ECE521 Assignment 3 Report

Date: 2017-03-26

Student: Winston Wong

Student Number: 1001614853

(Source code and data for each sub-question is attached to email)

Q1.1.1

Jensen Inequality tells us that the function f is convex if and only if

$$f(ta + (1-t)b) \leq tf(a) + (1-t)f(b)$$

The loss function is

$$L(\mu) = \sum_{n=1}^{B} \min_{k = 1 \ldots . K} \|x_n - \mu_k\|_2^2$$

Now, let

$$\mu = ta + (1-t)b$$

And let $\mu = [2\ 4]^T, a = b = [1\ 2]^T, t = 0.5,\ x = [1\ 2]$

Notice that the data points x lies exactly on a and b. Thus, $L(a) = L(b) = 0$. However, the same cannot be said for μ. Concretely, we have that

$$RHS = tf(a) + (1-t)f(b) = 0.5L(a) + 0.5L(b) = 0$$

And

$$LHS = f(ta + (1-t)b) = L(\mu) = 1 \geq RHS$$

Which contradicts Jensen's Inequality. Therefore, the objective function is not convex.

Q1.1.2

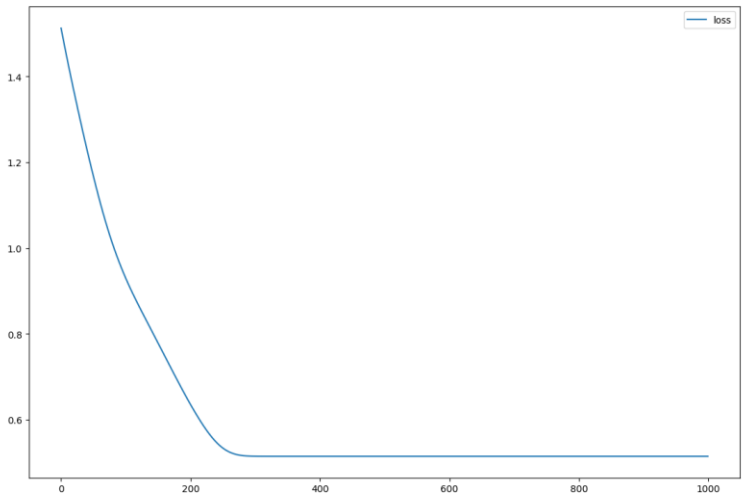


Figure 1 Loss vs. Number of Updates

Q1.1.3

For $K = 1$, 10000 data points fall into cluster 1. Percentage of data points belonging to cluster 1 is 100%.

