sandipanweb

Simply Data Science

Hard & Soft Clustering with K-means, Weighted K-means and GMM-EM in Python

MARCH 19, 2017APRIL 7, 2017 / SANDIPAN DEY

The following problems appeared as a project in the *edX course ColumbiaX: CSMM.102x Machine Learning*.

In this assignment the following clustering algorithms will be implemented:

1. Hard clustering with **K-means**
2. Soft clustering with
   a. **Weighted K-means**
   b. **Gaussian mixture models** with **Expectation Maximization**.

Some datasets with n data points $\{x_1,\ldots,x_n\}$ will be used for testing the algorithms, where each $x_i \in R^d$.

Hard Clustering

1. Each point is assigned to a one and only one cluster (hard

assignment).
2. With **K-means** we try to find *K* centroids {μ1,…,μK} and the corresponding assignments of each data point {c1,…,cn} where each ci∈{1,…,K} and c_i indicates which of the *K* clusters the observation x_i belongs to. The objective function that we seek to minimize can be written as

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbf{1}(c_i = k) \|x_i - \mu_k\|^2$$

Soft Clustering

(1) Each point is assigned to all the clusters with different weights or probabilities (soft assignment).

(2) With **Weighed K-means** we try to compute the weights φ_i(k) for each data point *i* to the cluster *k* as minimizing the following objective:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \phi_i(k) \frac{\|x_i - \mu_k\|^2}{\beta}$$
$$\phi_i(k) > 0. \ \sum_{k=1}^{K} \phi_i(k) = 1. \ \beta > 0.$$

(3) With **GMM-EM** we can do soft clustering too. The **EM** algorithm can be used to learn the parameters of a Gaussian mixture model. For this model, we assume a generative process for the data as follows:

$$x_i | c_i \sim Normal(\mu_{c_i}, \Sigma_{c_i}), \quad c_i \sim Discrete(\pi).$$

In other words, the *i_th* observation is first assigned to one of *K* clusters according to the probabilities in vector π, and the value of observation *x_i* is then generated from one of *K* multivariate Gaussian distributions, using the mean and covariance indexed by c_i. The EM algorithm seeks to maximize

$$p(x_1,\ldots,x_n|\pi,\boldsymbol{\mu},\boldsymbol{\Sigma}) = \prod_{i=1}^{n} p(x_i|\pi,\boldsymbol{\mu},\boldsymbol{\Sigma})$$

over all parameters $\pi,\mu 1,\ldots,\mu K,\Sigma 1,\ldots,\Sigma K$ using the cluster assignments $c1,\ldots,cn$ as the hidden data.

The following figures show the algorithms that are going to be implemented for clustering.

HARD CLUSTERING MODELS

Taken from the slides of the edX course ColumbiaX: CSMM.102x Machine Learning

Review: K-means clustering algorithm

Given: Data $x_1,\ldots,x_n$, where $x \in \mathbb{R}^d$

Goal: Minimize $\mathcal{L} = \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}\{c_i = k\}\|x_i - \mu_k\|^2$.

- Iterate until values no longer changing
  1. Update $\boldsymbol{c}$: For each i, set $c_i = \arg\min_k \|x_i - \mu_k\|^2$
  2. Update $\boldsymbol{\mu}$: For each k, set $\mu_k = \left(\sum_i x_i \mathbb{1}\{c_i = k\}\right) / \left(\sum_i \mathbb{1}\{c_i = k\}\right)$

WEIGHTED K-MEANS (SOFT CLUSTERING EXAMPLE)

Weighted K-means clustering algorithm

Given: Data $x_1,\ldots,x_n$, where $x \in \mathbb{R}^d$

Goal: Minimize $\mathcal{L} = \sum_{i=1}^{n}\sum_{k=1}^{K} \phi_i(k)\frac{\|x_i-\mu_k\|^2}{\beta}$ over ϕ_i and μ_k

Conditions: $\phi_i(k) > 0$ and $\sum_{k=1}^{K}\phi_i(k) = 1$. Set parameter $\beta > 0$.

- Iterate the following
  1. Update ϕ: For each i, update the word allocation weights

$$\phi_i(k) = \frac{\exp\{-\frac{1}{\beta}\|x_i - \mu_k\|^2\}}{\sum_j \exp\{-\frac{1}{\beta}\|x_i - \mu_j\|^2\}}, \quad \text{for } k = 1,\ldots,K$$

  2. Update $\boldsymbol{\mu}$: For each k, update μ_k with the *weighted* average

$$\mu_k = \frac{\sum_i x_i \phi_i(k)}{\sum_i \phi_i(k)}$$

Taken from the slides of the edX course ColumbiaX: CSMM.102x Machine Learning

EM FOR THE GMM

Taken from the slides of the edX course ColumbiaX: CSMM.102x Machine Learning

Algorithm: Maximum likelihood EM for the GMM

Given: $x_1, \ldots, x_n$ where $x \in \mathbb{R}^d$

Goal: Maximize $\mathcal{L} = \sum_{i=1}^{n} \ln p(x_i|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

- Iterate until incremental improvement to $\mathcal{L}$ is "small"

1. **E-step**: For $i = 1, \ldots, n$, set

$$\phi_i(k) = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x_i|\mu_j, \Sigma_j)}, \quad \text{for } k = 1, \ldots, K$$

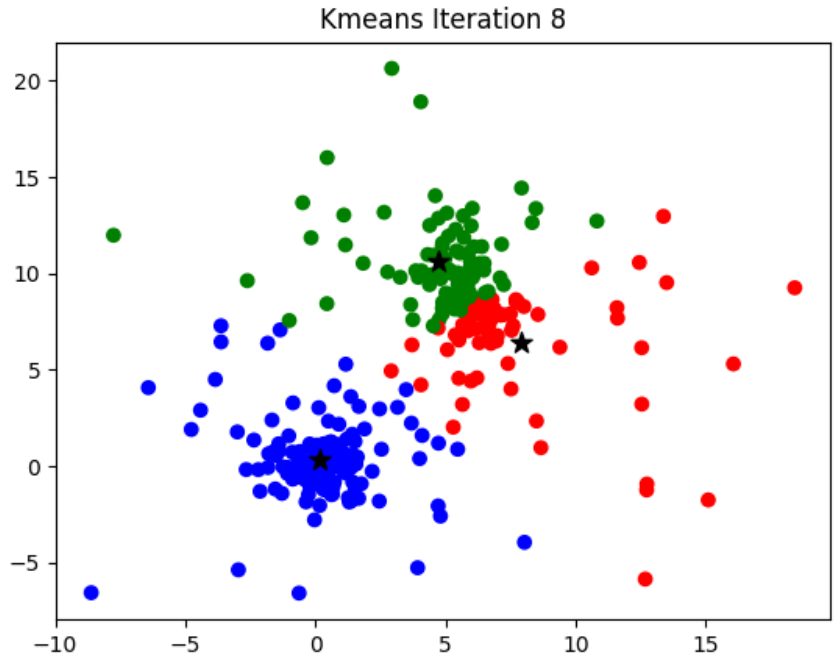
2. **M-step**: For $k = 1, \ldots, K$, define $n_k = \sum_{i=1}^{n} \phi_i(k)$ and update the values

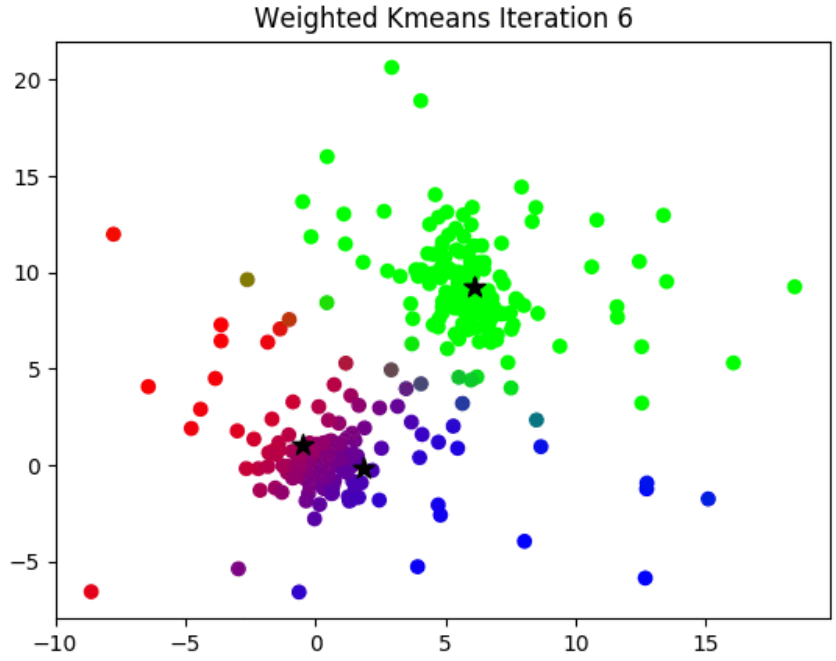
$$\pi_k = \frac{n_k}{n}, \quad \mu_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k) x_i \quad \Sigma_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k)(x_i - \mu_k)(x_i - \mu_k)^T$$

