# ECE 421 Assignment 3

Jerry He 1003979180
Contribution: 33%
Calvin Ma 1003803805
Contribution: 33%
Eric Li 1004015852
Contribution: 33%

March 27, 2020

Jerry He 1003979180
Calvin Ma 1003803805
Eric Li 1004015852

ECE 421
March 27, 2020

# Assignment 3

## 1 K-means

In this report, matrices will be denoted with bolded symbols ($\boldsymbol{X}$), vectors will be denoted with underlined symbols ($\underline{y}$), and scalars will be denoted with no accent ($z$).

### 1.1 Learning K-means

K-means clustering is an unsupervised learning algorithm that groups $D$-dimensional data points into $K$ clusters, each with a "cluster center". In this section, we use $\boldsymbol{\mu}$ to represent the $K \times D$ matrix of cluster centers, $\underline{\mu_k}$ to represent the $k^{\text{th}}$ cluster center in $\boldsymbol{\mu}$, $\boldsymbol{X}$ to represent the $N \times D$ matrix with $N$ $D$-dimensional data points, and $\underline{x_n}$ to represent the $n^{\text{th}}$ data point in $\boldsymbol{X}$.

1. Implementation
   We first implement the required *distanceFunc()*, which calculates pairwise distances between data points and cluster centers.

```python
def distanceFunc(X, MU):
    # Inputs
    # X: is an NxD matrix (N observations and D dimensions)
    # MU: is an KxD matrix (K means and D dimensions)
    # Outputs
    # pair_dist: is the squared pairwise distance matrix (NxK)
    K = MU.shape[0]
    N = X.shape[0]
    X = tf.tile(tf.expand_dims(X,1), [1,K,1])
    MU = tf.tile(tf.expand_dims(MU,0), [N,1,1])
    pair_dist = tf.reduce_sum(tf.math.squared_difference(X, MU), 2)
    return pair_dist
```

   We then define a loss function

$$\mathcal{L}(\boldsymbol{\mu}) = \sum_{n=1}^{N} \min_{1 \le k \le K} \left\| \underline{x_n} - \underline{\mu_k} \right\|_2^2 \tag{1}$$

   which could be easily evaluated using *distanceFunc()*.

   We want to minimize the loss using an Adam Optimizer, specified by assignment sheet. We also initialize $\boldsymbol{\mu}$ from the standard Gaussian distribution. These can be observed from our code.

```python
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
is_valid = 0

# Loading data
```

```
data = np.load('data2D.npy')
#data = np.load('data100D.npy')
[num_pts, dim] = np.shape(data)

# For Validation set
if is_valid:
  valid_batch = int(num_pts / 3.0)
  np.random.seed(45689)
  rnd_idx = np.arange(num_pts)
  np.random.shuffle(rnd_idx)
  val_data = data[rnd_idx[:valid_batch]]
  data = data[rnd_idx[valid_batch:]]

n_iterations = 150
K = 3
D = dim
N_train = num_pts
N_validate = num_pts
X_train_ = data
X_validate_ = data

graph = tf.Graph()
loss = np.zeros((n_iterations,1))

with graph.as_default():
     # Init
     MU = tf.Variable(tf.initializers.truncated_normal(mean=0, stddev
    =1)(shape=(K,D)))
     X_train = tf.placeholder(dtype='float32', shape=[N_train, D])
     X_validate = tf.placeholder(dtype='float32', shape=[N_validate, D
    ])

     # Calculate Training Loss
     pair_dist_train = distanceFunc(X_train, MU)
     L_train = tf.reduce_sum(tf.reduce_min(pair_dist_train, 1))

     optimizer = tf.train.AdamOptimizer(learning_rate=0.1, beta1=0.9,
    beta2=0.99, epsilon=1e-5).minimize(L_train)

     # Calculate Pair Distance for Validation Set
     pair_dist_validate = distanceFunc(X_validate, MU)

with tf.Session(graph=graph) as session:
     # Init
     tf.global_variables_initializer().run()
     tf.local_variables_initializer().run()

     # Train
     for n in range(n_iterations):
         optimizer_, MU_, pair_dist_validate_, L_validate_ = session.
    run([optimizer, MU, pair_dist_validate, L_validate],{X_train:
    X_train_, X_validate: X_validate_})
         loss[n] = L_validate_

 # Label Data
 min_dist = np.tile(np.amin(pair_dist_validate_, 1), [K,1]).transpose()
 y = np.where(pair_dist_validate_ == min_dist)[1]

 # Count Points in Cluster
 print('Cluster IDX      Portion')
 print('------------------------')
 for i in range(K):
     print('          ', i, '        ', np.count_nonzero(y==i)/float(
    N_validate))

 # Plot
 fig = plt.gcf()
 fig.set_size_inches(10, 4)

 plt.subplot(1,2,1)
 plt.plot(np.arange(n_iterations), loss)
 plt.xlabel('Number of Updates')
 plt.ylabel('Training Loss')
 plt.title('Training Curve for KMeans Clustering')

 plt.subplot(1,2,2)
 plt.scatter(X_validate_[:, 0], X_validate_[:, 1], c=y)
 plt.scatter(MU_[:, 0], MU_[:, 1], marker='x', c='k')
 plt.xlabel('x')
 plt.ylabel('y')
 plt.title('Clusters on Scatter Plot')
```

Using $K = 3$, the loss converges in around 50 iterations, as seen in Figure 1. For the following parts, we will run the algorithm for 150 iterations.

