

Step 3

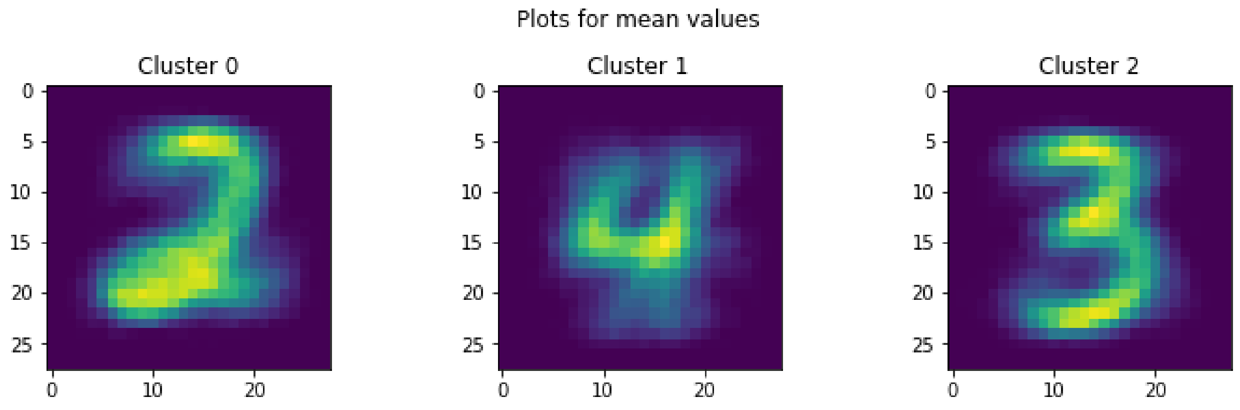
Update μ for each cluster as

$$\mu_k = \frac{\sum_{i=1}^{n} x_i * z_{ik}}{\sum_{i=1}^{n} z_{ik}}$$

Go again to step 2

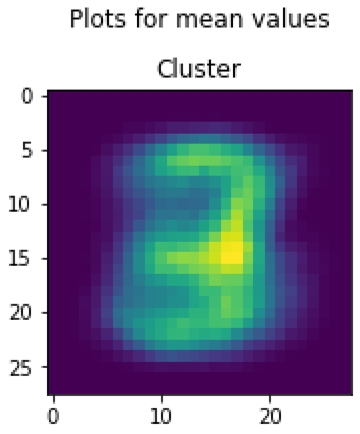
b. Results

After getting the estimates, we can plot the mean values



V. Single Multivariate Bernoulli Distribution

Estimating parameters using MLE for single multivariate bernoulli distribution.



VI. Gradient Descent

Now, we use Gradient descent on the likelihood function to separate the classes from mixture densities.

a. Equations

Likelihood function

$$log(p(D/\mu,\lambda)) \;=\; \sum_{i=1}^{n} ln(\sum_{j=1}^{j=k} \lambda_j * p(x_i/\mu_j))$$

Gamma function

$$\gamma_{ij} = \frac{\lambda_j * p(x_i/\mu_j)}{\sum_{j=1}^{k} \lambda_j * p(x_i/\mu_j)}$$

Derivatives w.r.t μ_{jt}

$$\frac{\partial}{\partial \mu_{jt}}(log(p(D/\mu,\lambda))) = \sum_{i=1}^{n} \gamma_{ij} * \; [x_{it} + \mu_{jt} - 2 * x_{it} * \mu_{jt}])$$

Derivative w.r.t λ_j

$$\frac{\partial}{\partial \lambda_j}(log(p(D/\mu,\lambda))) \;=\; \frac{\sum_{i=1}^{n} \gamma_{ij}}{\lambda_j}$$

Updating Equations

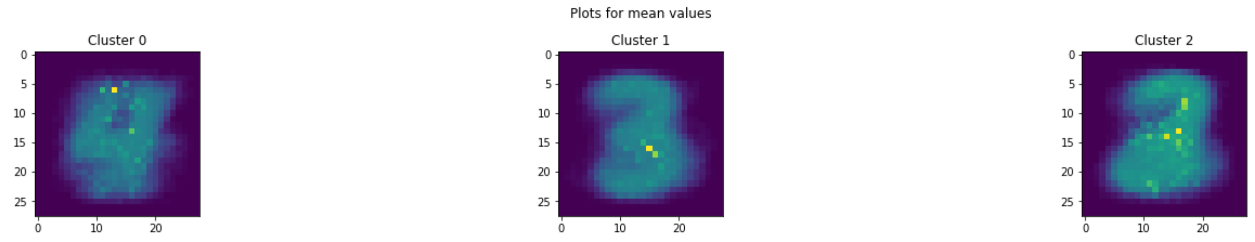
$$\mu_{jt}^{(1)} = \mu_{jt}^{(0)} \; + \; \eta * \frac{\partial}{\partial \mu_{jt}}(log(p(D/\mu,\lambda)))$$

$$\lambda_{j}^{(1)} = \lambda_{j}^{(0)} \; + \; \eta * \frac{\partial}{\partial \lambda_{j}}(log(p(D/\mu,\lambda)))$$

We use these updating equations to get $\mu_{jt}^{(t)}$ and $\lambda_j^{(t)}$

b. Results

Plotting the mean values after using gradient descent



VII. Conclusion

a. We were able to validate the results presented in the section 9.3.3 from "Pattern Recognition and Machine Learning" by Christopher Bishop.
b. Log likelihood function in EM increased as iterations were performed using the E and M step.
c. K-means and EM were run for 10 iterations, both performed equally well, however K-means required less computational power than EM.
d. Gradient Descent took 30 iterations to converge to the label images with 0.0001 as learning rate.
e. Single Multivariate gaussian was able to model the most occurring pixels among all the labels.