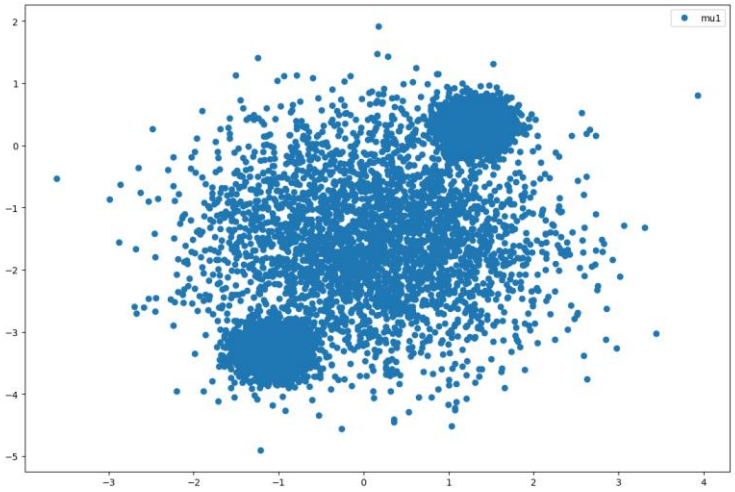


Figure 2 2D K-means Scatter Plot; K=1

For $K = 2$,

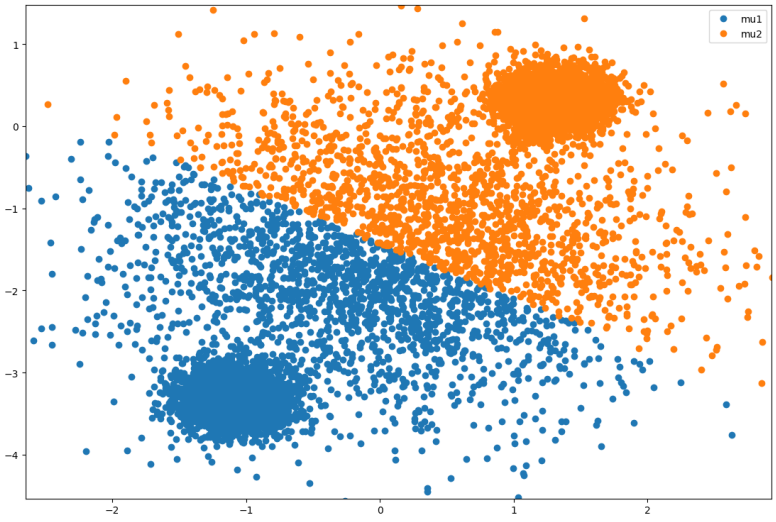


Figure 32D K-means Scatter Plot; K=2

	cluster 1	cluster 2
Percentage	0.5033	0.4967

For $K = 3$,

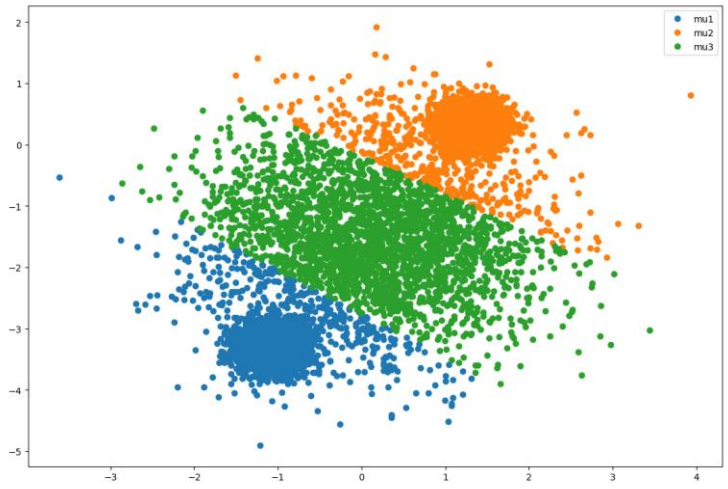


Figure 4 2D K-means Scatter Plot; K=3

	cluster 1	cluster 2	cluster 3
Percentage	0.3789	0.3789	0.2422

For $K = 4$,

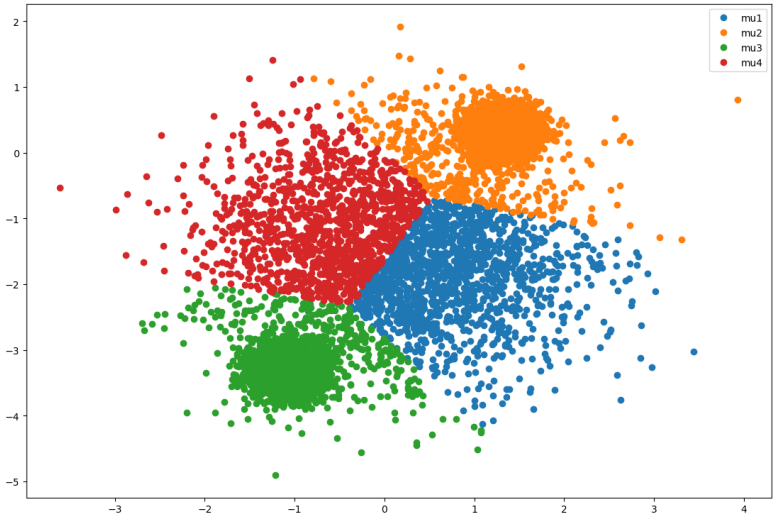


Figure 5 2D K-means Scatter Plot; K=4

	cluster 1	cluster 2	cluster 3	Cluster 4
Percentage	0.1372	0.372	0.3692	0.1216

For $K = 5$,

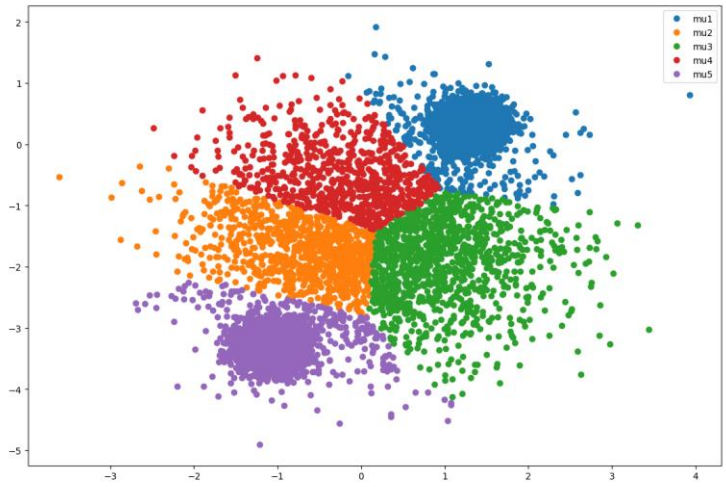


Figure 6 2D K-means Scatter Plot; K=5

	cluster 1	cluster 2	cluster 3	Cluster 4	Cluster 5
Percentage	0.3569	0.0914	0.1118	0.0778	0.3621

Zooming into the scatter plot,

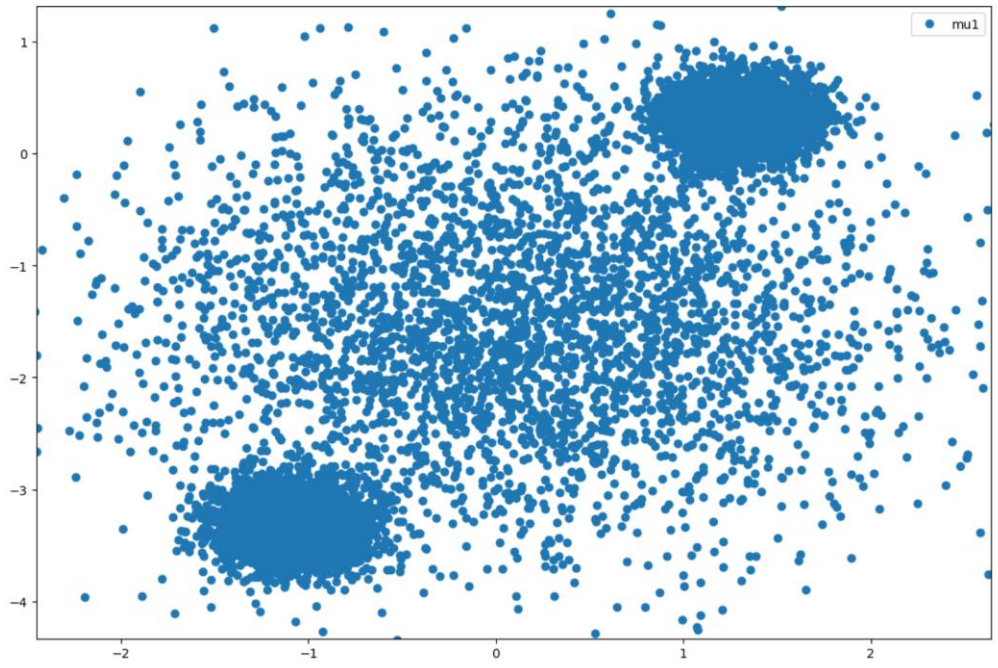
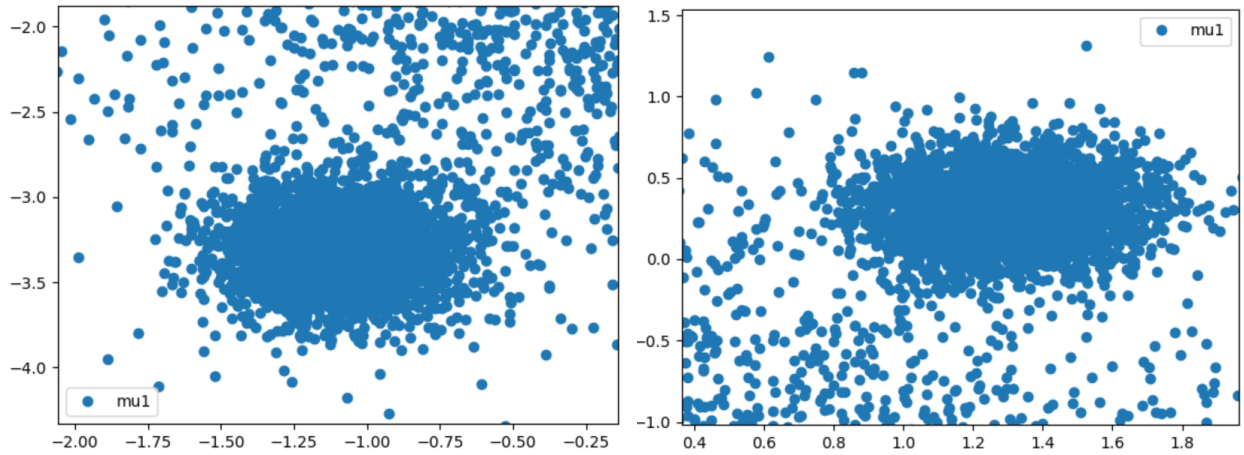