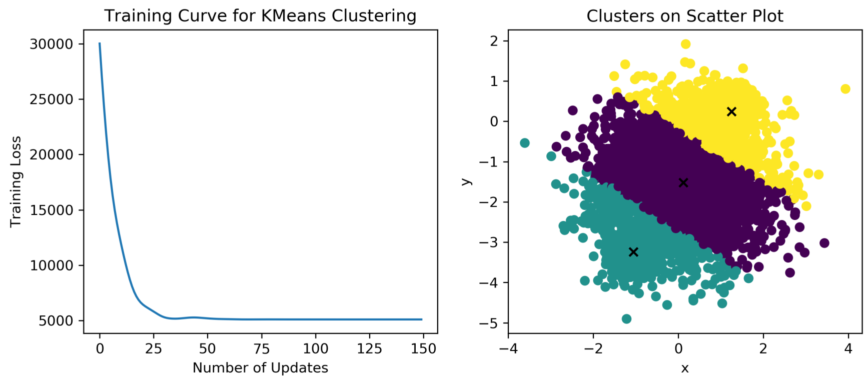
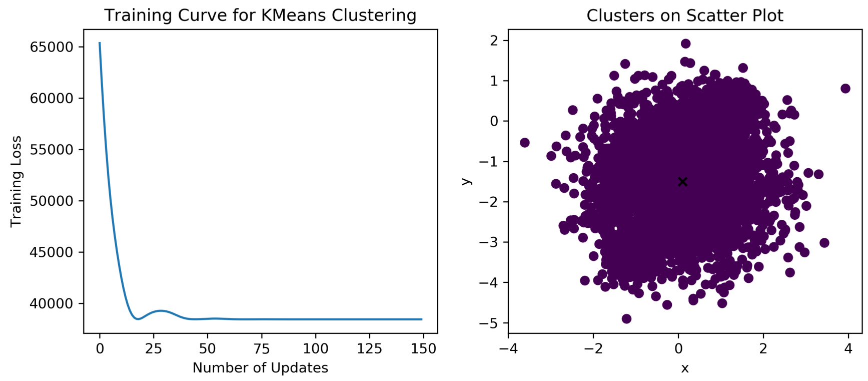
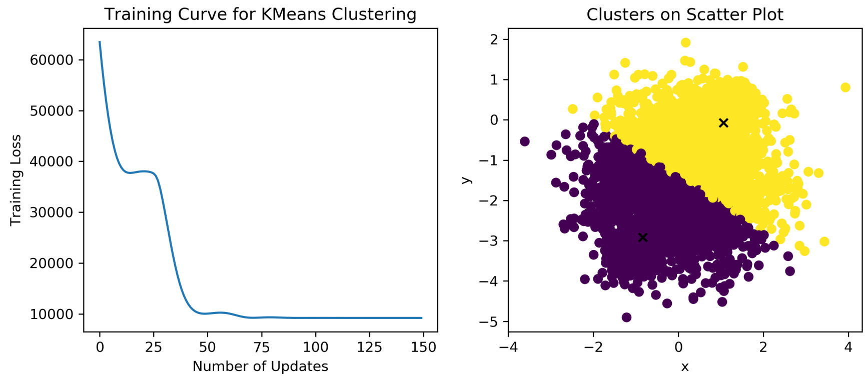


Figure 1: Training Curve and Clusters Plot with 3 Clusters

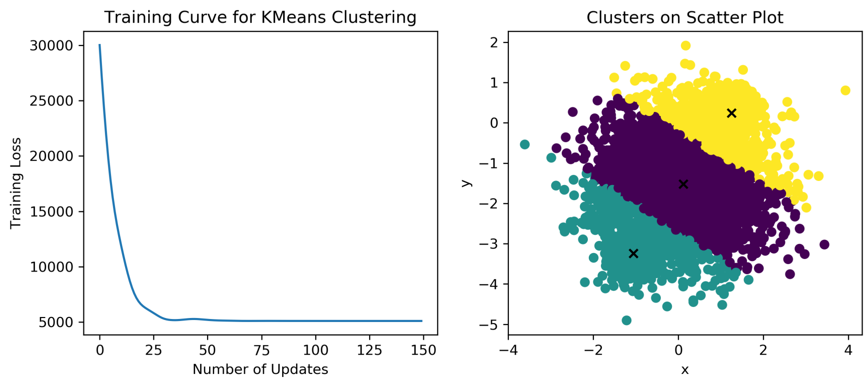
2. Optimal Number of Clusters
   Next, we study K-means clustering as we vary $K$, while all other parameters are fixed. The results are shown in Figure 2. The number of points in each cluster is summarized in Table 1. Note that in the table, "–" is used to denote the cluster does not exist.



