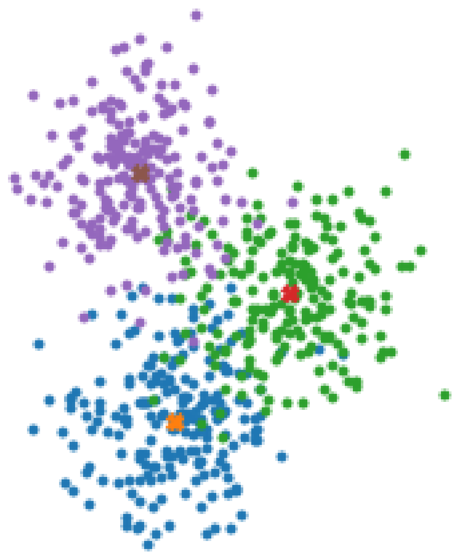
Course name: Machine Learning Sessional

Course No. CSE 472

Assignment 2: Expectation-Maximization Algorithm for Gaussian Mixture Model



Submitted by:

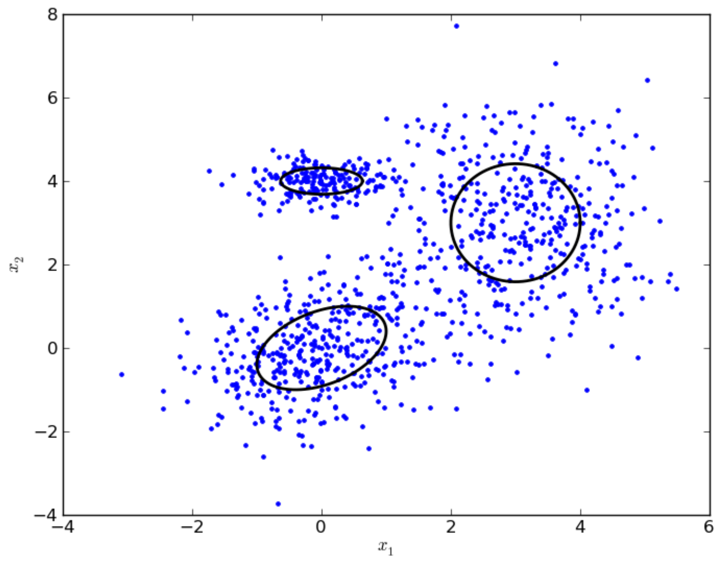
Tanmoy Sarkar Pias

ID: 1305055

Section A

Submission date: 18 May 2018

1. **Why should you use a Gaussian mixture model (GMM) in the above scenario?**



If we observer this plot of data, we can obviously see that the data set are sparse. So k mean clustering algorithm won't work efficiently as it tries to fit into a hard boundary. But for GMM it uses soft boundary. Actually in K means the data points are always assigned 0 or 1 value which means a particular data point can be of one cluster at a time. But in GMM the data points are assigned a probabilistic value. So a data point can have some value for every cluster. And so in this scenario using GMM is better than other algorithms.

2. How will you model your data for GMM?

Model:

- Make each feature of the data an axis of the sample space
- The data will cluster in the sample space
- Every cluster has some weight which represents how many data point is assigned to that cluster
- Now we have to try to fit the data into some ellipse efficiently

  For example if the data has 2 attributes the sample space will be two dimensional where each axis will represent the value of each attribute. And the samples will be scattered into 2D space.

Data generation:

```
For j=1 ………….. N
    i = cluster(w)
    Xj = N(i, mean[i], covariance[i])
    Data.append(xj)
```

3. What are the intuitive meaning of the update equations in M step?

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{N} p_{ij}\mathbf{x}_j}{\sum_{j=1}^{N} p_{ij}}$$

$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^{N} p_{ij}(\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{N} p_{ij}}$$

$$w_i = \frac{\sum_{j=1}^{N} p_{ij}}{N}$$

In the M step we are taking a sum of weighted probability. So the probability of a data point being in a cluster becomes more or less gradually. So after a few iteration the mean and covariance matrix will converge and become saturated. The main concept is to take sum with a probabilistic weight.

4. Derive the log-likelihood function in step 4.

$$\mathrm{l} = p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w})$$

$$\mathrm{l} = p(\mathbf{x}_1|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w})p(\mathbf{x}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w}) \ldots \ldots p(\mathbf{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w})$$

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w}) = \sum_{j=1}^{N} \ln p\left(\mathbf{x}_j \middle| \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w}\right) = \sum_{j=1}^{N} \ln\left(\sum_{i=1}^{k} w_i N_i(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\right)$$

Actually log is taken for scaling the value to a smaller range. As log reserves the relative value of each elements, so it is easier to work with smaller values rather than bigger ones.

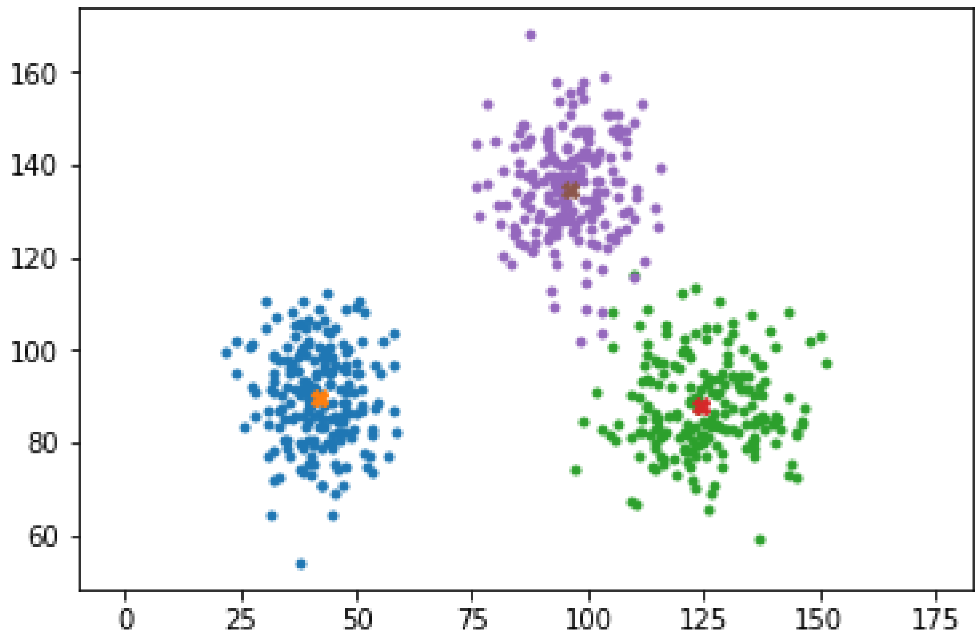
5. Implement the above pseudocode and estimate the location of enemy ships.

As the ships are scattered we can use the GMM to cluster them.

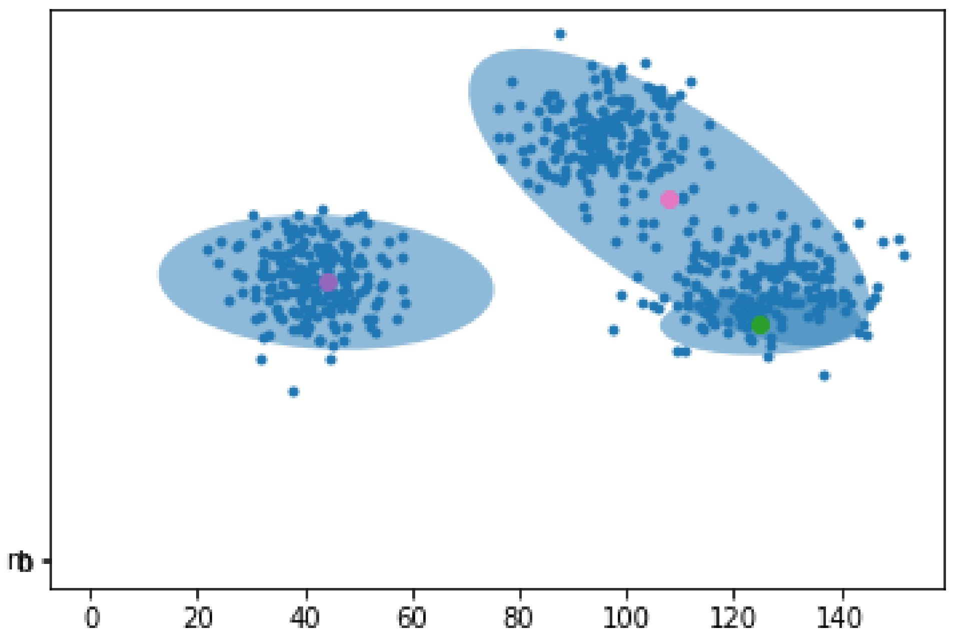
Implemented!

Case study: 1

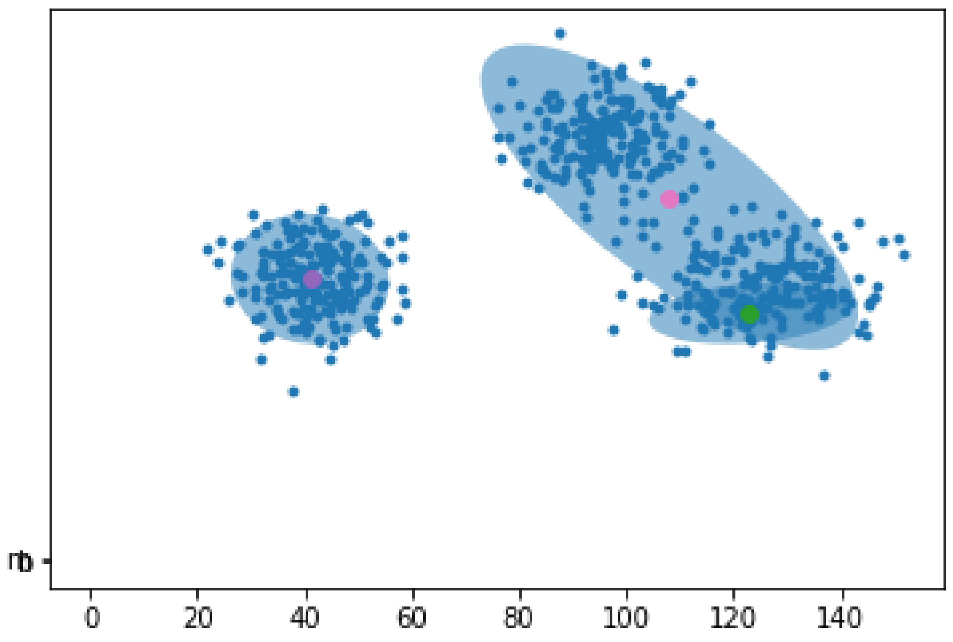
```
u_source:
[[42, 90], [124, 88], [96, 135]]

E_source:
 [[[ 62.   0.]
  [  0. 108.]]

 [[108.   0.]
  [  0. 102.]]

 [[ 61.   0.]
  [  0. 112.]]]
```