Figure 7 2D K-means Scatter Plot (zoom-in)

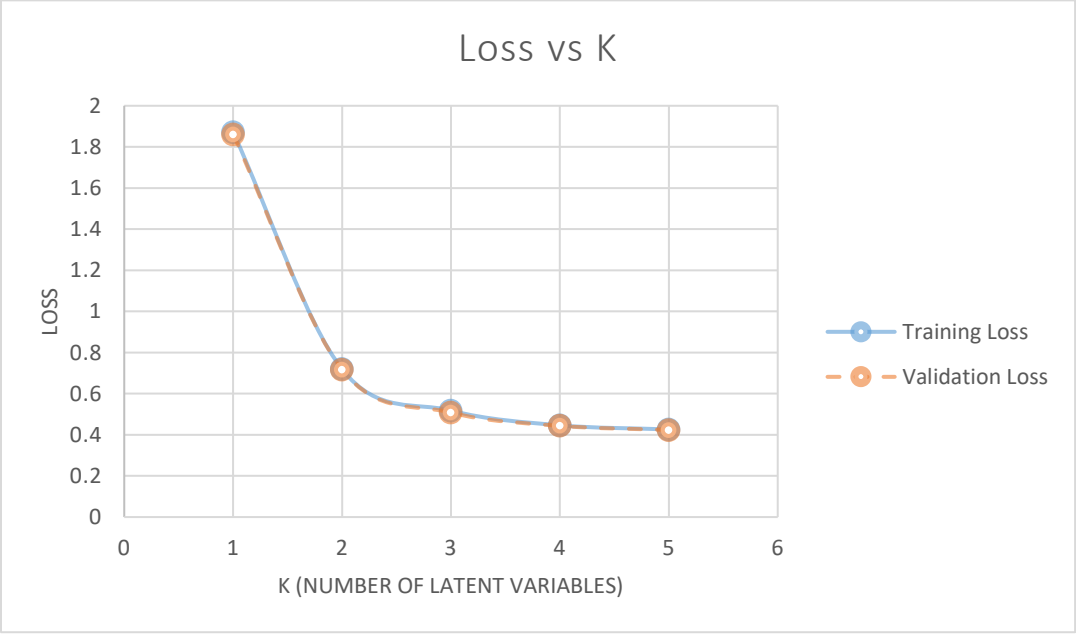


We observed that there are 3 clusters. 2 dense clusters centred at around $(-1, -3)$ and $(1.3, 0.5)$, and one sparse cluster center at around $(0, -1)$. Therefore, the optimum K is 3.

Q1.1.4

After running 1000 epochs, the results for K = 1, 2, 3, 4, 5 were obtained.

	Training loss	Validation loss
K = 1	1.87131	1.86053
K = 2	0.719057	0.717169
K = 3	0.517802	0.508331
K = 4	0.446221	0.443572
K = 5	0.426028	0.423196



The losses starts converging at K = 3. Thus, K =3 is the best.

Q2.1.1

$$P(z|\boldsymbol{x}) = \frac{p(z,\boldsymbol{x})}{p(\boldsymbol{x})} = \frac{p(x|z)p(z)}{p(\boldsymbol{x})}$$

In the question, it is given that

$$p(z) = \pi_k$$

Also, given z = k, x is normally distributed with mean μ_k and σ_k. Therefore,

$$p(x|z) = N(\mu_k, \sigma_k)$$

Marginalizing the probability, we have that

$$p(\boldsymbol{x}) = \sum_z p(\boldsymbol{x}|z)p(z) \sum_{k=1}^{K} N(\boldsymbol{x}; \mu_k, \sigma_k)\pi_k$$

And finally, after obtaining $p(x|z), P(z)$ and $p(\boldsymbol{x})$, we have that

$$P(z|\boldsymbol{x}) = \frac{\pi_k N(\boldsymbol{x}; \mu_k, \sigma_k)}{\sum_{k=1}^{K} N(\boldsymbol{x}; \mu_k, \sigma_k)\pi_k}$$

Q2.1.2

Recall

$$N(\boldsymbol{x}; \mu_k, \sigma_k) = (2\pi)^{-\frac{n}{2}}(\det\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

$$\log N(\boldsymbol{x}; \mu_k, \sigma_k) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log(\det\Sigma) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$$

Where n is the number of data dimensions.

