Topic No. 8

1. K-means
2. GMM
3. EM
4. DASH publishing in Linux
5. Preparing Proposal Presentation

Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where c_i is the centroid or medoid of cluster C_i)

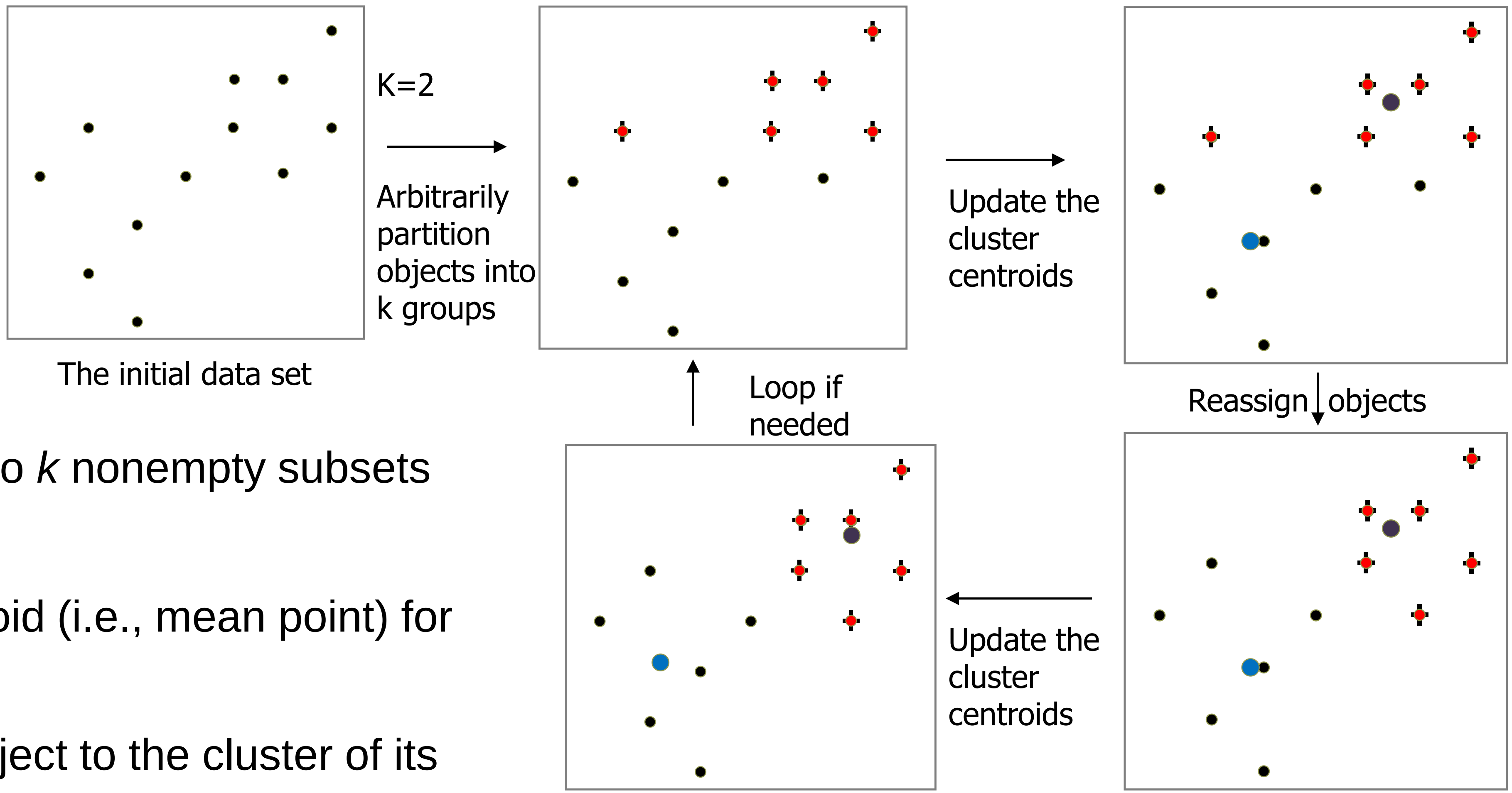
$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (p - c_i)^2$$

- Given *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when the assignment does not change