SOLUTION 2

EM Algorithm for Univariate and Multivariate Gaussian Mixture Models (GMMs)

I. Dataset

Sampled 1500 points from a Gaussian Mixture Model (GMM).

- **Parameters**
  - **Univariate (2 Components) -**
    - Number of components, K = 2
    - Means

      $\mu_1 = 1, \ \mu_2 = 4$
    - Variances

      $\sigma_1^2 = 0.7670192403518237, \ \sigma_2^2 = 0.2979386319204766$
    - Priors

      $\lambda_1 = 0.6, \lambda_2 = 0.4$
  - **Univariate (5 Components) -**
    - Number of components, K = 5
    - Means

      $\mu_1 = 1, \ \mu_2 = 2, \ \mu_3 = 3, \ \mu_4 = 4, \ \mu_5 = 5$
    - Variances

[0.459034464689231, 0.207702519428098, 0.23489427229510515, 0.17162219198716364, 0.5122055673069296]

- Priors

  $\lambda_1 = 0.1,\ \lambda_2 = 0.2,\ \lambda_3 = 0.4,\ \lambda_4 = 0.15,\ \lambda_5 = 0.15$

- **Multivariate -**
  - Number of components, K = 3
  - Feature Dimension = 2
  - Means

    $\mu_1 = (-4,\ -5),\ \mu_2 = (0,\ 0),\ \mu_3 = (4,\ 5)$
  - Covariance matrices.

    $\Sigma_1 = I,\ \Sigma_2 = I,\ \Sigma_3 = I$
  - Priors

    $\lambda_1 = 0.2,\ \lambda_2 = 0.5,\ \lambda_3 = 0.3$

II. EM Algorithm for GMM

1. Univariate

i. Randomly Initialize parameters

ii. Iteratively update the parameters while maximizing the likelihood.

$$\gamma_{ij}^{(t+1)} = \frac{\lambda_j^{(t)} N(x_i, \mu_j^{(t)}, \sigma_j^{2(t)})}{\sum_{k=1}^{K} \lambda_k^{(t)} \ N(x_i, \mu_k^{(t)}, \sigma_k^{2(t)})}$$

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

$$\sigma_j^{2(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ij\hat{}}^{(t)} (x_i - \mu_j)^2}{\sum_{i=1}^{n} \gamma_{ij}^{(t)}}$$

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

iii. Repeat 2 until epsilon convergence.

2. **Multivariate**
   i. Randomly Initialize parameters
   ii. Iteratively update the parameters while maximizing the likelihood.

$$\gamma_{ij}^{(t+1)} = \frac{\lambda_j^{(t)} N(x_i, \mu_j^{(t)}, \sigma_j^{2(t)})}{\sum_{k=1}^{K} \lambda_k^{(t)} \; N(x_i, \mu_k^{(t)}, \sigma_k^{2(t)})}$$

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

$$\Sigma_j^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ij\hat{}}^{(t)} (x_i - \mu_j^{(t+1)}) \; (x_i - \mu_j^{(t+1)})}{\sum_{i=1}^{n} \gamma_{ij}^{(t)}}$$

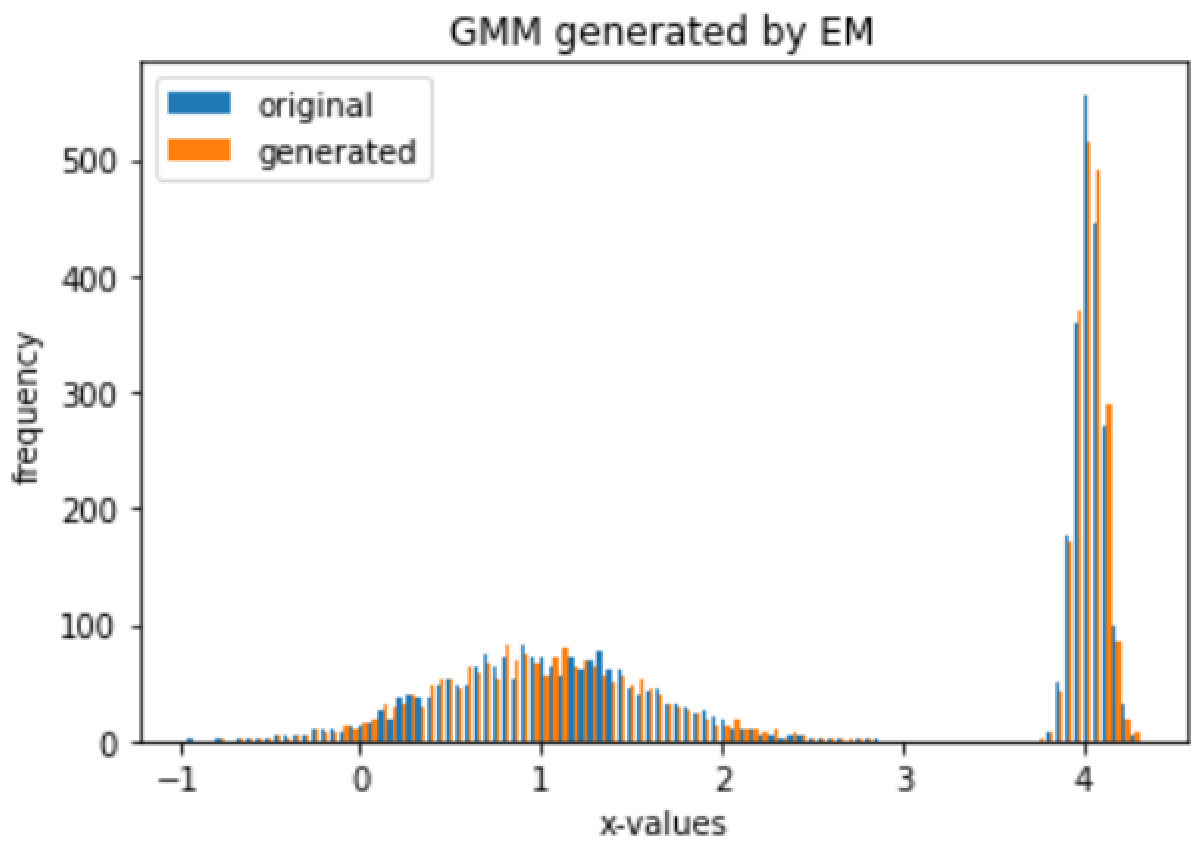
$$\lambda_j^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ij}^{(t)}}{n}$$

iii. Repeat 2 until epsilon convergence.