Given that data dimensions are independent and each with standard deviation σ_k, we have that

$$\Sigma_{ij} = \sigma_k^2 I \qquad \text{and} \qquad \det(\sigma_k^2 I) = {\sigma_k^2}^n$$

$$\log N(\boldsymbol{x}; \mu_k, \sigma_k) = -\frac{1}{2\sigma_k^2}\|\boldsymbol{x}-\mu_k\|_2^2 - \frac{1}{2}\log({\sigma_k^2}^n) - \frac{n}{2}\log 2\pi = -\frac{1}{2\sigma_k^2}\|\boldsymbol{x}-\mu_k\|_2^2 - \frac{n}{2}\log(\sigma_k^2) - \frac{n}{2}\log 2\pi$$

(Eq.1)

Using Tensorflow broadcasting and the relationship above (Eq.1), the following python code is written to compute the log probability for all pairs of N data points and K clusters.

```
#Q2.1.2
#log_prob_x_given_z of size [B,K] has the probability of xn given normal distribution mu_k and var_k
def compute_log_prob_x_given_z(x, mu_T, var, B,D):
    # dim(x) = [B,1,2], dim(mu_T) = [1,K,2]
    distance_matrix = tf.reduce_sum(tf.square(x - mu_T),2)# B by K
    # [B,K]                    [B,K] <--- ([B,K]   [1,K])             [1,K]                  [1,1]
    log_prob_x_given_z = -0.5*tf.div(distance_matrix, var) - 0.5*D*tf.log(var) - 0.5*D*tf.log(2*math.pi)
    return log_prob_x_given_z
```

Q2.1.3

$$P(z = k|\boldsymbol{x}) = \frac{\pi_k N(\boldsymbol{x}; \mu_k, \sigma_k)}{\sum_{k=1}^{K} N(\boldsymbol{x}; \mu_k, \sigma_k)\pi_k}$$

$$\log P(z = k|\boldsymbol{x}) = \log\left(\frac{\pi_k N(\boldsymbol{x}; \mu_k, \sigma_k)}{\sum_{k=1}^{K} N(\boldsymbol{x}; \mu_k, \sigma_k)\pi_k}\right)$$

$$\log P(z = k|\boldsymbol{x}) = \log(\pi_k) + \log N(\boldsymbol{x}; \mu_k, \sigma_k) - \log \sum_{k=1}^{K} N(\boldsymbol{x}; \mu_k, \sigma_k)\pi_k$$

$$\log \sum_{k=1}^{K} N(\boldsymbol{x}; \mu_k, \sigma_k)\pi_k = \log \sum_{k=1}^{K} e^{\log N(\boldsymbol{x};\mu_k,\sigma_k)\pi_k} = \log \sum_{k=1}^{K} e^{\log N(\boldsymbol{x};\mu_k,\sigma_k)+\log \pi_k}$$

Let $x_k = \log N(\boldsymbol{x}; \mu_k, \sigma_k) + \log \pi_k$

$$\log \sum_{k=1}^{K} N(\boldsymbol{x}; \mu_k, \sigma_k)\pi_k = \log \sum_{k=1}^{K} e^{x_k} = LSE(x_1 \dots x_k)$$

Thus

$$\log P(z = k|\boldsymbol{x}) = \log(\pi_k) + \log N(\boldsymbol{x}; \mu_k, \sigma_k) - LSE(x_1 \dots x_k) \quad \text{(Eq.2)}$$

Based on Eq.2, the following python code is written:

```
#Q2.1.3
def compute_log_prob_z_given_x(log_prob_x_given_z,log_pi):
    #[B,K]     [1,K]          [B,K]                    [B,1]
    return log_pi + log_prob_x_given_z - reduce_logsumexp(tf.add(log_prob_x_given_z, log_pi),keep_dims=True)
```

Putting everything together, we have

```
#Q2.1.2
#log_prob_x_given_z of size [B,K] has the probability of xn given normal distribution mu_k and var_k
def compute_log_prob_x_given_z(x, mu_T, var, B,D):
    # dim(x) = [B,1,2], dim(mu_T) = [1,K,2]
    distance_matrix = tf.reduce_sum(tf.square(x - mu_T),2)# B by K
    # [B,K]                    [B,K] <--- ([B,K]   [1,K])          [1,K]                [1,1]
    log_prob_x_given_z = -0.5*tf.div(distance_matrix, var) - 0.5*D*tf.log(var) - 0.5*D*tf.log(2*math.pi)
    return log_prob_x_given_z

#Q2.1.3
def compute_log_prob_z_given_x(log_prob_x_given_z,log_pi):
    #[B,K]     [1,K]          [B,K]                    [B,1]
    return log_pi + log_prob_x_given_z - reduce_logsumexp(tf.add(log_prob_x_given_z, log_pi),keep_dims=True)


def build_graph(B, K, learning_rate, D):

    #Parameters
    x = tf.placeholder(tf.float32,[B,1,D],name="x") # input data
    mu_T = tf.Variable(tf.random_normal([1,K,D], mean = 0, stddev = 0.001), [1,K,D],name="mu_T") # mu
    tranpose
    phi = tf.Variable(tf.random_normal([1,K], mean=-1, stddev=0.1), [1,K], name="phi")
    psi = tf.Variable(tf.random_normal([1,K], mean=10, stddev=0.1), [1,K], name="psi")
    var = tf.exp(phi) #variance [0, inf)
    log_pi = logsoftmax(psi) # sum(log_pi) = 1

    #Q2.1.2
    log_prob_x_given_z= compute_log_prob_x_given_z(x, mu_T, var, B,D)
    #Q2.1.3
    log_prob_z_given_x = compute_log_prob_z_given_x(log_prob_x_given_z, log_pi)

    argmaxs = tf.argmax(log_prob_z_given_x, 1)
```

It is important to use logsumexp to increase the accuracy and avoid underflow and overflow problems when the argument in the log function gets large.