An Example of *K-Means* Clustering

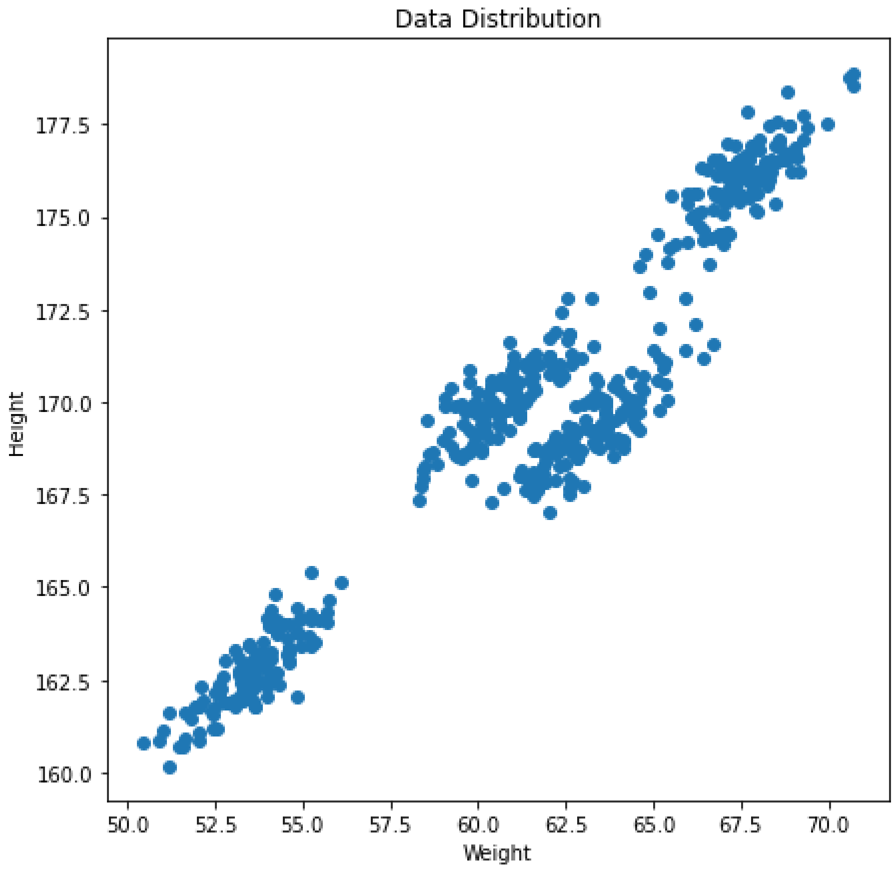


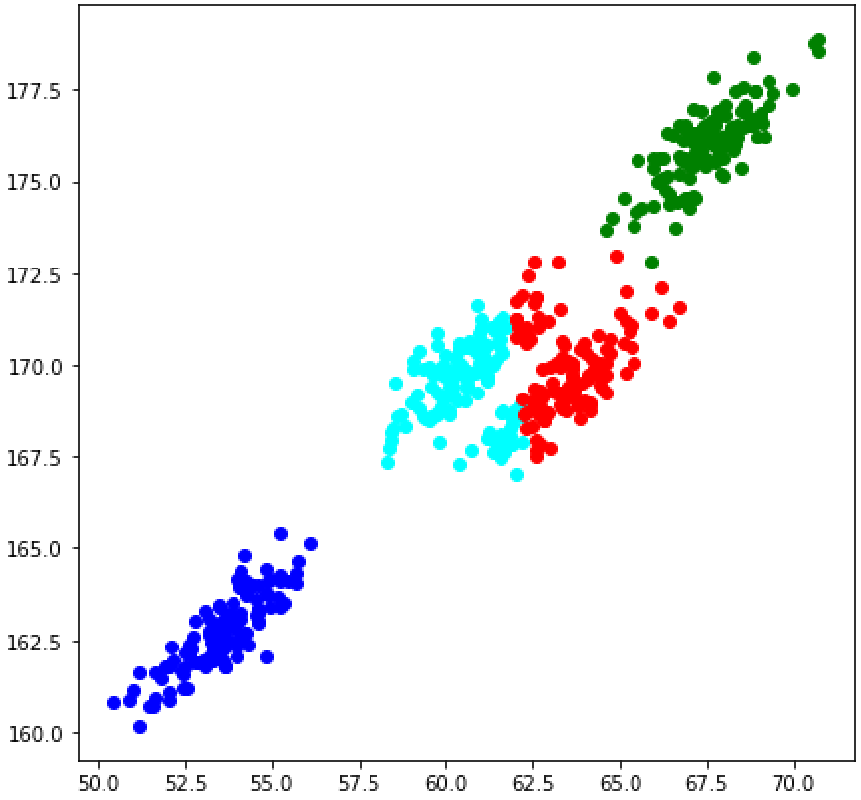
- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

What Is the Problem of the K-Means Method?

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data
- All the clusters created have a circular shape
  - Will not work when the distribution of points is ***not*** in a circular form

K-means


Data Distribution
177.5
175.0
172.5
170.0
167.5
165.0
162.5
160.0
Height
50.0
52.5
55.0
57.5
60.0
62.5
65.0
67.5
70.0
Weight

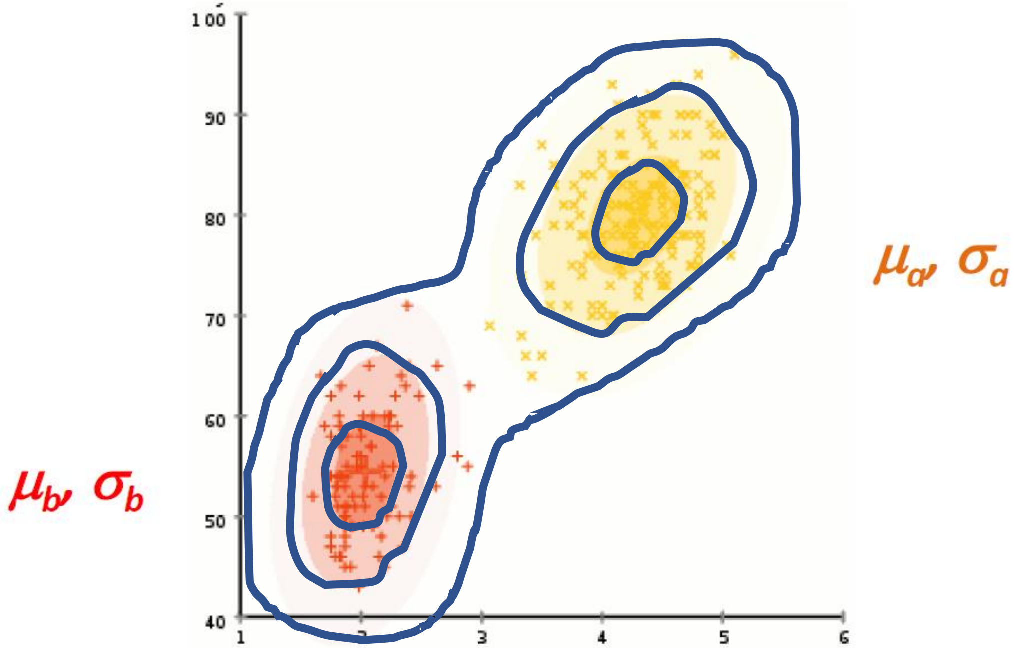

177.5
175.0
172.5
170.0
167.5
165.0
162.5
160.0
50.0
52.5
55.0
57.5
60.0
62.5
65.0
67.5
70.0

Gaussian Mixture Model

- Assume data points are from several Gaussian distributions

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leqslant \pi_k \leqslant 1$$

- Not necessarily the same Gaussian
- For each data point, estimate the corresponding Gaussian
- How?



Gaussian Mixture Model

- likelihood

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

- Log likelihood

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

Summation inside of log → makes it difficult to solve

EM — Expectation Maximization

1. EM
   - A popular iterative refinement algorithm
   - Assign each object to a cluster according to **a weight (prob. distribution).**
   - New means are computed based on weighted measures.
2. General idea
   - Starts with an initial estimate (“guess”) of **the parameter vector.**
   - Iteratively rescores the patterns against the mixture density produced by the parameter vector.
   - The rescored objects are used to update the parameter estimates.
   - Objects belong to the same cluster, if they are placed by their scores in a particular component distribution.

EM — Expectation Maximization

E-step

Gaussian distribution

Data points

mean

covariance

mixing probability

$$\gamma_j(\mathbf{x}_n) = \frac{\pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_k \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