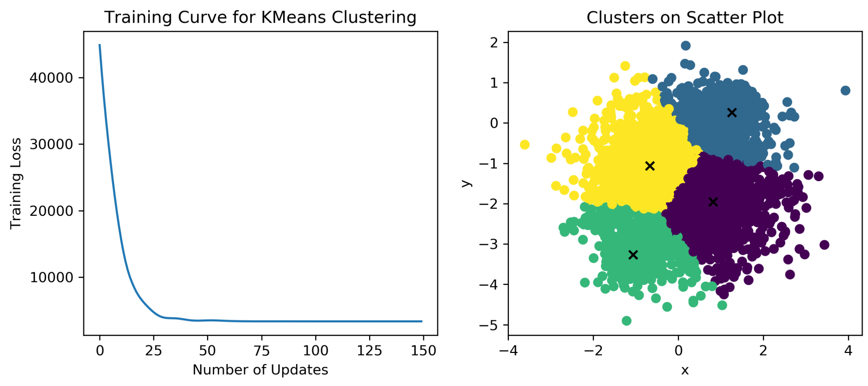
(a) Training Curve and Clusters Plot with 1 Cluster



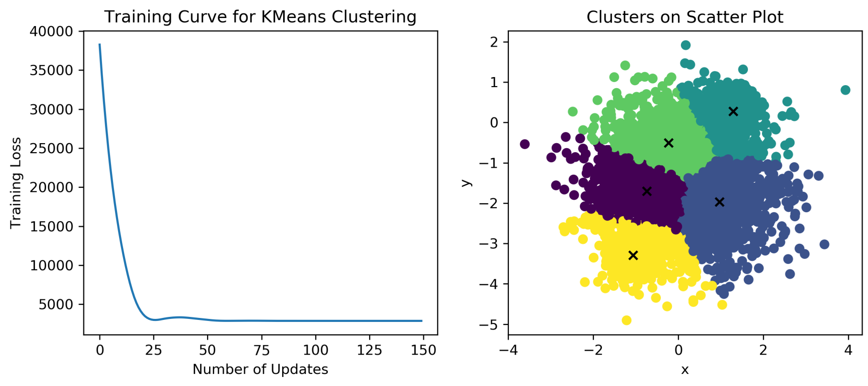
(b) Training Curve and Clusters Plot with 2 Clusters



(c) Training Curve and Clusters Plot with 3 Clusters



(d) Training Curve and Clusters Plot with 4 Clusters



(e) Training Curve and Clusters Plot with 5 Clusters

Figure 2: K Means Clustering with Varied Number of Clusters on Full Data Set

Table 1: Percentage of Points in Each Cluster (Full Data Set)

| K | % in Cluster 1 | % in Cluster 2 | % in Cluster 3 | % in Cluster 4 | % in Cluster 5 |
|---|---|---|---|---|---|
| 1 | 100.00 | - | - | - | - |
| 2 | 50.48 | 49.52 | - | - | - |
| 3 | 23.81 | 38.13 | 38.06 | - | - |
| 4 | 13.49 | 37.29 | 37.13 | 12.09 | - |
| 5 | 8.84 | 11.36 | 35.92 | 7.58 | 36.30 |

We are also asked to identify the "best" value of $K$. As is well-known in engineering design, "best" depends on metrics and criteria. Without any background information on what the data set actually presents, it is hard to speculate the objective in doing this clustering in the first place.

However, with the information we are asked to compute, we could speculate perhaps the objective is to ensure a roughly equal number of points fall into each bin. Then, from Table 1, $K = 3$ appears a natural choice. Indeed, from the scatter plot in Figure 2c, the area and the number of points assigned to each cluster look even and balanced. If this is our objective, then having 3 clusters is most reasonable amongst the five cases considered.

More commonly, however, we perform clustering to see if a group of data differs vastly from another. In other words, most points should be significantly closer to its cluster center, than the distance to the center it is not a part of. For instance, in Figure 3, we can create a hypothetical data set, which involves numbers drawn from two jointly Gaussian distributions. Even visually we can conclude from Figure 3 this data set should have two clusters. In the given data set, unfortunately, we do not see clear boundaries from which one cluster can be easily drawn out. If we want to identify special groups of data, perhaps in the two dimensions considered, the data we were given do not and should not be clustered at all; we just want $K = 1$.

In summary, if we want roughly equal areas, we should choose $K = 3$. If we want clusters to be well-separated from each other, we should choose $K = 1$.