Q2.2.1

$$p(x) = \sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)$$

$$\log p(x) = \log\left(\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)\right)$$

$$\nabla_u \log p(x) = \nabla_u \log\left(\sum_{k=1}^{K} \pi_k N\left(x; \mu_k, \sigma_k^2\right)\right)$$

$$= \left(\frac{\frac{\partial}{\partial \mu_1}\left(\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)\right)}{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}, \frac{\frac{\partial}{\partial \mu_2}\left(\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)\right)}{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}, \dots, \frac{\frac{\partial}{\partial \mu_k}\left(\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)\right)}{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}\right)$$

$$= \left(\frac{\frac{\partial}{\partial \mu_1}\left(\pi_1 N(x; \mu_1, \sigma_1^2)\right)}{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}, \frac{\frac{\partial}{\partial \mu_2}\left(\pi_2 N(x; \mu_2, \sigma_2^2)\right)}{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}, \dots, \frac{\frac{\partial}{\partial \mu_k}\left(\pi_k N\left(x; \mu_k, \sigma_k^2\right)\right)}{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}\right)$$

$$= \left(\frac{\frac{\partial}{\partial \mu_1}\left(\pi_1 N(x; \mu_1, \sigma_1^2)\right)}{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}, \frac{\frac{\partial}{\partial \mu_2}\left(\pi_2 N(x; \mu_2, \sigma_2^2)\right)}{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}, \dots, \frac{\frac{\partial}{\partial \mu_k}\left(\pi_k N\left(x; \mu_k, \sigma_k^2\right)\right)}{\sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}\right)$$

In the gradient, multiply numerator and denominator by $\pi_k N\left(x; \mu_k, \sigma_k^2\right)$, we have that

$$\nabla_u \log p(x) = \left(\frac{\frac{\partial}{\partial \mu_1}\left(\pi_1 N(x; \mu_1, \sigma_1^2)\right) \pi_1 N(x; \mu_1, \sigma_1^2)}{\pi_1 N(x; \mu_1, \sigma_1^2) \sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}, \dots, \frac{\frac{\partial}{\partial \mu_k}\left(\pi_k N\left(x; \mu_k, \sigma_k^2\right)\right) \pi_k N\left(x; \mu_k, \sigma_k^2\right)}{\pi_k N(x; \mu_k, \sigma_k^2) \sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k^2)}\right)$$

Notice that $\frac{\partial}{\partial \mu_k}(P(x, z = k)) = \frac{\partial}{\partial \mu_k}\left(\log \pi_k N\left(x; \mu_k, \sigma_k^2\right)\right) = \frac{\frac{\partial}{\partial \mu_k}\left(\pi_k N\left(x; \mu_k, \sigma_k^2\right)\right)}{\pi_k N\left(x; \mu_k, \sigma_k^2\right)}$

And $p(z = k|x) = \frac{\pi_k N\left(x; \mu_k, \sigma_k^2\right)}{\sum_{k=1}^{K} \pi_k N\left(x; \mu_k, \sigma_k^2\right)}$, Thus, this can be re-written as

$$\nabla_u \log p(x) = \left(p(z = 1|x) \frac{\partial}{\partial \mu_1}(P(x, z = 1)), \dots, p(z = K|x) \frac{\partial}{\partial \mu_k}(P(x, z = K))\right)$$

Decomposing the above into a linear combination of k basis vectors weighted by the corresponding $p(z = k|x)$, we have that,

$$\nabla_u \log p(x) = \left\{p(z = 1|x)\left(\frac{\partial}{\partial \mu_1}(P(x, z = 1)), 0, \dots, 0\right)\right\} + \left\{p(z = 2|x)\left(0, \frac{\partial}{\partial \mu_2}(P(x, z = 2)), 0, \dots, 0\right)\right\}$$
$$+ \dots + \left\{p(z = k|x)\left(0, \frac{\partial}{\partial \mu_k}(P(x, z = K)), 0, \dots, 0\right)\right\}$$

Also notice that

$$\nabla_{\mathrm{u}} P(x, z=1) = \left(\frac{\partial}{\partial \mu_1}(P(x, z=1)), 0,0,0 \ldots, 0,0,0\right)$$

$$\nabla_{\mathrm{u}} P(x, z=2) = \left(0, \frac{\partial}{\partial \mu_2}(P(x, z=2)), 0, \ldots, 0,0,0\right)$$

$$\vdots$$

$$\nabla_{\mathrm{u}} P(x, z=K) = \left(0,0,0, \ldots, 0,0,0 \frac{\partial}{\partial \mu_k}(P(x, z=K))\right)$$

Therefore

$$\nabla_u \log p(x) = \sum_{k=1}^{K} p(z=k|x) \nabla_{\mathrm{u}} P(x, z=k)$$

As required.

Q2.2.2

$$L = -\log p(\mathbf{X}) = -\log \prod_{n=1}^{B} \sum_{k=1}^{K} \pi_k N\left(x_n; \mu_k, \sigma_k^2\right) = -\sum_{n=1}^{B} \log \sum_{k=1}^{K} \pi_k N\left(x_n; \mu_k, \sigma_k^2\right)$$

Again, Let $x_k = \log N(\boldsymbol{x}; \mu_k, \sigma_k) + \log \pi_k$

$$\log \sum_{k=1}^{K} N(\boldsymbol{x}; \mu_k, \sigma_k) \pi_k = \log \sum_{k=1}^{K} e^{x_k} = LSE(x_1 \ldots x_k)$$

After running 1500 epochs, the parameters below are learnt.