ownership weight

each Gaussian model

M-step

$$\boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)}$$

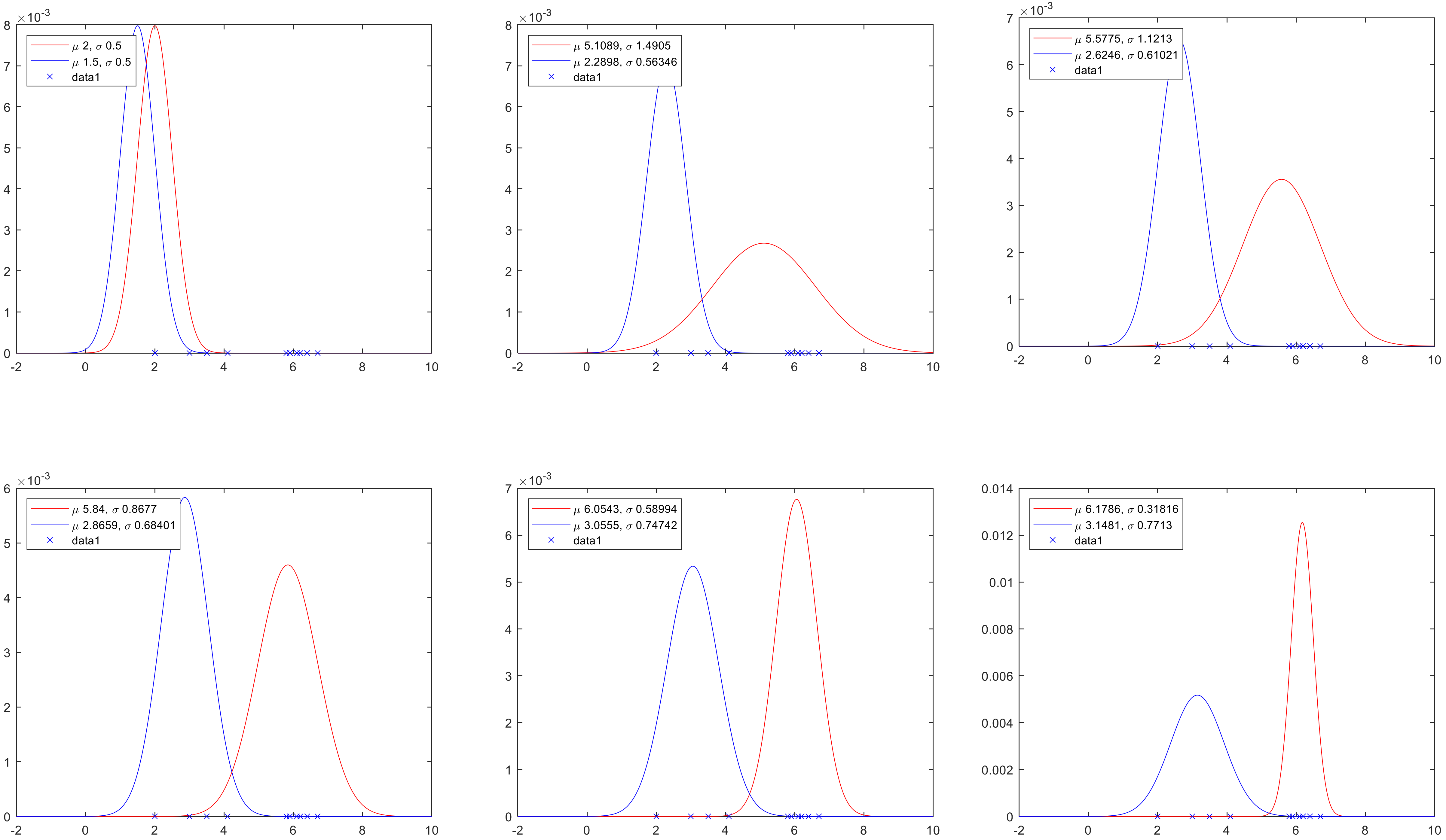
mean

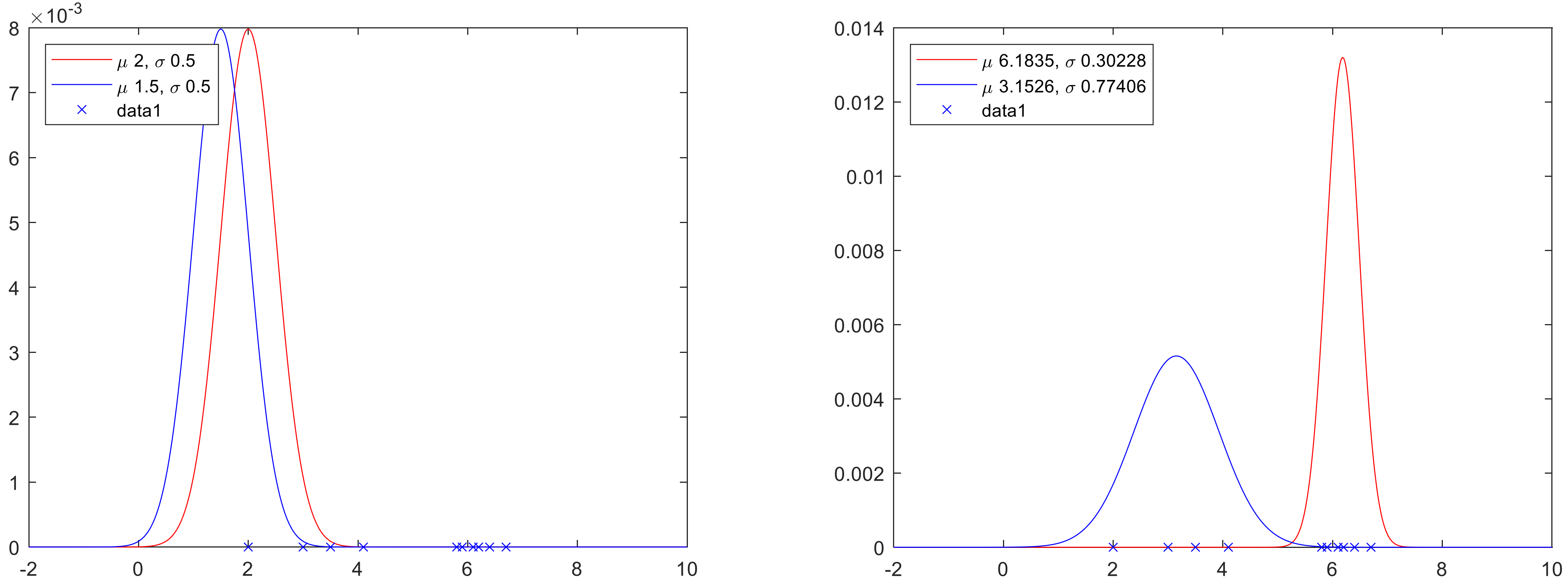
$$\boldsymbol{\Sigma}_j = \frac{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)(\mathbf{x}_n - \boldsymbol{\mu}_j)(\mathbf{x}_n - \boldsymbol{\mu}_j)^{\top}}{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)}$$