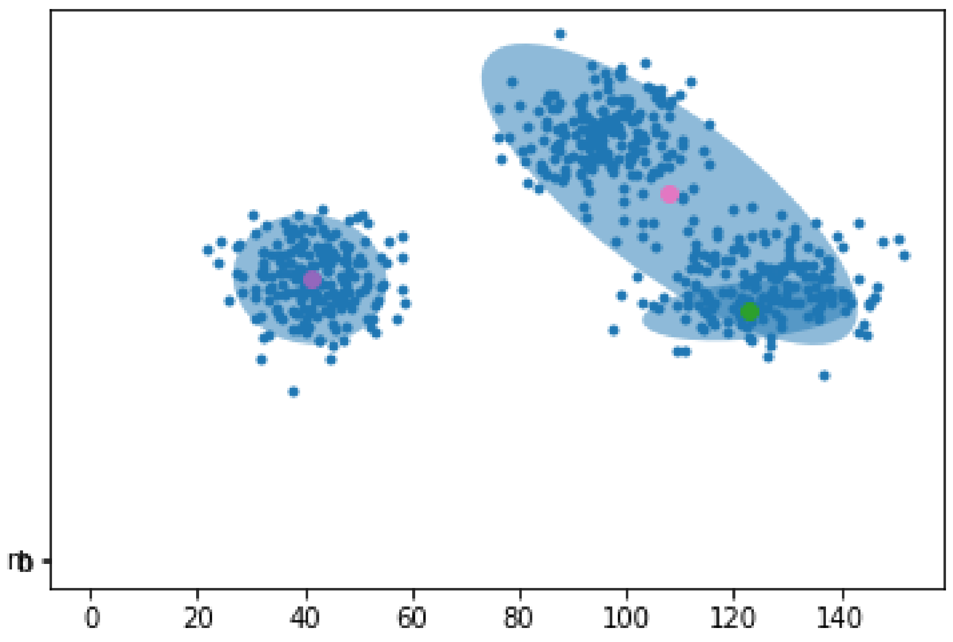
Iteration 0



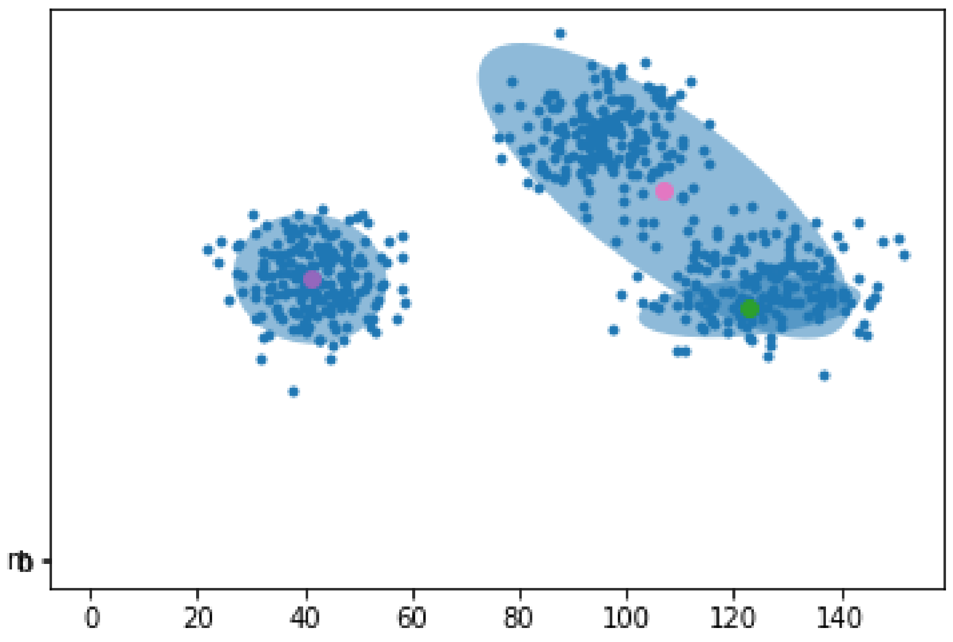
Iteration 1



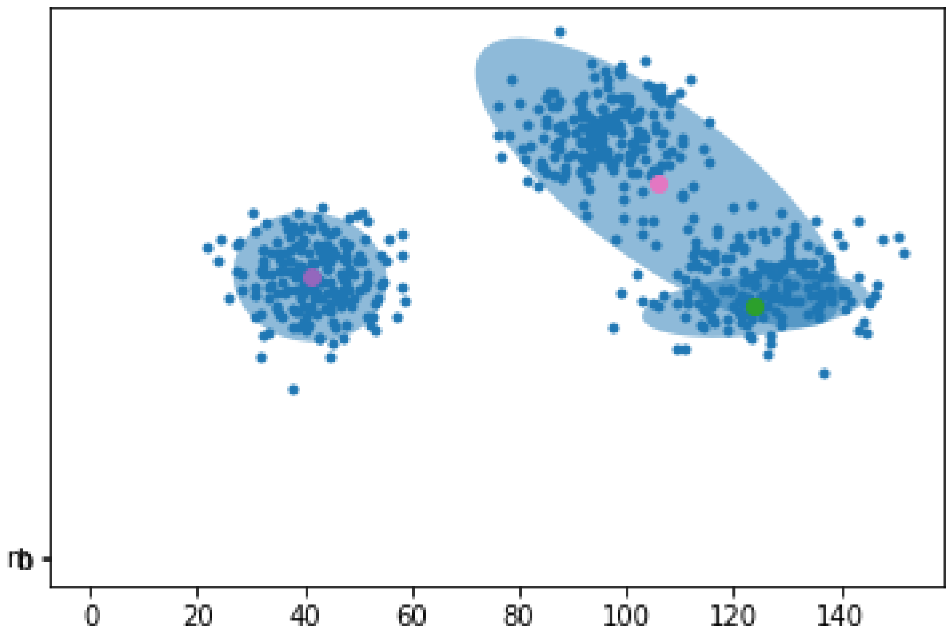
Iteration 2



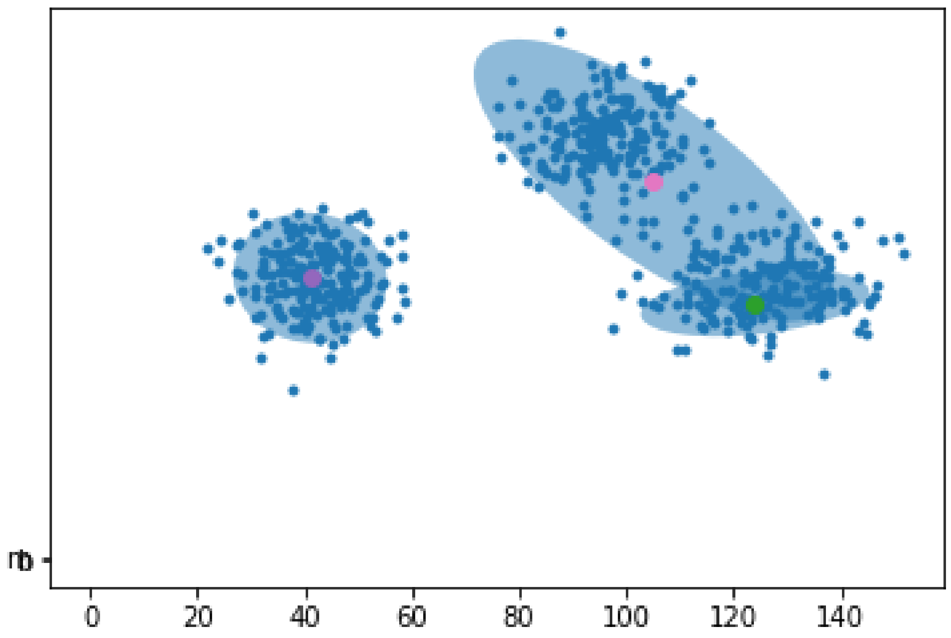
Iteration 3



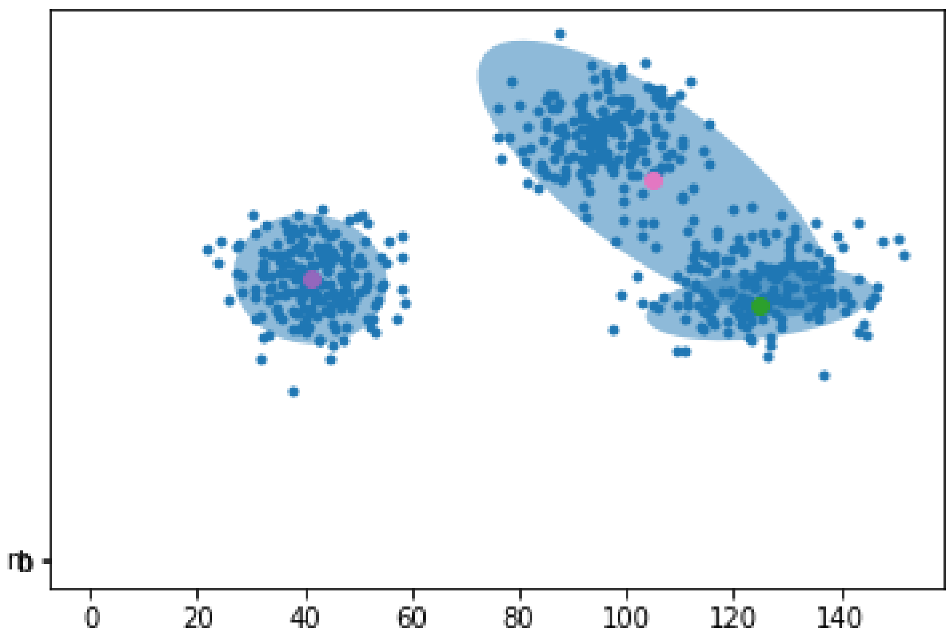
Iteration 4



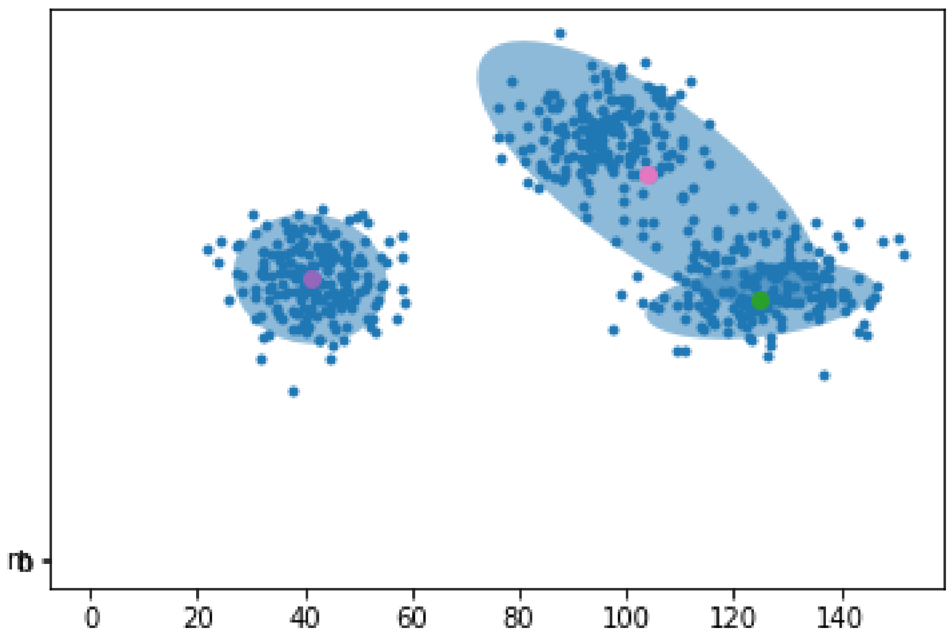
Iteration 5



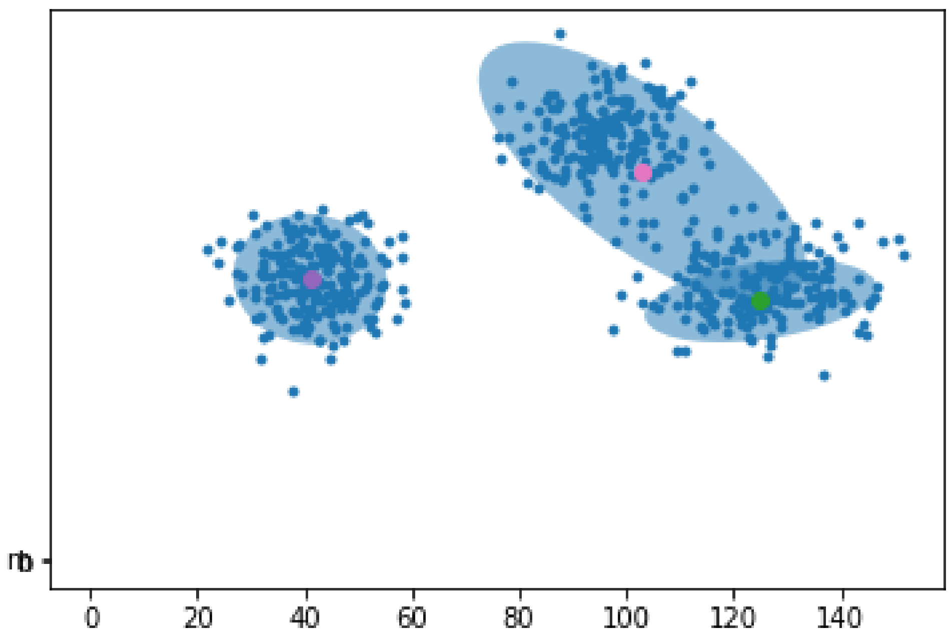
Iteration 6



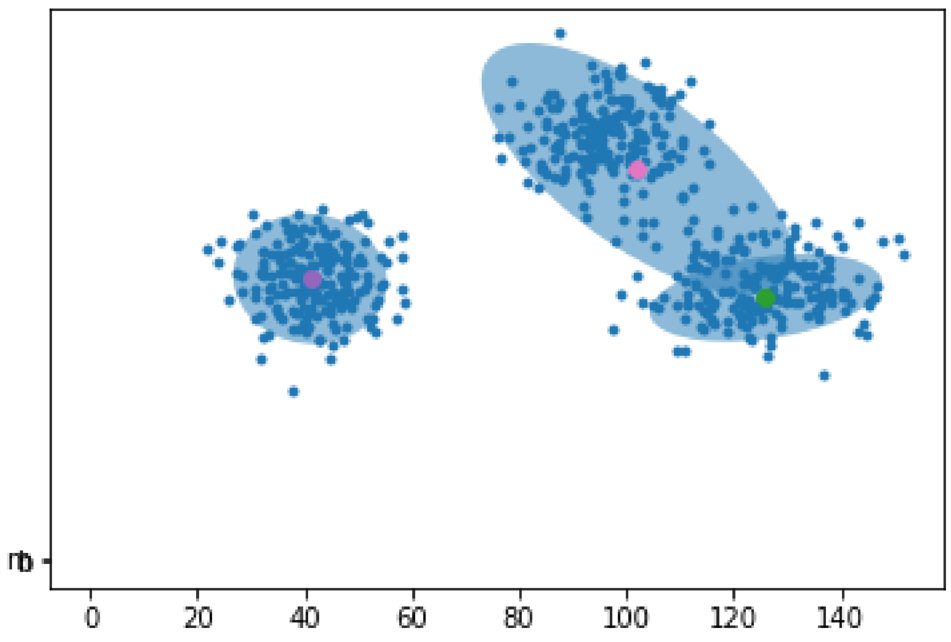
Iteration 7



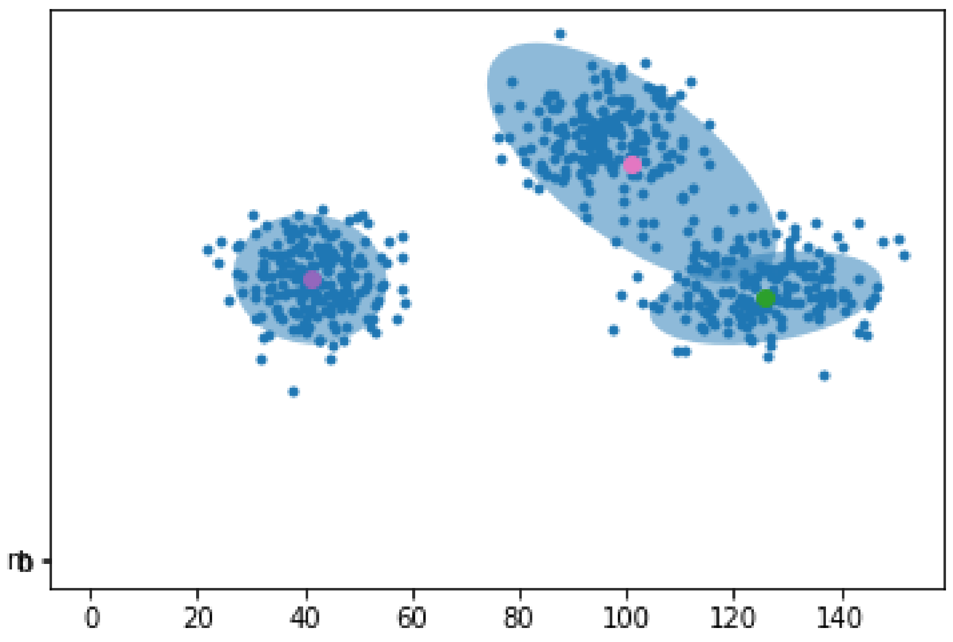
Iteration 8



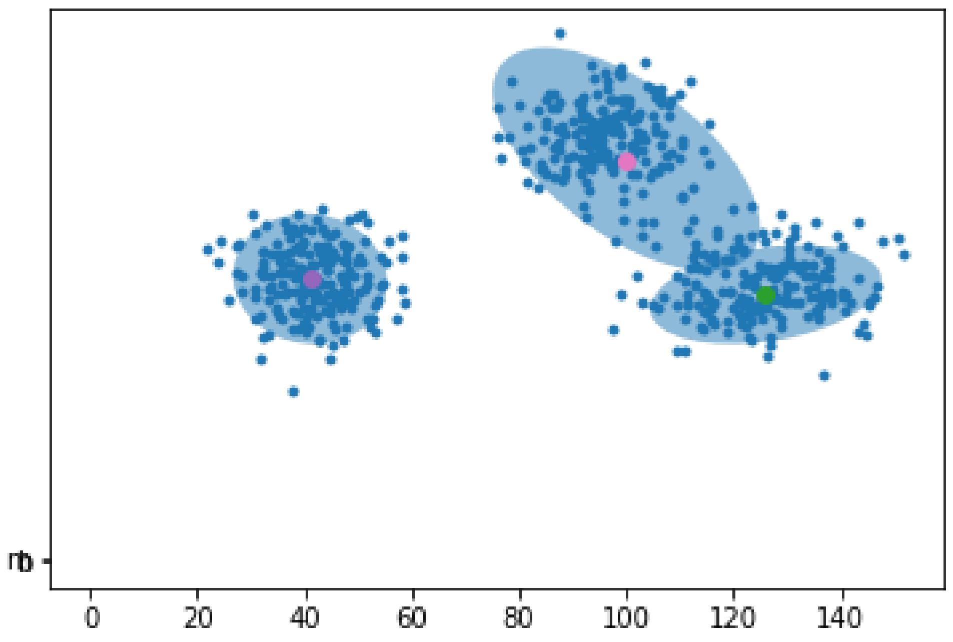
Iteration 9



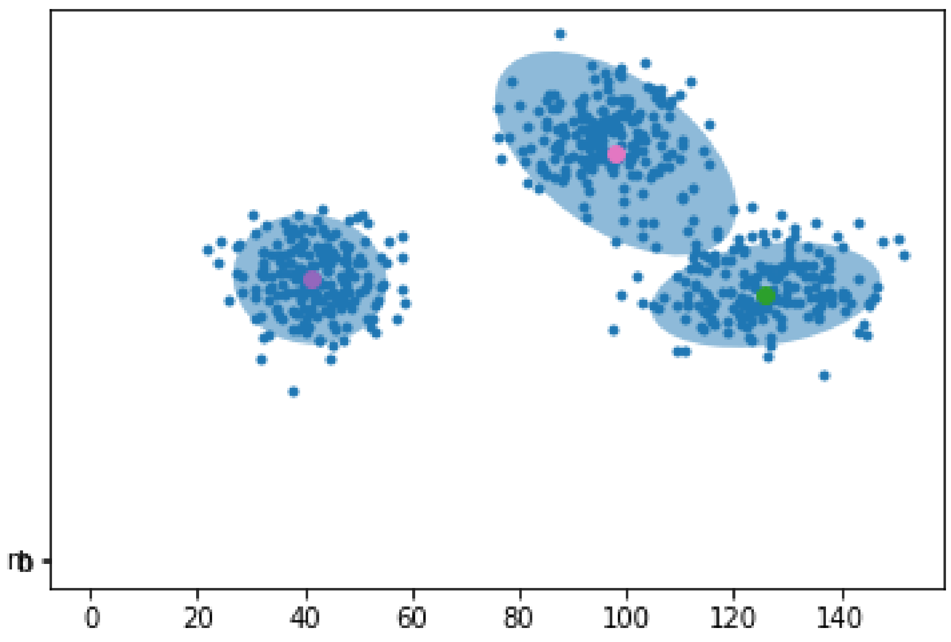
Iteration 10



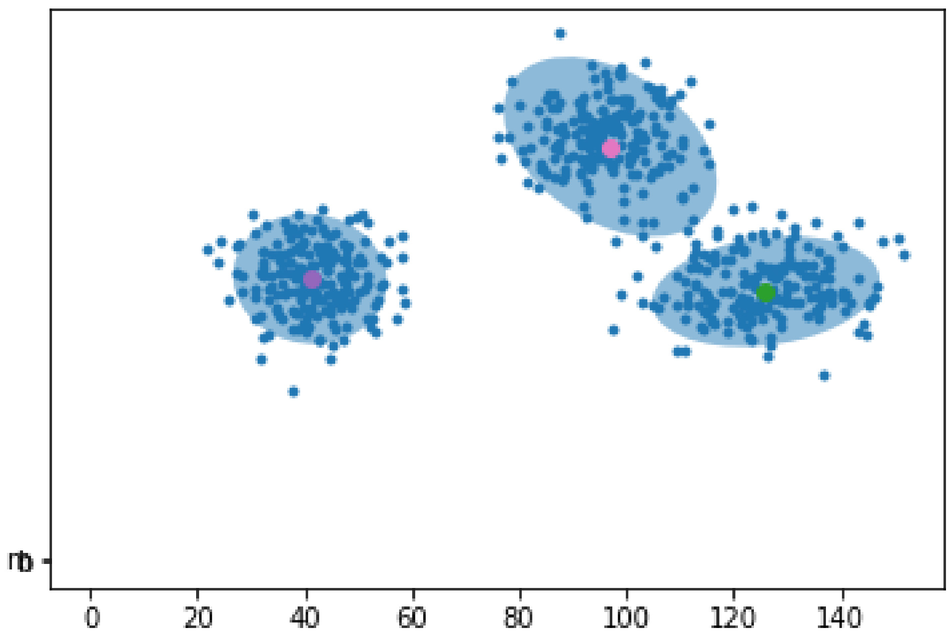
Iteration 11



