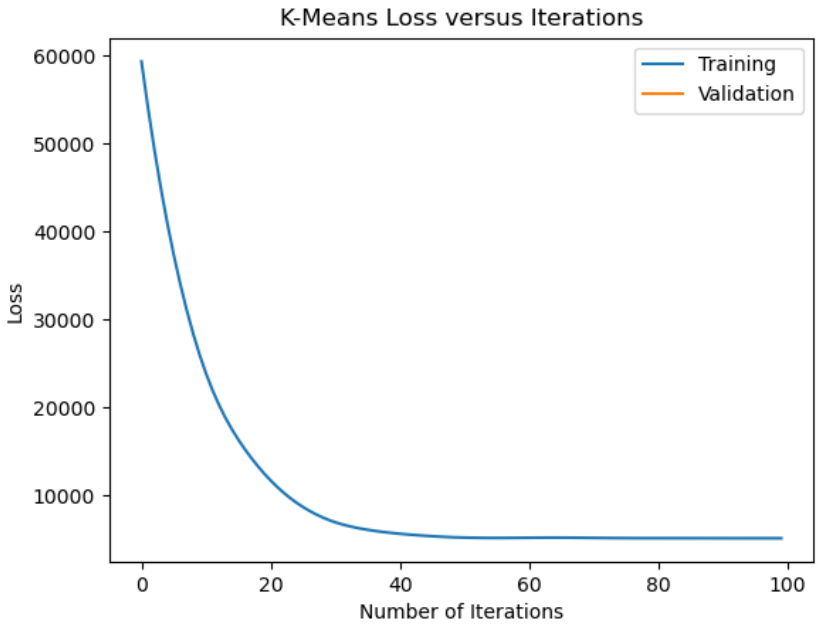
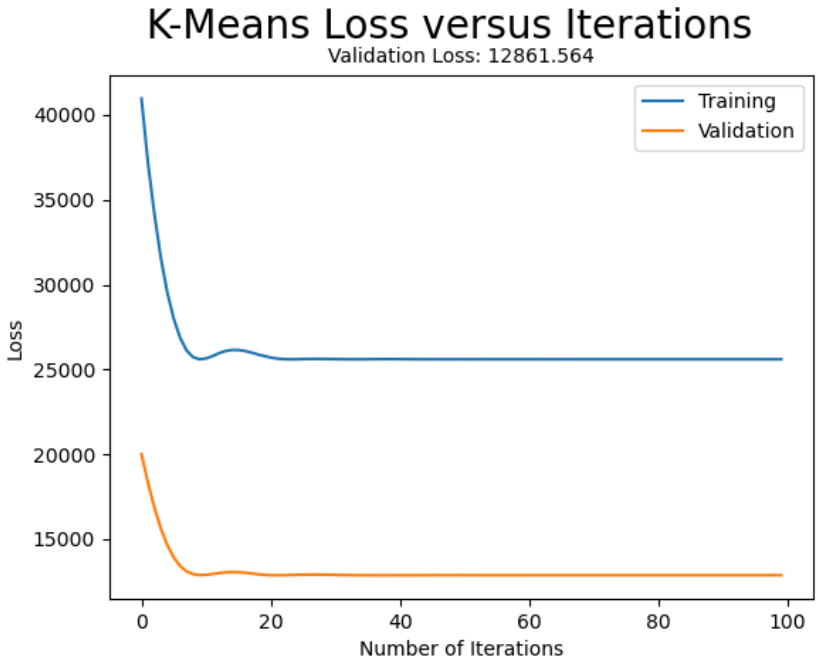
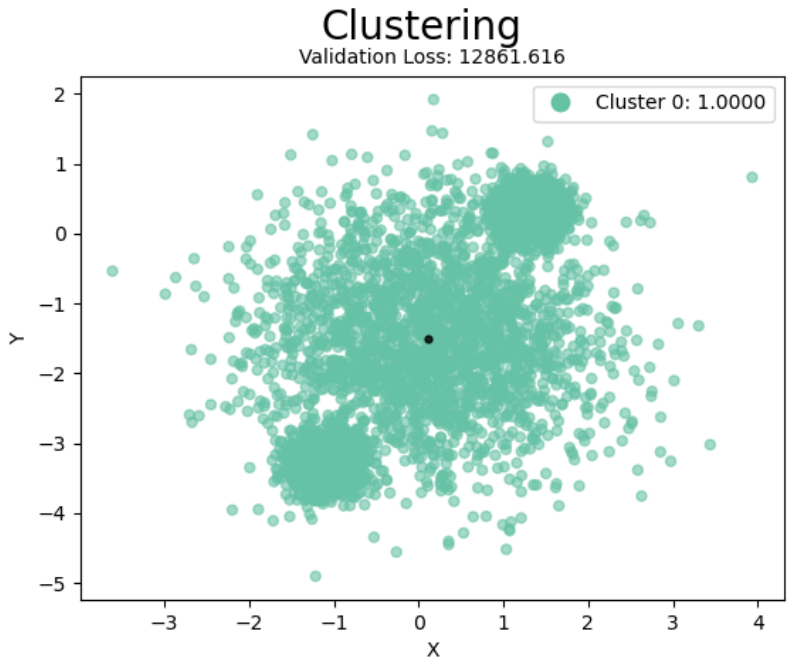
Part 1 – K-means