3. Results on Validation Set

Table 2: Validation Loss in K-means

| K | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Loss | 12870 | 2491 | 1629 | 1055 | 889 |

Once again, we are asked to identify the "best" $K$ value. While our analysis above still holds, this time, we speculate our objective might be to identify $K$ such that validation loss is minimal. While it is easy to conclude without any further thinking $K = 5$

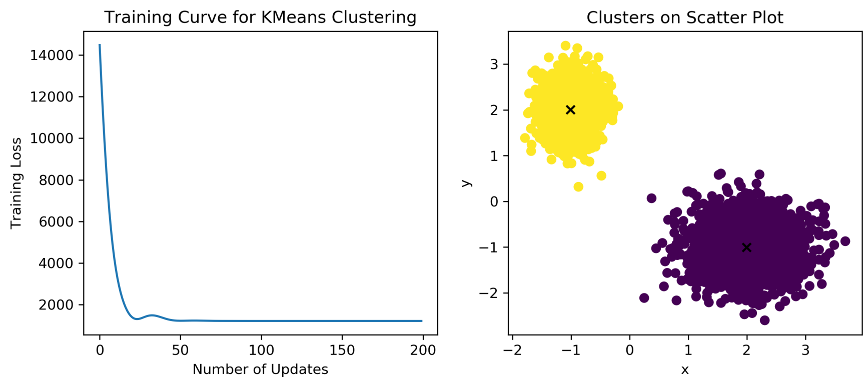


Figure 3: Training Curve and Clusters Plot from a Hypothetical Data Set

is optimal, we note that naturally loss should decrease given more clusters. For any given point, if there are more clusters, its distance to the nearest cluster center is likely smaller. If we do not desire the conclusion of having as many clusters as possible (i.e. $K = N$), we might want to find $K$ such that adding another group does not greatly reduce loss, in which case $K = 4$ might be the best.

For future extension on evaluating the ideal number of clusters, we can compute more relevant and more widely-accepted internal criteria than the ones suggested and required here in the assignment, such as the silhouette score, which compares inter-cluster and within-cluster distances.

## 2 Mixture of Gaussians

### 2.1 The Gaussian Cluster Model

1. The Probability Density Function (PDF) for a multivariate Gaussian distribution is as follows:

$$\mathcal{N}(\underline{x};\underline{\mu},\mathbf{\Sigma}) = \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{\det(\mathbf{\Sigma})}}\exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu})^T\mathbf{\Sigma}^{-1}(\underline{x}-\underline{\mu})\right\}$$

where $D$ is the dimension of the input vector $\underline{x} \in \mathbb{R}^{D\times 1}$, $\underline{\mu}$ is the mean vector $\in \mathbb{R}^{D\times 1}$, and $\mathbf{\Sigma}$ is the covariance matrix $\in \mathbb{R}^{D\times D}$. In this case, the covariance matrix $\mathbf{\Sigma}$ is $\sigma_k^2\mathbf{I}_D$. Thus, the PDF for a given data vector $\underline{x}_n$ and cluster $k$ becomes

$$\mathcal{N}(\underline{x}_n;\underline{\mu}_k,\sigma_k^2\mathbf{I}_D) = \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{\det[\sigma_k^2\mathbf{I}_D]}}\exp\left\{-\frac{1}{2}(\underline{x}_n-\underline{\mu}_k)^T(\sigma_k^2\mathbf{I}_D)^{-1}(\underline{x}_n-\underline{\mu}_k)\right\}$$

$(2\pi)^{\frac{D}{2}}\sqrt{\det[\sigma_k^2\mathbf{I}_D]}$ can be simplified using the identities

$$\det(c\mathbf{A}) = c^n\det(\mathbf{A}), \mathbf{A}\in\mathbb{R}^{n\times n}$$

and

$$\det(\mathbf{I}_n) = 1$$

The simplification is as follows:

$$\begin{aligned}(2\pi)^{\frac{D}{2}}\sqrt{\det[\sigma_k^2\mathbf{I}_D]} &= (2\pi)^{\frac{D}{2}}\sqrt{(\sigma_k^2)^D\det[\mathbf{I}_D]}\\ &= (2\pi)^{\frac{D}{2}}\sqrt{(\sigma_k^2)^D\cdot 1}\\ &= (2\pi)^{\frac{D}{2}}(\sigma_k^2)^{\frac{D}{2}}\\ &= (2\pi\sigma_k^2)^{\frac{D}{2}}\end{aligned}$$

$-\frac{1}{2}(\underline{x}_n-\underline{\mu}_k)^T(\sigma_k^2\mathbf{I}_D)^{-1}(\underline{x}_n-\underline{\mu}_k)$ can be simplified using the identities

$$(c\mathbf{A})^{-1} = c^{-1}\mathbf{A}^{-1}$$

and

$$(\mathbf{I}_n)^{-1} = \mathbf{I}_n$$

The simplification is as follows:

$$\begin{aligned}
-\frac{1}{2}(\underline{x}_n-\underline{\mu}_k)^T(\sigma_k^2\mathbf{I}_D)^{-1}(\underline{x}_n-\underline{\mu}_k) &= -\frac{1}{2}(\underline{x}_n-\underline{\mu}_k)^T\frac{1}{\sigma_k^2}(\mathbf{I}_D)^{-1}(\underline{x}_n-\underline{\mu}_k)\\
&= -\frac{1}{2\sigma_k^2}(\underline{x}_n-\underline{\mu}_k)^T\mathbf{I}_D(\underline{x}_n-\underline{\mu}_k)\\
&= -\frac{1}{2\sigma_k^2}(\underline{x}_n-\underline{\mu}_k)^T(\underline{x}_n-\underline{\mu}_k)\\
&= -\frac{1}{2\sigma_k^2}\|\underline{x}_n-\underline{\mu}_k\|_2^2
\end{aligned}$$

Note that $\|\underline{x}_n-\underline{\mu}_k\|_2^2$ is equivalent to the `distanceFunc(X, MU)` function written in Section 1.1.