covariance

$$\pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)$$

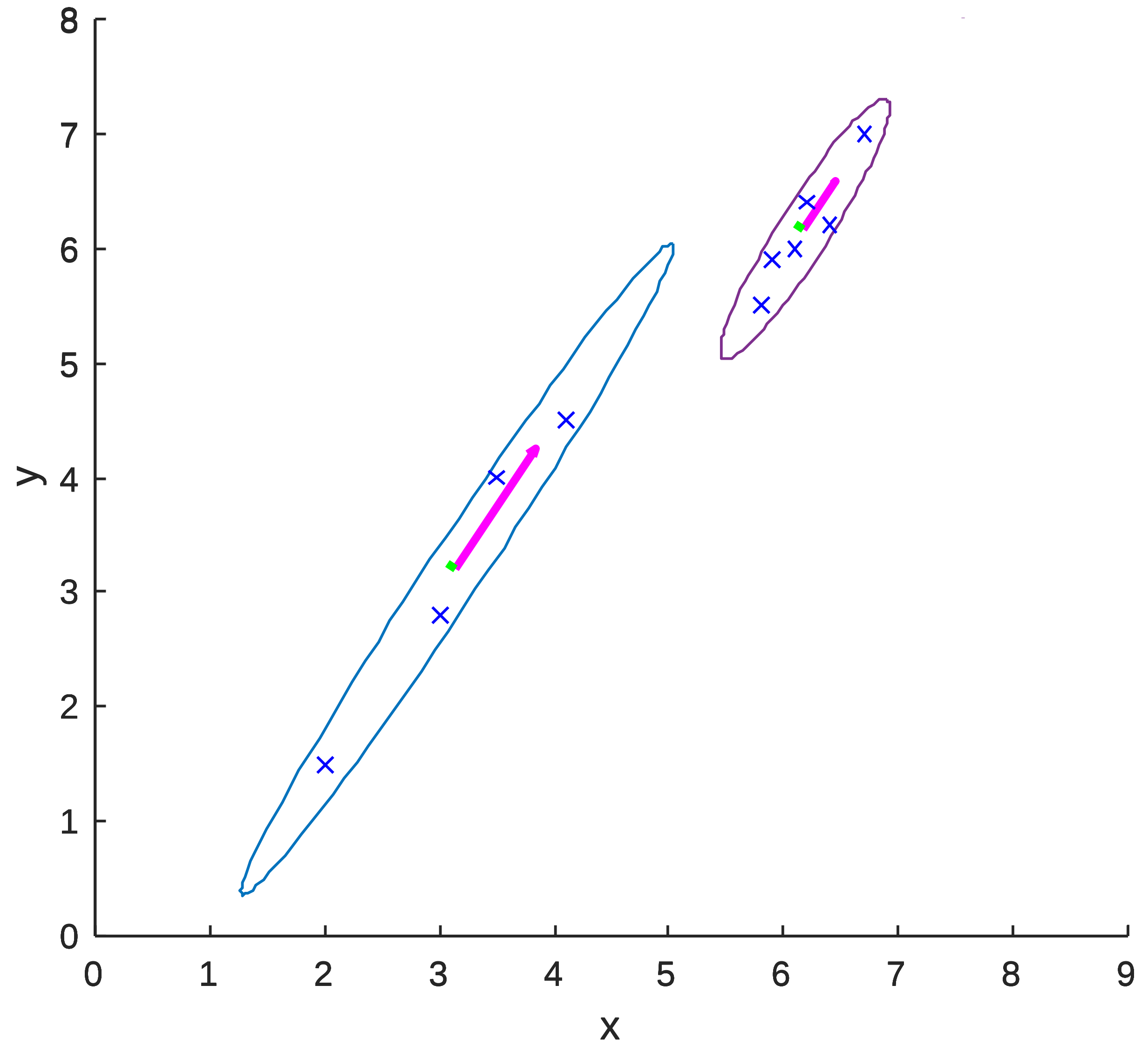
mixing probability


μ 2, σ 0.5
μ 1.5, σ 0.5
data1
μ 5.1089, σ 1.4905
μ 2.2898, σ 0.56346
data1
μ 5.5775, σ 1.1213
μ 2.6246, σ 0.61021
data1
μ 5.84, σ 0.8677
μ 2.8659, σ 0.68401
data1
μ 6.0543, σ 0.58994
μ 3.0555, σ 0.74742
data1
μ 6.1786, σ 0.31816
μ 3.1481, σ 0.7713
data1

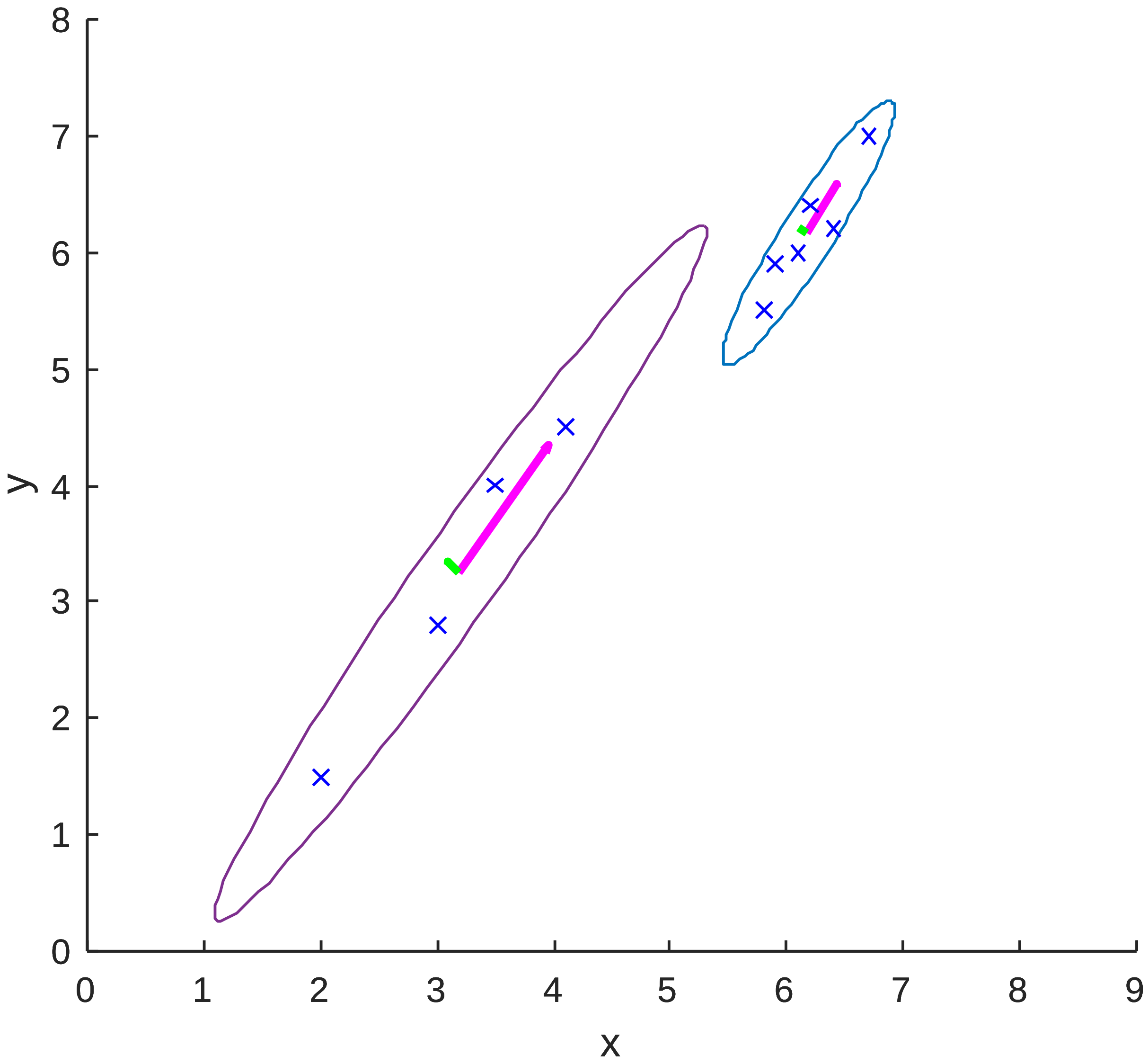


$\times 10^{-3}$
8
7
6
5
4
3
2
1
0
-2
0
2
4
6
8
10
μ 2, σ 0.5
μ 1.5, σ 0.5
data1
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
μ 6.1835, σ 0.30228
μ 3.1526, σ 0.77406
data1

