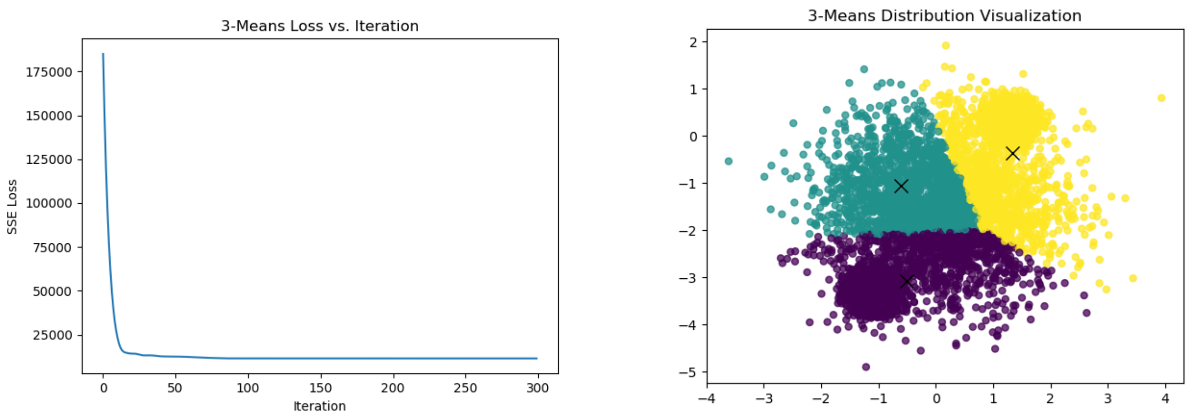
Osvald Nitski
Will Morley

Assignment 3

1 K-means [9 pts.]

1.1 Learning K-means

1. 2D K-means with $k = 3$



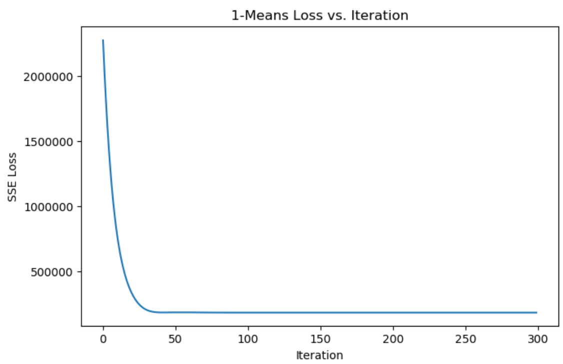
(a) Loss

(b) Visualization

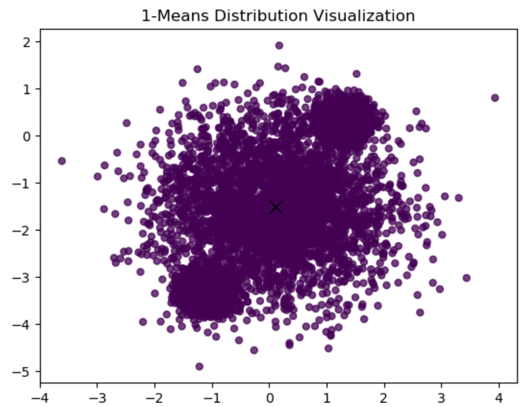
Figure 1: 2D K-means with $k = 3$

2. 2D K-means with $K \epsilon \{1, 2, 3, 4, 5\}$

Without any knowledge of what this data represents, 2 clusters appear to appropriately split the data into almost mirrored groups based on proximity to 'poles' at either end; these are the two dense areas on opposing ends of the entire distribution.
With more knowledge of what this distribution is modelling arguments could be made for k = 3,4,5 should they make for more useful results to a specific application. In the next section, MoG creates a more pleasing assignment for k = 3 which seems to split the data in a natural way; this result, however, did not occur for 3-means.
If finding a single mean is the goal of this exercise, k-means feels inappropriate.