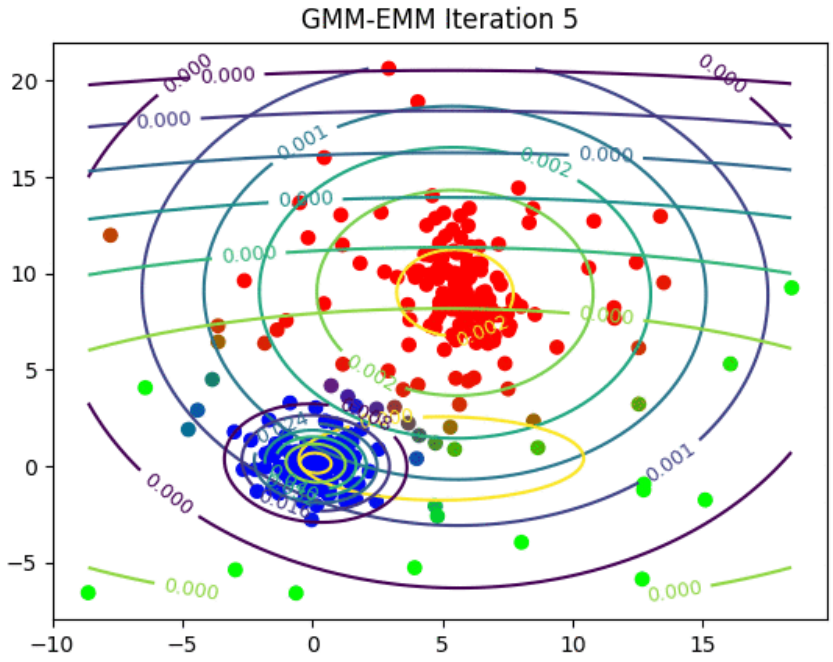
Comment: The updated value for μ_k is used when updating Σ_k.

The following animations show the output of the clustering algorithms (and how they converge with different iterations) on a few datasets (with *k=3* clusters), the weighted k-means is run with the stiffness-parameter *beta=10*.