III. Results

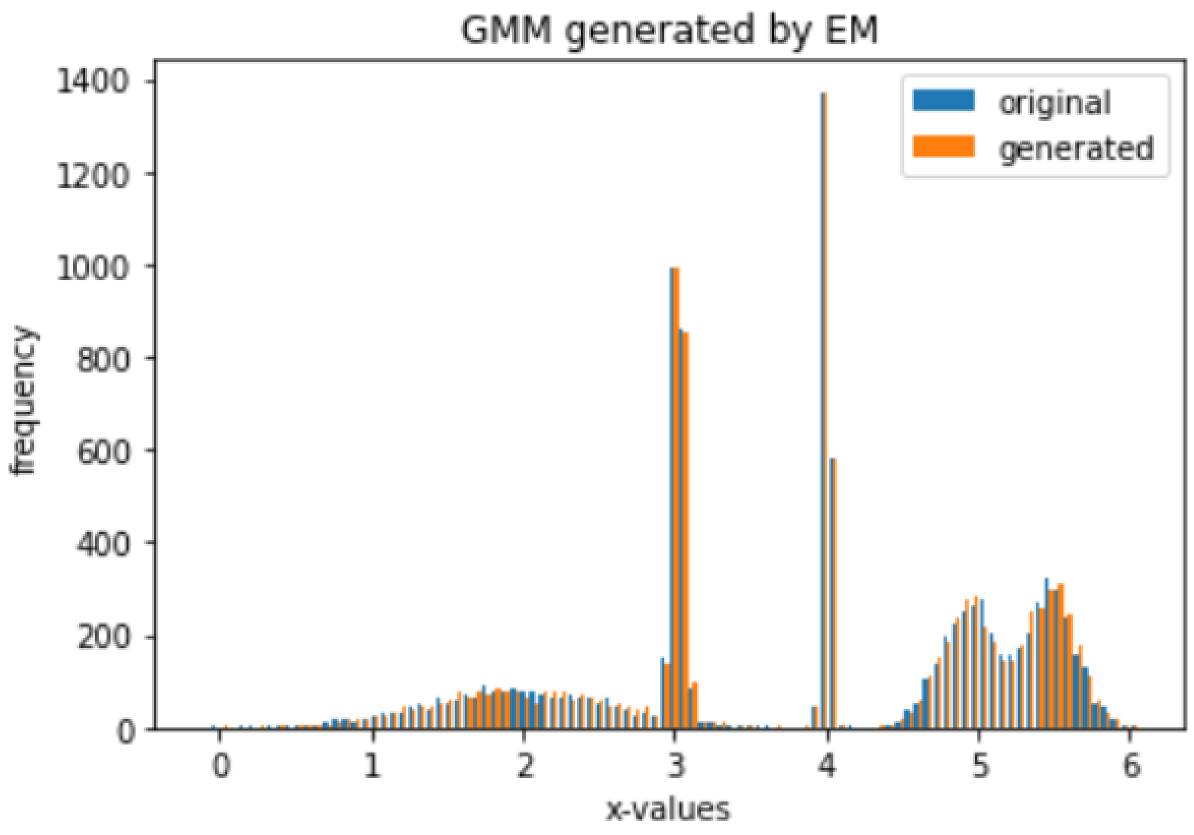
```
Univariate Results:
```

-
  ```
  K = 3
  means - [0.98541005 3.97427681],
  variances - [0.58584664 0.08486503],
  lambdas = [0.5957197 0.4042803]
  ```



-
  ```
  K = 5
  Results:
  means - [4.93188145 4.00888024 2.00035397 5.49257333 3.02821754],
  ```

```
variances - [0.18553479 0.02467827 0.61646364 0.16346885 0.0368454]
lambdas = [0.1155429  0.14763037 0.40559525 0.02991026 0.30132122]
```



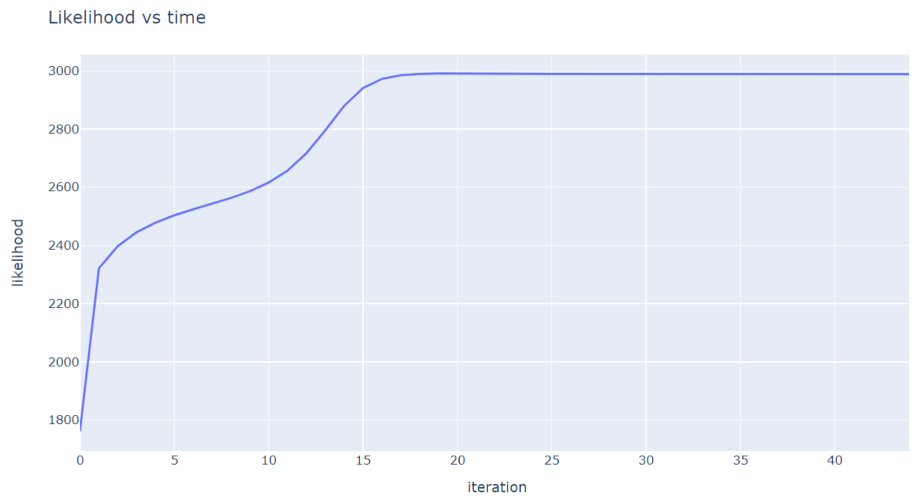
Which are very close to the original distribution.