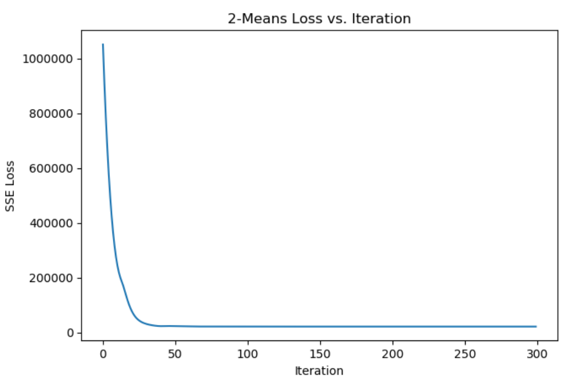


(a) Loss

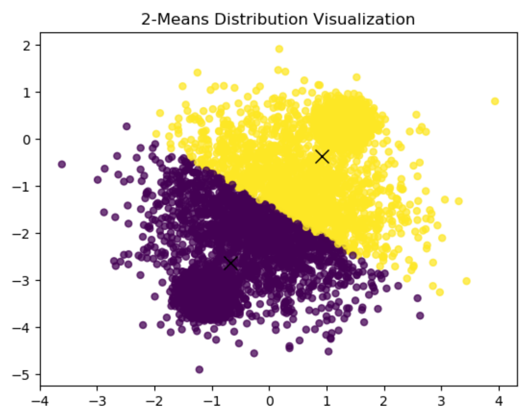


(b) Visualization

mean (0.1038, -1.5025): 100%

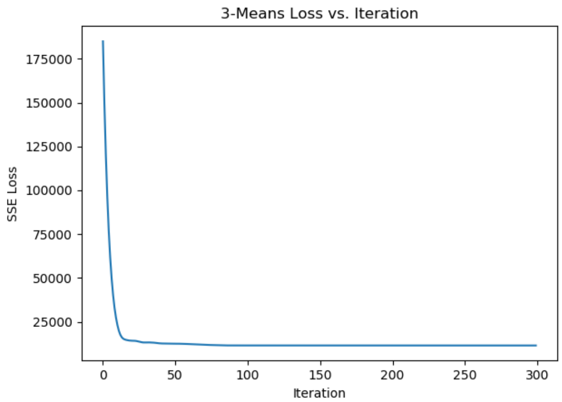


(a) Loss

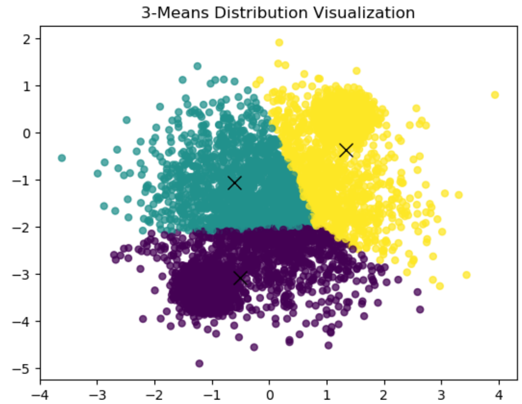


(b) Visualization

mean (-0.6749, -2.6380): 50.41%
mean (0.9161, -0.3711): 49.59%

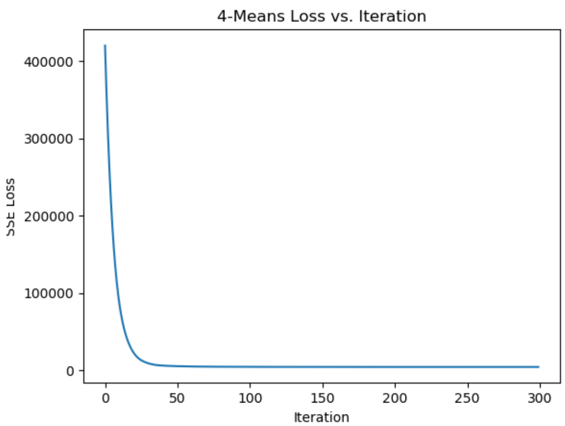


(a) Loss

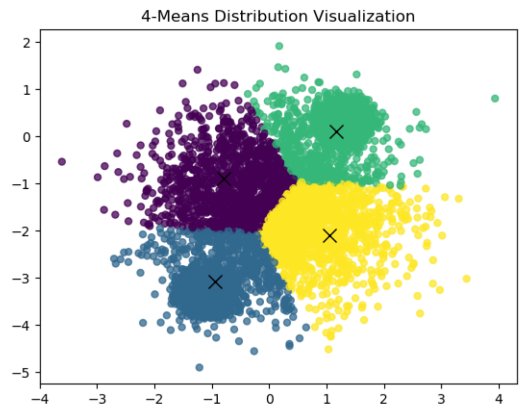


(b) Visualization

mean (-0.5002, -3.0742): 42.36%
mean (-0.6043, -1.0485): 14.83%
mean (1.3305, -0.3646): 42.81%

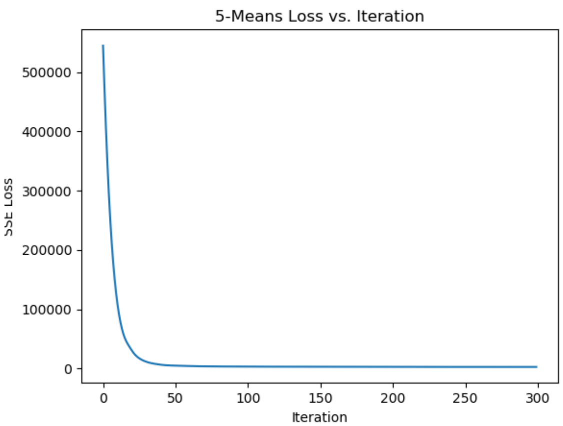


(a) Loss

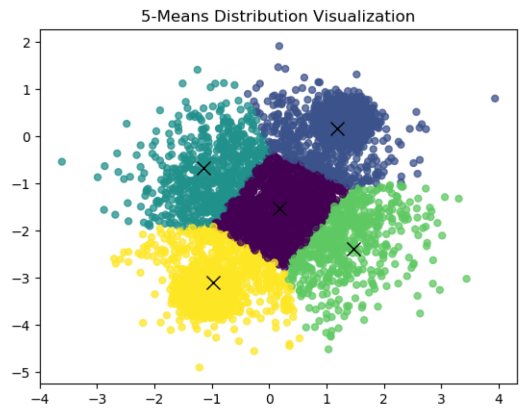


(b) Visualization

mean (-0.7969, -0.9004): 11.42%
mean (-0.9444, -3.0850): 38.46%
mean (1.1644, 0.1151): 37.95%
mean (1.0451, -2.1047): 12.17%



(a) Loss



(b) Visualization

mean (0.1762, -1.5342): 12.97%
mean (1.1774, 0.1602): 37.40%
mean (-1.1456, -0.6759): 6.33%
mean (1.4717, -2.3802): 5.57%
mean (-0.9752, -3.1052): 37.73%

3. Validation Losses
   k = 1
   validation loss: 60655.453
   k = 2
   validation loss: 6334.5747
   k = 3
   validation loss: 3197.8066
   k = 4
   validation loss: 1158.9004
   k = 5
   validation loss: 828.1911

   Based on the above data one might be tempted to say 5 means gives the best result; however, loss will generally decrease as means increase
   For this reason we believe that although loss decreases as k increases, beyond a certain k no use comes from the lower loss - the model is essentially overfitting. The most drastic loss decrease is at k = 2 where it drops an order of magnitude from k = 1. Since k = 2 is the last point of drastic loss decrease we believe it is the best k value for k-means on this data set.