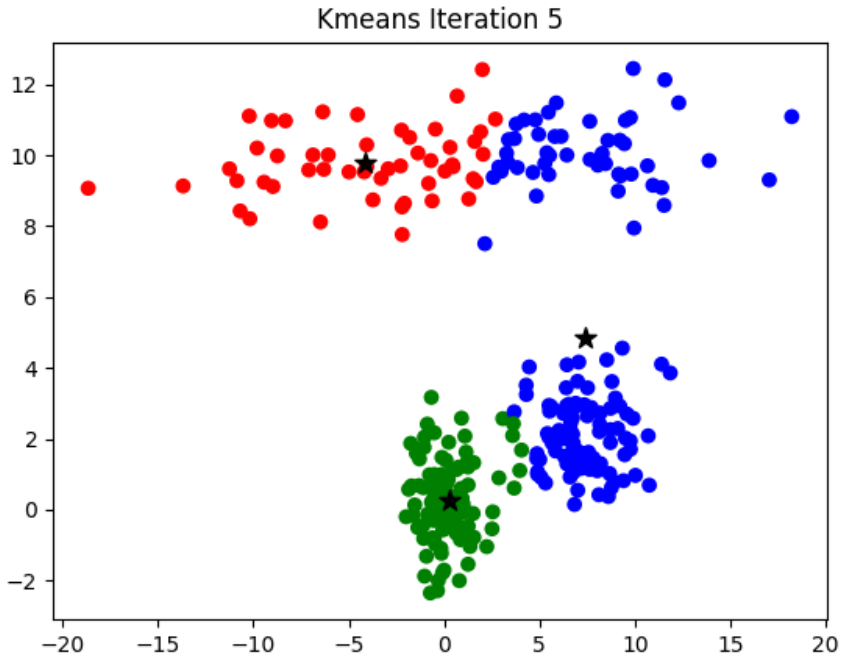
Dataset1

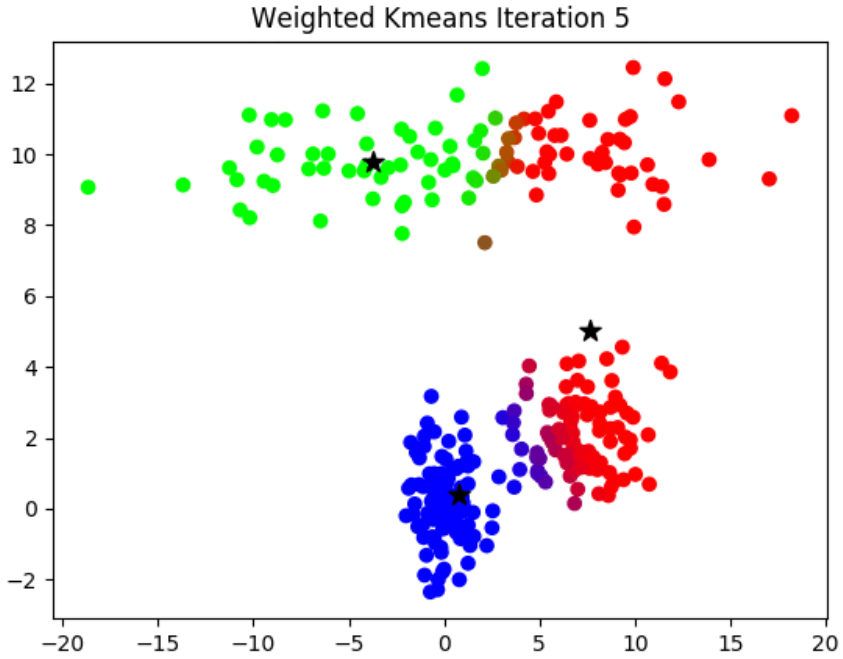

Kmeans Iteration 8
20
15
10
5
0
−5
−10
−5
0
5
10
15

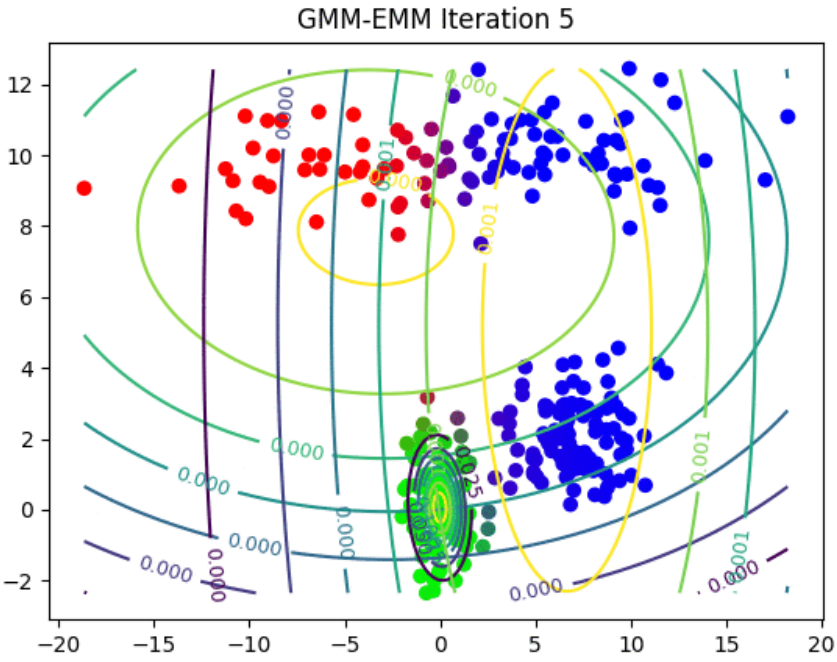

Weighted Kmeans Iteration 6
20
15
10
5
0
−5
−10
−5
0
5
10
15



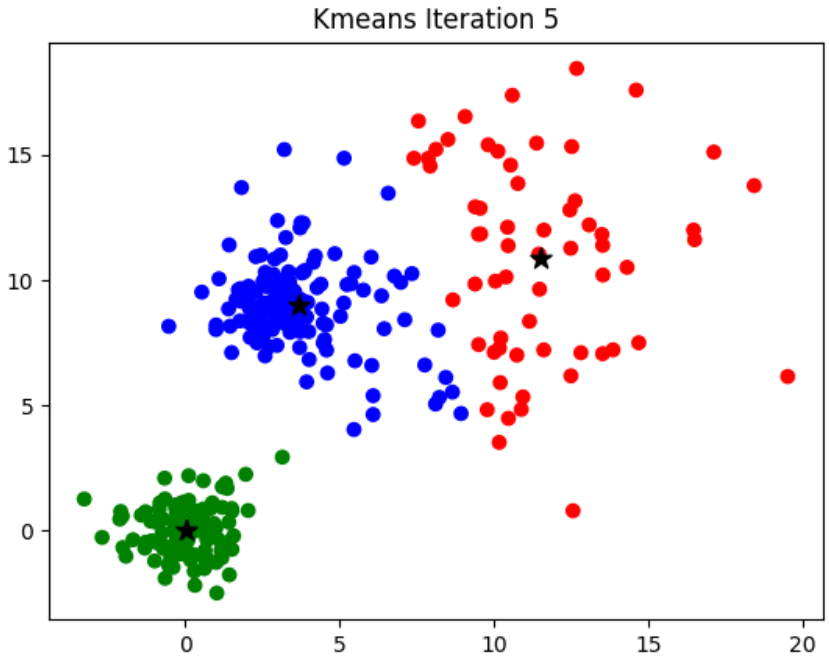
Dataset2

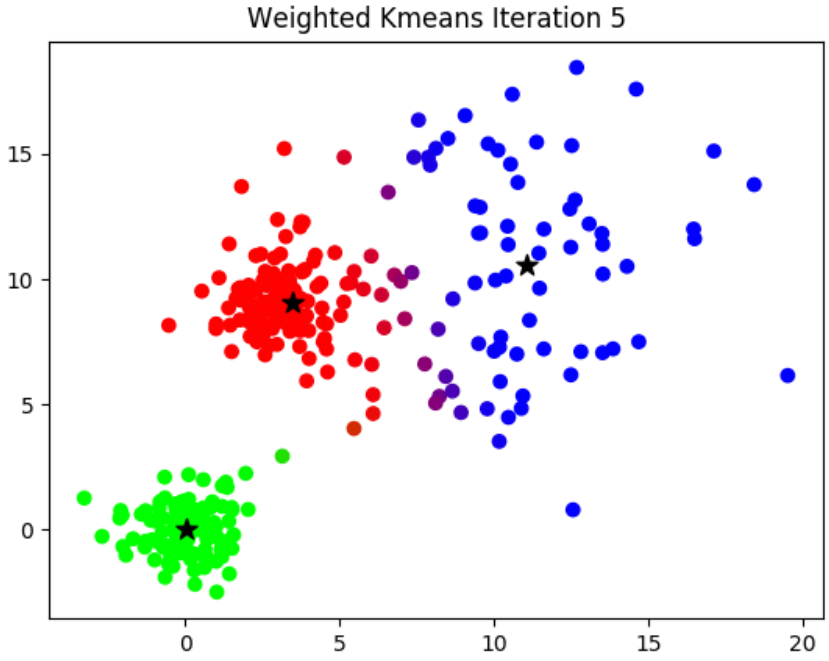

Kmeans Iteration 5
12
10
8
6
4
2
0
−2
−20
−15
−10
−5
0
5
10
15
20

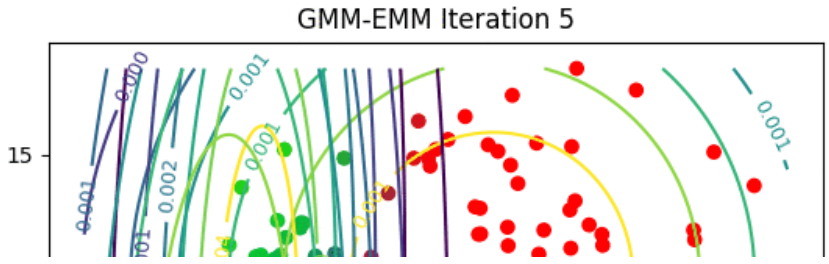

Weighted Kmeans Iteration 5
12
10
8
6
4
2
0
−2
−20
−15
−10
−5
0
5
10
15
20

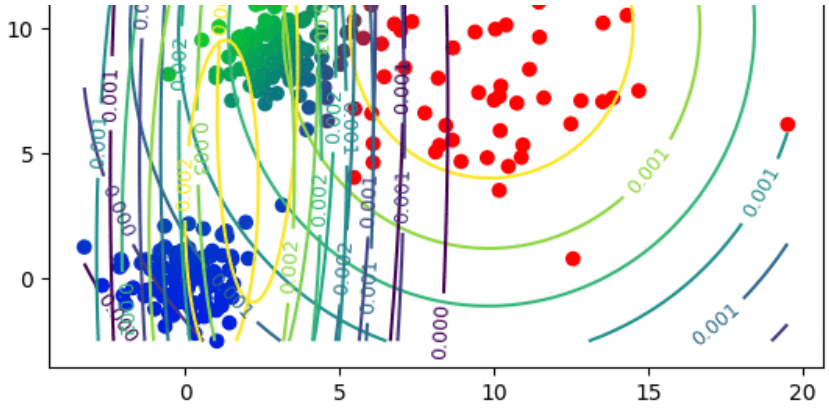


Dataset3

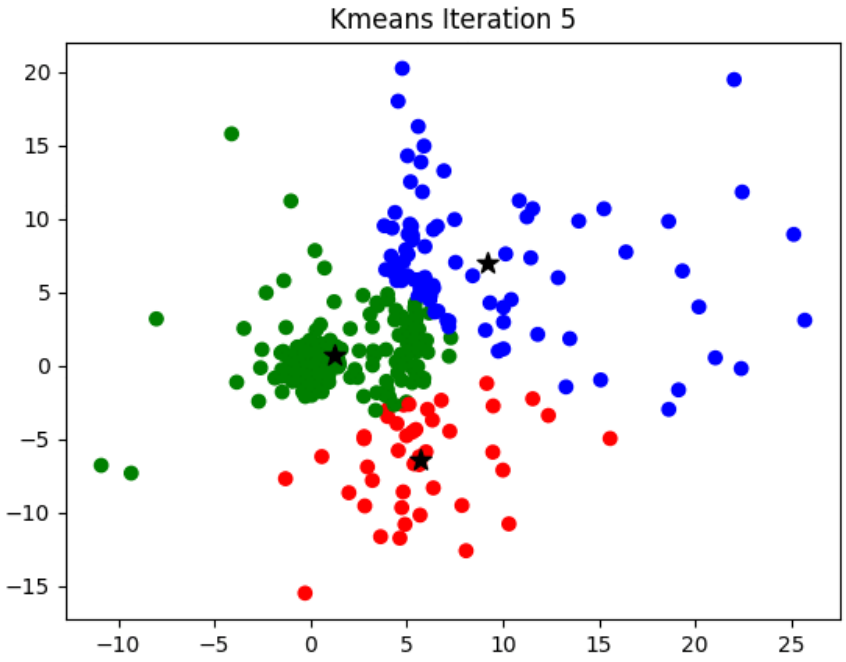

Kmeans Iteration 5
15
10
5
0
0
5
10
15