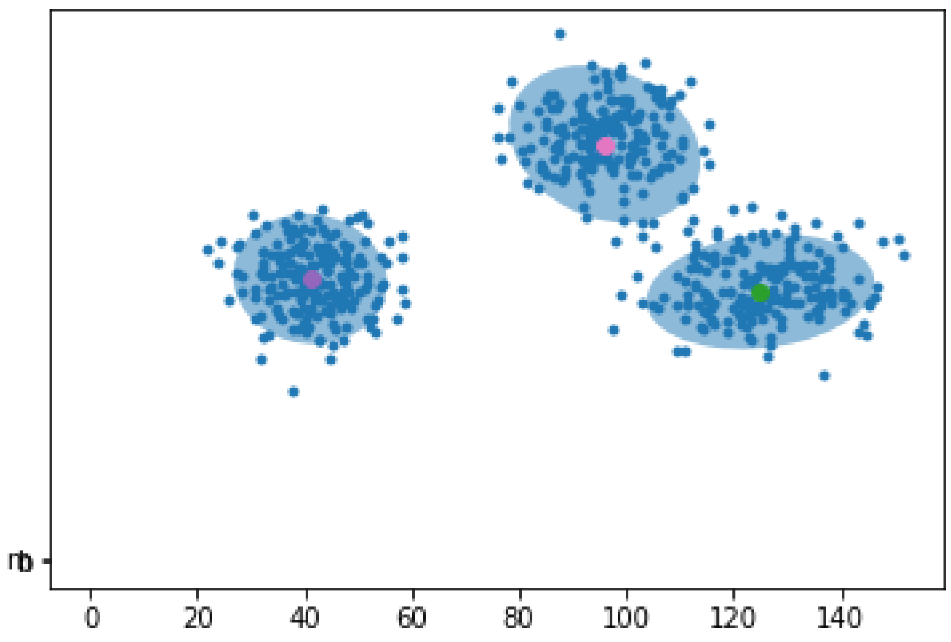
Iteration 12



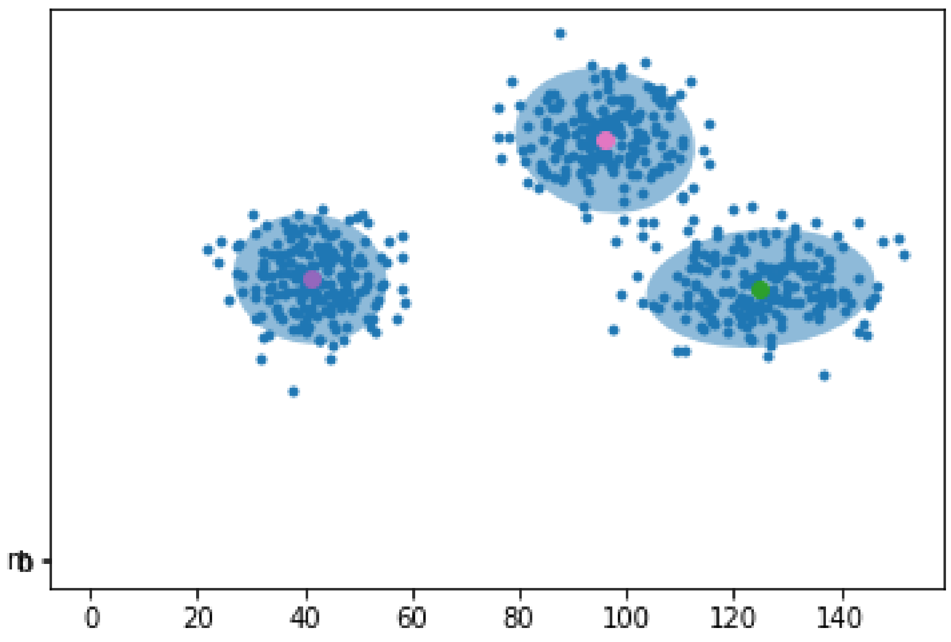
Iteration 13



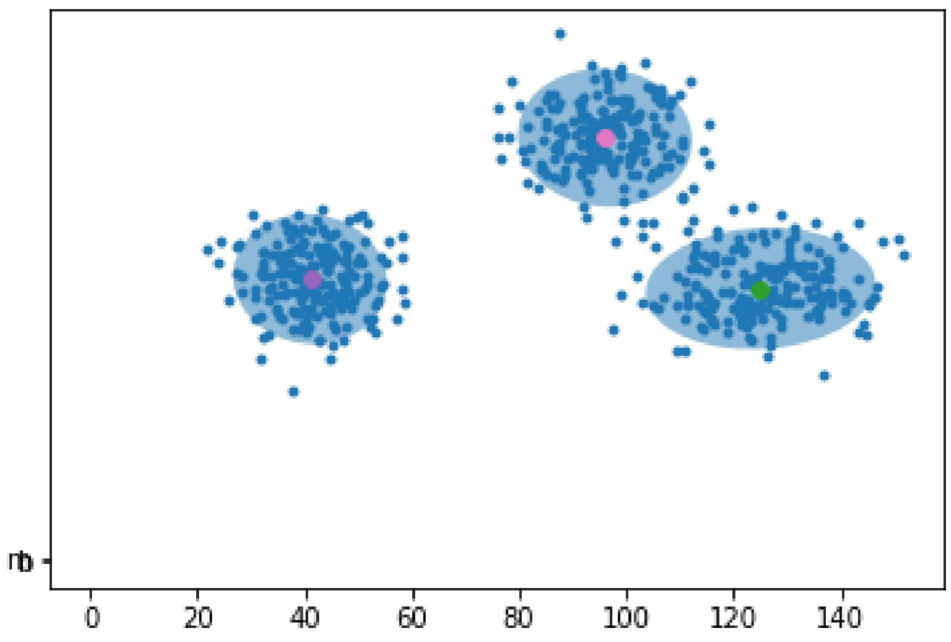
Iteration 14



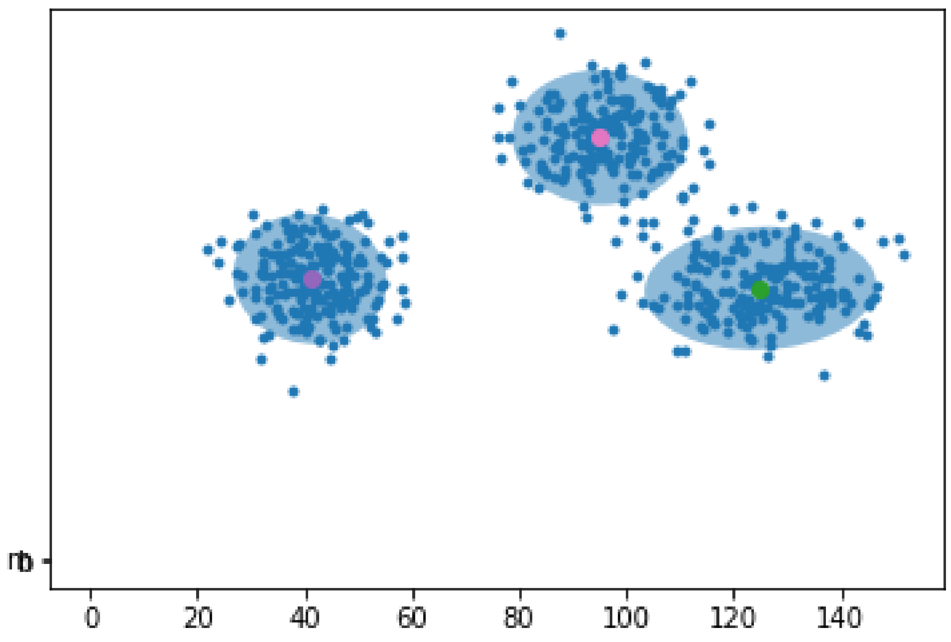
Iteration 15



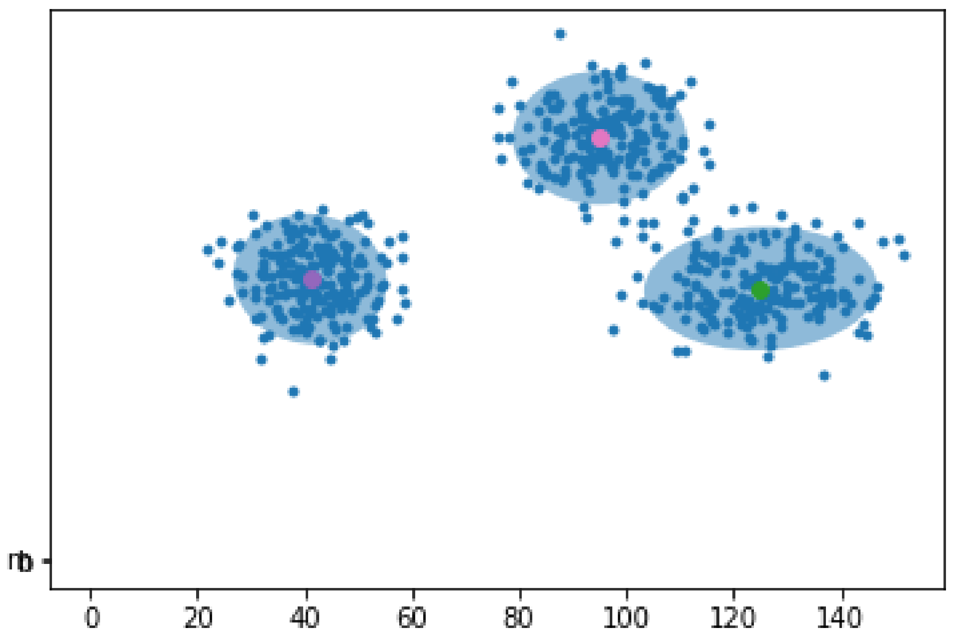
Iteration 16



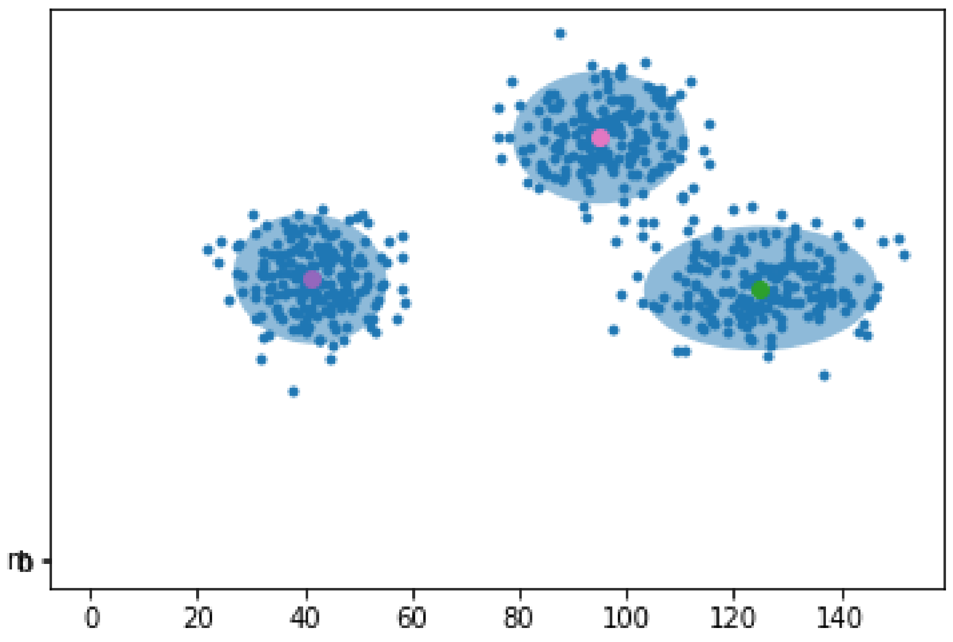
Iteration 17



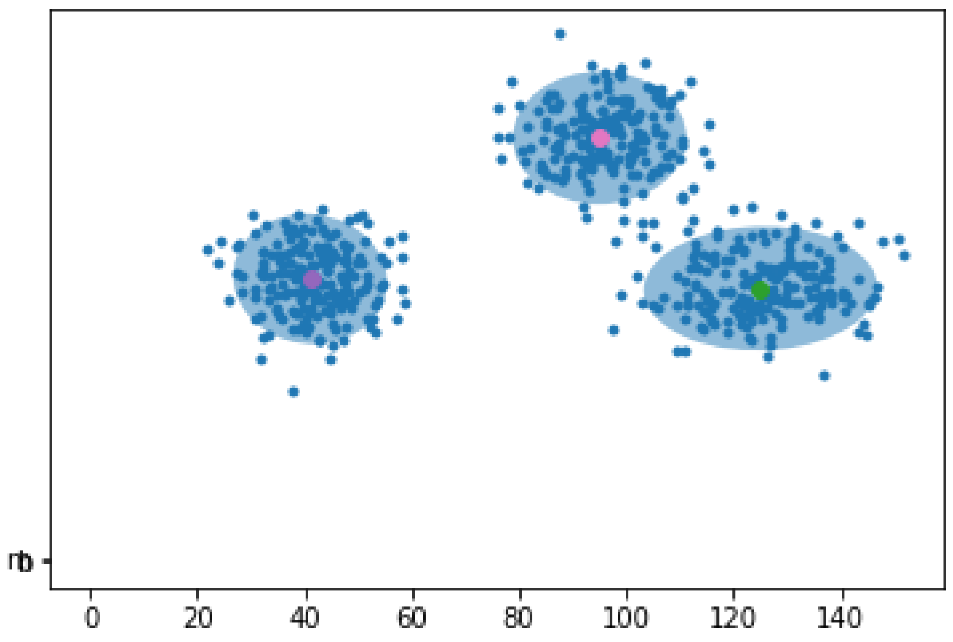
Iteration 18



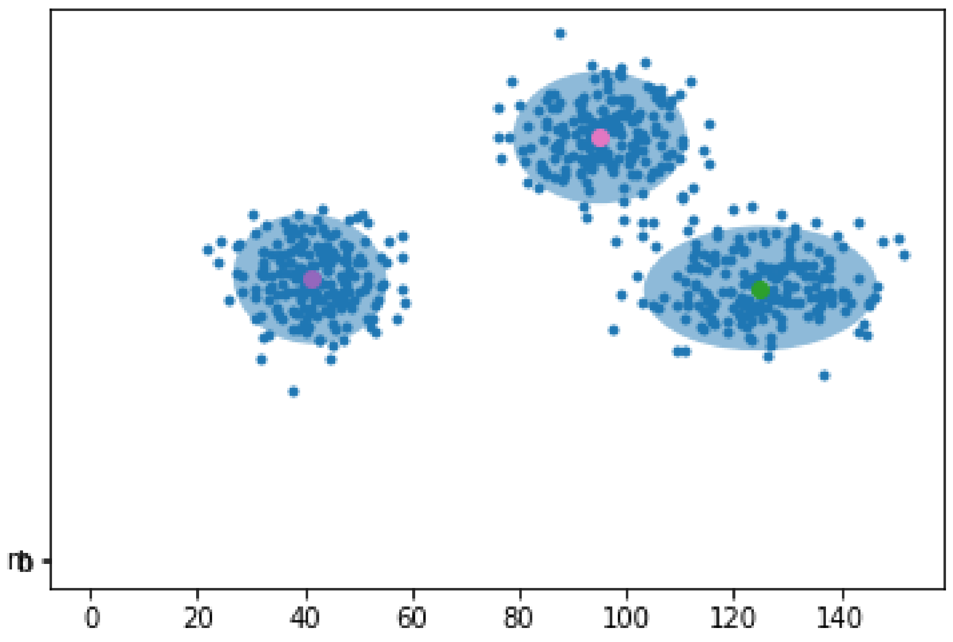
Iteration 19



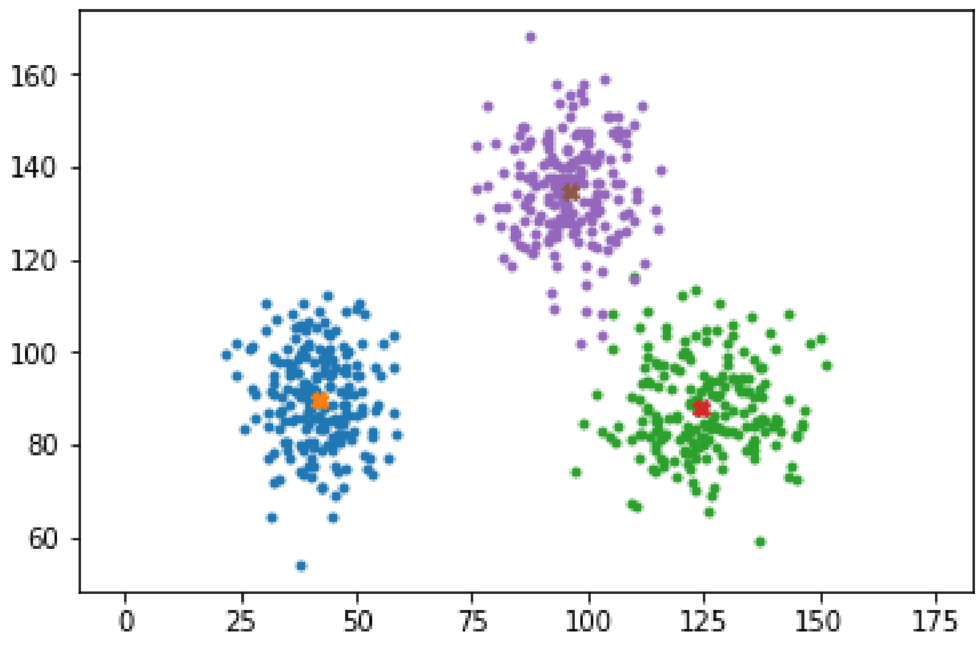
Iteration 20



Iteration 21



```
u:
[[125  87]
 [ 41  90]
 [ 95 135]]

E:
[[[118.34927386   3.16341255]
  [  3.16341255  98.24521864]]

 [[ 51.39676673  -3.66347456]