2 Mixture of Gaussians [16 pts.]

2.1 The Gaussian cluster mode [7 pts.]

1. This is our code to compute the log probability density function for cluster k.

Code Snippet 1: $log\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}^k, \sigma^{k^2})$ for all pair of N data points and K clusters

```
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

    dists = tf.transpose(distanceFunc(X, mu))
    log = tf.transpose(tf.log(2 * np.pi * sigma))
    sig = 1 / tf.transpose(sigma)

    dists_sig = dists * sig
    log_gaussPDF = -0.5 * (log + dists_sig)

    return log_gaussPDF
```

2. This is our code to compute the log probability of the cluster variable z given the data vector $\boldsymbol{x}$.
The reason for using the helper function reduce_logsumexp instead of just a reduce sum is to combat the underflow and overflow which often accompanies limited precision floating point calculations dealing with exponential functions.

Code Snippet 2: $log\mathcal{P}(z|\boldsymbol{x})$

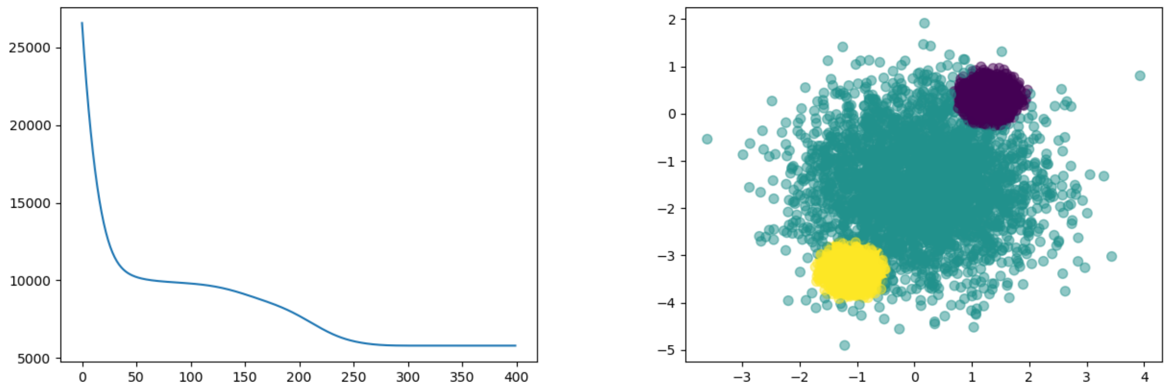
```
def log_posterior(log_PDF, log_pi):
# Input
# log_PDF: log Gaussian PDF N X K
# log_pi: K X 1

# Outputs
# log_post: N X K

log_piGauss = tf.transpose(log_pi) + log_PDF
log_post = log_piGauss - tf.reshape(
                hlp.reduce_logsumexp(log_piGauss), [-1, 1])
return log_post
```

2.2 Learning the MoG [9 pts.]

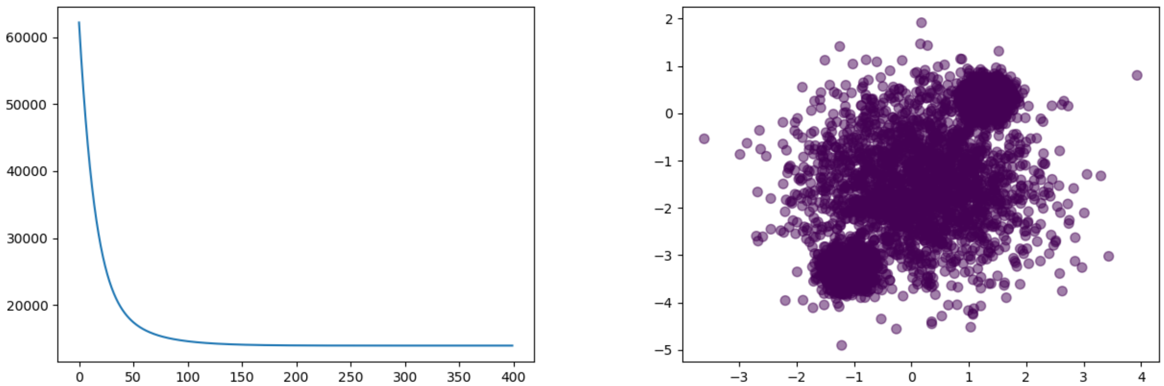
1. Our MoG model ended with a loss of 5803.36