20

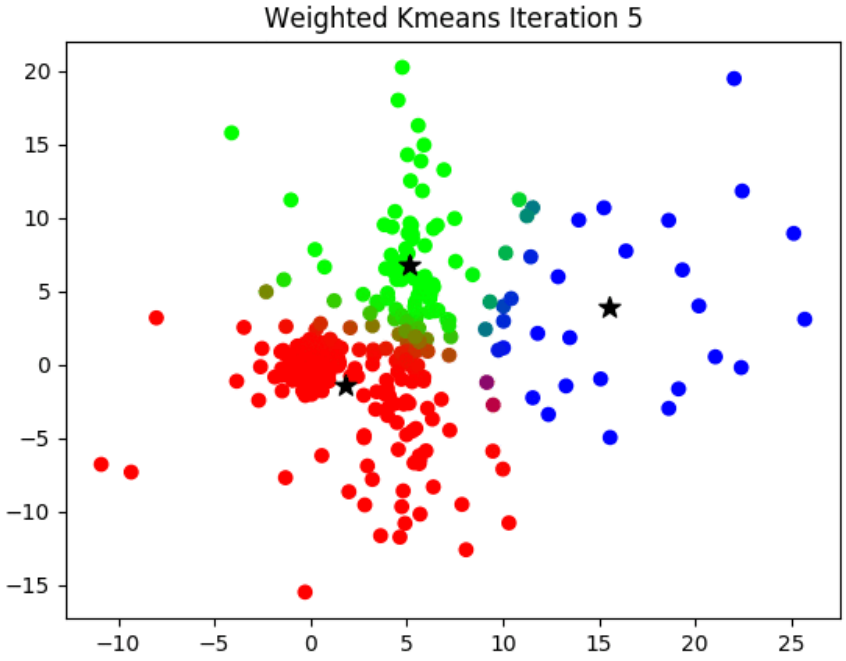

Weighted Kmeans Iteration 5
15
10
5
0
0
5
10
15
20

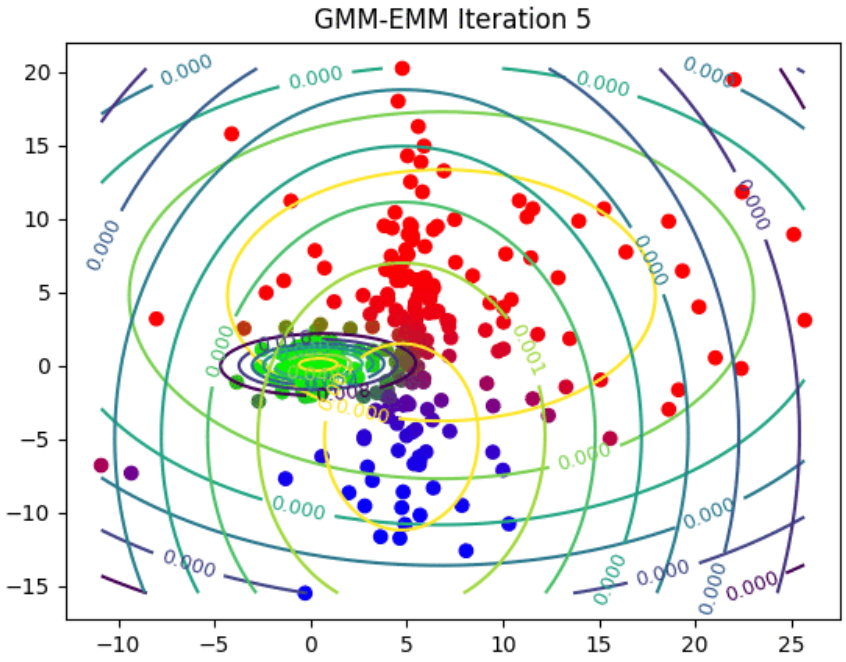

GMM-EMM Iteration 5
15
0.001
0.001
0.001
0.002
0.001



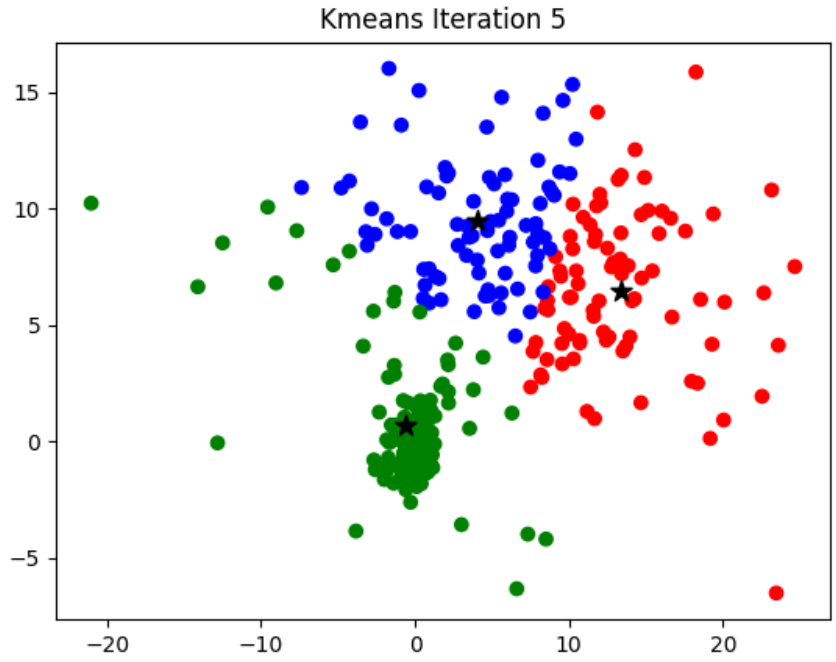
Dataset4

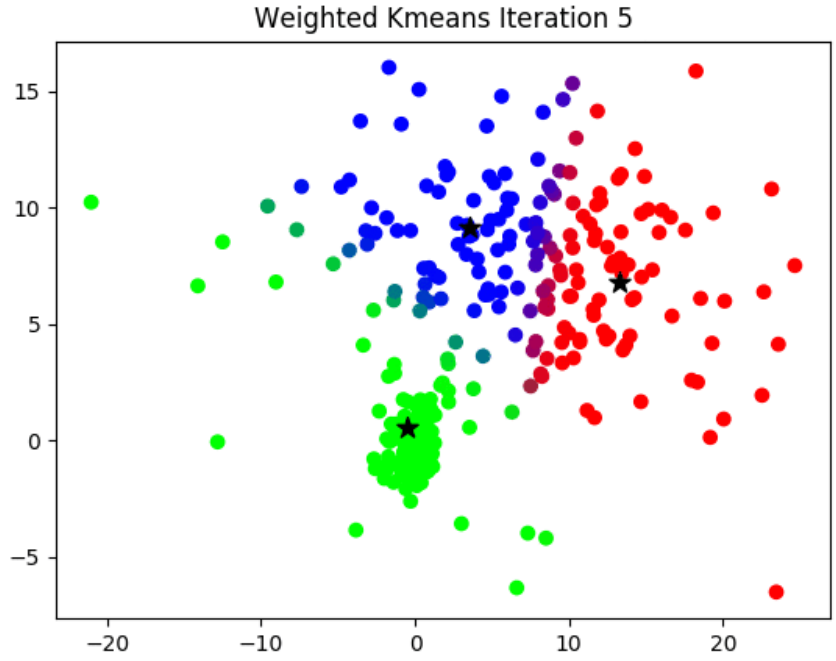

Kmeans Iteration 5
20
15
10
5
0
−5
−10
−15
−10
−5
0
5
10
15
20
25

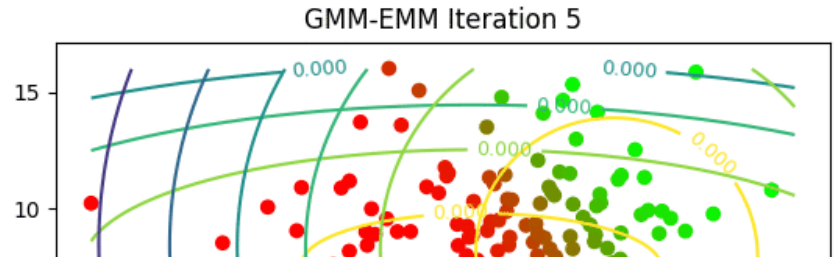

Weighted Kmeans Iteration 5
20
15
10
5
0
−5
−10
−15
−10
−5
0
5
10
15
20
25

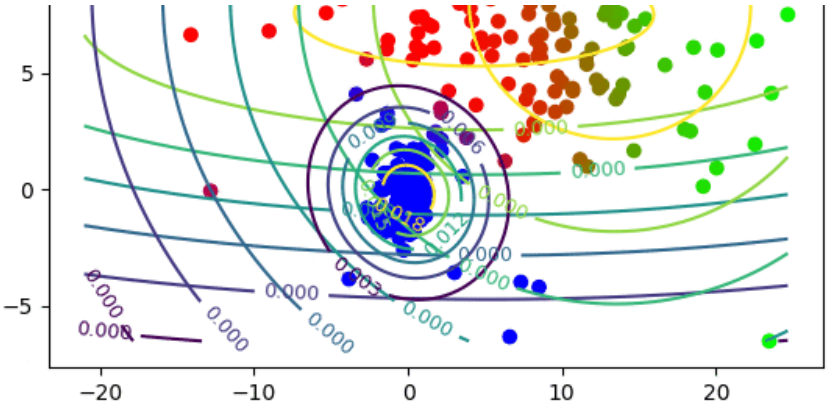


Dataset5

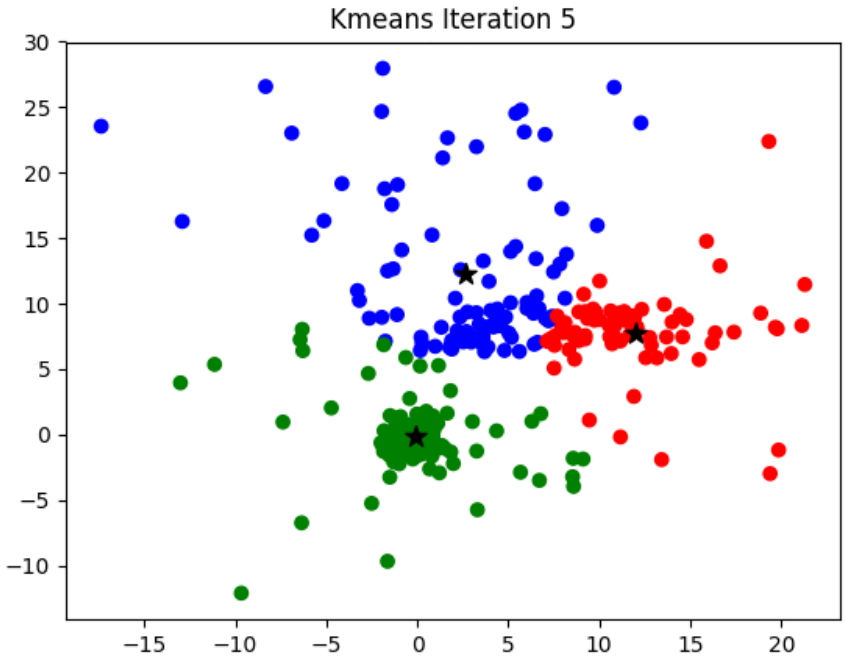

Kmeans Iteration 5
15
10
5
0
−5
−20
−10
0
10
20

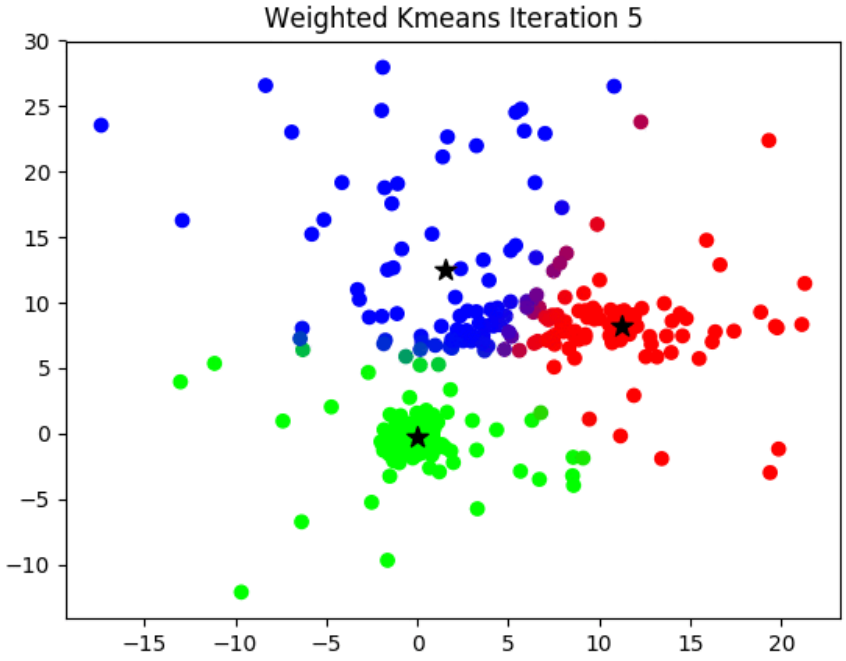