1.1: Learning K-means

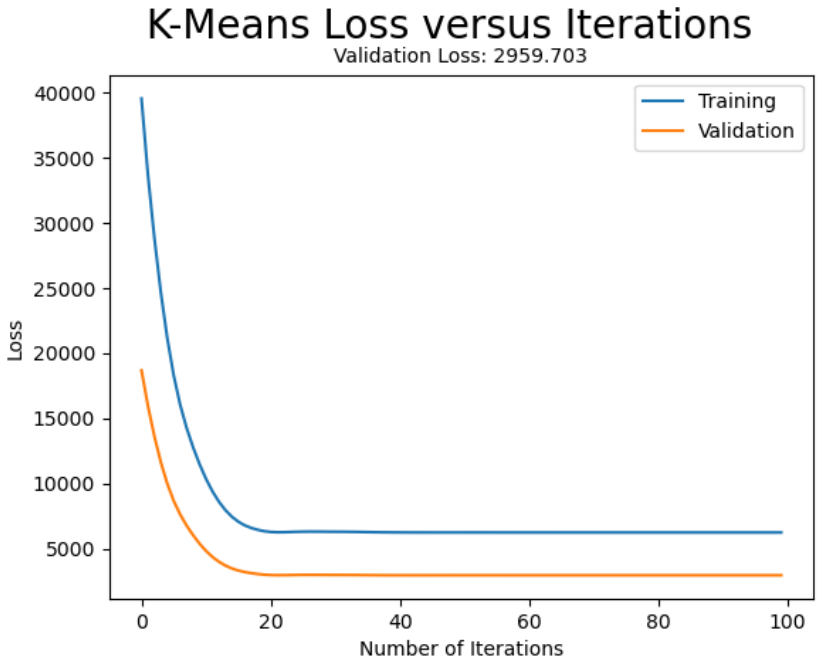


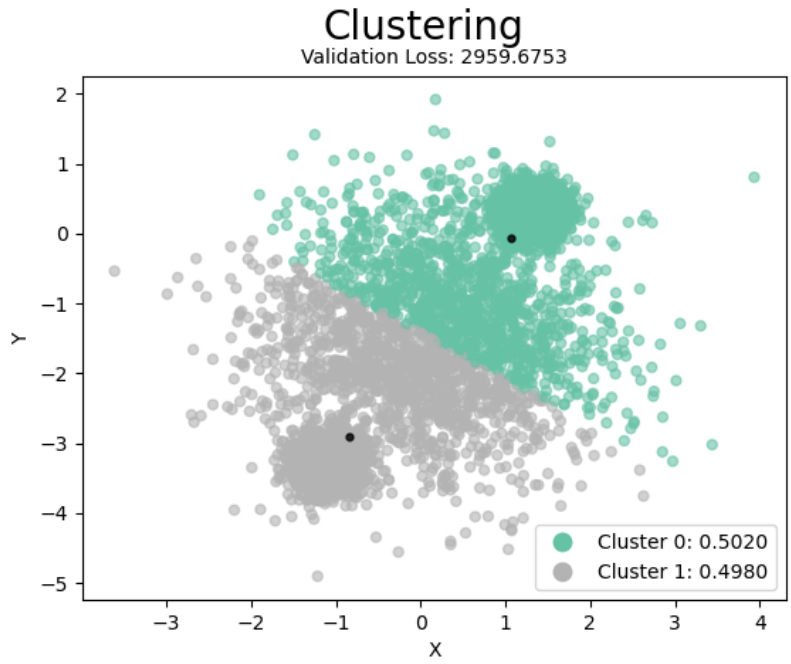
K=1



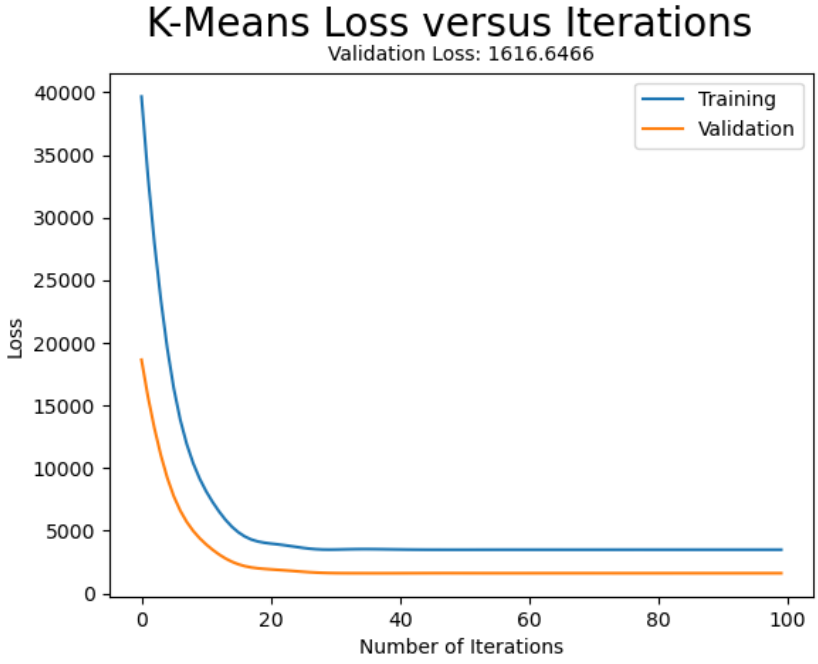

Clustering
Validation Loss: 12861.616