After these simplications, the PDF becomes

$$\mathcal{N}(\underline{x}_n;\underline{\mu}_k,\sigma_k^2\mathbf{I}_D) = \frac{1}{(2\pi\sigma_k^2)^{\frac{D}{2}}}\exp\left\{-\frac{1}{2\sigma_k^2}\|\underline{x}_n-\underline{\mu}_k\|_2^2)\right\} \tag{2}$$

Taking the log of (2) to find the log probability density function for a given data vector $\underline{x}_n$ and cluster $k$, the following is obtained:

$$\begin{aligned}
\log\left[\mathcal{N}(\underline{x}_n;\underline{\mu}_k,\sigma_k^2\mathbf{I}_D)\right] &= \log\left[\frac{1}{(2\pi\sigma_k^2)^{\frac{D}{2}}}\exp\left\{-\frac{1}{2\sigma_k^2}\|\underline{x}_n-\underline{\mu}_k\|_2^2)\right\}\right]\\
&= \log\left[\frac{1}{(2\pi\sigma_k^2)^{\frac{D}{2}}}\right] + \log\left[\exp\left\{-\frac{1}{2\sigma_k^2}\|\underline{x}_n-\underline{\mu}_k\|_2^2)\right\}\right]\\
\log[P(\underline{x}_n|z_n=k)] &= -\frac{D}{2}\log\left[2\pi\sigma_k^2\right] - \frac{1}{2\sigma_k^2}\|\underline{x}_n-\underline{\mu}_k\|_2^2) \qquad (3)
\end{aligned}$$

The vectorized Tensorflow Python implementation of (3) over all $N$ data points and $K$ clusters is shown below:

```python
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

    D = tf.to_float(X.shape[1])
    sigma = tf.squeeze(sigma)
    log_term = -D/2 * tf.log(2 * np.pi * sigma)

    X = tf.dtypes.cast(X, tf.float32)
    x_dist = distanceFunc(X, mu)
    exp_term = - x_dist / (2 * sigma)

    return log_term + exp_term
```

2. To compute the probability that a given data vector $\underline{x}_n$ belongs to a cluster $k$, one can apply Baye's rule:

$$\begin{aligned} P(z_n = k|\underline{x}_n) &= \frac{P(\underline{x}_n|z_n = k)P(z_n = k)}{P(\underline{x}_n)} \\ &= \frac{P(\underline{x}_n|z_n = k)P(z_n = k)}{\sum_{i=1}^{K} P(\underline{x}_n|z_n = i)P(z_n = i)} \end{aligned} \tag{4}$$

Taking the log of (4), the following is obtained:

$$\begin{aligned} \log\left[P(z_n = k|\underline{x}_n)\right] &= \log\left[\frac{P(\underline{x}_n|z_n = k)P(z_n = k)}{\sum_{i=1}^{K} P(\underline{x}_n|z_n = i)P(z_n = i)}\right] \\ &= \log[P(\underline{x}_n|z_n = k)] + \log[P(z_n = k)] \\ &\quad - \log\left[\sum_{i=1}^{K} P(\underline{x}_n|z_n = i)P(z_n = i)\right] \\ &= \log[P(\underline{x}_n|z_n = k)] + \log[P(z_n = k)] \\ &\quad - \log\left[\sum_{i=1}^{K} \exp[\log(P(\underline{x}_n|z_n = i)) + \log(P(z_n = i))]\right] \end{aligned} \tag{5}$$

Substituting $\pi_k = P(z_n = k)$ into (5), the following is obtained:

$$\begin{aligned} \log[P(z_n = k|\underline{x}_n)] = \log[P(\underline{x}_n|z_n = k)] + \log[\pi_k] \\ - \log\left[\sum_{i=1}^{K} \exp[\log(P(\underline{x}_n|z_n = i)) + \log(\pi_i)]\right] \end{aligned} \tag{6}$$

where $\log[P(\underline{x}_n|z_n = k)]$ is the result in (3). The vectorized Tensorflow Python implementation of (6) over all $N$ data points and $K$ clusters is shown below:

```
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_pi = tf.squeeze(log_pi)

    sum_term = hlp.reduce_logsumexp(log_PDF + log_pi,keep_dims=True)
    return tf.add(log_pi, log_PDF) - sum_term
```

It is important to use `reduce_logsumexp()` instead of `tf.reduce_sum()` because `reduce_logsumexp()` takes the maximum value in each row of the input tensor and subtracts that maximum value from every element of its row. Then, it takes the exponential of the normalized tensor, calls `tf.reduce_sum()`, and takes the log of the result. Without normalizing the exponents like this, the exponential operation can easily overflow and break the program.