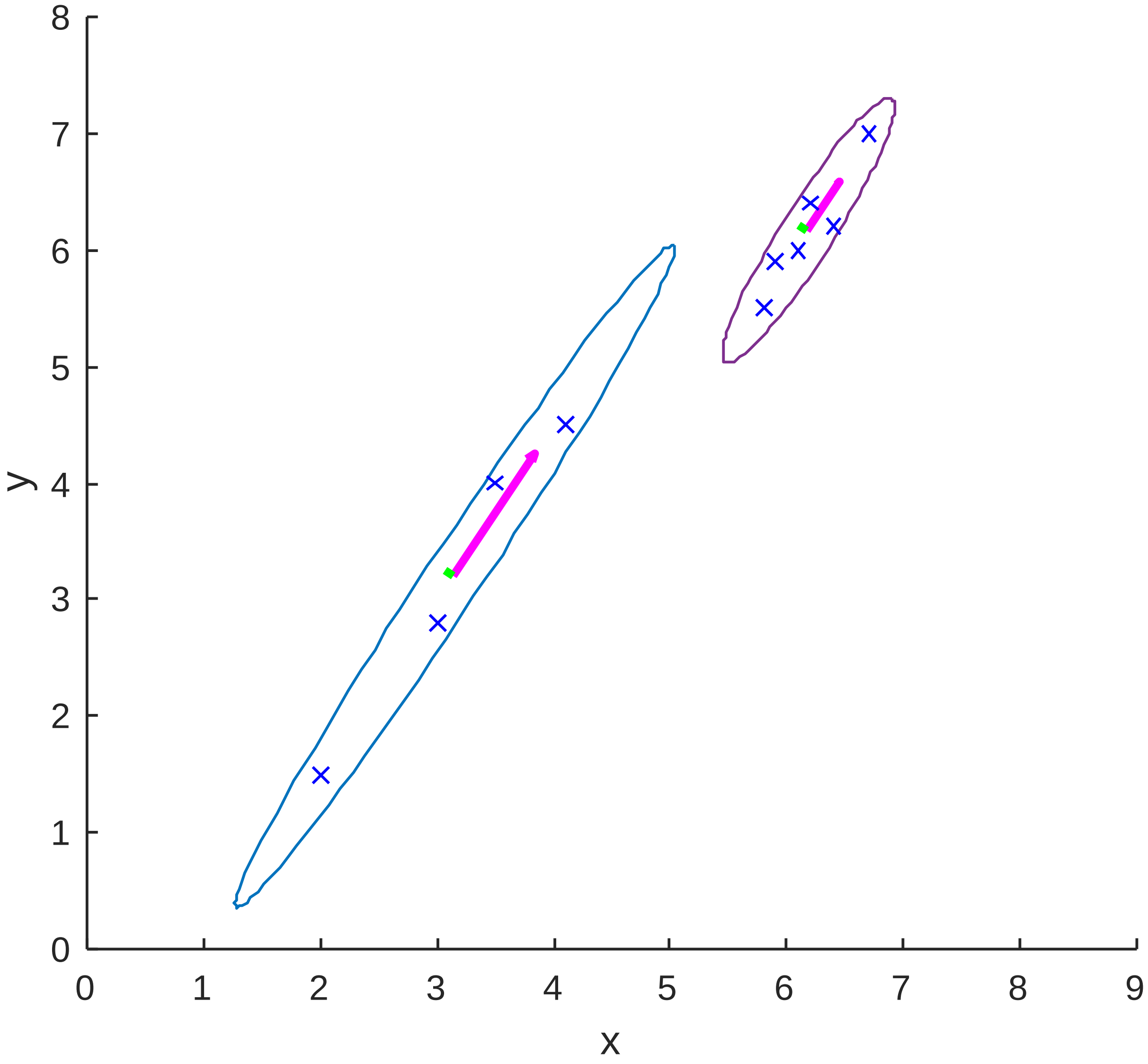
Example of 2D Gaussian Mixture using EM

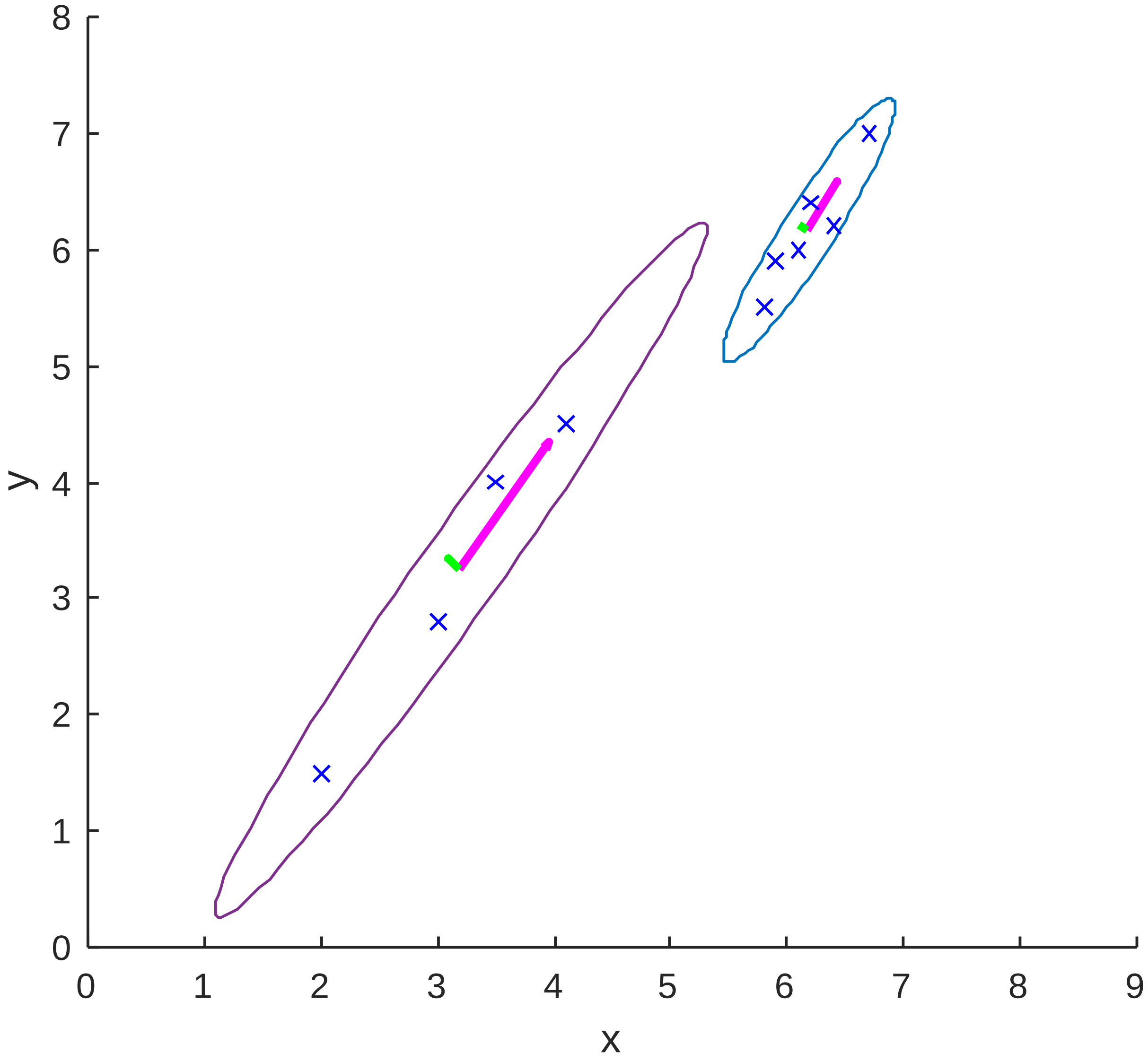


Example of 2D Gaussian Mixture using EM

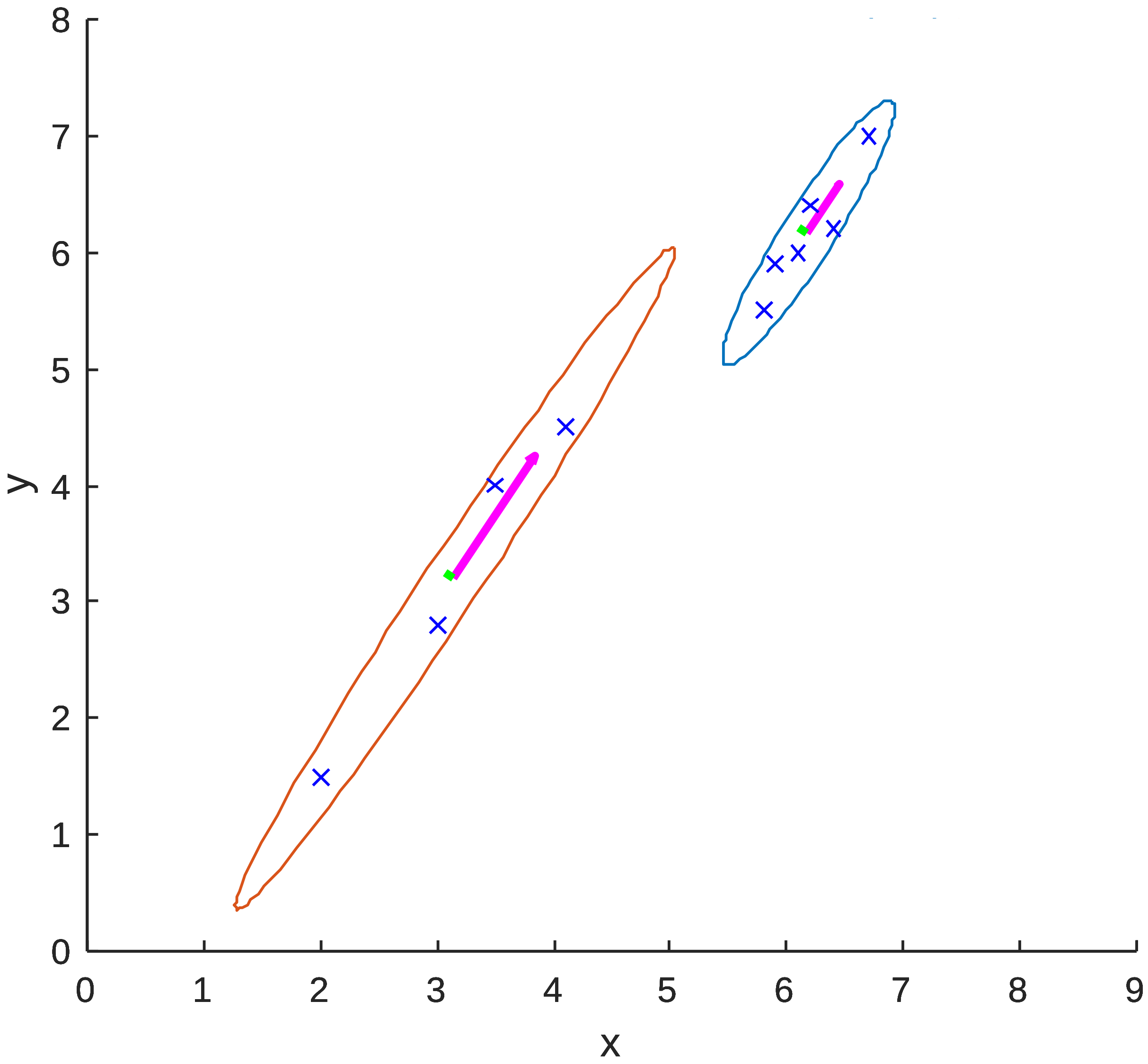


Example of 2D Gaussian Mixture using EM

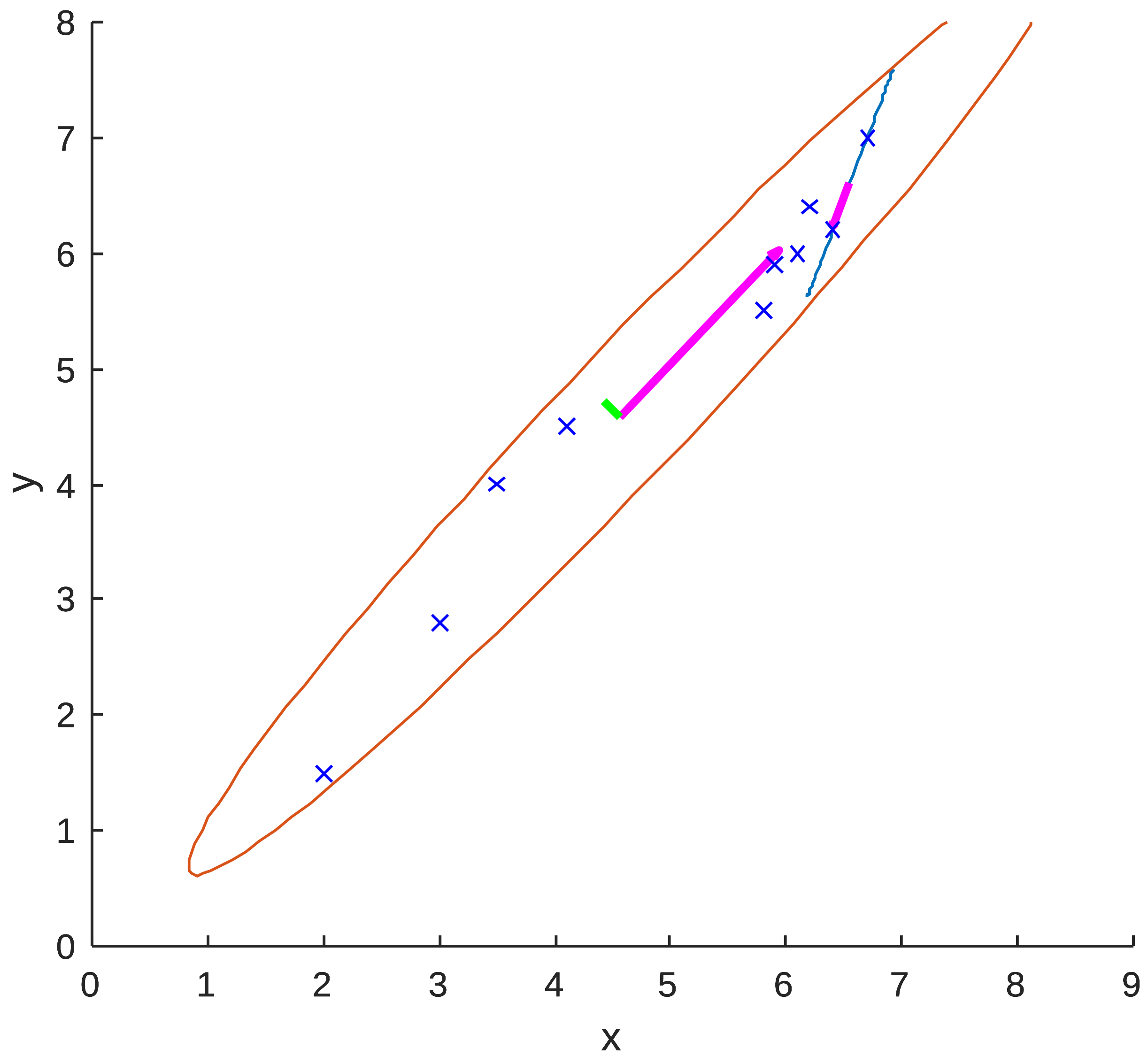


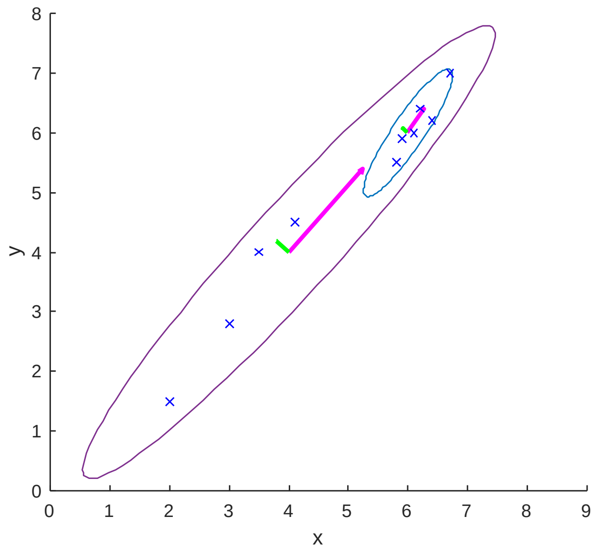


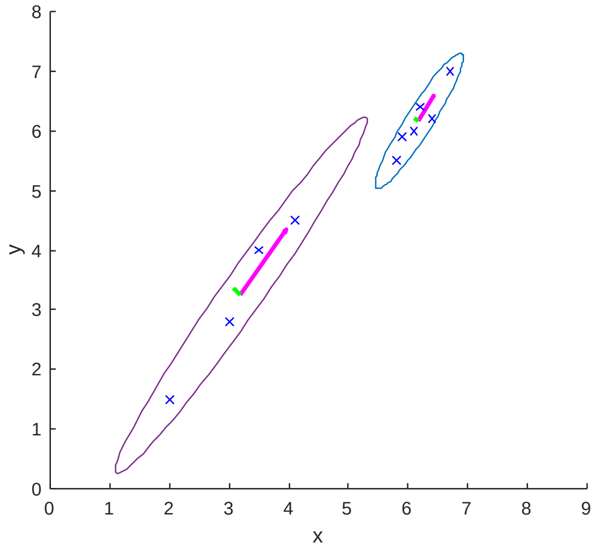
Example of 2D Gaussian Mixture using EM