Cluster 0: 1.0000
Y
X

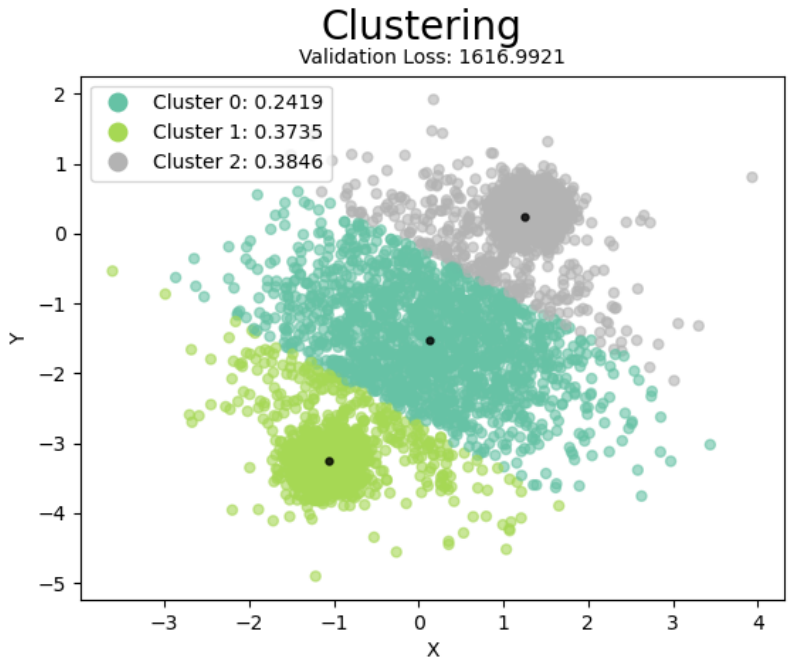
K=2


K-Means Loss versus Iterations
Validation Loss: 2959.703
Training
Validation
Loss
Number of Iterations