  [ -3.66347456 107.84988078]]

 [[ 65.47773308   0.57327508]
  [  0.57327508 111.57560682]]]


u_source:
[[42, 90], [124, 88], [96, 135]]

E_source:
 [[[ 62.   0.]
  [  0. 108.]]

 [[108.   0.]
  [  0. 102.]]

 [[ 61.   0.]
  [  0. 112.]]]
```

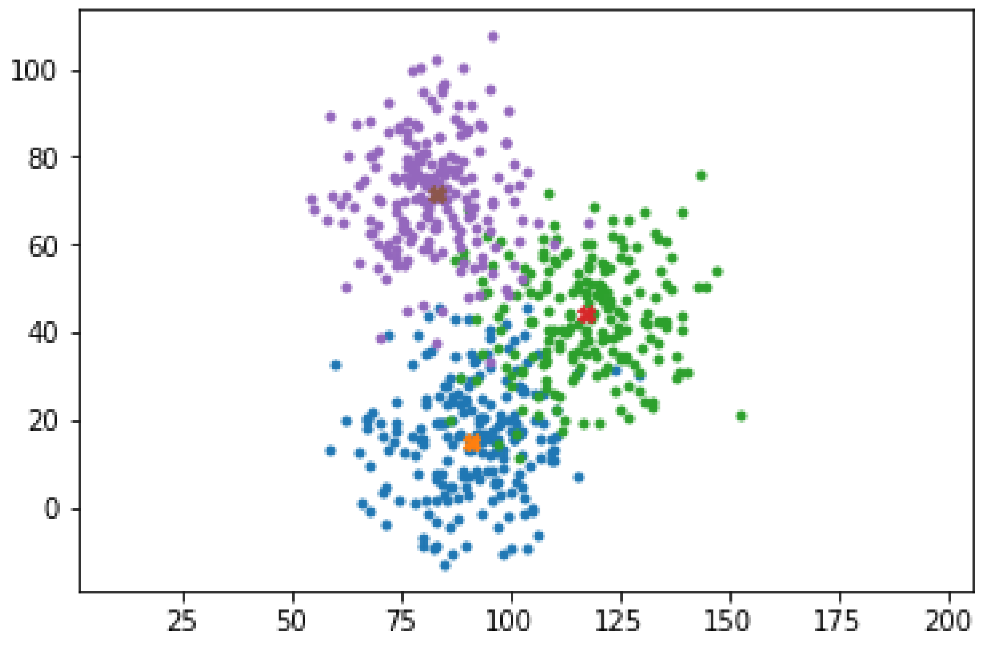


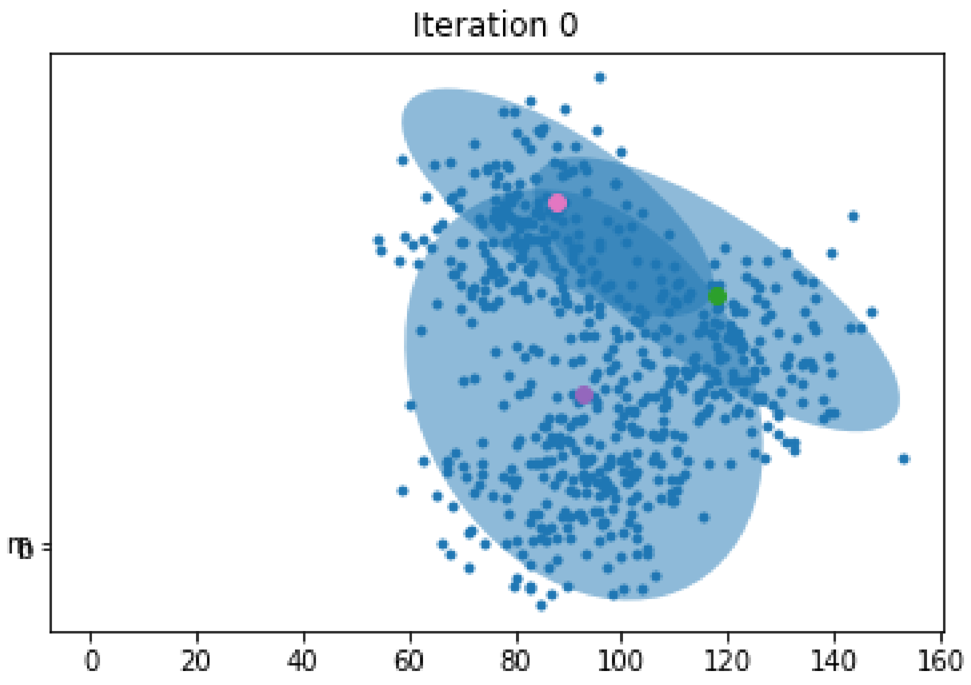
Case study: 2

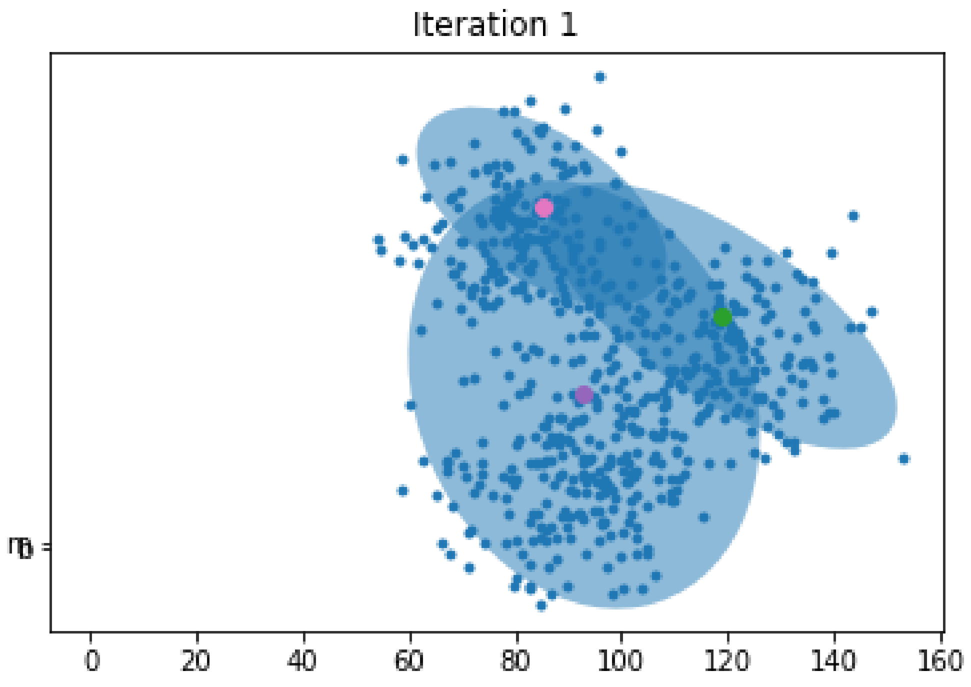
numberOfDatapoint = 600

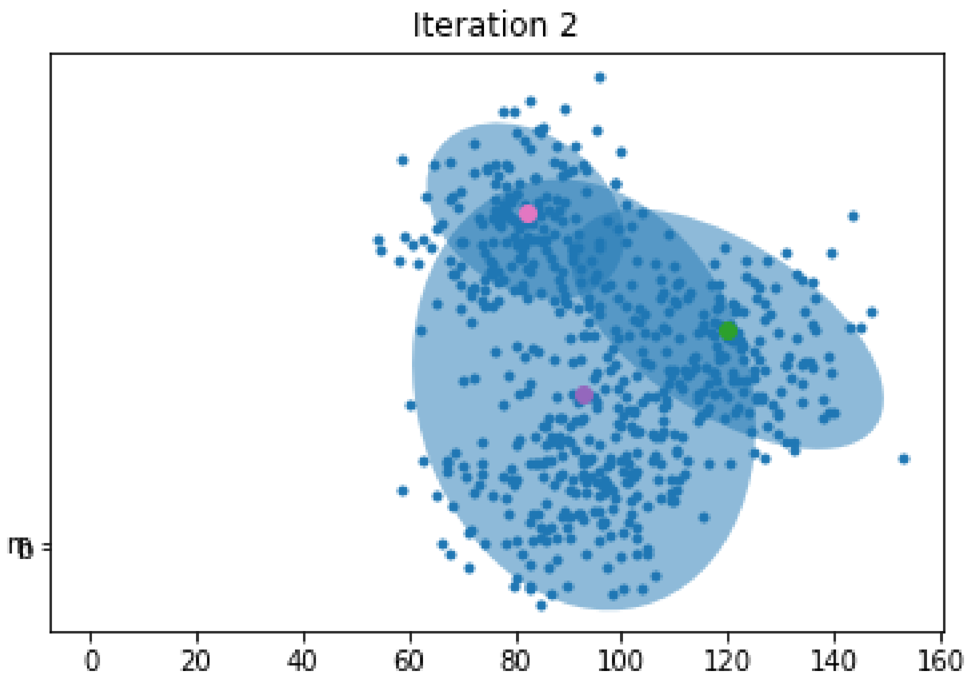
numberOfClusters = 3

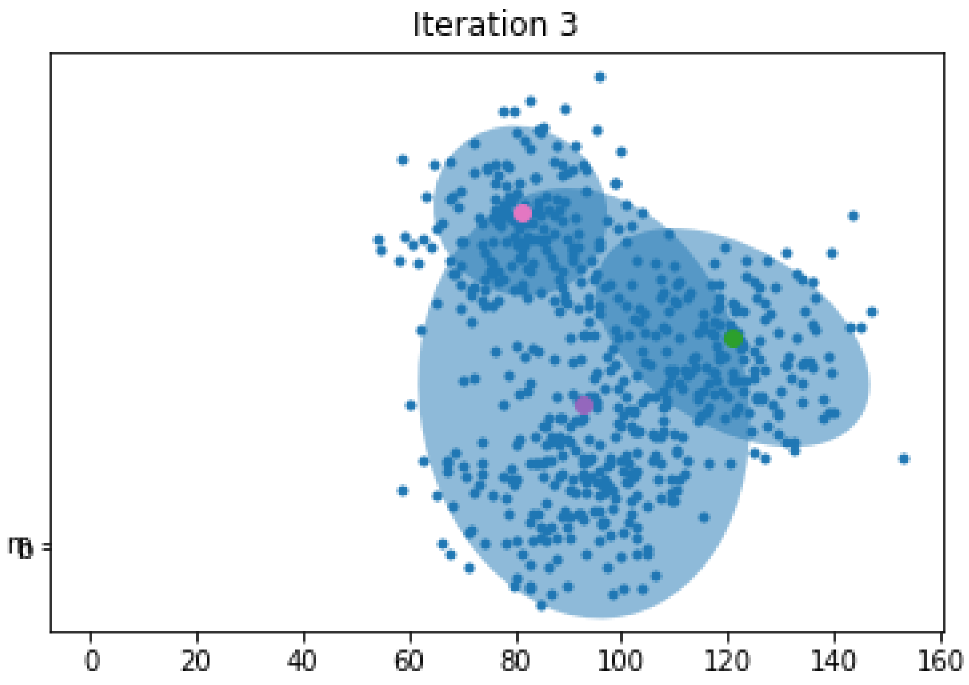
```
u_source:
[[91, 15], [117, 44], [83, 72]]

E_source:
 [[[161.   0.]
  [  0. 189.]]

 [[195.   0.]
  [  0. 174.]]

 [[128.   0.]
  [  0. 185.]]]
```