Multivariate Results:

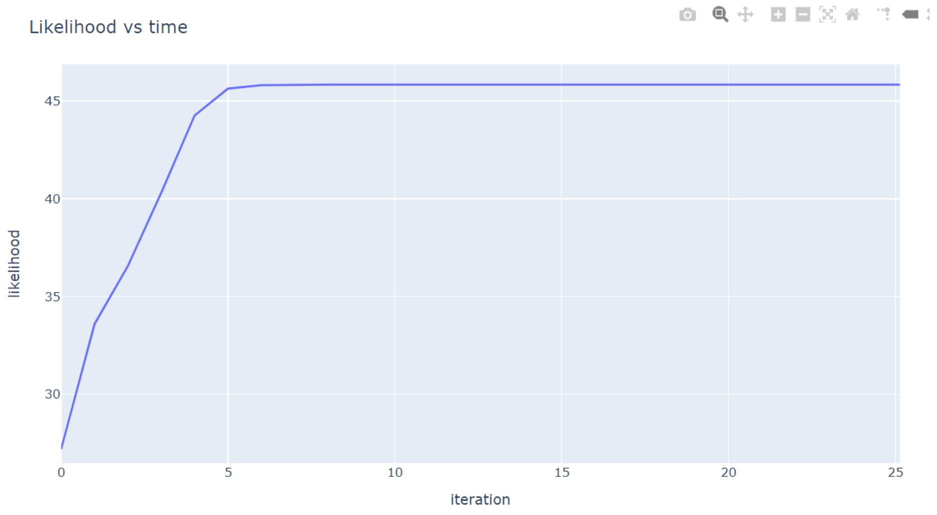
```
means - [[-4.116174   -4.99465148]
 [-0.03083184  0.05894495]
 [ 3.9375611   5.01749467]],
variances - [[[ 0.97883548 -0.04590425]
  [-0.04590425  0.89823411]]

 [[ 0.91472256  0.03783833]
  [ 0.03783833  0.94821342]]

 [[ 1.03326969  0.02982488]
  [ 0.02982488  1.07705795]]],
lambdas = [0.20027376 0.49914707 0.30057917]
```

Which is very close to the original distribution.

Clearly, we see from the below plots that the likelihood increases with time. For the univariate case with 5 components -



For the multivariate case with 3 components -



IV. Conclusion

The EM algorithm is successfully able to retrieve the mixture densities in case of GMM although the results and the speed of convergence depend on the random initialization. Compared to Parzen window estimation, in this scenario we already know the number of components in the mixture and are able to get good estimates for the true mixture.

Parzen Window density estimation

Parzen window estimate is a non-parametric density estimation technique. This technique intuitively sums up the contribution of each data point through a window function. This density is estimated essentially in a volume governed by h.

For data points D = {x1, x2, x3, x4....xn}

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V} \varphi \frac{(x - x_i)}{h}$$

The final density function is calculated by summing up all the contributions at a particular point x.

Considering a case for a univariate gaussian distribution with K components

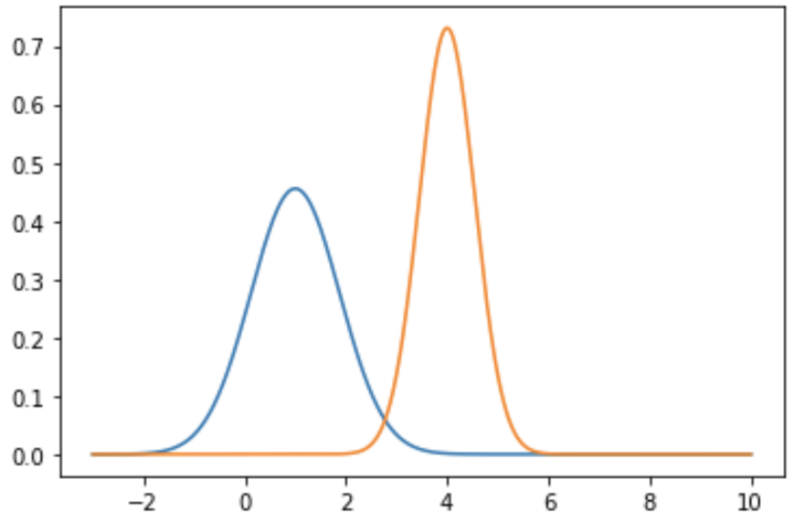
$$p(x) = \sum_{i=1}^{K} \phi_i \mathcal{N}(x \mid \mu_i, \sigma_i)$$

$$\mathcal{N}(x \mid \mu_i, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(x - \mu_i)^2}{2\sigma_i^2}\right)$$

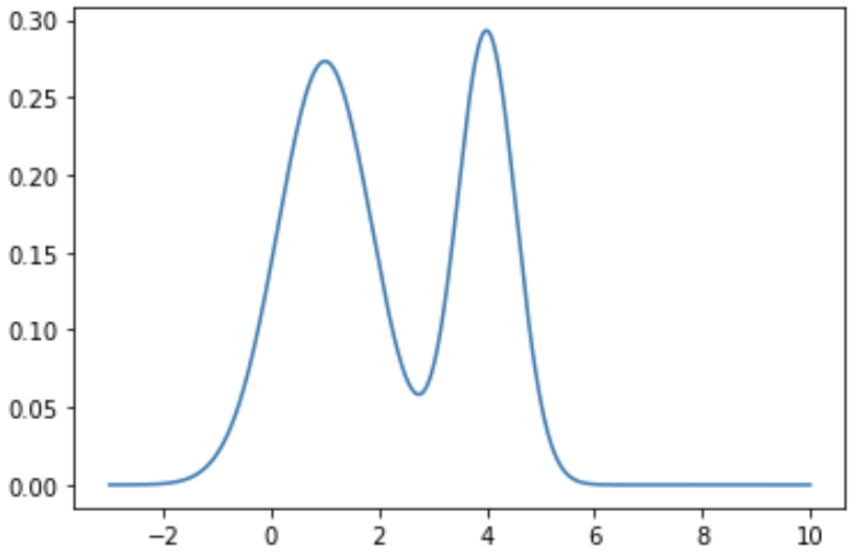
$$\sum_{i=1}^{K} \phi_i = 1$$

Where K is the number of mixtures, μ and σ are the parameters of each gaussian component and ϕ is the mixing probability of each component.