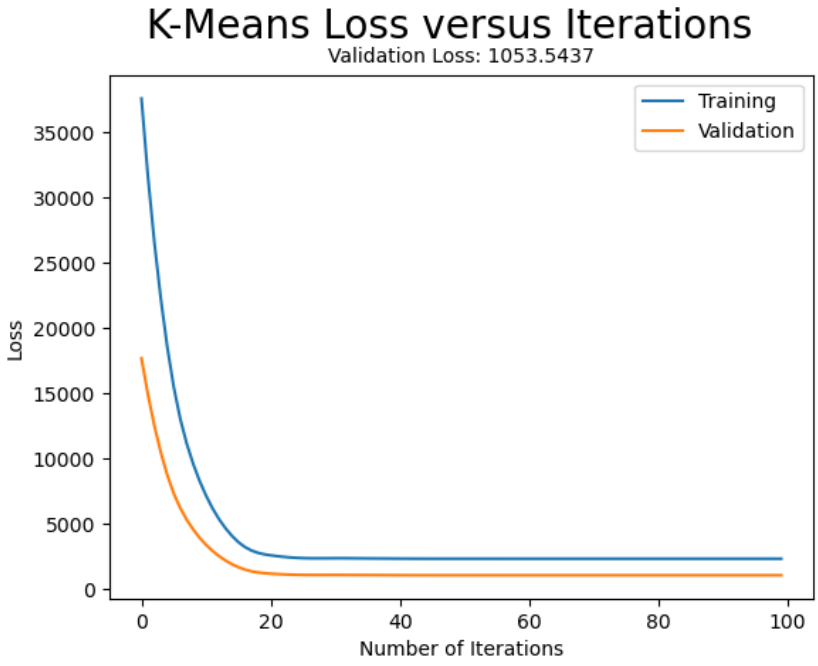

Clustering
Validation Loss: 2959.6753
Cluster 0: 0.5020
Cluster 1: 0.4980
X
Y

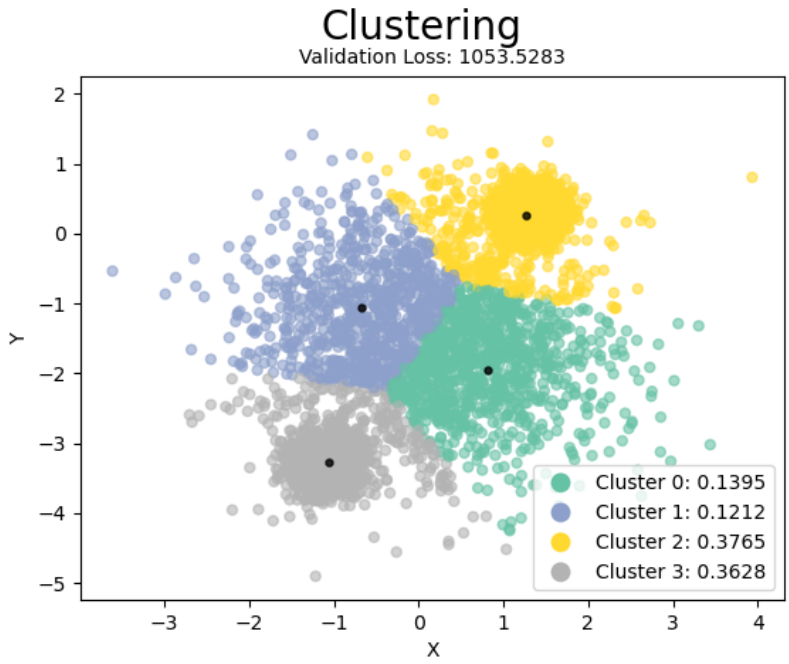
K=3


K-Means Loss versus Iterations
Validation Loss: 1616.6466
Training
Validation
Loss
Number of Iterations

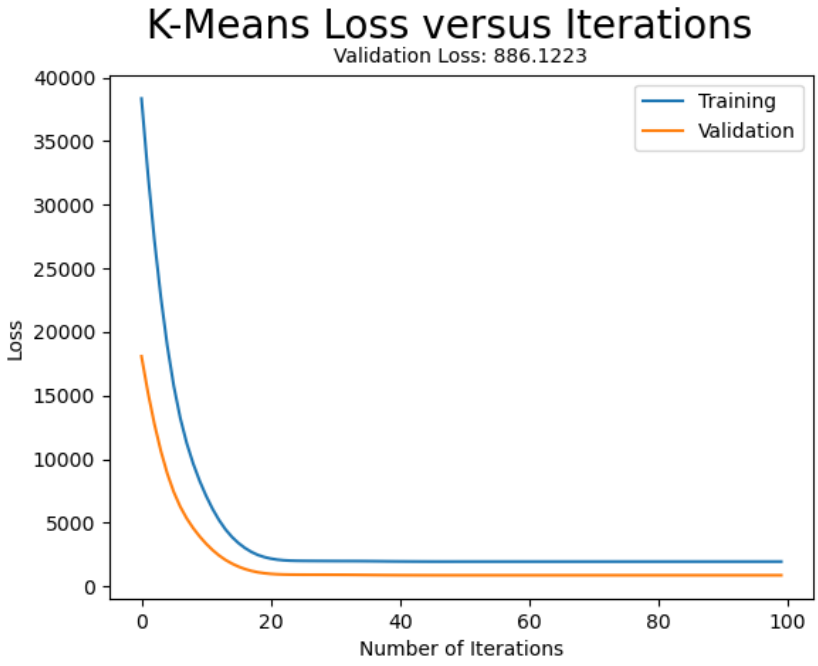

Clustering
Validation Loss: 1616.9921
Cluster 0: 0.2419
Cluster 1: 0.3735
Cluster 2: 0.3846
X
Y