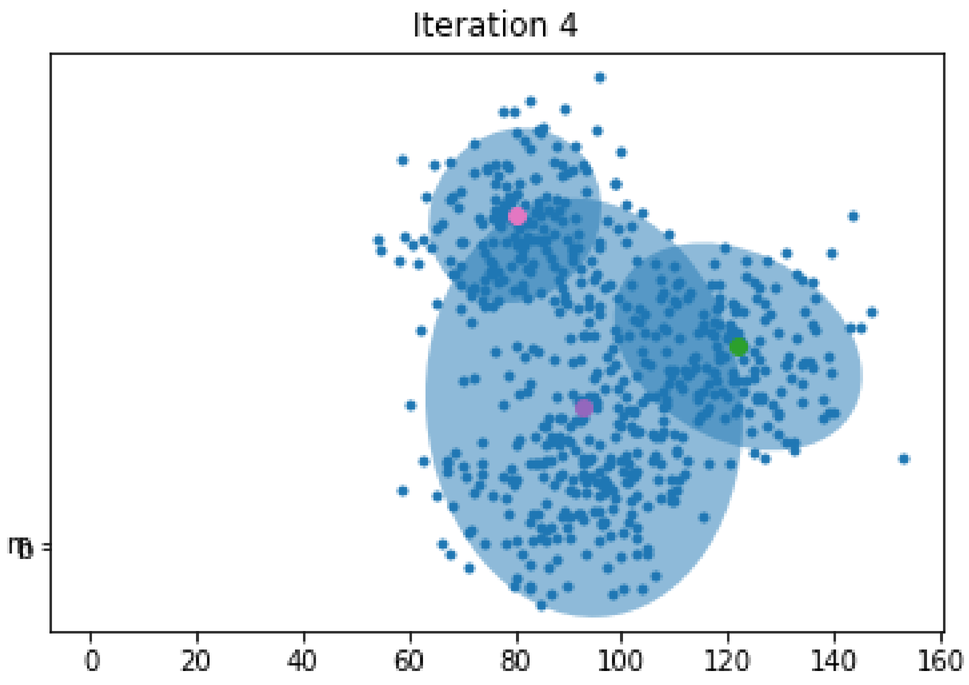


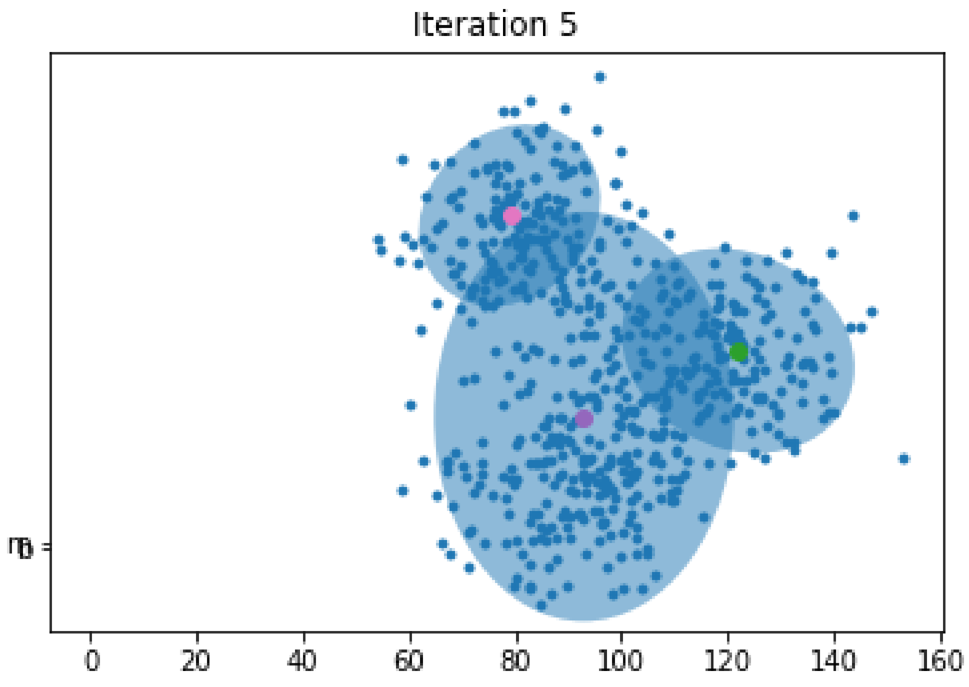

Iteration 0
0
0
20
40
60
80
100
120
140
160


Iteration 1
0
0
20
40
60
80
100
120
140
160

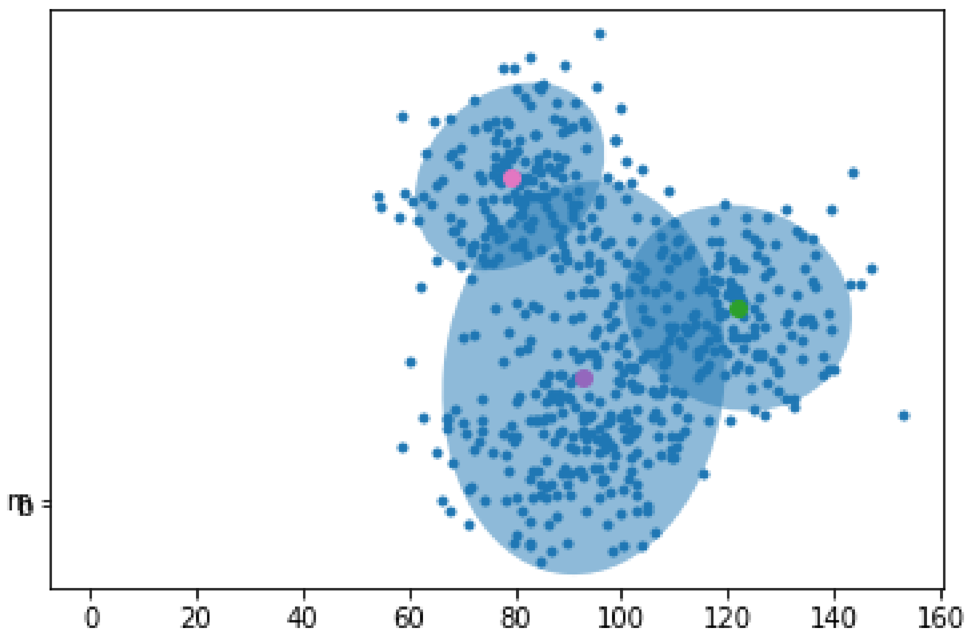

Iteration 2
0
20
40
60
80
100
120
140
160


Iteration 3
0
20
40
60
80
100
120
140
160

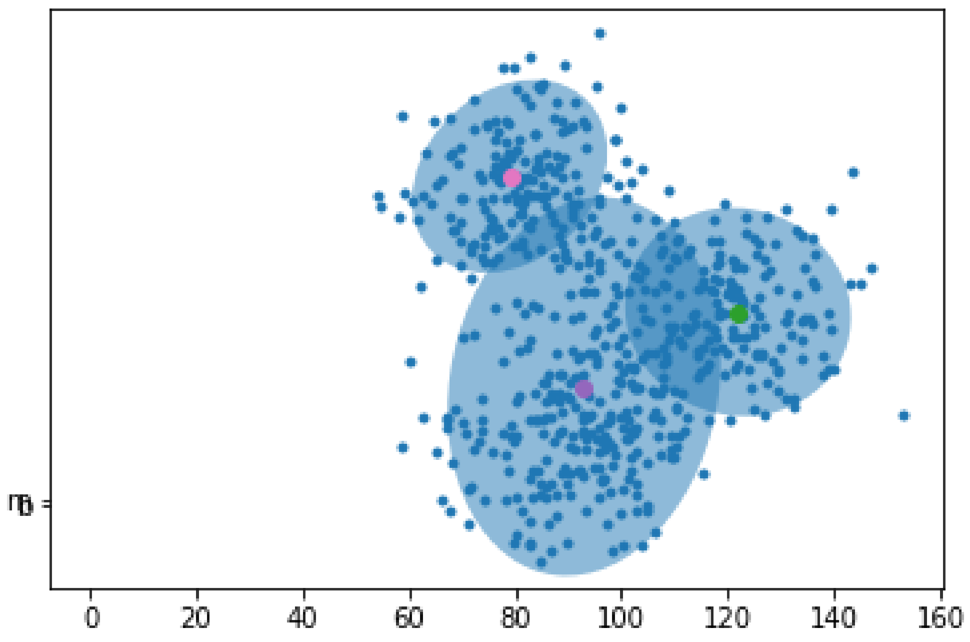

Iteration 4
0
20
40
60
80
100
120
140
160

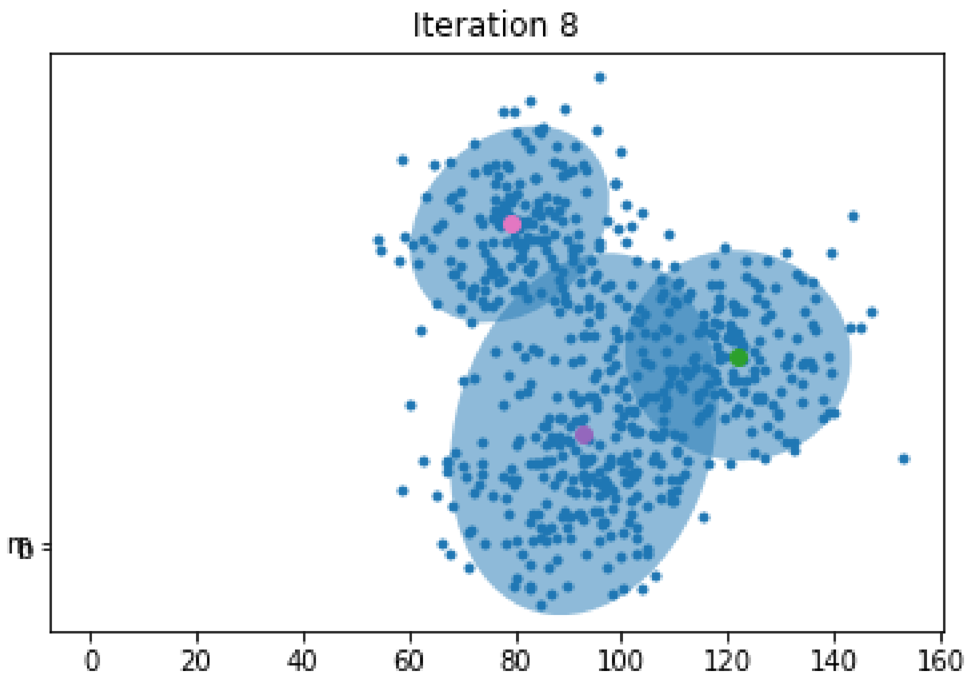

Iteration 5
0
20
40
60
80
100
120
140
160

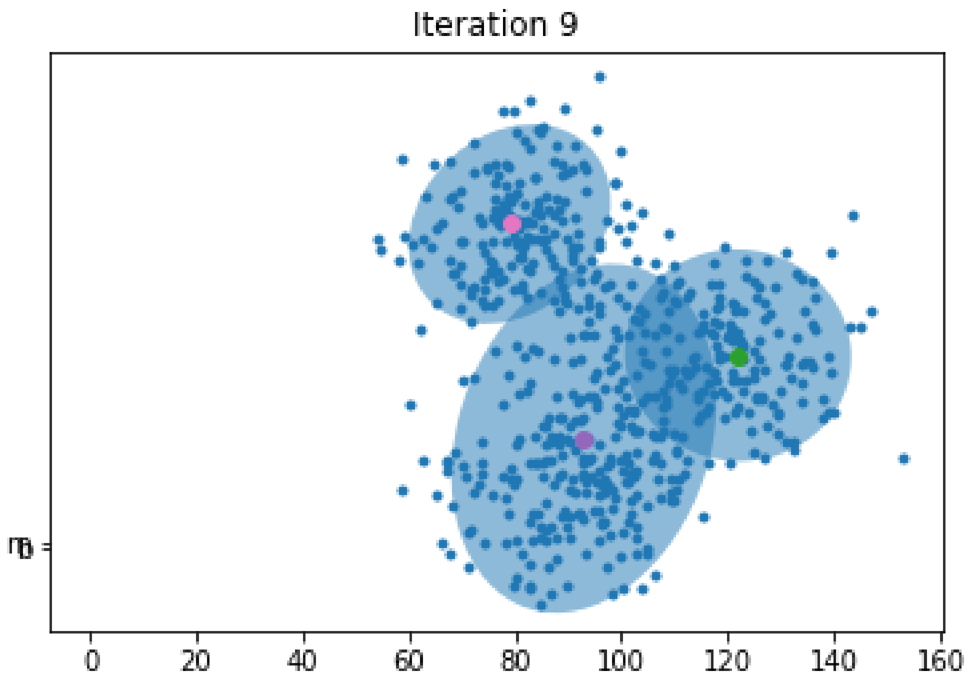
Iteration 6

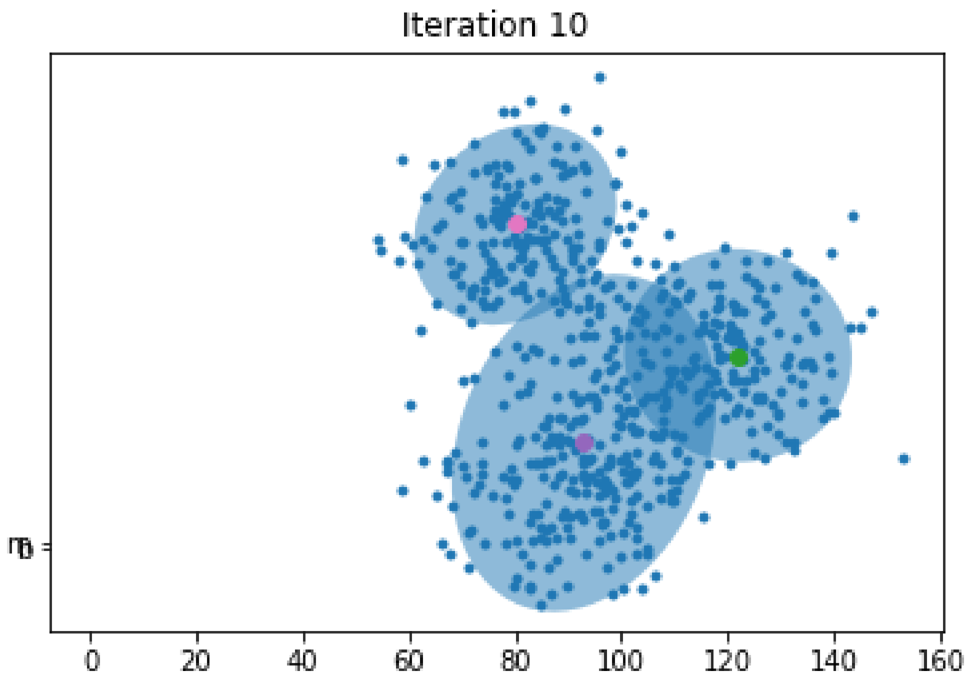


Iteration 7

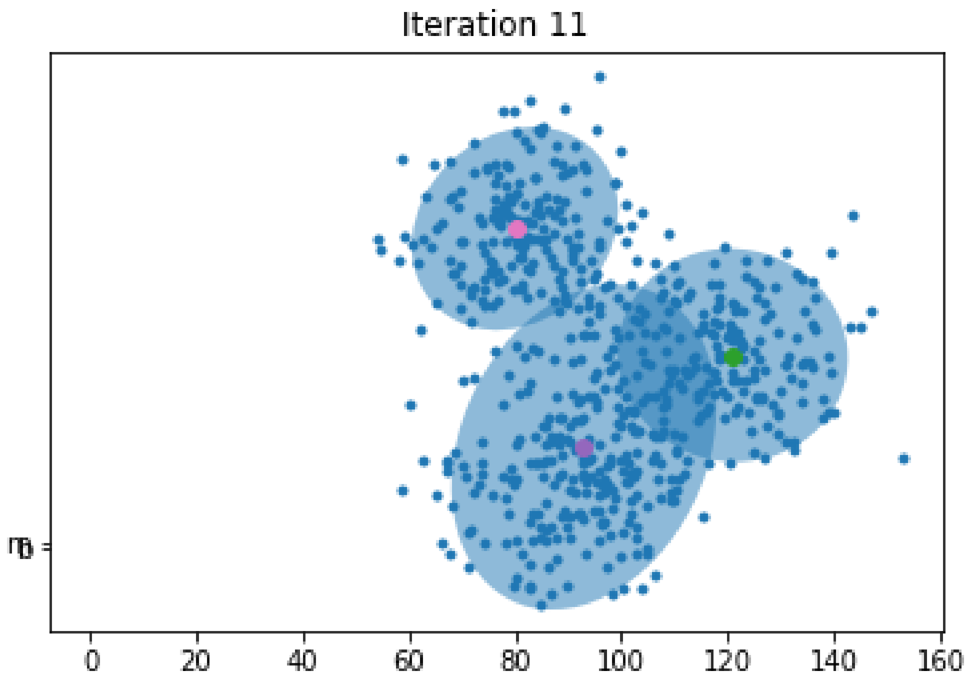



Iteration 8
0
20
40
60
80
100
120
140
160


Iteration 9
0
20