As the first step, we generate a GMM with 2 components, with a mixing probability of 0.6 and 0.4, mean of 1 and 4 and a variance of 0.77 and 0.3 respectively.
The individual components gaussian components are as given below.



And as a mixture of components, they are as given below.

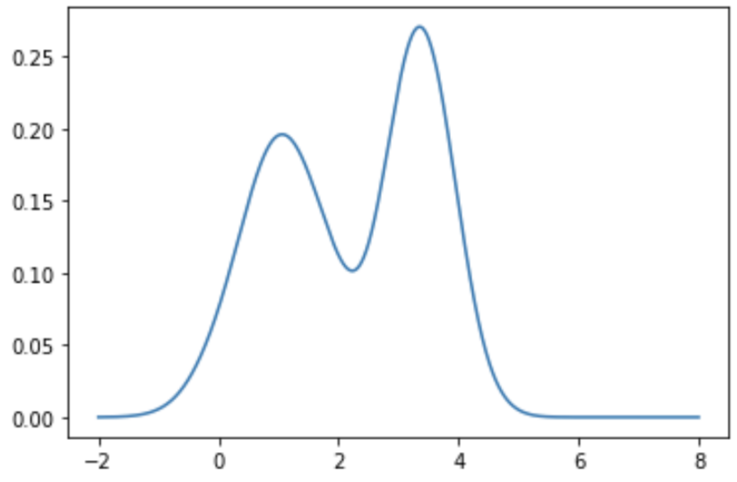


Sampling

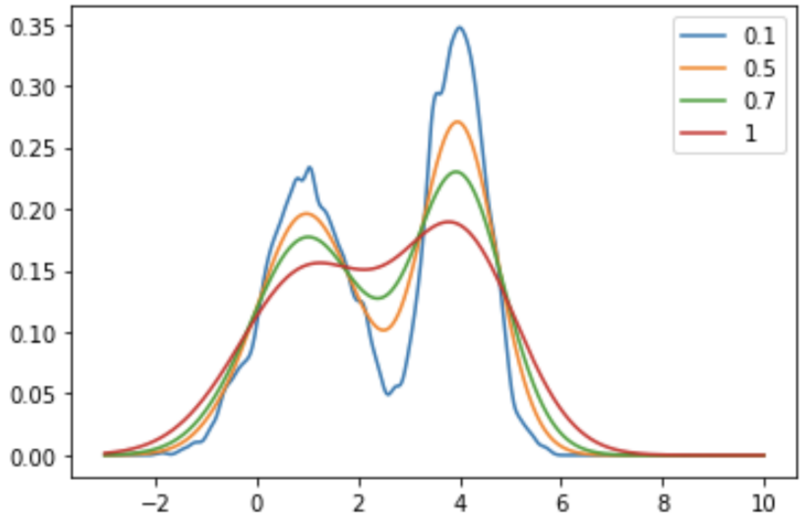
To sample points from the GMM, we generate samples from both gaussians and sample them based on their mixing probabilities.

Gaussian Kernel

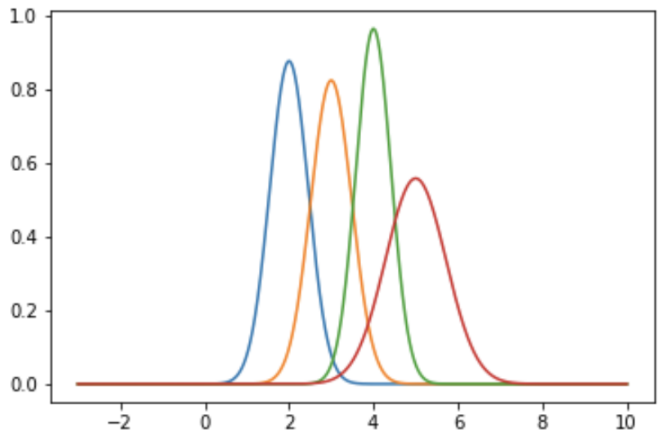
We first use a standard normal kernel as the window function, to estimate the density. Essentially, we consider the mixture to be a combination of n gaussians with equal mixing probabilities of 1/n. We experiment with different window sizes, to arrive at the optimum h, which is the closest approximation to the actual density function. The volume V, is determined to be h^d.
The density estimate for a h = 0.5



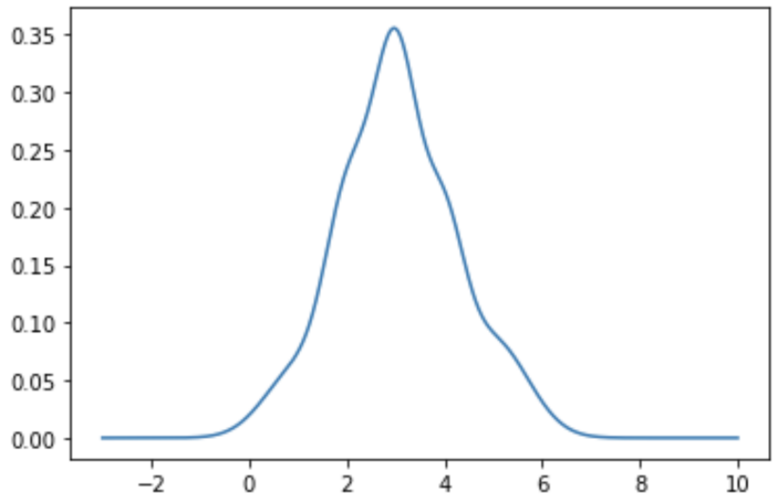
Parzen window density estimated with varying window size