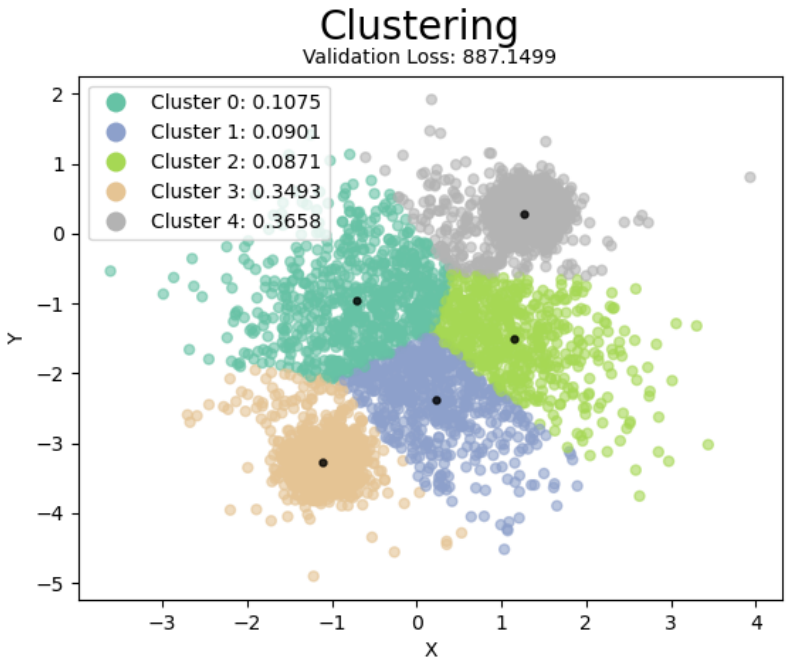
K=4


K-Means Loss versus Iterations
Validation Loss: 1053.5437
Training
Validation
Loss
Number of Iterations


Clustering
Validation Loss: 1053.5283
Cluster 0: 0.1395
Cluster 1: 0.1212
Cluster 2: 0.3765
Cluster 3: 0.3628
X
Y

K=5


K-Means Loss versus Iterations
Validation Loss: 886.1223
Training
Validation
Loss
Number of Iterations



In looking at the scatter plots, using 3 clusters is best number to use. This is because there is a relatively even distribution of points belonging to each cluster, however, when using 4 or 5 clusters there is significant imbalance between the distribution. Therefore, k=3 is the best choice of gaining the best representation of all of the data.

Part 2 – Mixtures of Gaussians

2.1: The Gaussian Cluster Mode

```
def log_GaussPDF(X, mu, sigma):
    # Inputs
    # X: N X D
    # mu: K X D
    # sigma: K X 1

    # Outputs:
    # log Gaussian PDF N X K

    distance = distanceFunc(X,mu)
    exp = distance / (2*tf.squeeze(sigma))
    coef = -0.5*tf.to_float(tf.rank(X))*tf.log(2*np.pi*tf.squeeze(sigma))
    return coef - exp
```

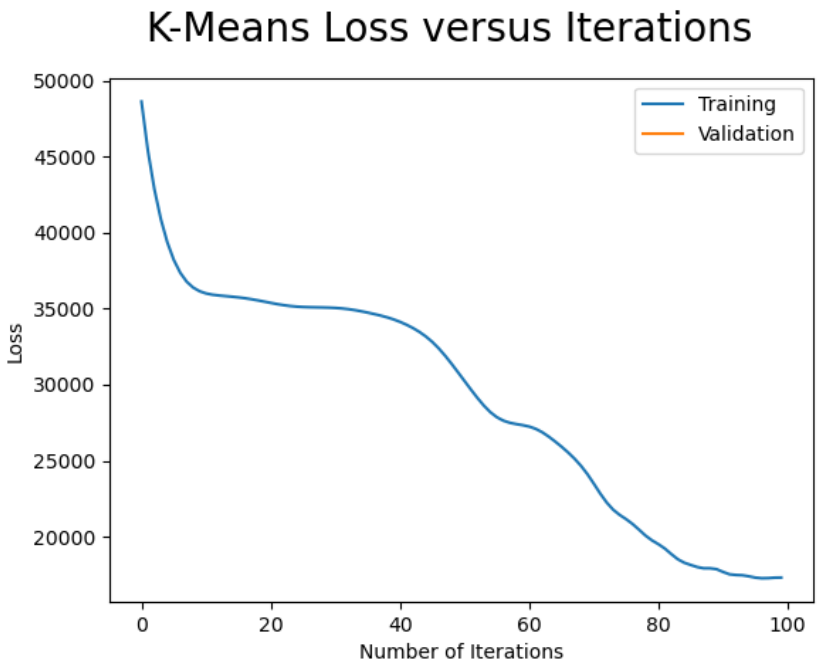
```
def log_posterior(log_PDF, log_pi):
    # Input
    # log_PDF: log Gaussian PDF N X K
    # log_pi: K X 1
```

```
    # Outputs
    # log_post: N X K

    log_probability = tf.squeeze(log_pi) + log_PDF
    log_sum = hlp.reduce_logsumexp(log_probability + tf.squeeze(log_pi),
keep_dims=True)
    return log_probability - log_sum
```