40
60
80
100
120
140
160

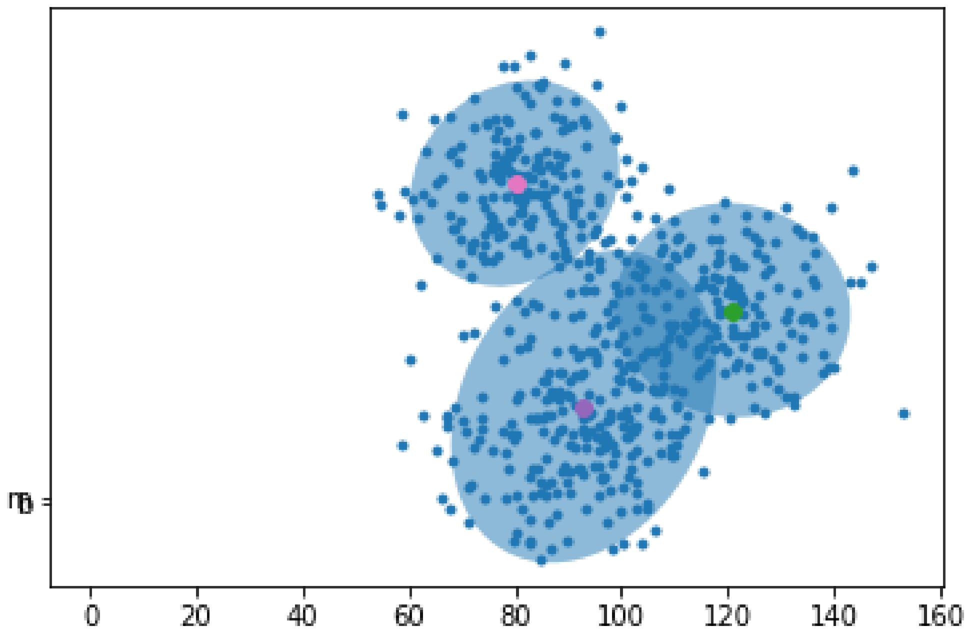


Iteration 10
0
20
40
60
80
100
120
140
160

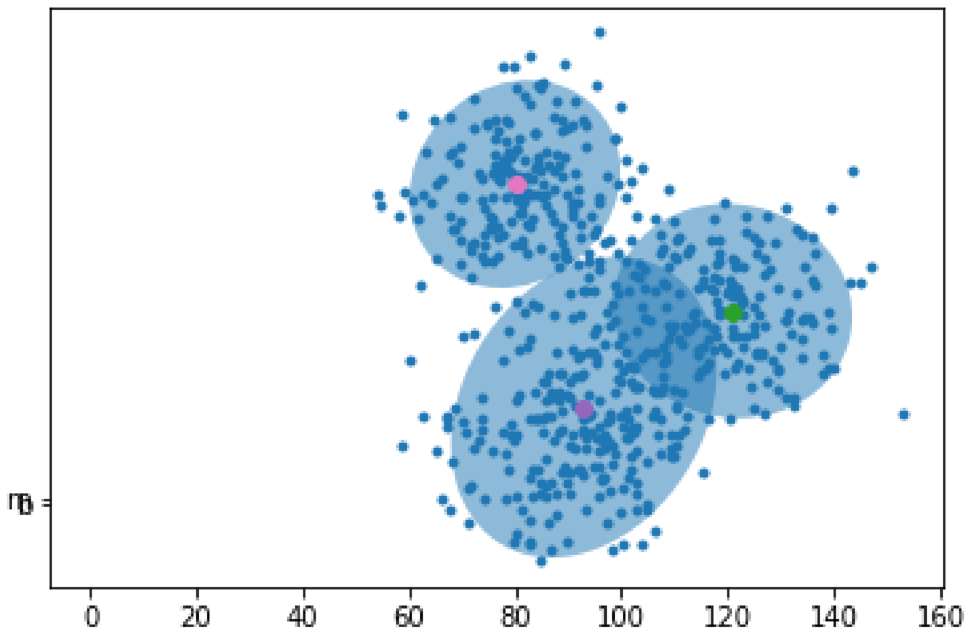


Iteration 11
0
20
40
60
80
100
120
140
160

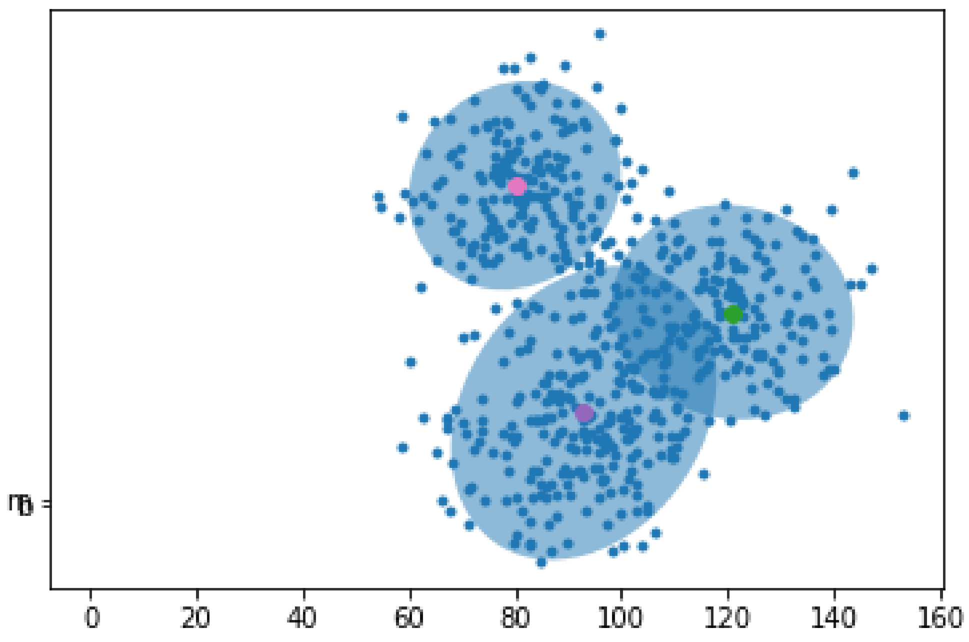
Iteration 12



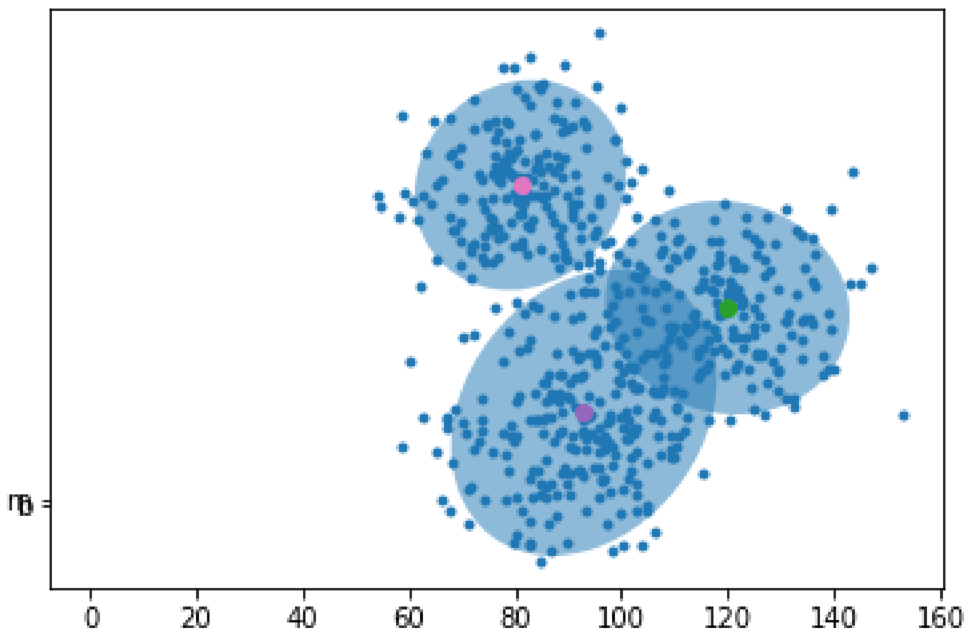
Iteration 13



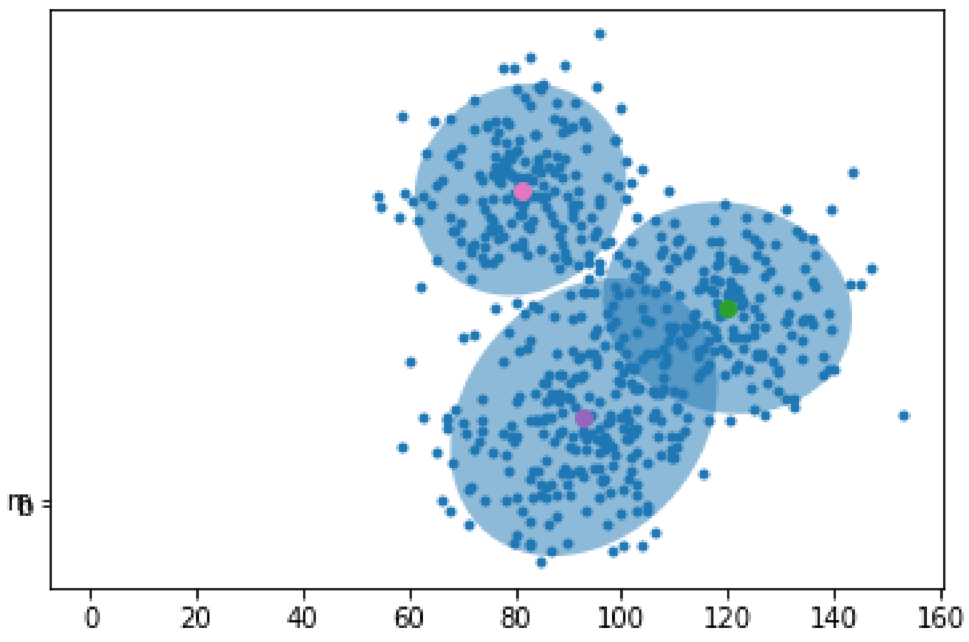
Iteration 14



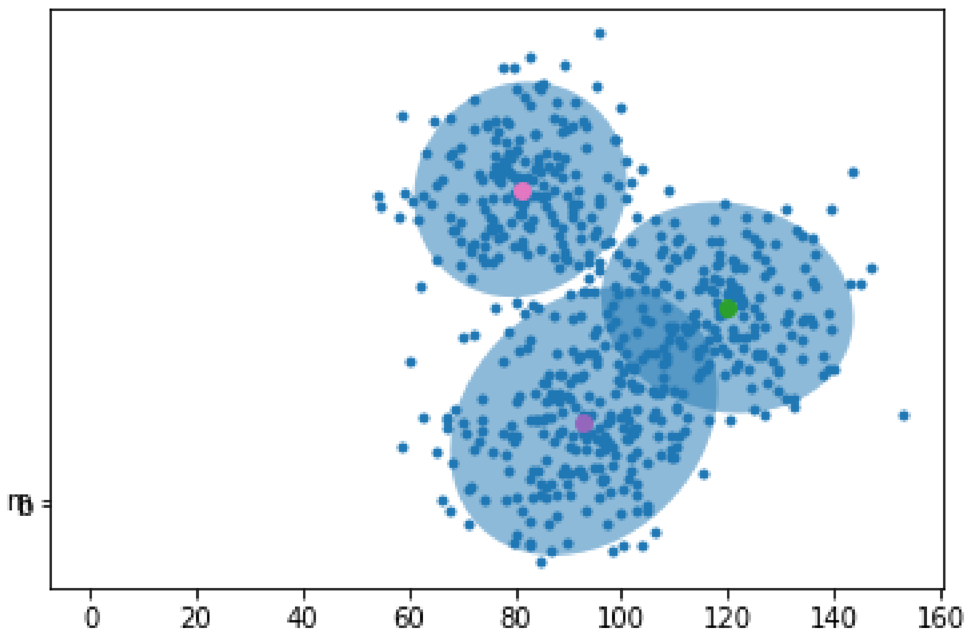
Iteration 15



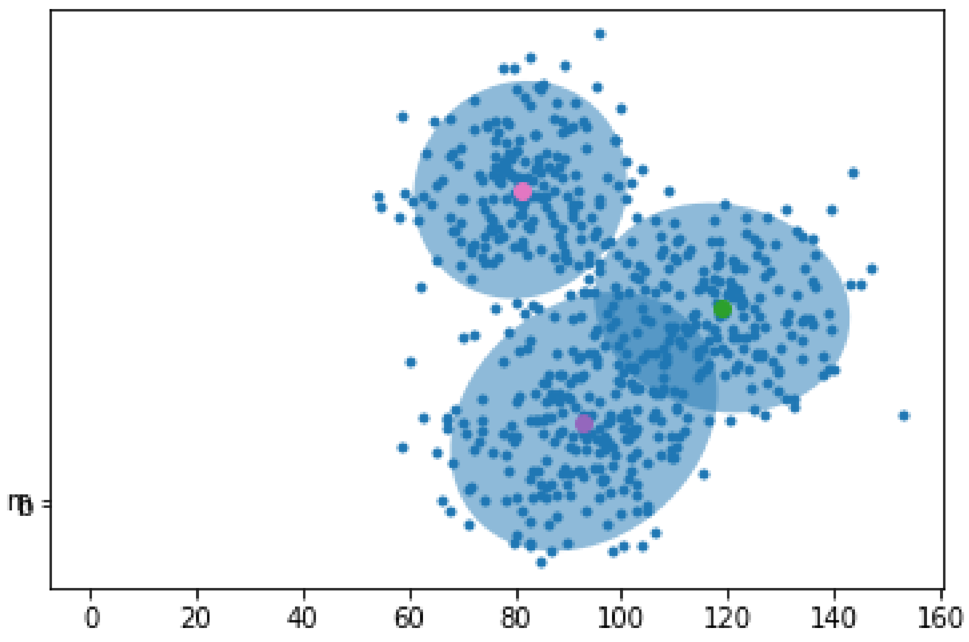
Iteration 16



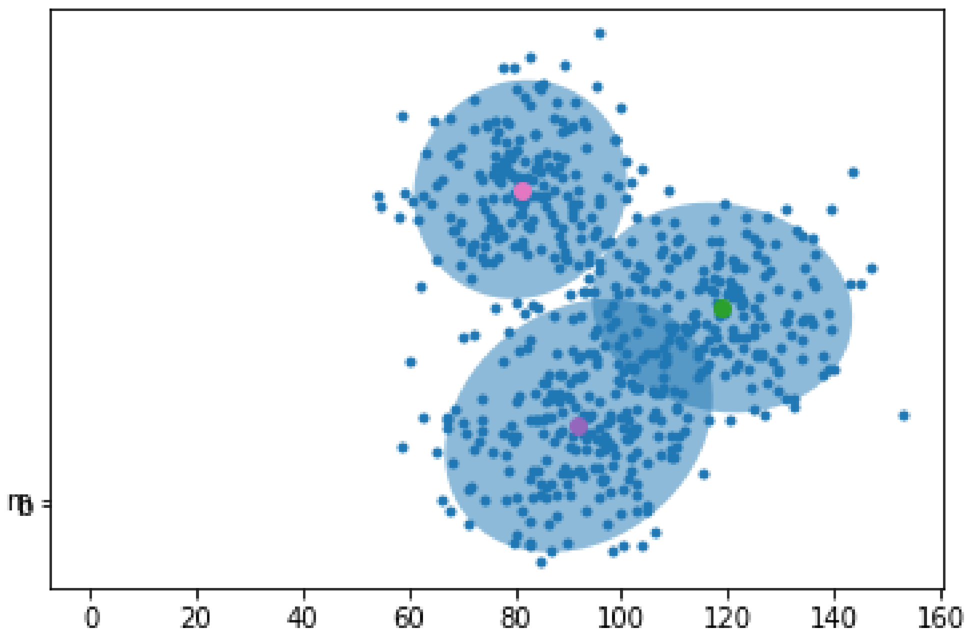
Iteration 17



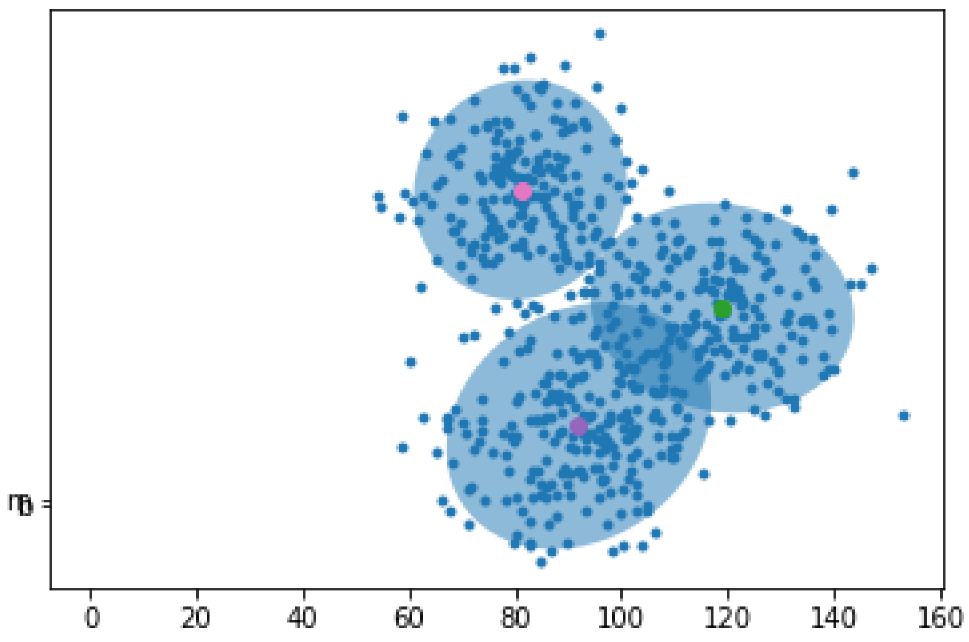
Iteration 18



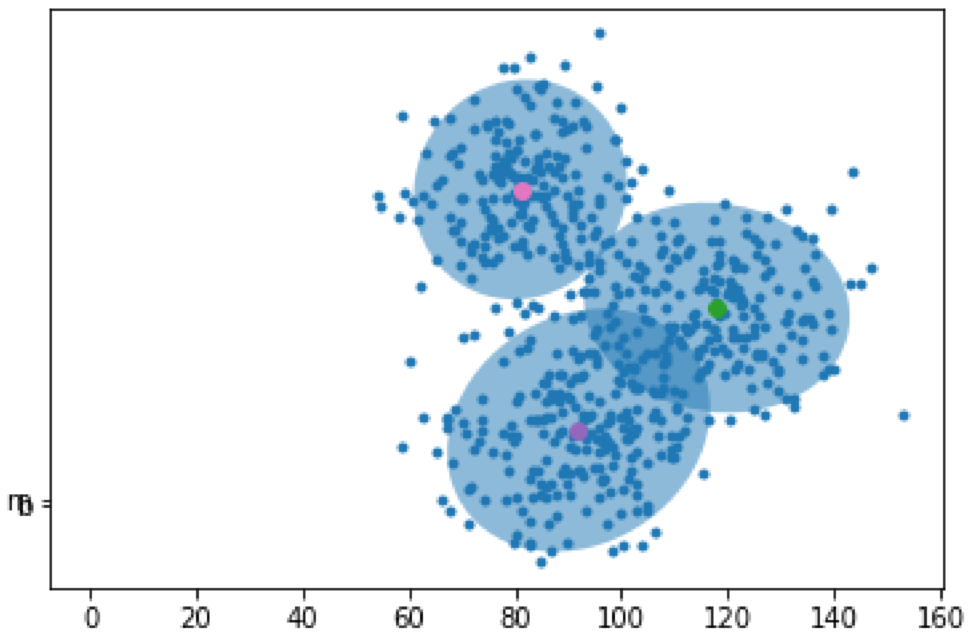
Iteration 19



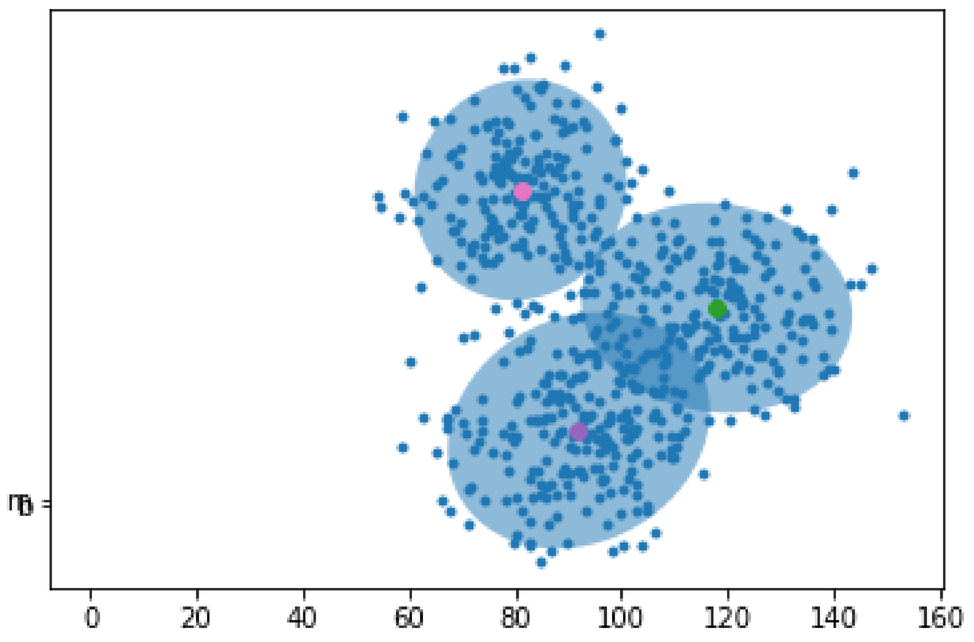