## 2.2 Learning the MoG

1. Best Model Parameters and Validation Loss and Scatter Plots for K=3 MoG model

Table 3: Best Model Parameters Learned K=3

| | $\mu$ | $\sigma$ | $\pi$ |
|---|---|---|---|
| Cluster 1 | [ 1.2975621, 0.31892934] | 0.03895941 | -1.098111 |
| Cluster 2 | [-1.1039814, -3.3082023 ] | 0.039111 | -1.1009928 |
| Cluster 3 | [ 0.10936161, -1.5266703 ] | 0.98683137 | -1.0967376 |

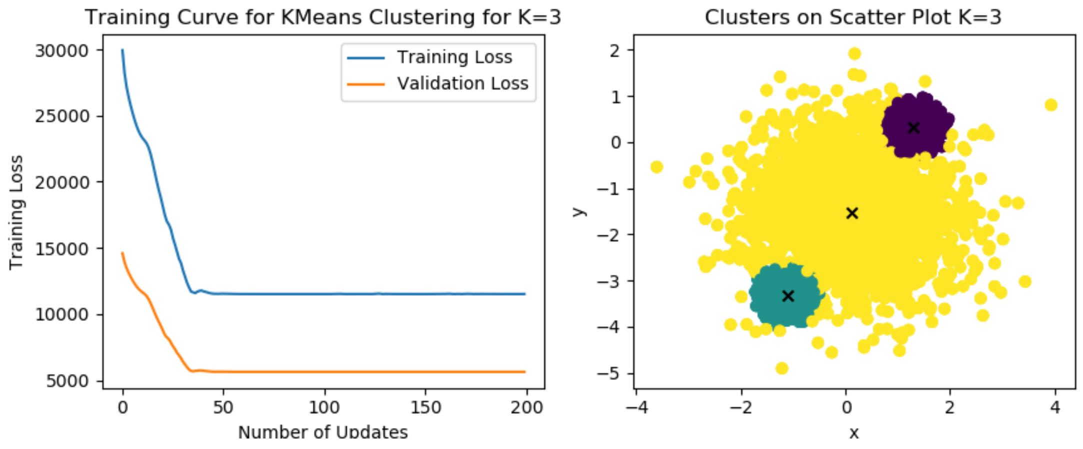


Figure 4: Validation Loss vs number of updates (Left) and Scatter Plot (Right) for K=3 on the data2D.npy dataset

2. Validation Loss Functions and Scatter Plots for K = 1,2,3,4,5 MoG Model

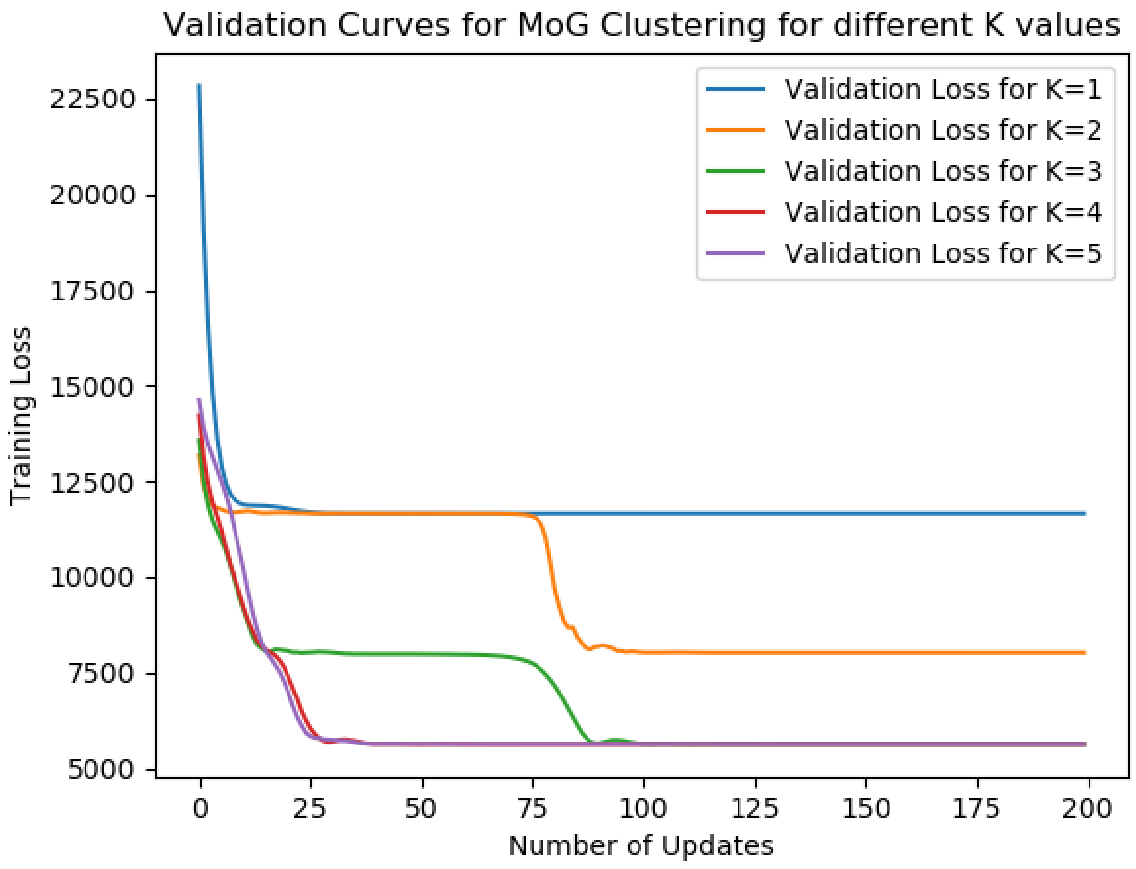


Figure 5: Validation Loss vs number of updates for K=1,2,3,4,5 on the data2D.npy dataset