Weighted Kmeans Iteration 5
15
10
5
0
−5
−20
−10
0
10
20

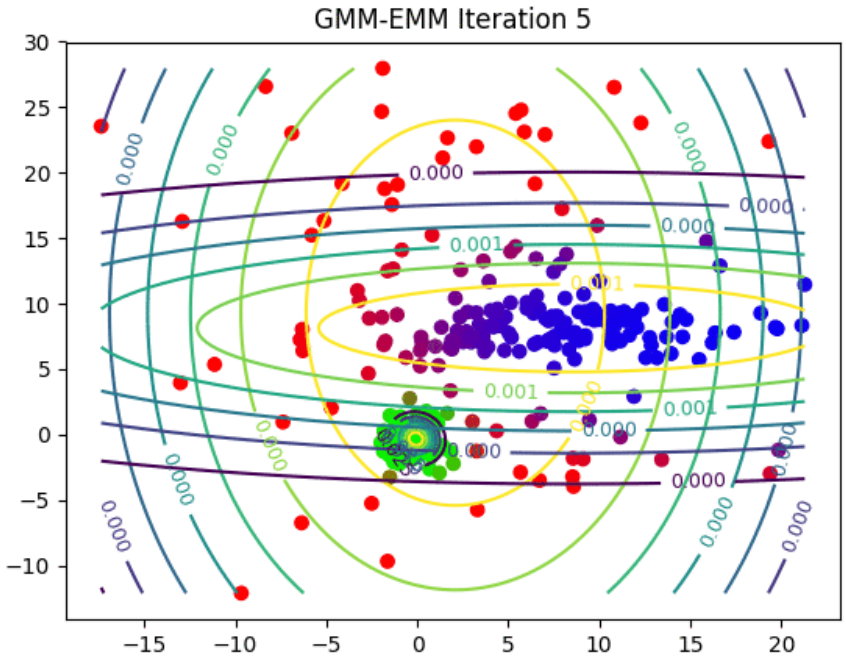

GMM-EMM Iteration 5
15
10
0.000
0.000
0.000
0.000
0.000



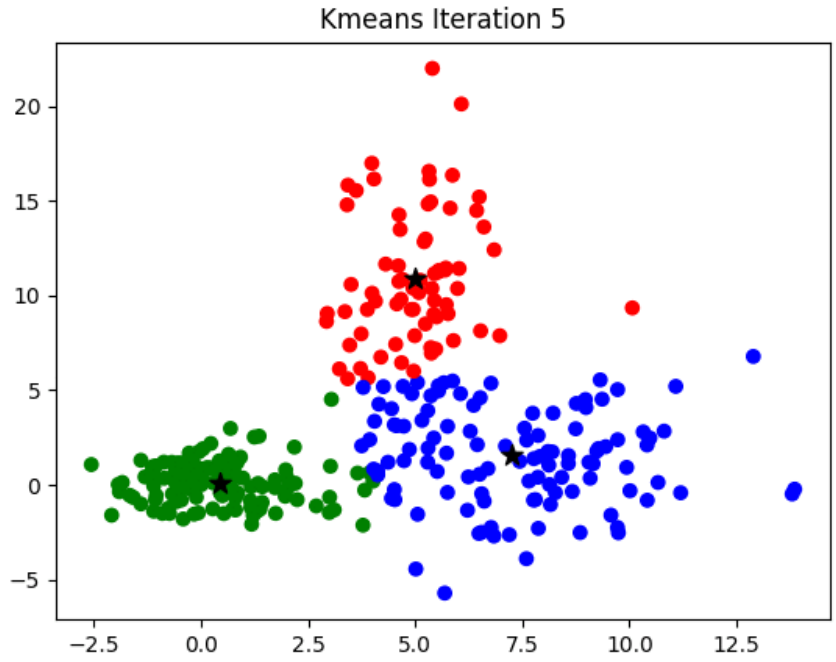
Dataset6

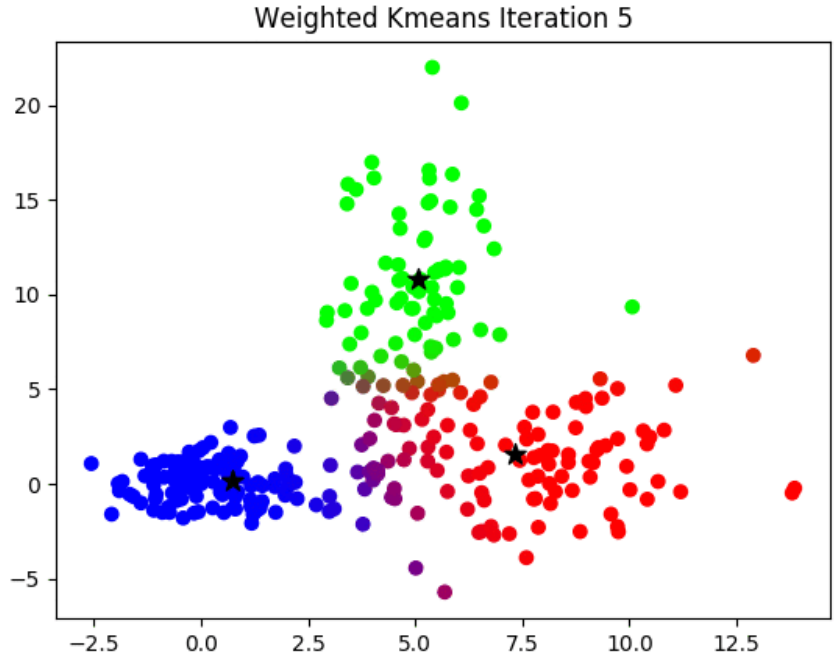

Kmeans Iteration 5
30
25
20
15
10
5
0
−5
−10
−15
−10
−5
0
5
10
15
20

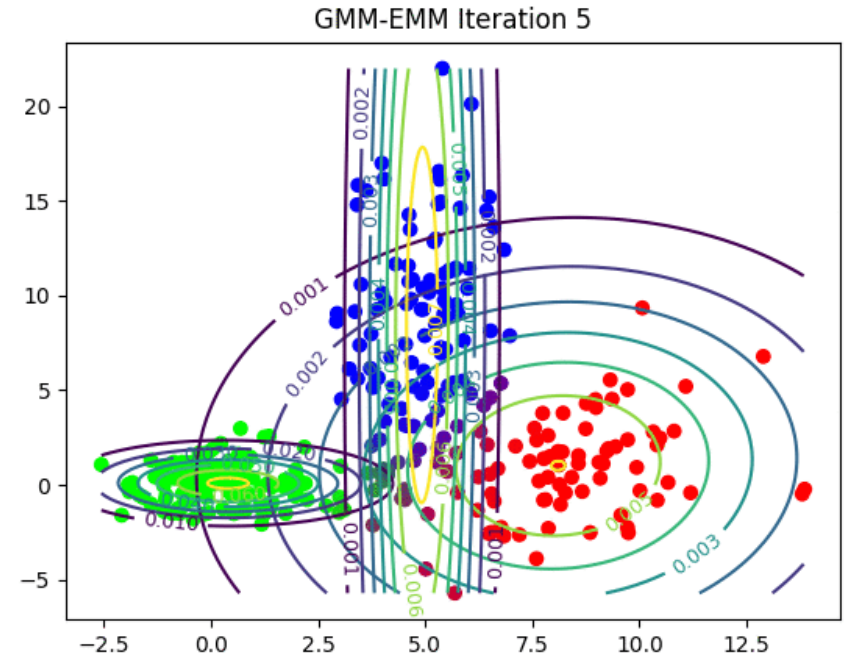

Weighted Kmeans Iteration 5
30
25
20
15
10
5
0
−5
−10
−15
−10
−5
0
5
10
15
20


GMM-EMM Iteration 5
30
25
20
15
10
5
0
−5
−10
−15
−10
−5
0
5
10
15
20
0.000
0.001

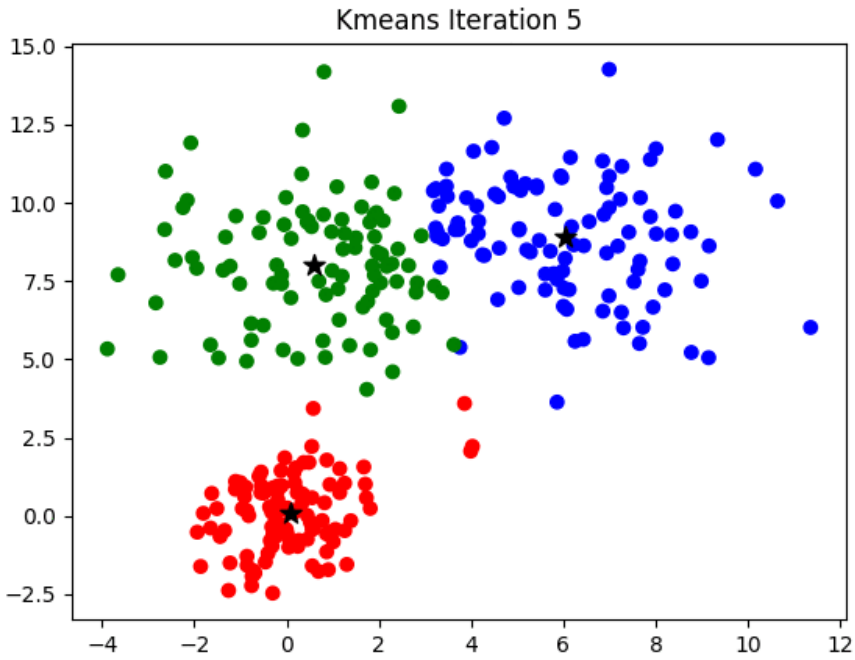
Dataset7


Kmeans Iteration 5
20
15
10
5
0
−5
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