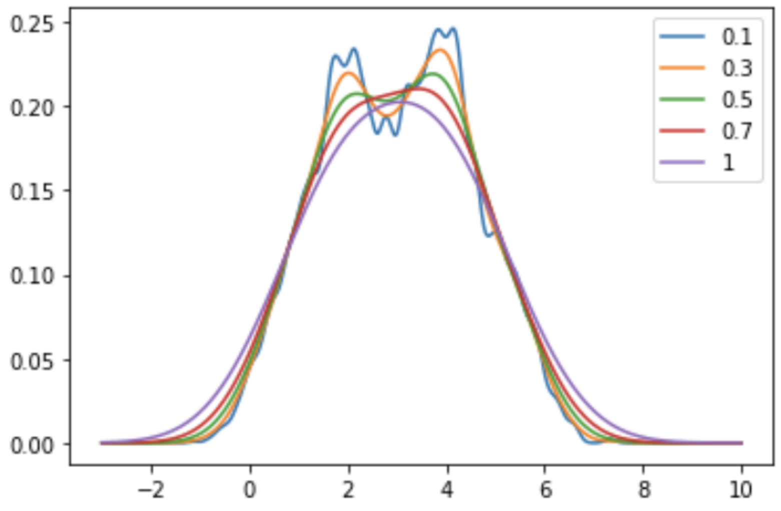
The same idea can be extended to GMMs with 5 components -



GMM model



Parzen window density estimate with varying window sizes

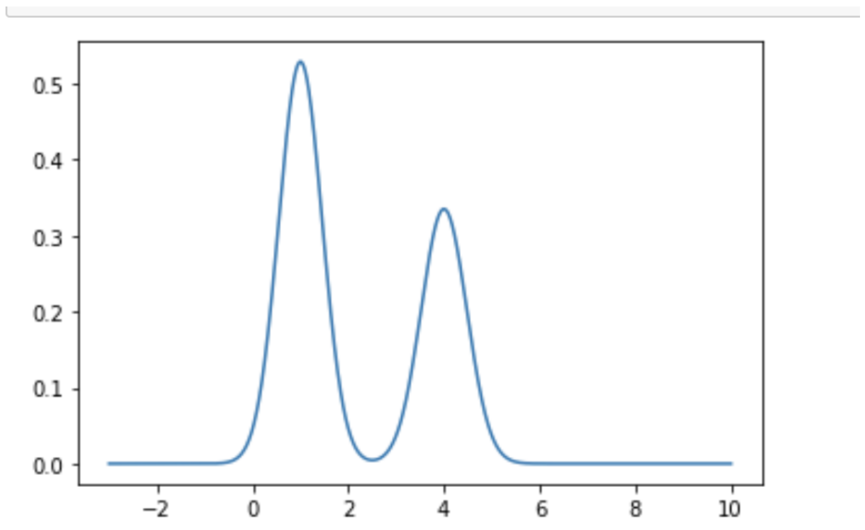


Uniform Kernel

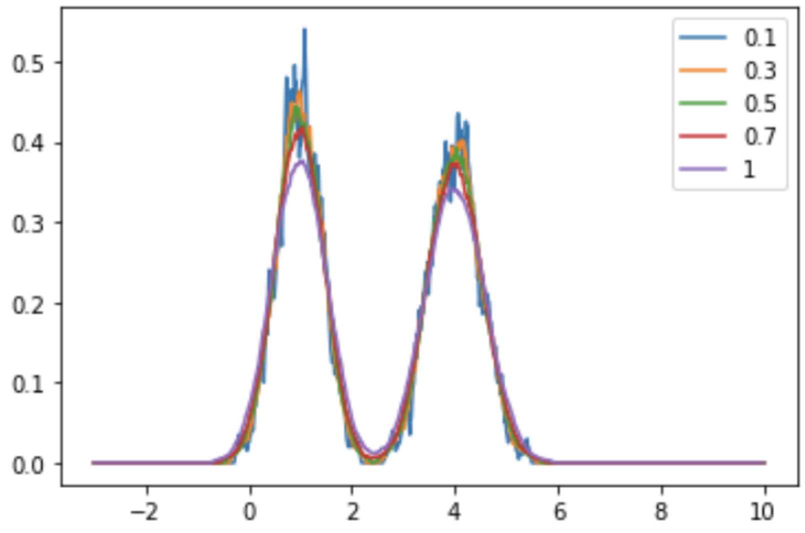
We use the uniform function as the window function. The function is defined as follows

$\phi(x-x_0) = 1 \quad \text{if } \{x_0 - h/2 <= x <= x_0 + h/2\}$

$\qquad\qquad 0 \quad \text{otherwise}$



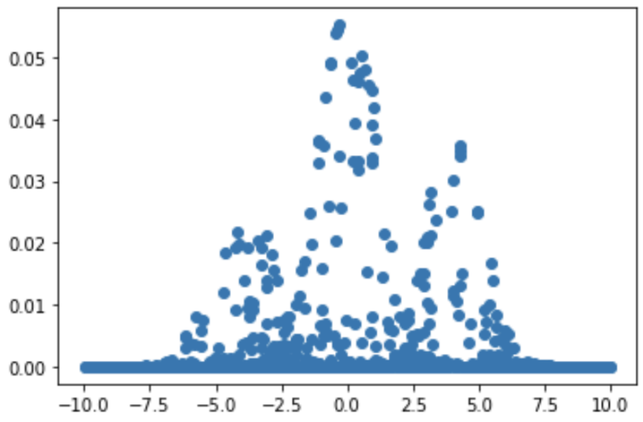
Following the procedure as in the previous kernel function, by varying the window sizes the following functions graph is observed for a mixture