Iteration 20



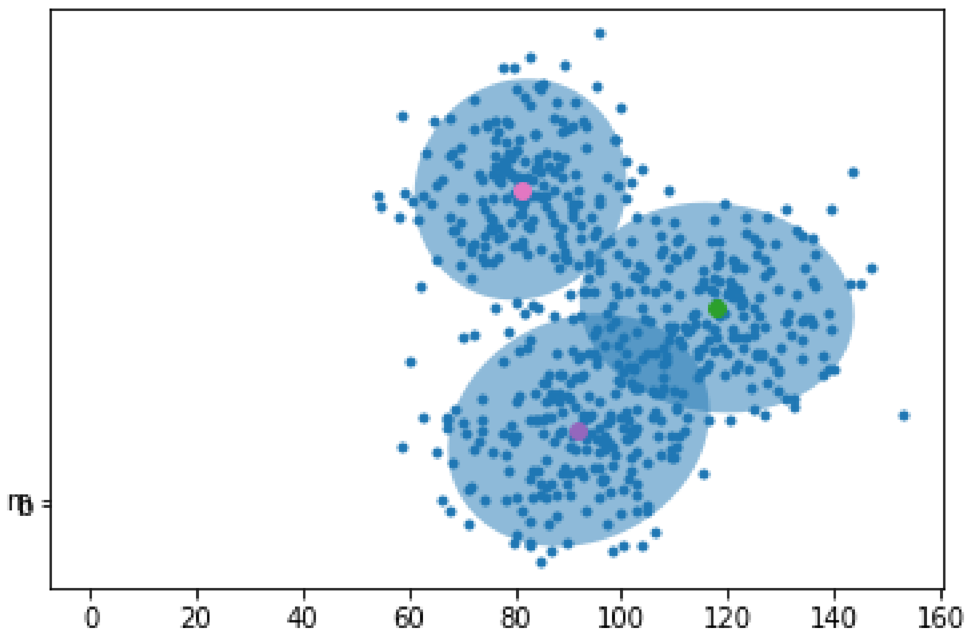
Iteration 21



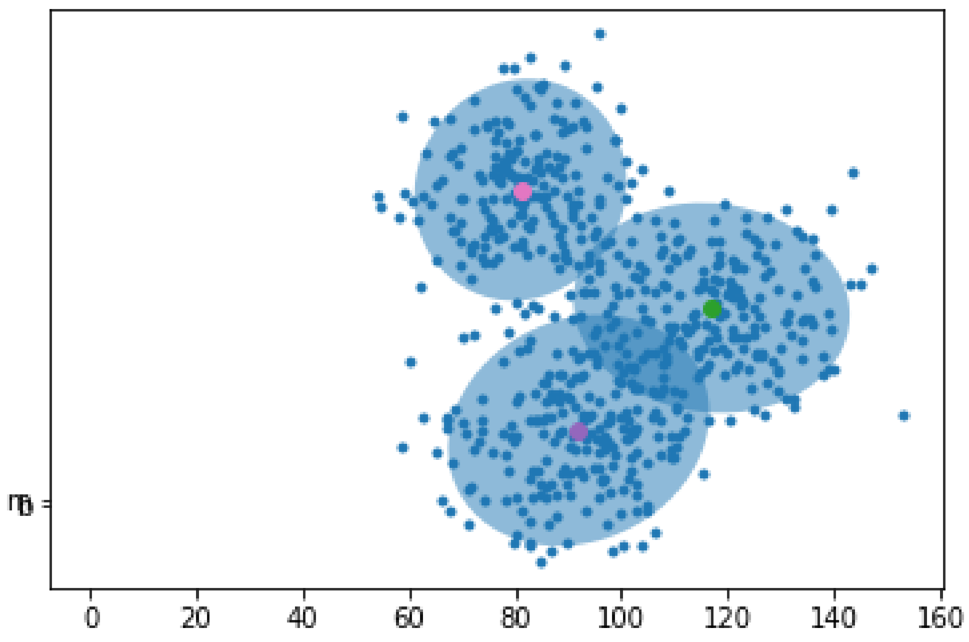
Iteration 22



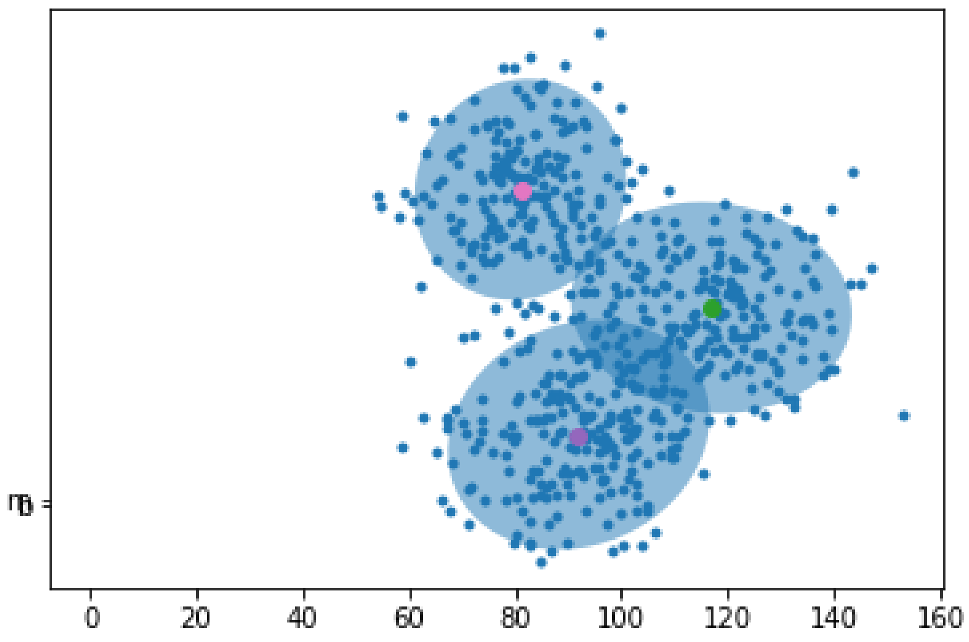
Iteration 23



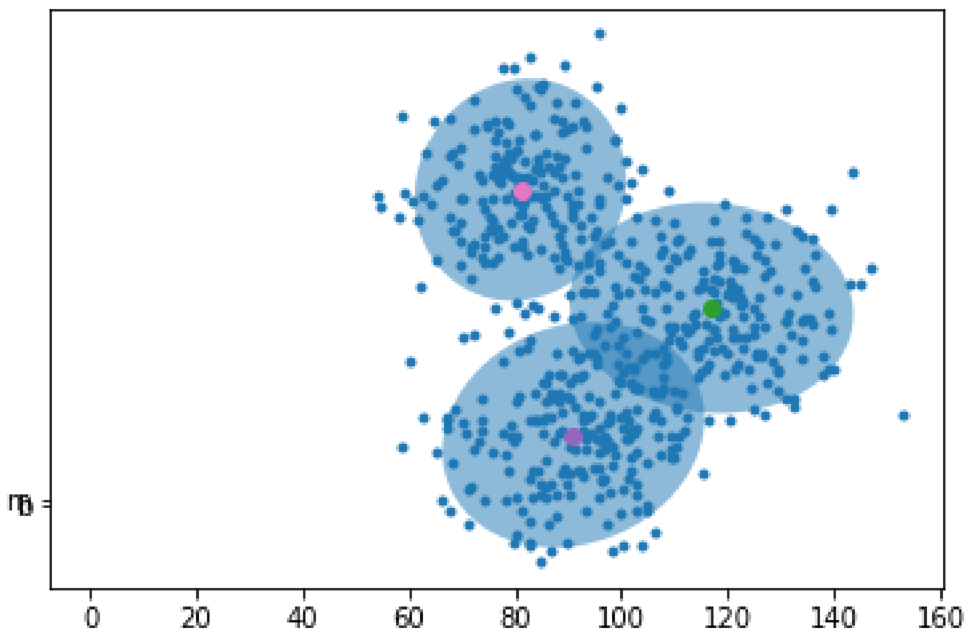
Iteration 24



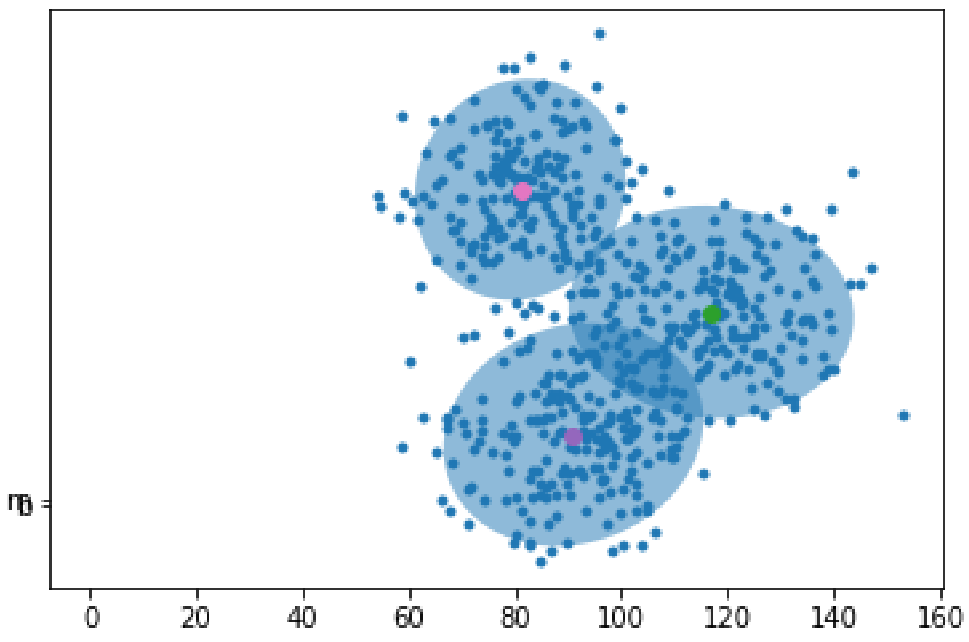
Iteration 25



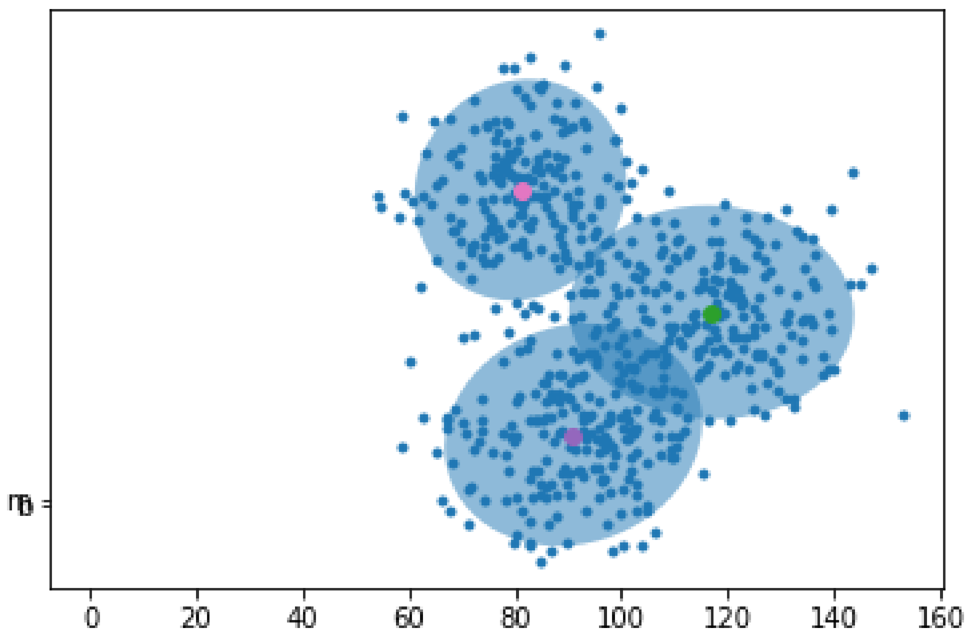
Iteration 26



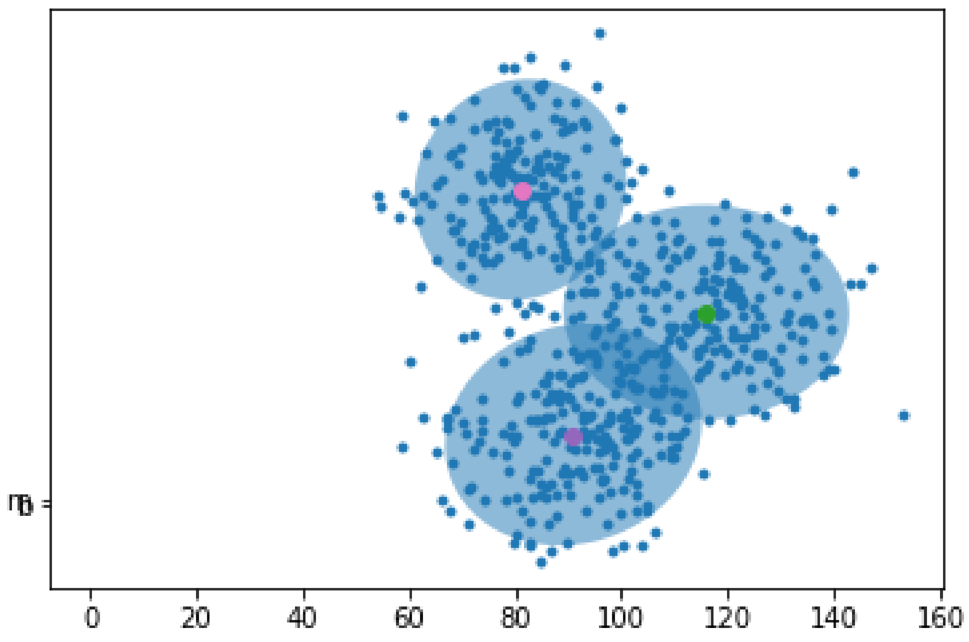
Iteration 27



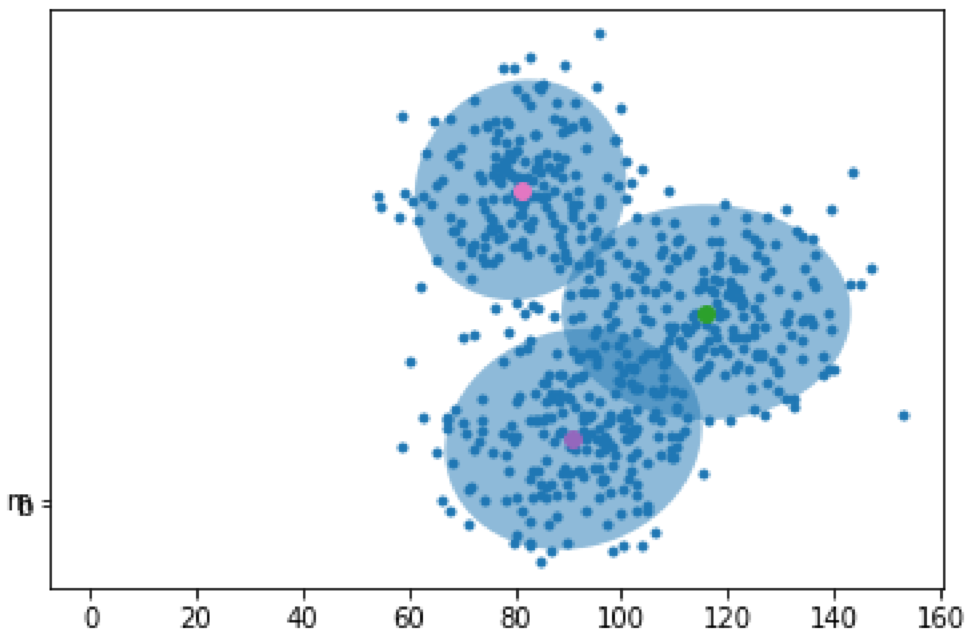
Iteration 28



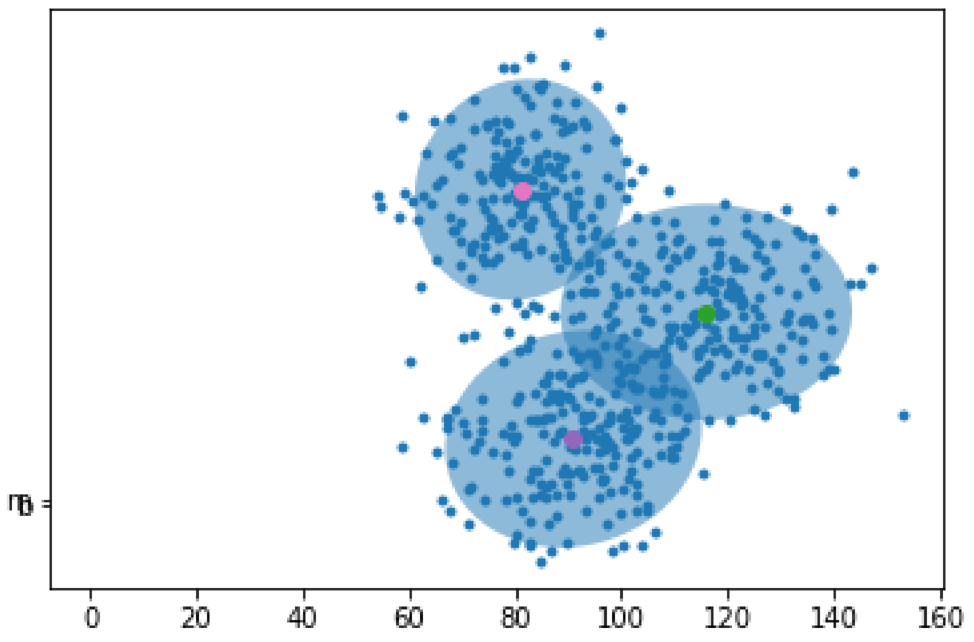
Iteration 29

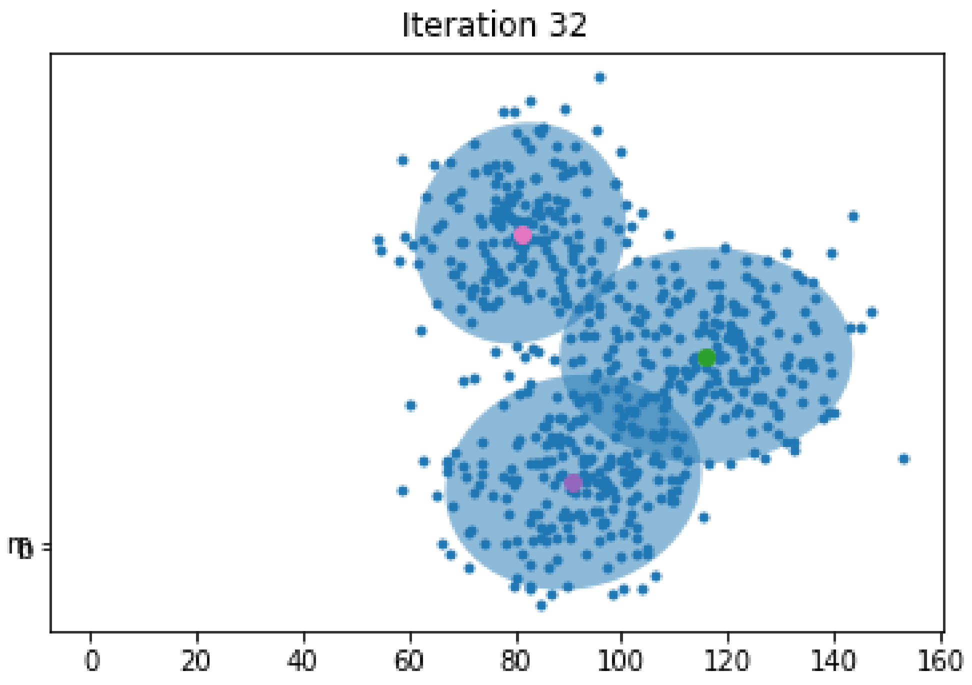


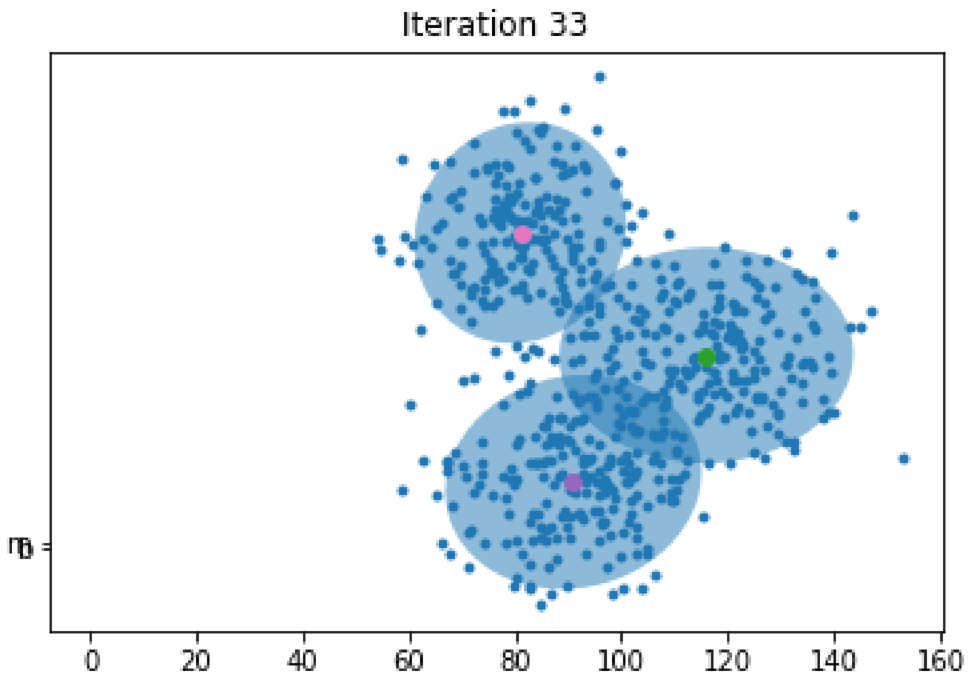
Iteration 30

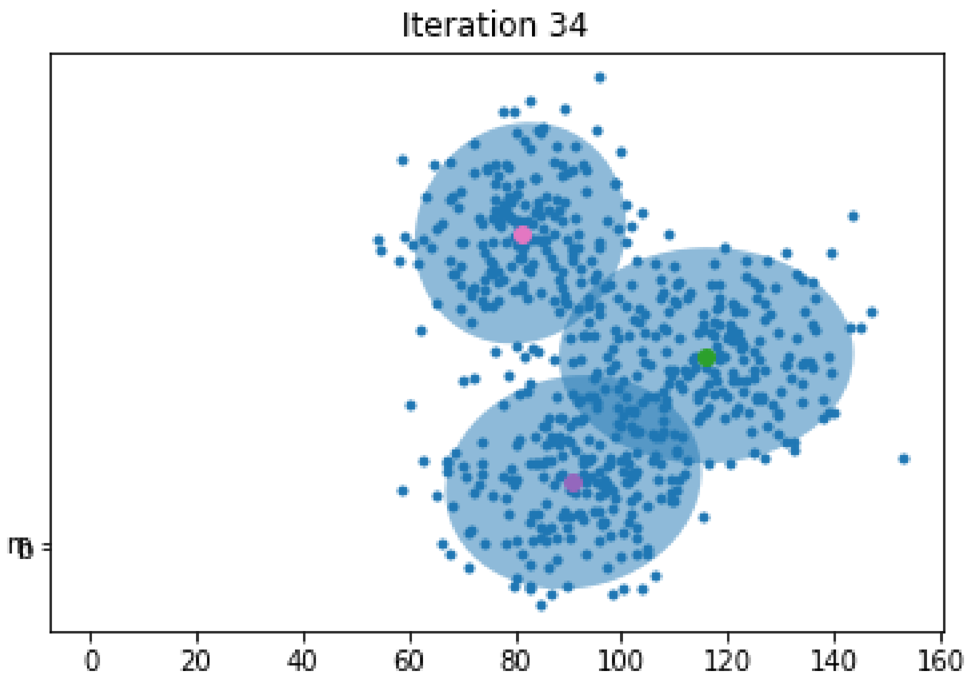