It is important to use the log-sum-exp function because the posterior probability is obtained using Bayes Rule, and these probabilities therefore need to be normalized. The denominator in the equation used is equivalent to the marginalization of P(x) and ensures the sum across the function is equal to 1, therefore normalized. In los_posterior, the inputs are in the log domain which then need to be normalized in the original domain of the inputs that are used. The result is transferred back to the original domain by exponentiating these values and summing across the axis to create a Nx1 array, which is logged again to bring the result back to the log domain.

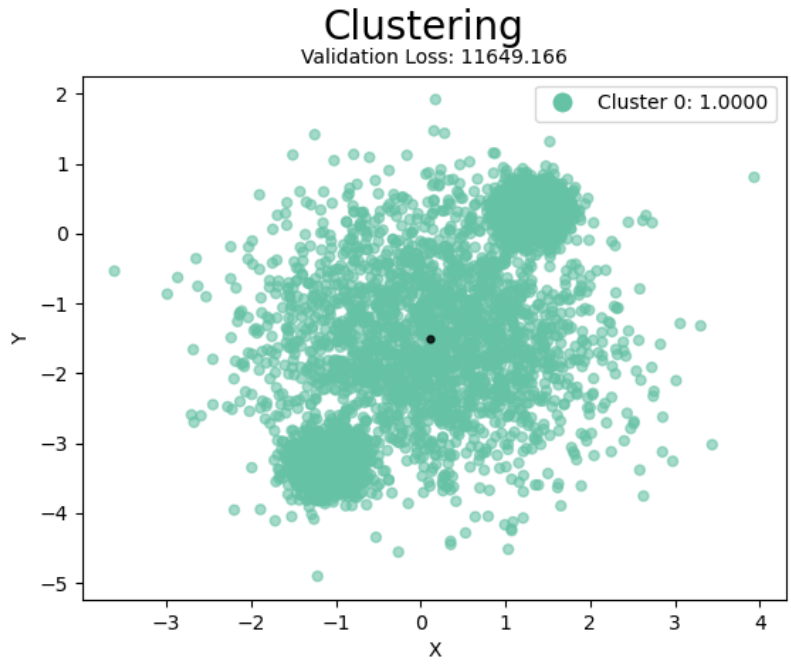
2.2: Learning the MoG



Cluster	1	2	3
π	-1.0966995	-1.1003758	-1.0987651
σ	0.0316915	0.04316588	1.0054854
μ	-1.0991392 -3.299245	1.2996538 0.31289315	0.10272314 -1.6202357

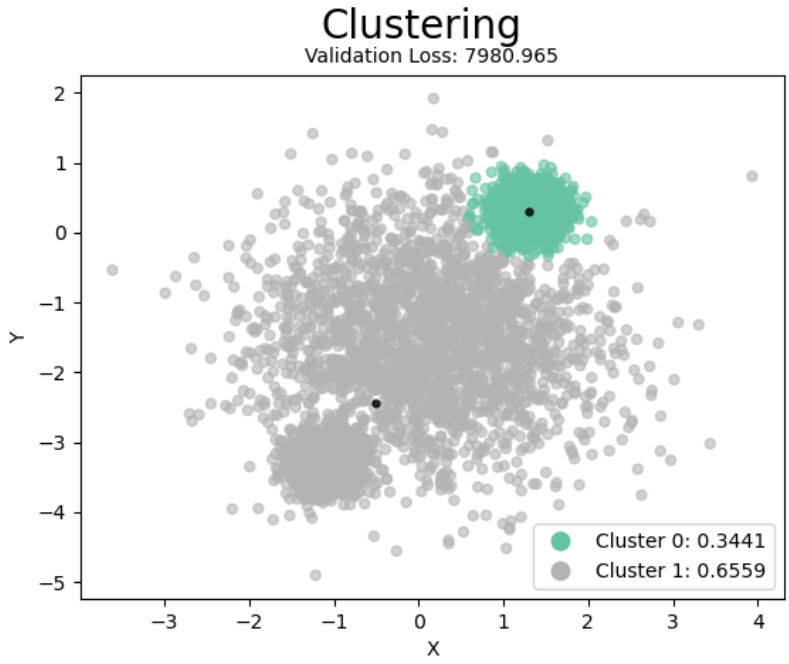
K=1


K-Means Loss versus Iterations
Validation Loss: 11649.16
Training
Validation
30000
25000
20000
15000
Loss
0
20
40
60
80
100
Number of Iterations