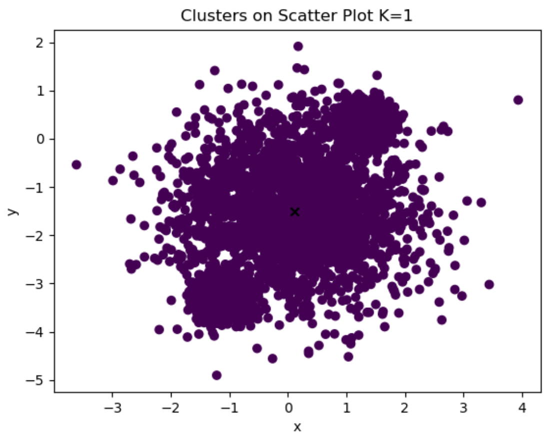
Table 4: Final Validation losses for $K = 1, 2, 3, 4, 5$

| K | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Final Validation Loss | 11649.18 | 8009.49 | 5625.29 | 5621.61 | 5624.19 |

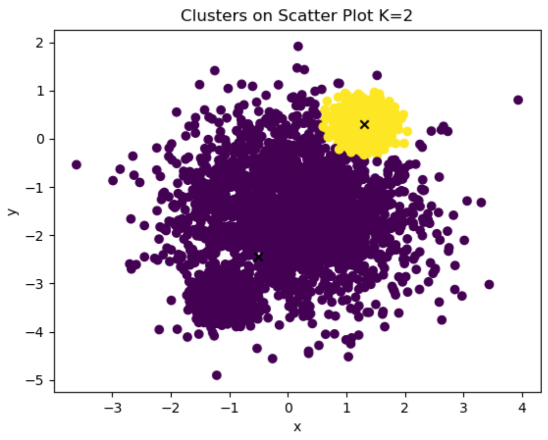
By referring to Table 4, it is seen that the best K value is $K = 3$. The final validation loss does not decrease significantly between K=3, 4, and 5. It can also be seen in Figure 5 that the validation losses also plateau at similar values and do not decrease further for $K = 3$, $K = 4$, and $K = 5$.

3. K-Means and MoG Algorithms on the data100D.npy dataset for K=5,10,15,20,30

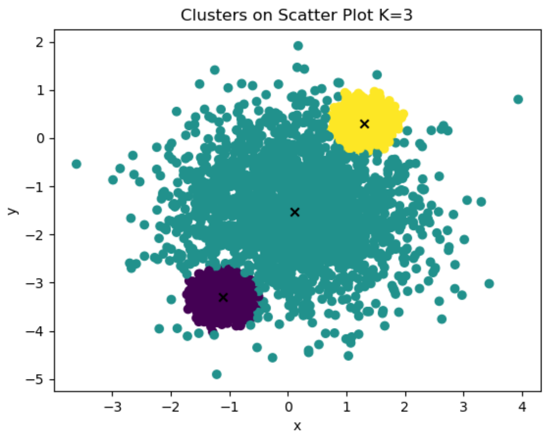
   Referencing Figure 7, it can be guessed that there are K=10 clusters in the dataset. This distinction is mainly seen in the K-Means model where the validation loss stop significantly decreasing after passing K=10, whereas in the MoG model, all the K values result in similar losses.