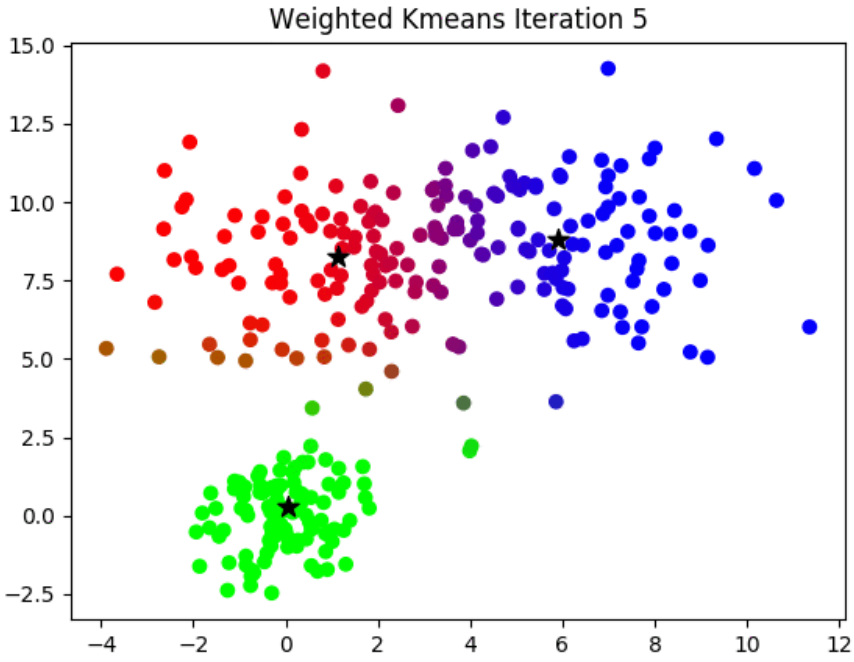

Weighted Kmeans Iteration 5
20
15
10
5
0
−5
−2.5
0.0
2.5
5.0
7.5
10.0
12.5



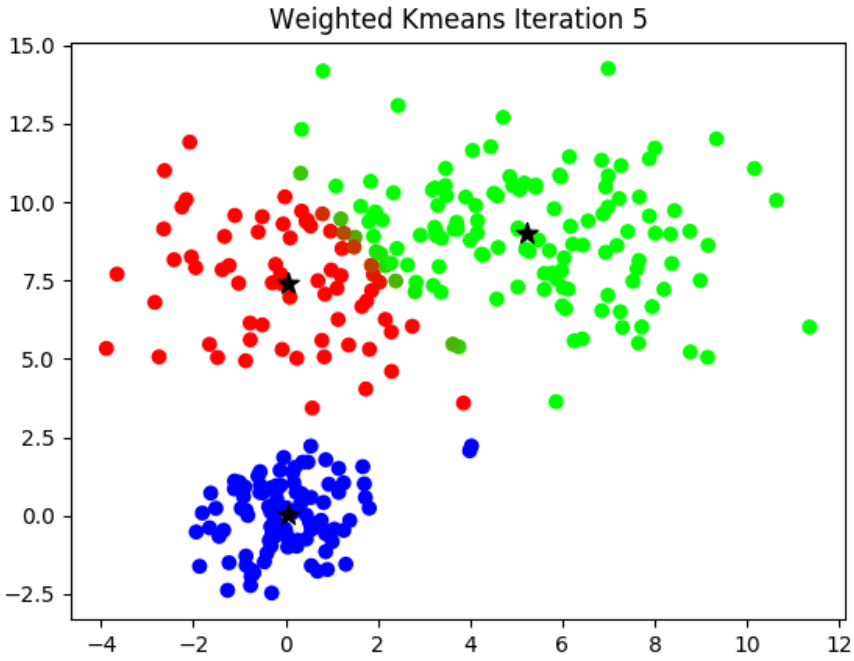
Dataset8

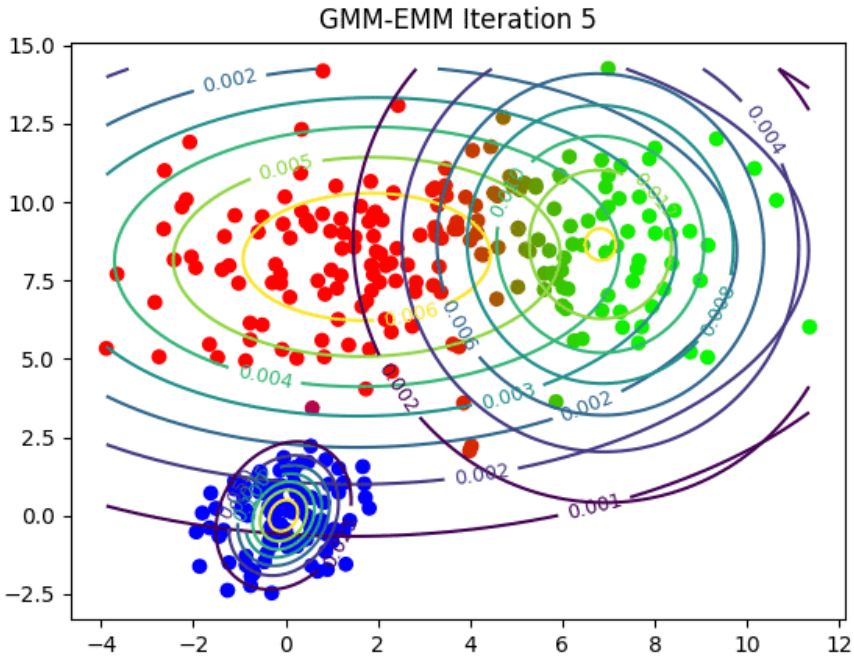


The next animation shows the results with **weighted K-means** with stiffness parameter **beta=10**

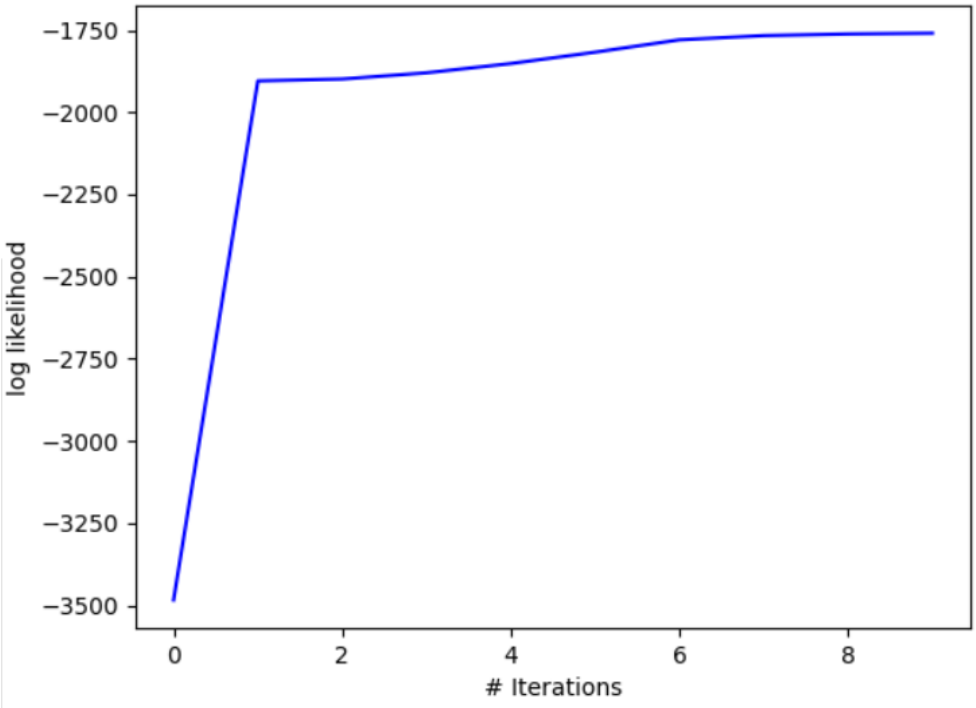


The next animation shows the results with **weighted K-means** with stiffness parameter **beta=1**