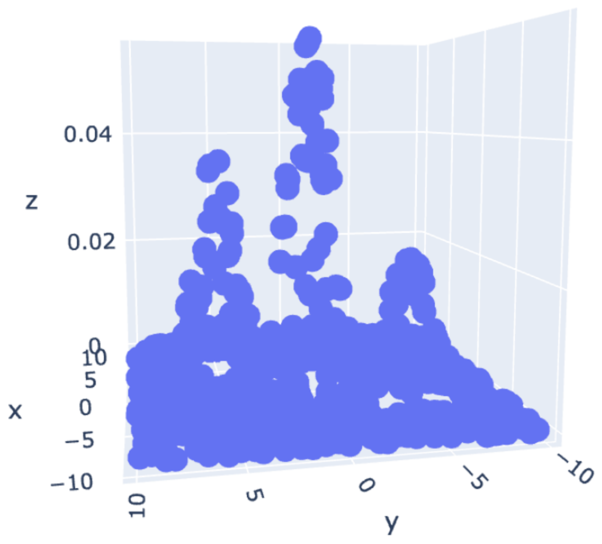


The volume of the density function is h^d which is the same result obtained for the gaussian kernel function.

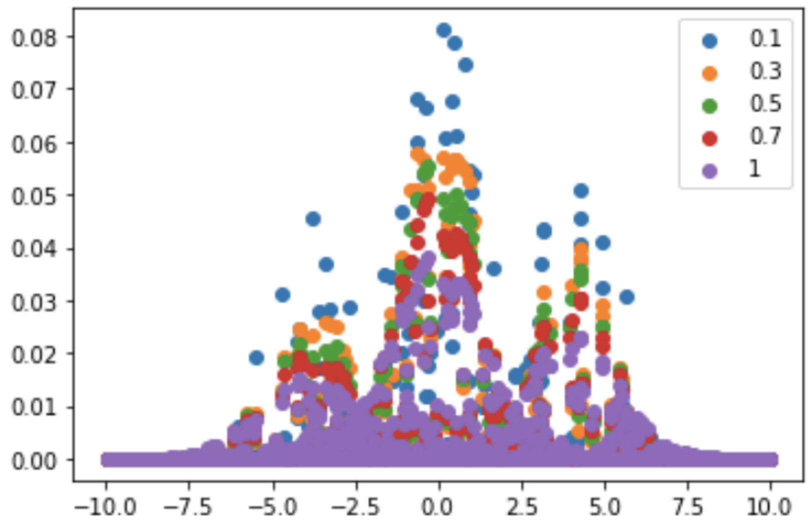
Multivariate Parzen Window

Extending the idea of the univariate case, we consider a two-dimensional GMM model whose parameters we estimate using the Parzen window density estimate technique.
The GMM consists of 3 components with means at [-4, -5] , [0,0] and [4, 5] with mixture probabilities 0.2, 0.5 and 0.3 respectively.

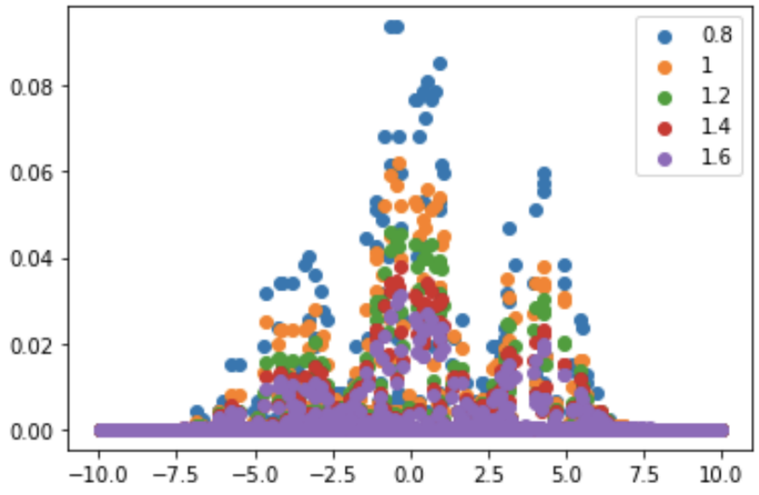




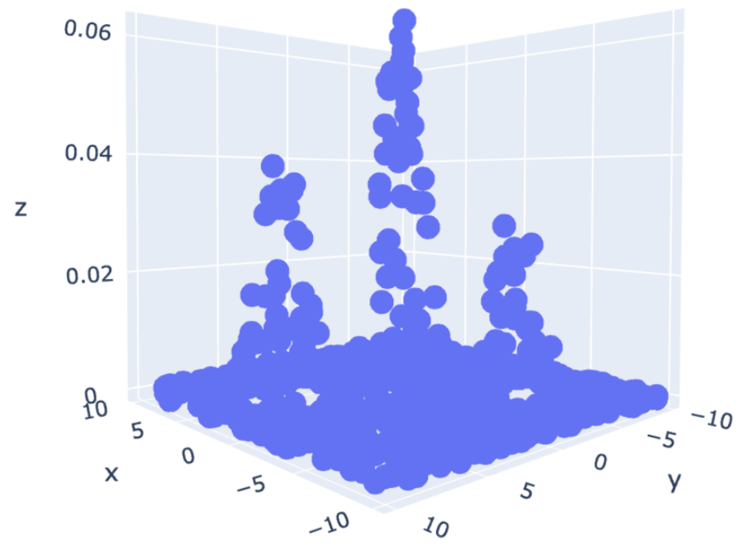
Multivariate Parzen window estimates with varying window sizes with a gaussian kernel