$\mu = \begin{bmatrix} \mu_1^T \\ \mu_2^T \\ \mu_3^T \end{bmatrix}, \mu_k \in R^2$	[[0.10560007 -1.52742445] [1.29843318 0.30928019] [-1.10161233 -3.30632114]]
$\pi = [\pi_1 \quad \pi_2 \quad \pi_3], \pi_k \in R$	[[0.33463621 0.33333665 0.33202699]]
$\sigma = [\sigma_1 \quad \sigma_2 \quad \sigma_3], \sigma_k \in R$	[[0.98719239 0.03885011 0.03908068]]

The loss converges to 1.7132, as illustrated in the Figure below.

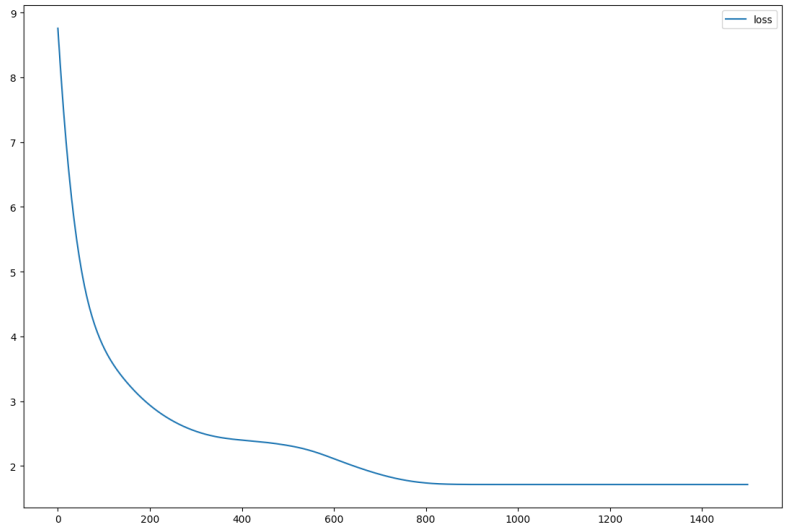


Figure 8 Loss vs Number of Updates

Q2.2.3

For $K = 1$ validation loss is 3.4866662

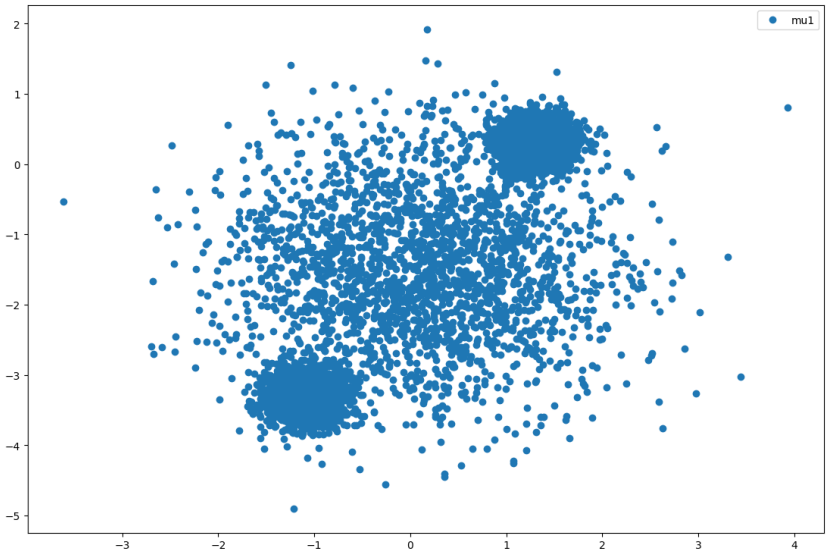


Figure 9 2D MoG Scatter Plot; K=1

For $K = 2$, validation loss is 2.4167347.

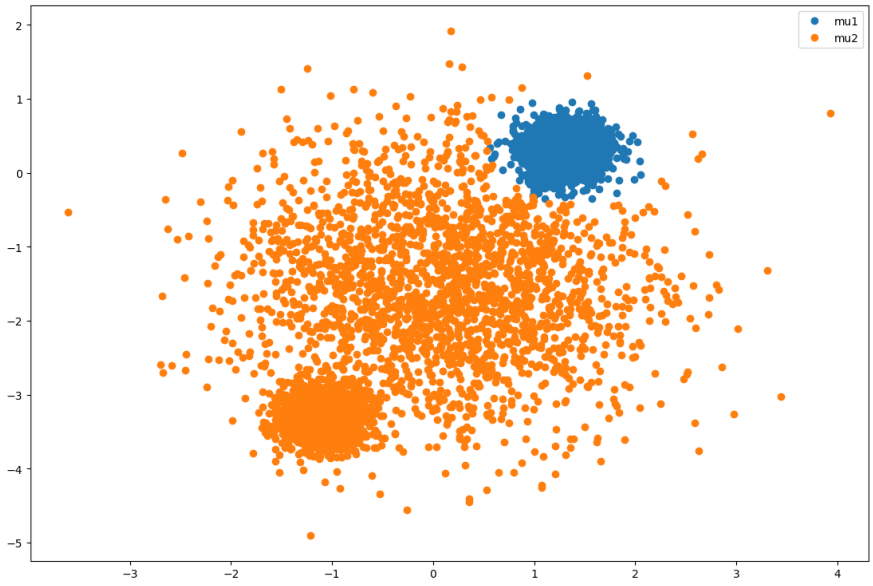


Figure 10 2D MoG Scatter Plot; K=2

For $K = 3$, validation loss is 1.7055.