(a) Loss (b) Visualization

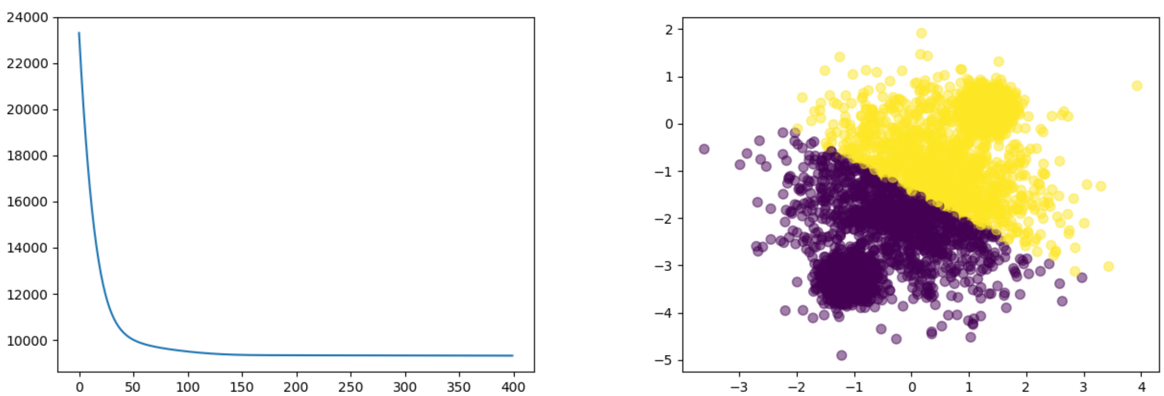
mean $(1.299, 0.309)$: 33.85%
mean $(0.105, -1.523))$: 32.37%
mean $(-1.102, -3.306)$: 33.78%

2. We believe $k = 3$ was best based on the scatter plot. The 2d data seemed to have 3 distinct clusters: two dense poles and a central, more spread out cluster. With higher k values, the validation loss was lower, but the scatter plot split a visually obvious cluster into multiple sections. Similarly when $k < 3$ the splits looked fairly arbitrary and not representative of any underlying pattern. When $k = 3$ these 3 clusters get nicely separated in the scatter plot.