Multivariate Parzen window estimates with varying window sizes with a uniform kernel function



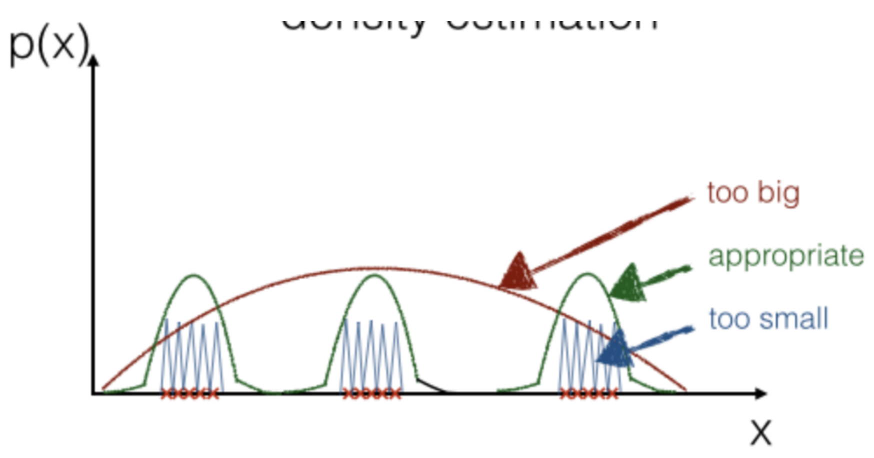
Parzen window estimates for a uniform kernel function with a h = 1



Inferences :

The parzen window estimates are strongly dependent on the window size, especially in case of gaussian kernels. Larger window sizes leads to discontinuity and averaging of estimates in the window and smaller window sizes have very high variance. It is a general trend to decrease the enclosing volume with increasing number of points sampled from the distribution.

The uniform kernel function requires a higher h value to obtain a more accurate estimate of the density as compared to a gaussian kernel.



K-Means clustering

K-Means is a very popularly known unsupervised iterative clustering algorithm that groups similar data points together.

The objective function in k-means is to cluster all data points such that we achieve the minimum sum of euclidean distances among each point and their corresponding cluster means.

$$\arg\min_{\mathcal{Z},A} \sum_{i=1}^{N} ||x_i - z_{A(x_i)}||^2$$

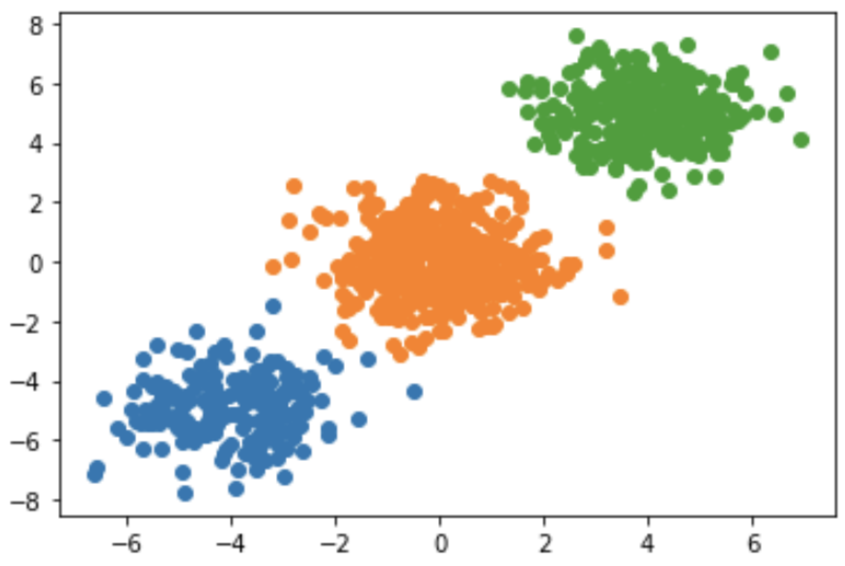
Univariate and Multivariate K-Means Clustering

We assume the number of clusters to be the number of components.

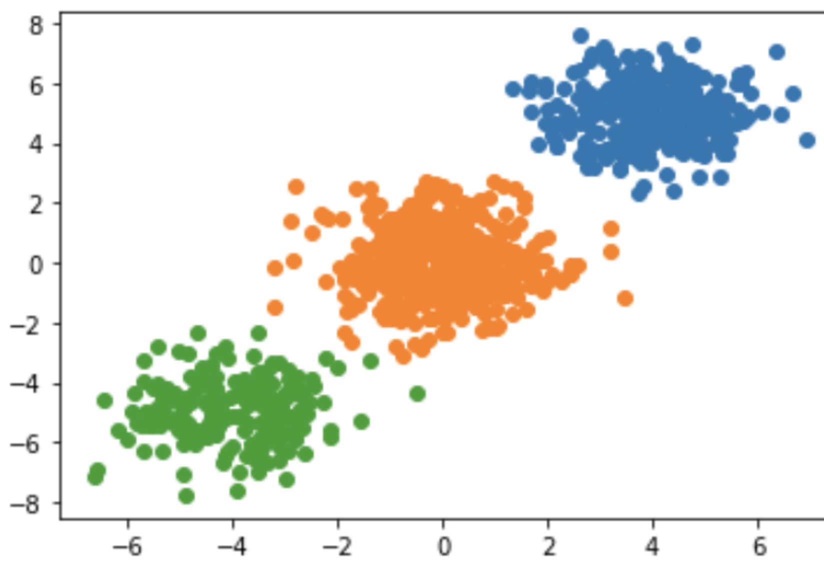
The K-means clustering technique works well when there is no spiralling of data and the data points are far apart from each other.

For the above data distribution, the k-means clustering algorithm appropriately identifies the components when the means of each of these components are further apart.

Actual data distribution



K-means clustering output


8
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6