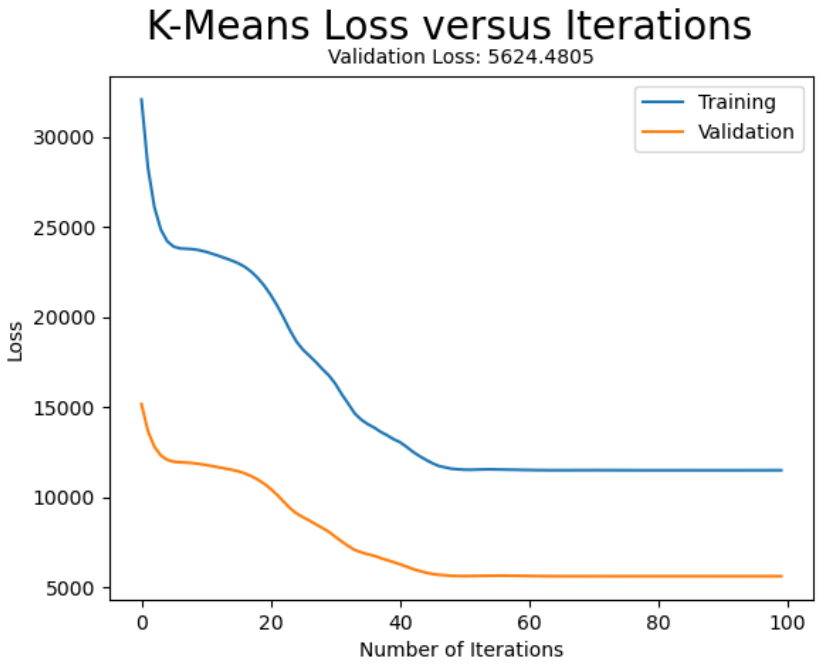

Clustering
Validation Loss: 11649.166
Cluster 0: 1.0000
2
1
0
−1
−2
−3
−4
−5
Y
−3
−2
−1
0
1
2
3
4
X

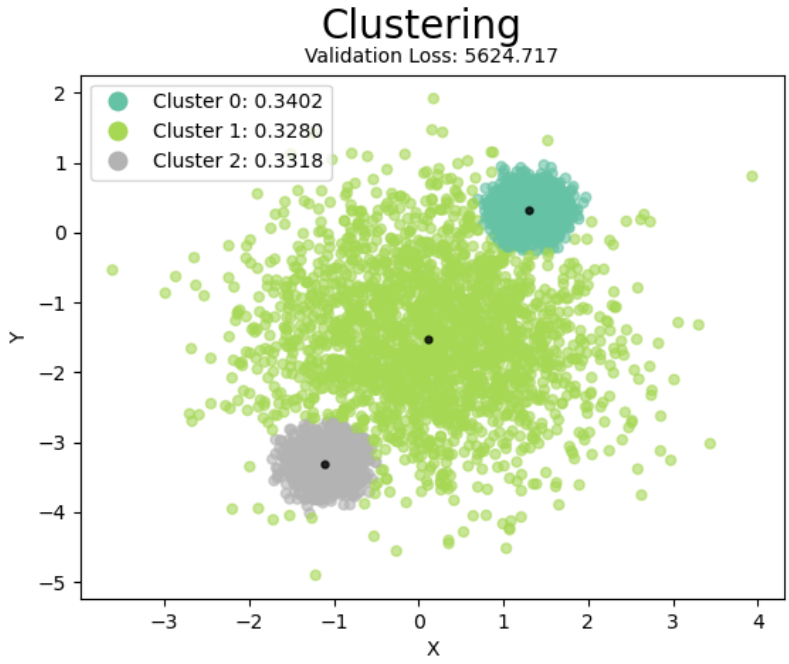
K=2


K-Means Loss versus Iterations
Validation Loss: 7980.8
Training
Validation
Loss
30000
25000
20000
15000
10000
0
20
40
60
80
100
Number of Iterations