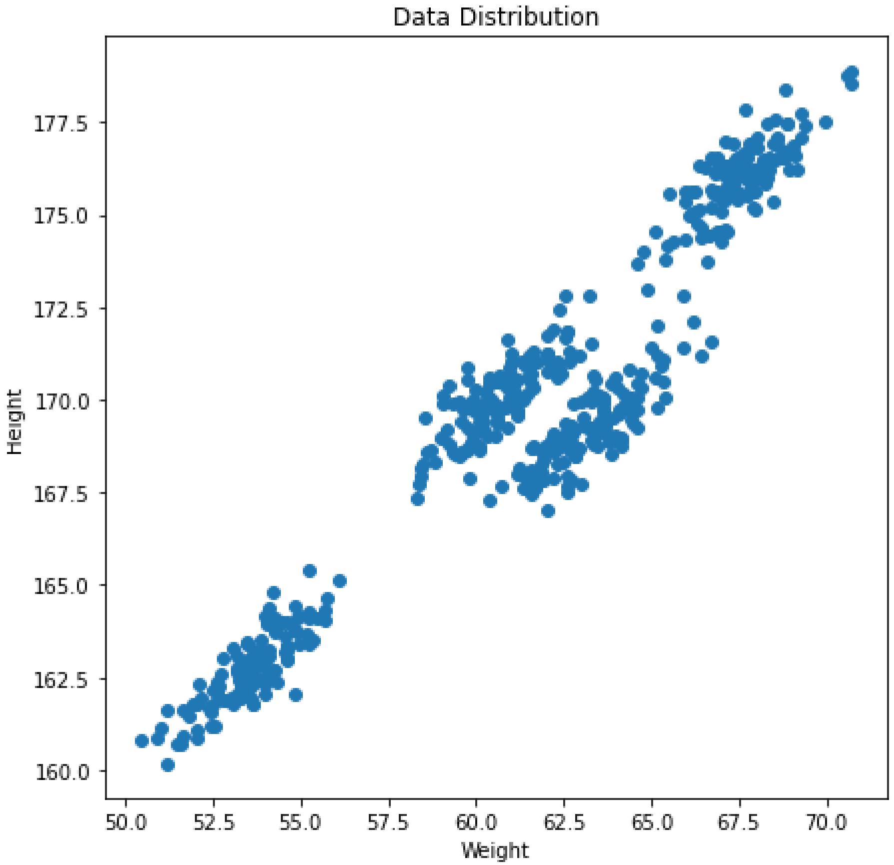


Example of 2D Gaussian Mixture using EM

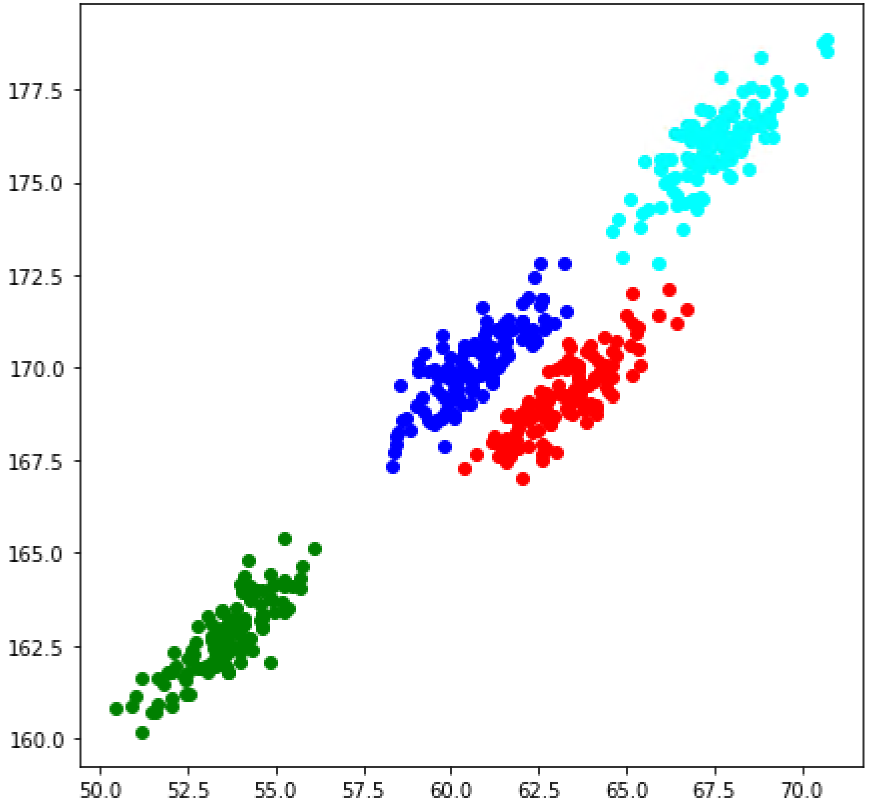






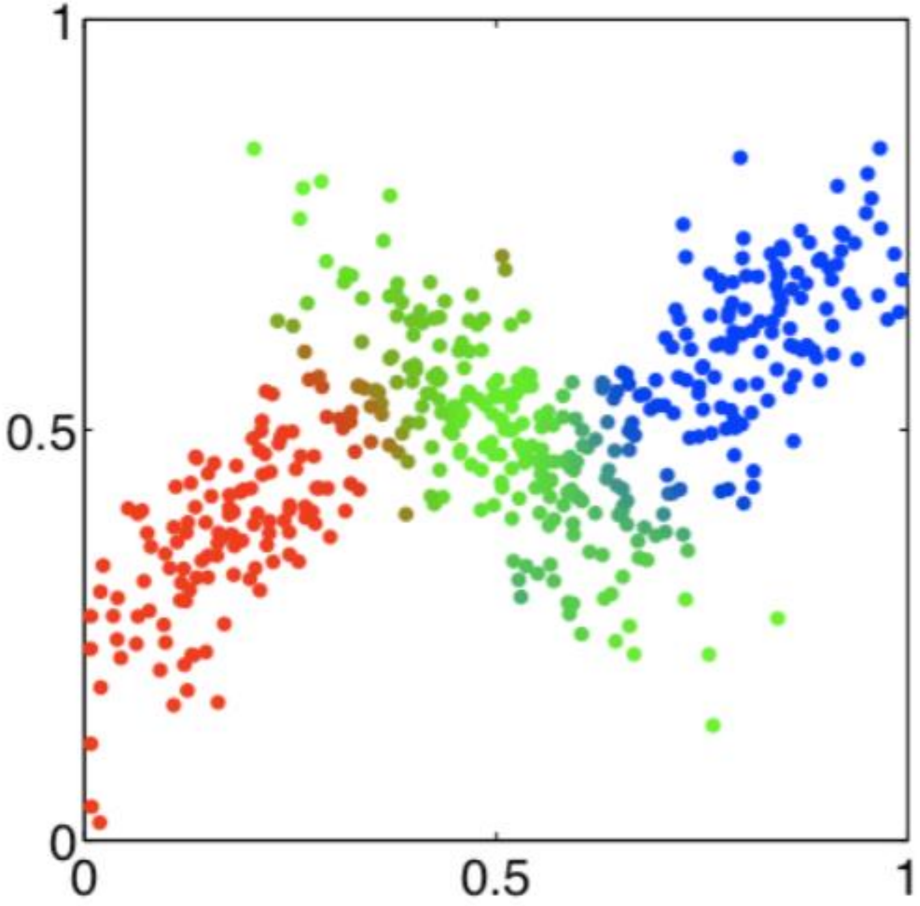


Data Distribution
Height
Weight
177.5
175.0
172.5
170.0
167.5
165.0
162.5
160.0
50.0
52.5
55.0
57.5
60.0
62.5
65.0
67.5
70.0



177.5
175.0
172.5
170.0
167.5
165.0
162.5
160.0
50.0
52.5
55.0
57.5
60.0
62.5
65.0
67.5
70.0

EM estimates weighted contributions of data point to the clusters



http://www.cse.psu.edu/~rtc12/CSE586/lectures/EMLectureFeb3.pdf

EM and *k*-means

k-means clustering is a special case of GMM-EM

1. Constant mixing probability $\pi_j = \frac{1}{k}$
2. Constant covariance matrix $\Sigma = I$
3. Only update mean values

```
from sklearn.mixture import GaussianMixture
import matplotlib.pyplot as plt

data = pd.read_csv('GMM.csv')
gmm = GaussianMixture(n_components=4)
gmm.fit(data)

#predictions from gmm
labels = gmm.predict(data)
frame = pd.DataFrame(data)
frame['cluster'] = labels
frame.columns = ['Weight', 'Height', 'cluster']

plt.figure(figsize=(7,7))
color=['blue', 'green', 'cyan', 'red']
for k in range(0,4):
   data = frame[frame["cluster"]==k]
   plt.scatter(data["Weight"],data["Height"],c=color[k])
plt.show()
```

https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/