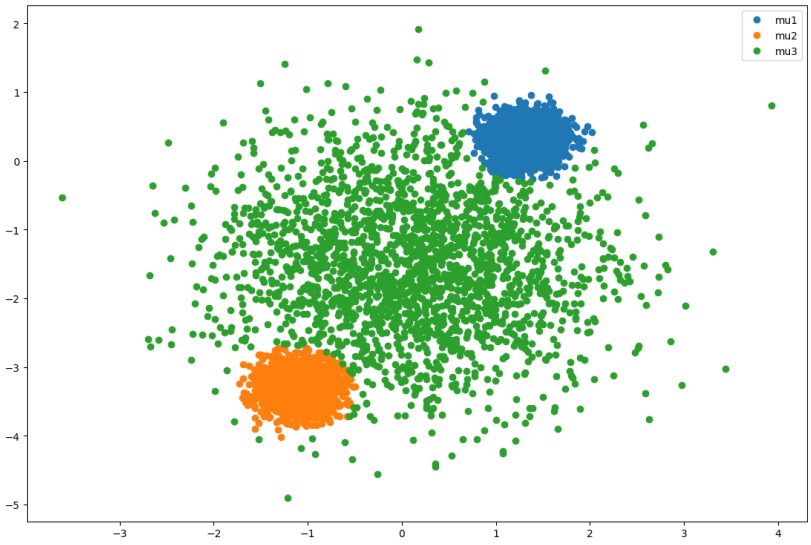


Figure 112D MoG Scatter Plot; K=3

For $K = 4$, validation loss is 1.7055.

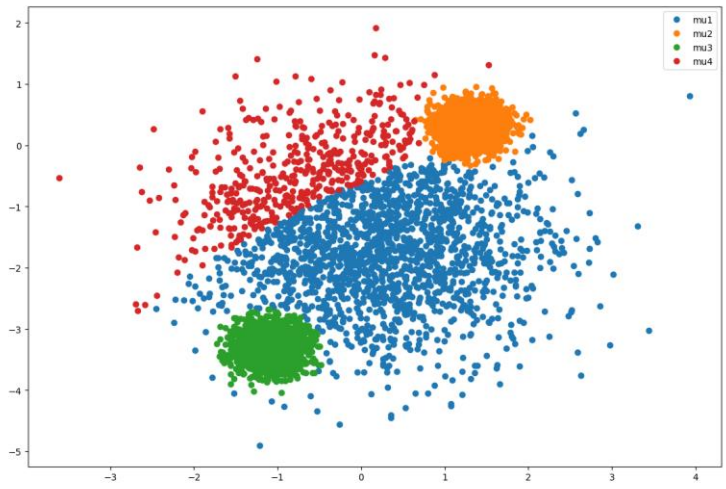


Figure 12 2D MoG Scatter Plot; K=4

For $K = 5$, validation loss is 1.7057.

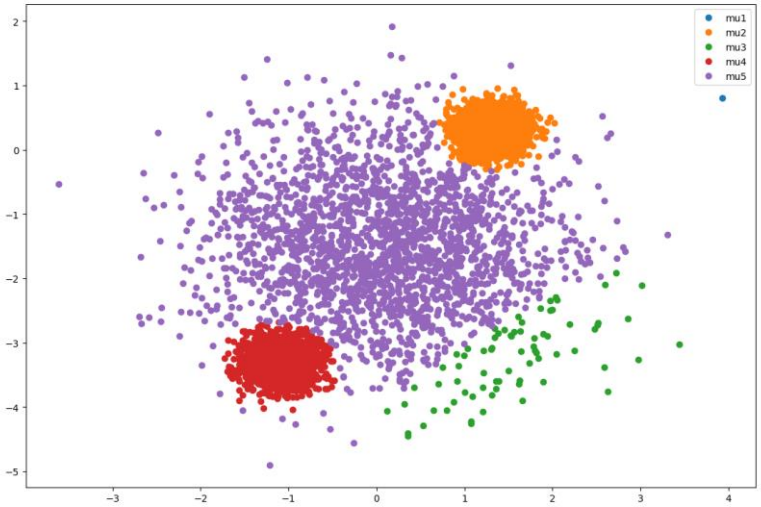


Figure 13 2D MoG Scatter Plot; K=5

At K = 3, the validation converges to around 1.7057. From the scatter plot K=5, we see that some of the clusters degenerate into 1 single point. Thus, the best K is K=3.

Q2.2.4

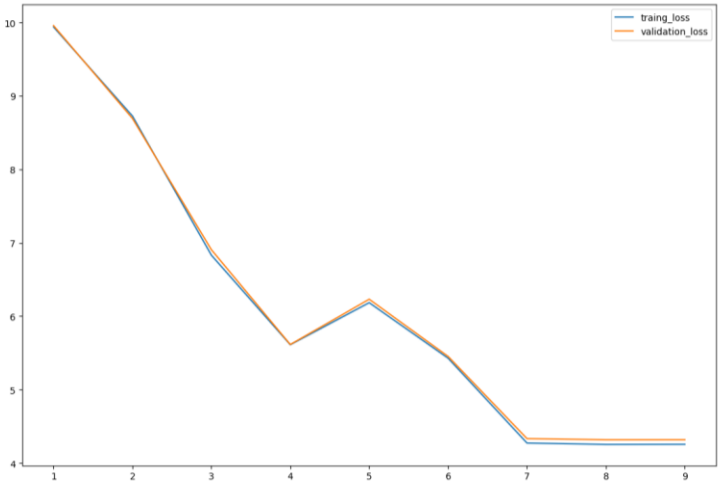


Figure 14 Training and Validation Negative Log-likelihood vs K (K-means)

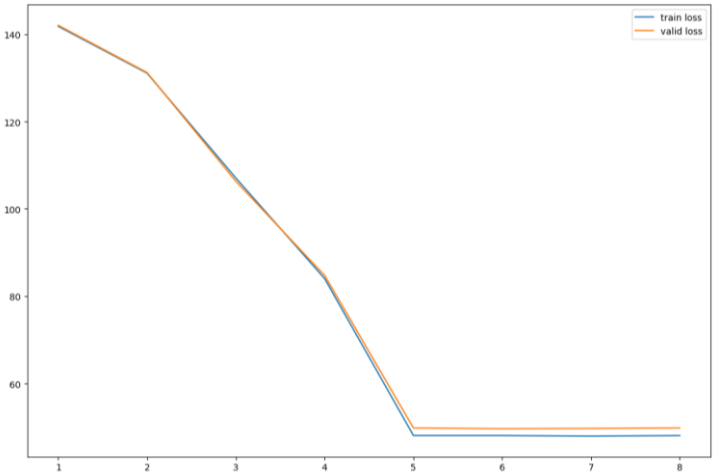


Figure 15 Training and Validation Negative Log-likelihood vs K (MoG)

In K- means, the training loss and validation seemingly converges at 7. However, depending on how the cluster points are initialized, we may end up in situations where the loss may increase (e.g. K=6). Moreover, minimizing the square distance does not necessarily tell where the clusters are located.