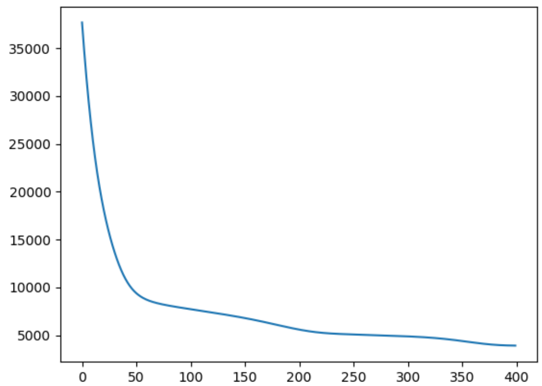
(a) Loss (b) Visualization

Figure 8: K = 1
Final validation loss: 6980.85

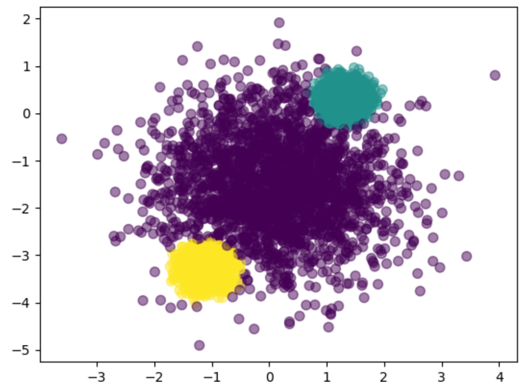


(a) Loss (b) Visualization

Figure 9: K = 2
Final validation loss: 4670.98

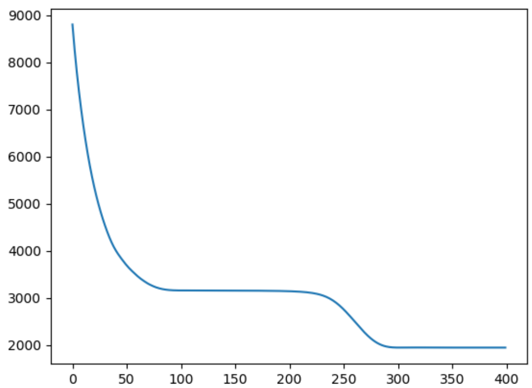


(a) Loss

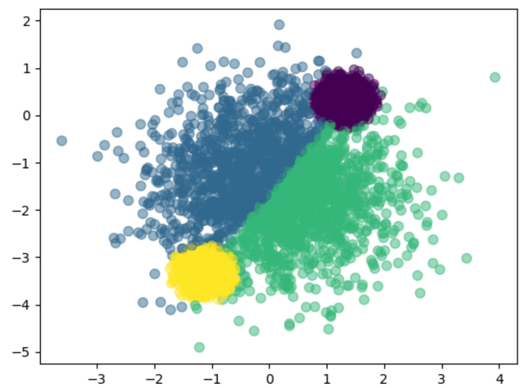


(b) Visualization

Figure 10: K = 3
Final validation loss: 1885.81

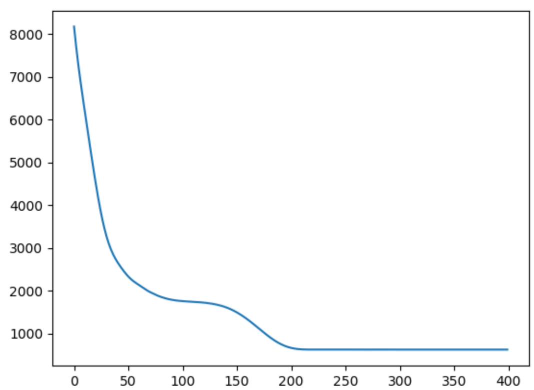


(a) Loss

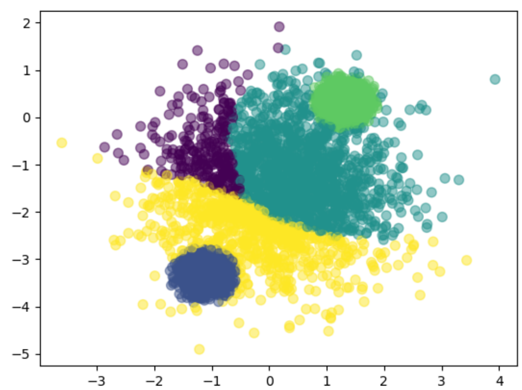


(b) Visualization

Figure 11: K = 4
Final validation loss: 928.46



(a) Loss



(b) Visualization

Figure 12: K = 5
Final validation loss: 285.11

3. The table below outlines the K-means and MoG training loss for the 100-Dimension data set. When run with $K = \{5,\ 10,\ 15,\ 20,\ 30\}$ we experience a general trend of decreasing loss with increasing K. There is a significant decrease in the K-means loss between $k = 5$ and $k = 10$ followed by marginal decreases for $k > 10$. This implies that the correct number of bins lies somewhere between 5 and 10 and that adding additional clusters simply groups small insignificant fringe groups of data in what could be called "overfitting"

K	K-means loss	MoG loss
5	18302634.0	29166.01
10	6810376.0	32220.57
15	6757137.5	26099.89
20	6962632.0	20115.72
30	6696336.0	22149.94