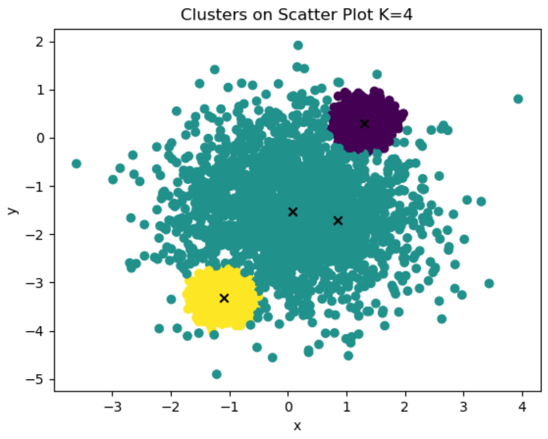
(a) Scatter Plot for K=1



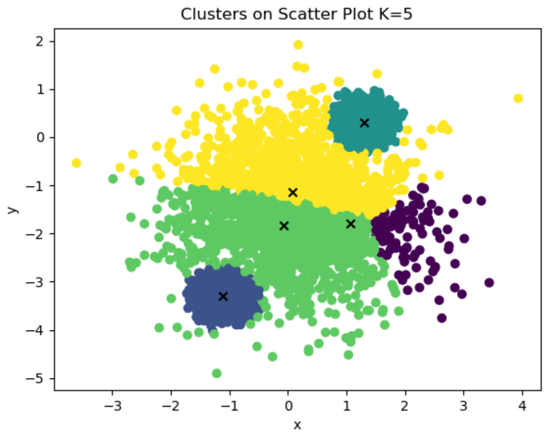
(b) Scatter Plot for K=2



(c) Scatter Plot for K=3

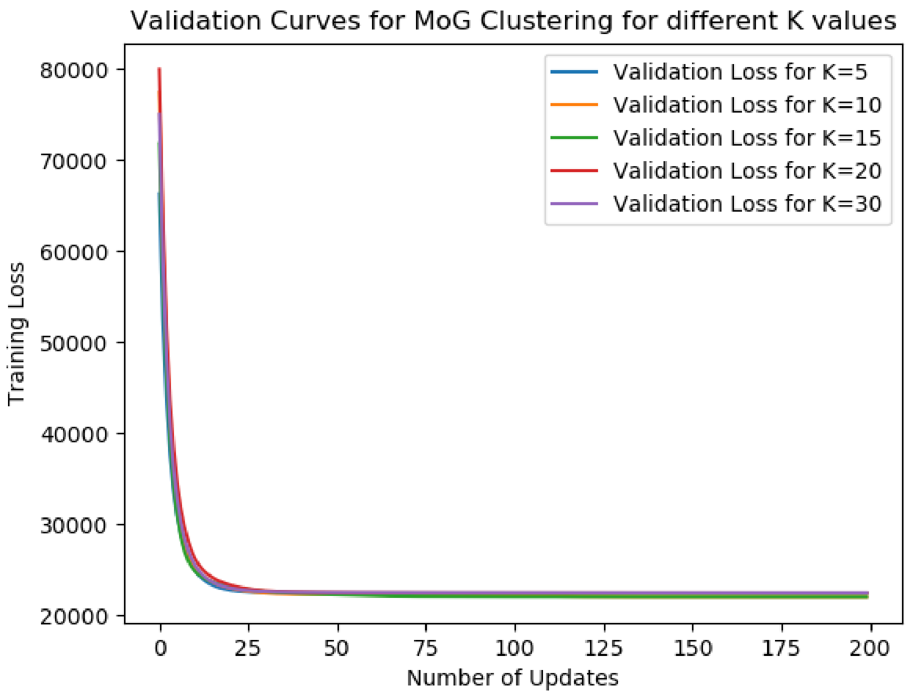


(d) Scatter Plot for K=4

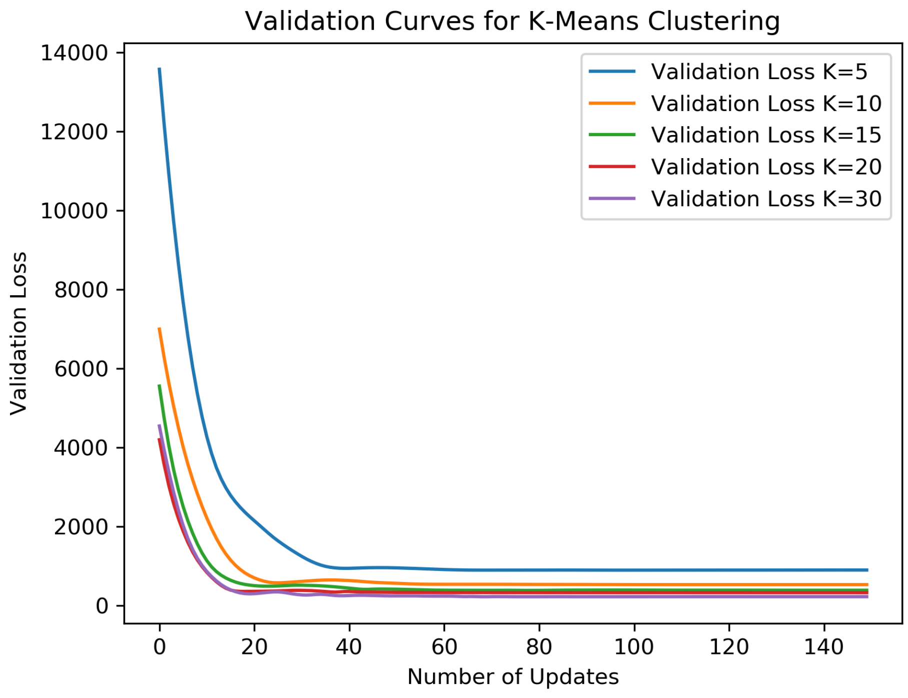


(e) Scatter Plot for K=5

Figure 6: Scatter plots with colored data points for K=1,2,3,4,5



(a) Validation Loss plots for MoG model given K=5,10,15,20,30



(b) Validation Loss plots for K-means model given K=5,10,15,20,30

Figure 7: Validation losses for the MoG model (left) and K-means model (right)