To locate the clusters, we should use the Mixture of Gaussian model. In MoG, the training loss and validation loss converges at K =5, with the loss converges to 48. Thus, adding more clusters will not increase the likelihood of observing a data within the additional cluster. Therefore, there are 5 clusters in the dataset.

In the MoG loss vs k plot, the loss remains the same at least until K=15. Also, even though different initialization may lead to different optimum solution, we verified that K =5 is point the convergence by running multiple tests at K=4, 5, 6 (that is, around the point of convergence).

Q3.1.1

$$\log p(x) = \log \int_{s_n} p(s_n) p(x_n|s_n) ds_n$$

Given that

$$p(s) = N(s; 0, I)$$

$$p(x|s) = N(x; Ws + \mu, \Psi)$$

Therefore, from equation 3 given in the handout,

$$P(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Lambda^{-1}) \quad (1)$$
$$P(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}; \mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \quad (2)$$
$$P(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\Lambda^{-1}\mathbf{A}^T) \quad (3)$$
$$P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}; \Sigma\{\mathbf{A}^T\mathbf{L}(\mathbf{y} - \mathbf{b}) + \Lambda\boldsymbol{\mu}, \Sigma) \quad (4)$$
$$\text{where, } \Sigma = (\Lambda + \mathbf{A}^T\mathbf{L}\mathbf{A})^{-1} \quad (5)$$

In our case, we have that $y = x,\ x = s,\ b = \mu,\ L^{-1} = \Psi,\ \Lambda^{-1} = I, \mu = 0, A = W$. Thus, by making the substitution, we get

$$p(x) = N(x; W0 + \mu, \Psi + WI^{-1}W^T)$$

Therefore,

$$p(x) = N(x; \mu, \Psi + WW^T)$$

As required.

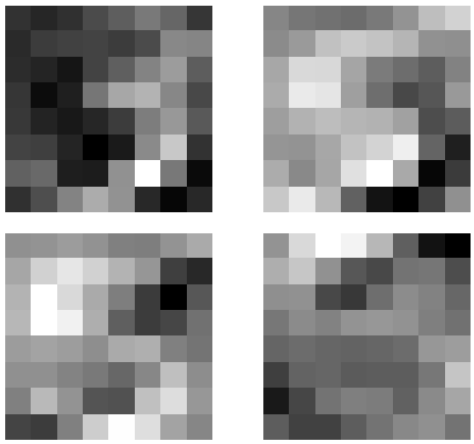
Q3.1.2

Based on the equation below

$$\log N(\boldsymbol{x};\mu,\Sigma) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log(\det\Sigma) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$$

The training, validation, and test log-likelihood is reported below.

Training loss (negative likelihood)	10.1367
Validation loss (negative likelihood)	9.86124
Test loss (negative likelihood)	10.5444



The Factor analysis has capture 4 latent causes as depicted above. First, the top left weight captures the general shape of “3” with an accented bottom. The top right weight captures the bottom part of “5”. The bottom left weight illustrates the figure of “5” with the left part more pronounced and the bottom right weight captures the top left corner.

Q3.1.3

Using PCA, the single principal component u_1 is obtained by solving the following eigenvalue-eigenvector problem.

$$Su_1 = \lambda u_1$$

Where sample covariance matrix S and the sample mean is $\bar{x}$ are given by

$$S = \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T, \bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

The eigenvalue λ is the maximum eigenvalue of the set of eigenvalues.

Using the built-in Tensorflow function *tf.self_adjoint_eig(S_PCA, name="eigenvector"),* we have obtained the following eigenvalue-eigenvector pairs.

λ_i	u_i
2.10714848e-06	[-7.07107306e-01 -7.07103431e-01 -1.97936571e-03]
1.95731175e+00	[7.07106233e-01 -7.07104564e-01 -1.97765953e-03]
9.99493408e+01	[1.23179052e-06 2.79801618e-03 -9.99996066e-01]

Since the principal component is the eigenvector with the maximum corresponding eigenvalue, the principal component is:

$$u_1 = \begin{bmatrix} 1.23179052\text{e}-06 \\ 2.79801618\text{e}-03 \\ -9.99996066\text{e}-01 \end{bmatrix}$$

We observe that the principal component has a magnitude$|u| = 1$, and value of 3rd dimension is much greater than the other 2. Therefore, the experiment agrees with our expectation that PCA learns in the x_3 direction (i.e. the maximum variance direction).

Using Factor Analysis,

$$W_{proj} \triangleq \Sigma W^T \Psi^{-1} = (I + W^T \Psi^{-1} W)^{-1} W^T \Psi^{-1}$$

Illustrates the direction that FA learns.

After running 10000 epochs, the following are the learnt results.

μ	[[-0.00641742 -0.00641738 -0.03771915]]
Ψ	[[8.98865073e-06 0.00000000e+00 0.00000000e+00] [0.00000000e+00 8.14545092e-06 0.00000000e+00] [0.00000000e+00 0.00000000e+00 2.03695267e+02]]
W	[[1.49340177] [1.50076866] [0.70065057]]

Thus,

$$W_{proj} = \begin{bmatrix} 3.1685975 * 10^{-1} \\ 3.51192564 * 10^{-1} \\ 6.55642651 * 10^{-9} \end{bmatrix}$$

The first 2 dimensions with similar magnitude dominates over the 3rd dimension. Thus, we conclude that Factor Analysis model learns the maximum correlation direction (i.e. $x_1 + x_2$ direction).