Clustering
Validation Loss: 7980.965
Y
X
Cluster 0: 0.3441
Cluster 1: 0.6559

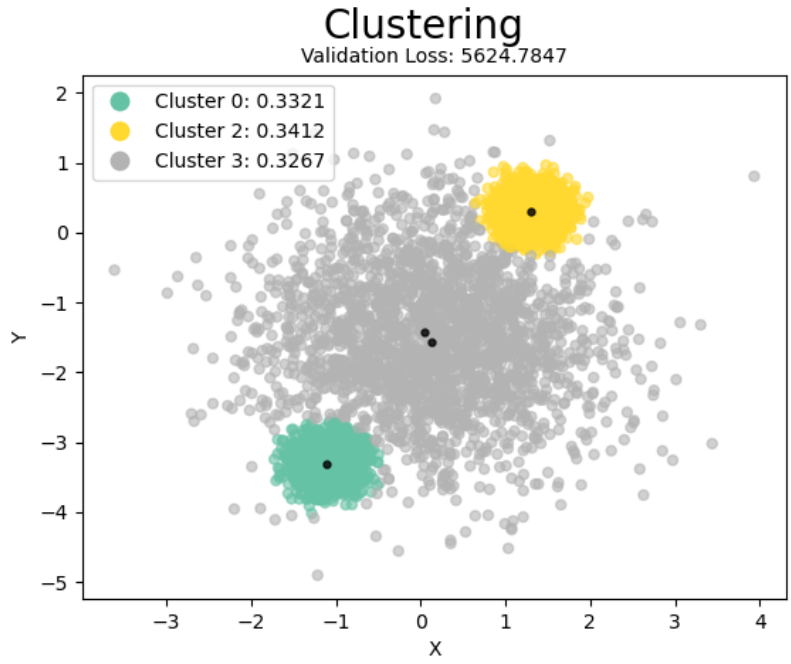
K=3


K-Means Loss versus Iterations
Validation Loss: 5624.4805
Training
Validation
Loss
Number of Iterations

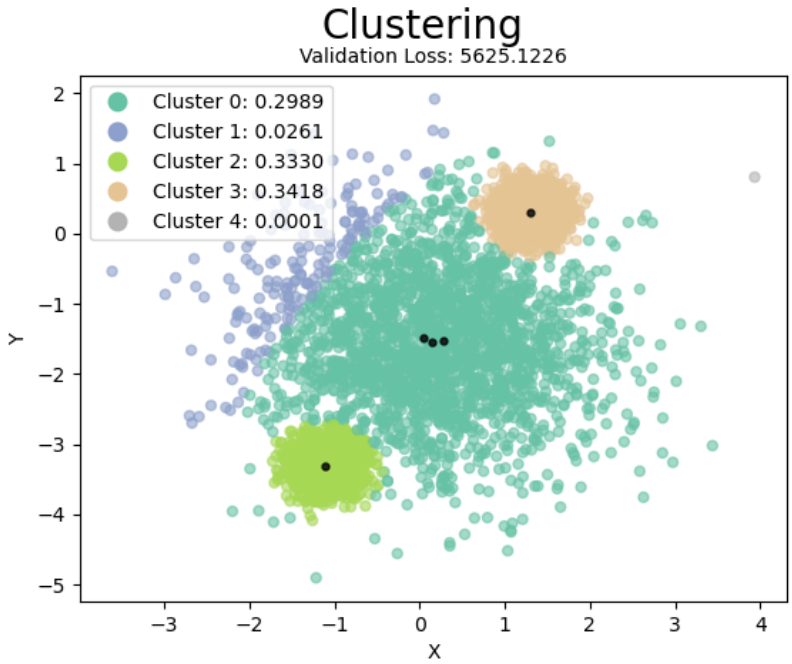

Clustering
Validation Loss: 5624.717
Cluster 0: 0.3402
Cluster 1: 0.3280
Cluster 2: 0.3318
X
Y

K=4


K-Means Loss versus Iterations
Validation Loss: 5624.855
Training
Validation
Loss
Number of Iterations
30000
25000
20000
15000
10000
5000
0
20
40
60
80
100


Clustering
Validation Loss: 5624.7847
Cluster 0: 0.3321
Cluster 2: 0.3412
Cluster 3: 0.3267
X
Y
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1
2
3
4

K=5



Based on the validation loss, using K=3 is the best value with a validation loss of 5624.48 which is a slight advantage over both K=4 and K=5, however, there is very little to separate the performance of these three parameters.

Using data100D.npy:

	K=5	K=10	K=15	K=20	K=30
K-means	122359.73	70837.46	70734.3	68188.35	67870.34
MoG	21822.365	21401.67	21364.451	21356.42	22083.299

As shown by the validation loss for MoG, there appears to be somewhere between 15 and 20 clusters in the dataset as evident by their validation losses. The validation loss begins to increase after this again which indicates the number of clusters lies somewhere in that range. Using K-means to conclude the number of clusters is ineffective because increasing the number of clusters used will always cause the validation loss to decrease.