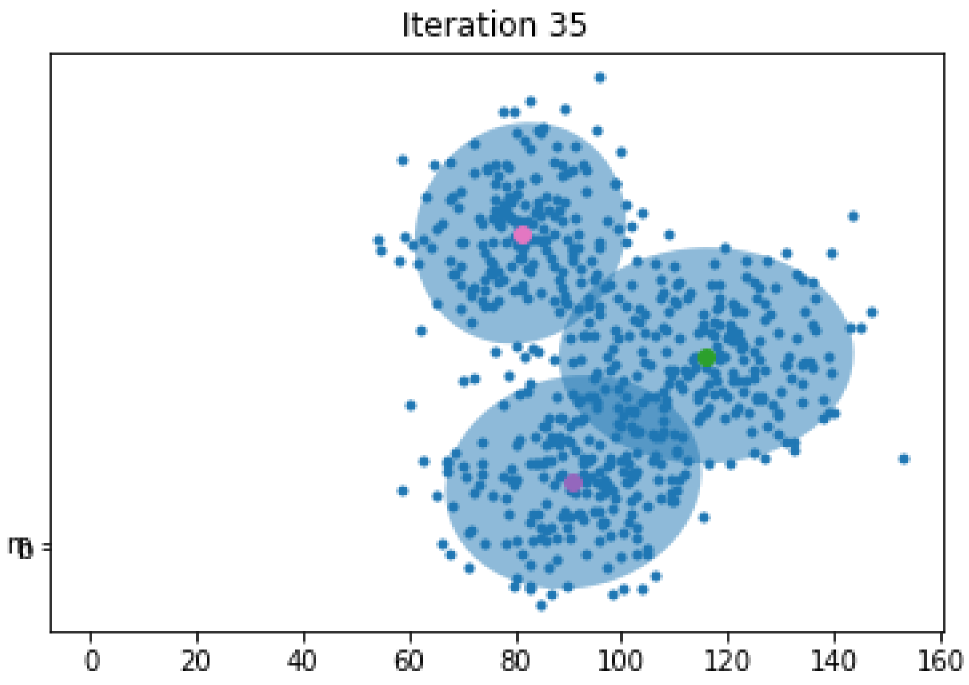
Iteration 31

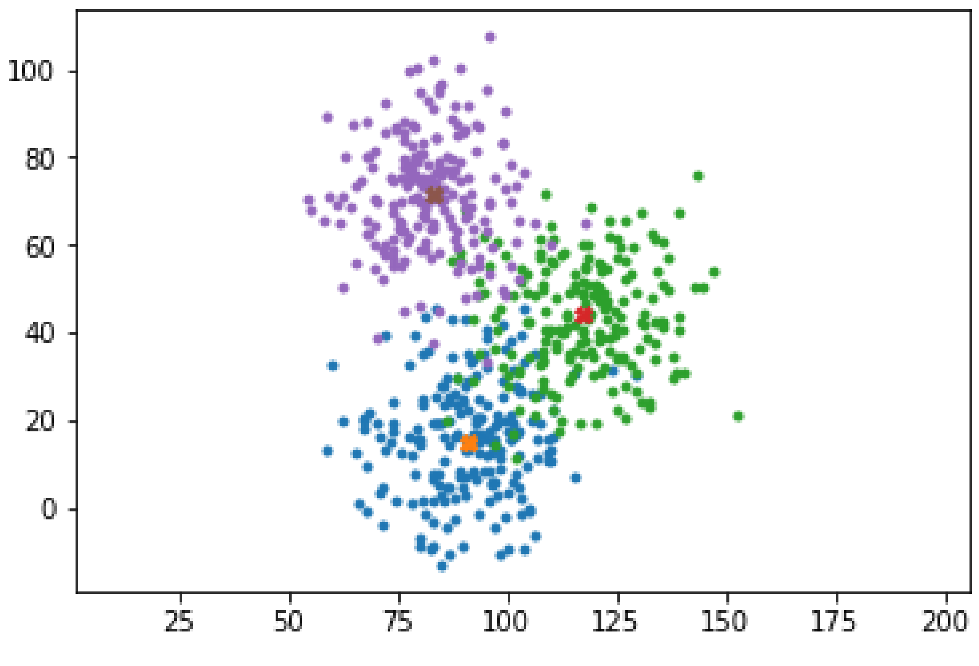



Iteration 32
0
20
40
60
80
100
120
140
160


Iteration 33
0
20
40
60
80
100
120
140
160


Iteration 34
0
0
20
40
60
80
100
120
140
160


Iteration 35
0
20
40
60
80
100
120
140
160


100
80
60
40
20
0
25
50
75
100
125
150
175
200

```
u:
[[116  44]
 [ 91  15]
 [ 81  72]]

E:
[[[192.74449773   2.67667052]
  [  2.67667052 152.2489376 ]]

 [[145.35040389  13.28286339]
  [ 13.28286339 149.39266173]]

 [[ 99.28714093  10.38288232]
  [ 10.38288232 160.18429122]]]




u_source:
[[91, 15], [117, 44], [83, 72]]

E_source:
 [[[161.   0.]
  [  0. 189.]]

 [[195.   0.]
  [  0. 174.]]

 [[128.   0.]
  [  0. 185.]]]
```