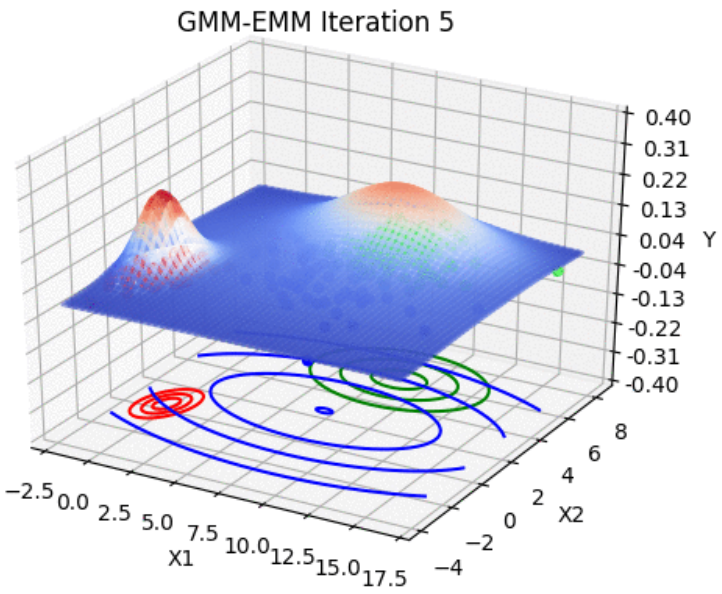


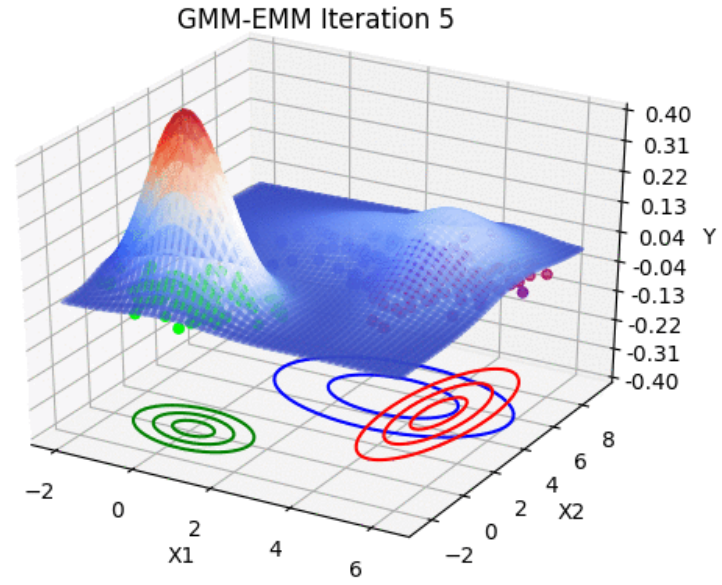


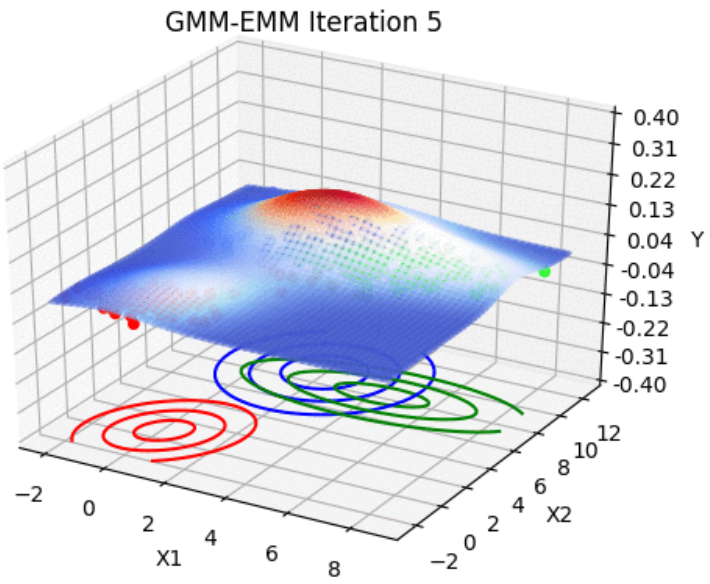
Here is how the **log-likelihood** for **EM** continuously increases over the iterations for a particular dataset:

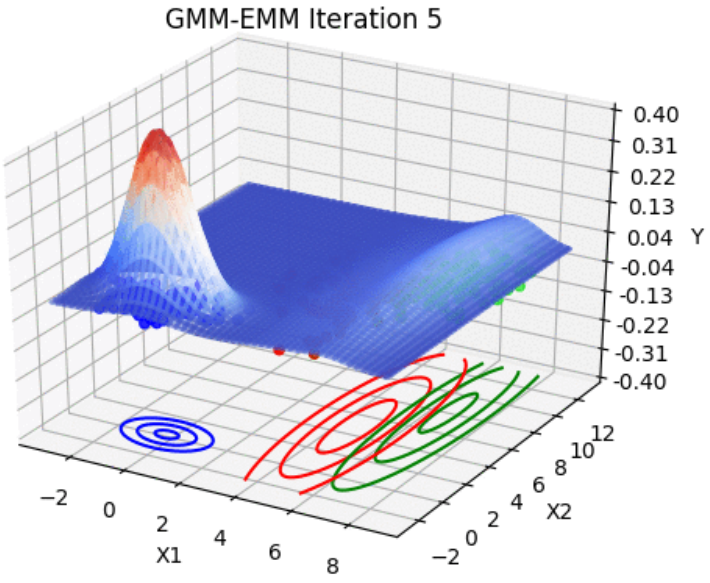


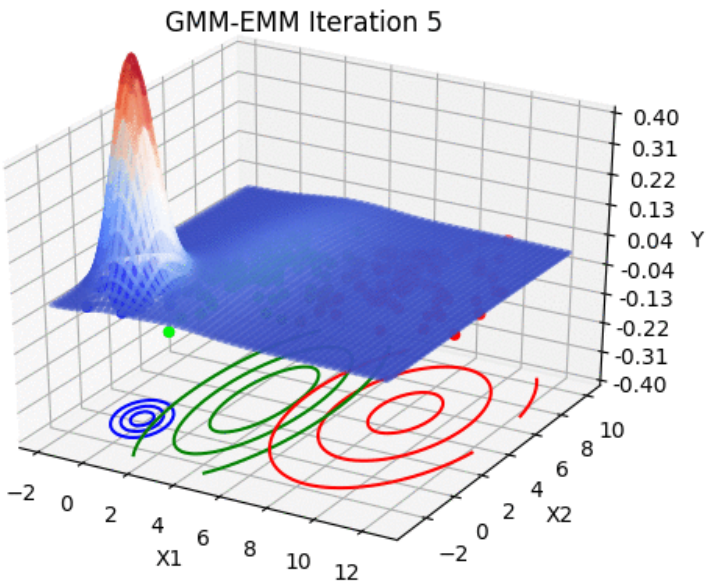
The following animations show **GMM EM convergence** on a few more datasets:

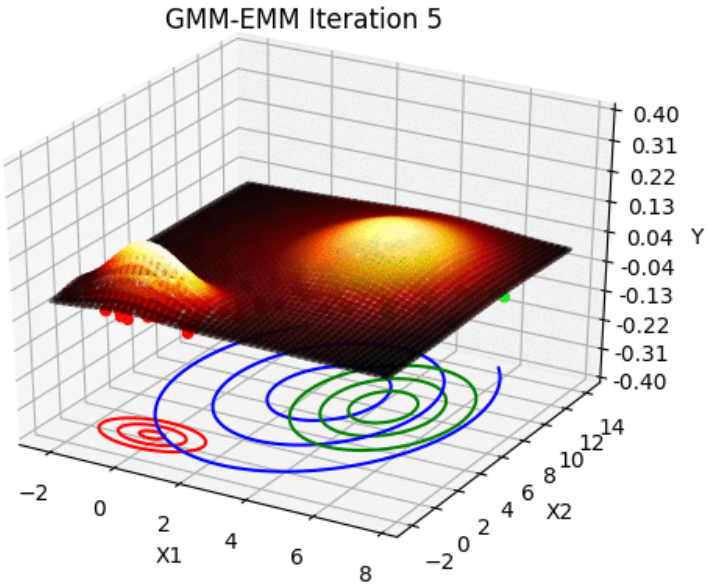

GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
X1
−4
−2
0
2
4
6
8
X2

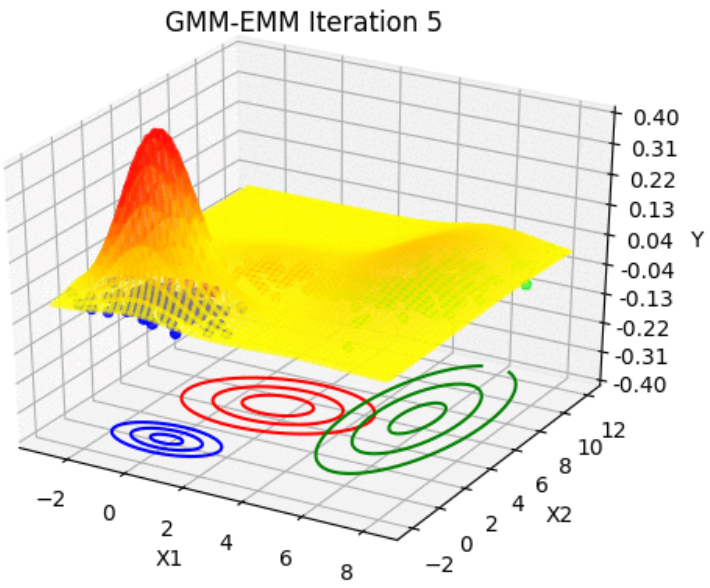

GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
X1
−2
0
2
4
6
8
X2

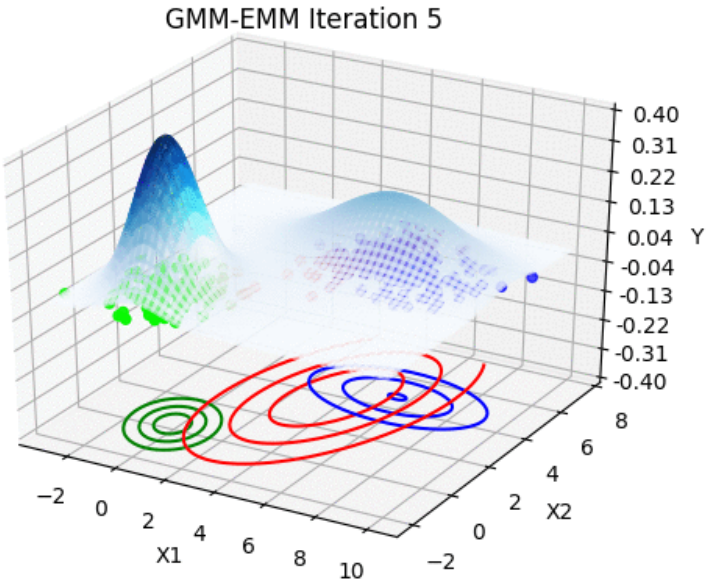

GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
X1
−2
0
2
4
6
8
10
12
X2

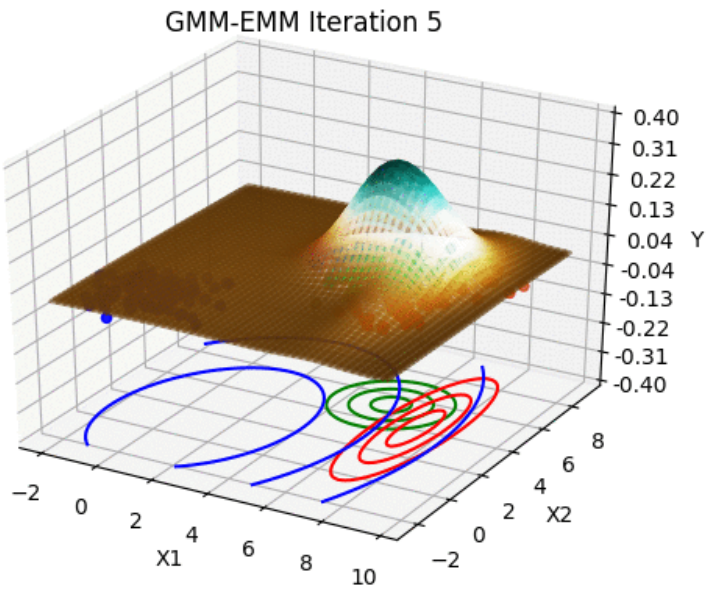

GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
X1
−2
0
2
4
6
8
10
12
X2

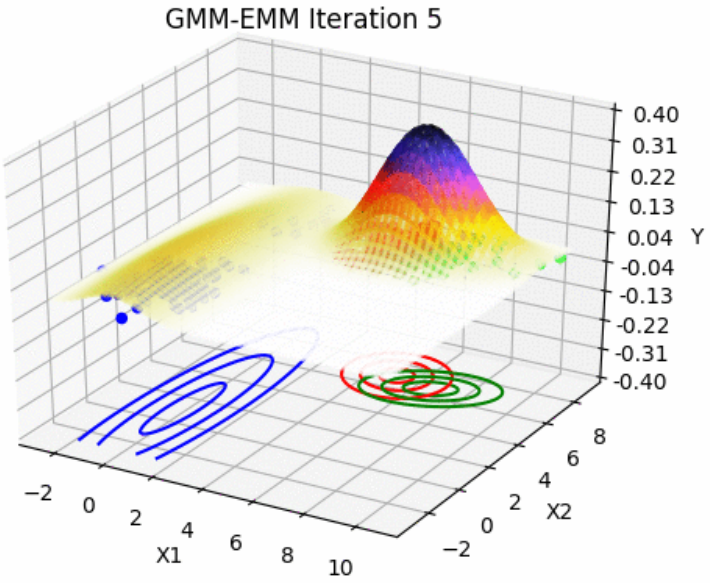

GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
10
12
X1
−2
0
2
4
6
8
10
X2

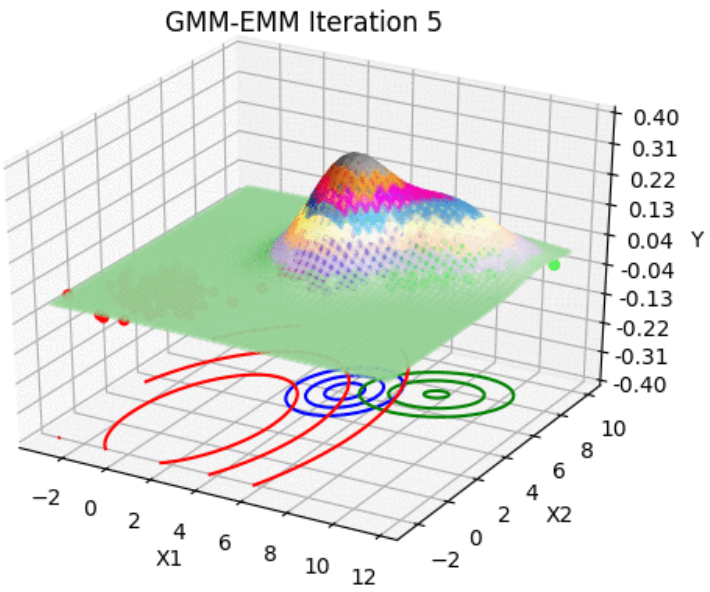

GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
X1
−2
0
2
4
6
8
10
12
14
X2

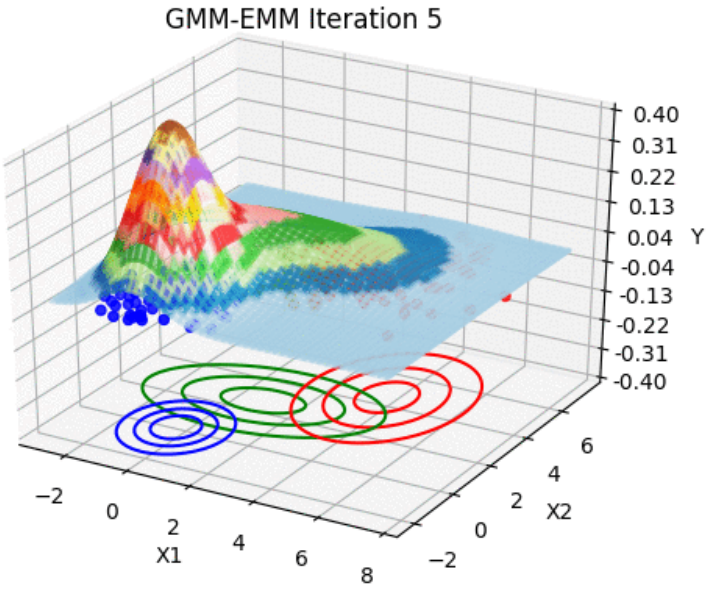

GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
X1
−2
0
2
4
6
8
10
12
X2

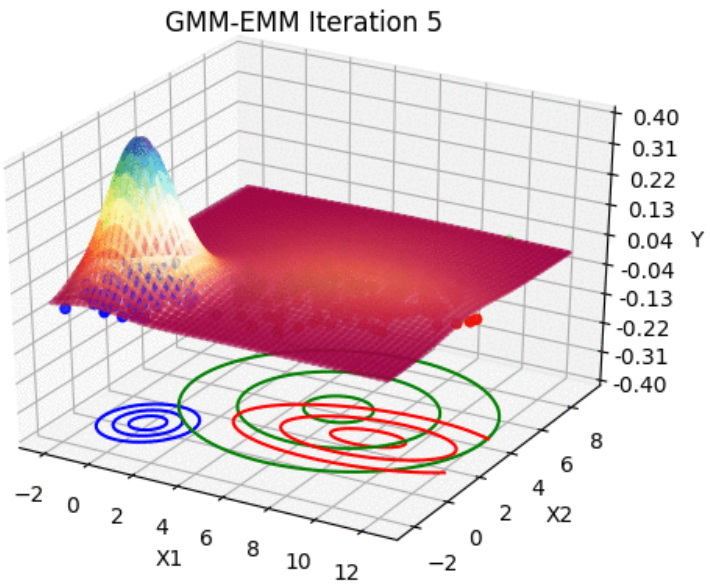

GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
10
X1
−2
0
2
4
6
8
X2


GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
10
X1
−2
0
2
4
6
8
X2


GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
10
X1
−2
0
2
4
6
8
X2


GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
10
12
X1
−2
0
2
4
6
8
10
X2


GMM-EMM Iteration 5
0.40
0.31
0.22
0.13
0.04
-0.04
-0.13
-0.22
-0.31
-0.40
Y
−2
0
2
4
6
8
X1
−2
0
2
4
6
X2



Advertisements
ML, Python, Uncategorized

POWERED BY WORDPRESS.COM.