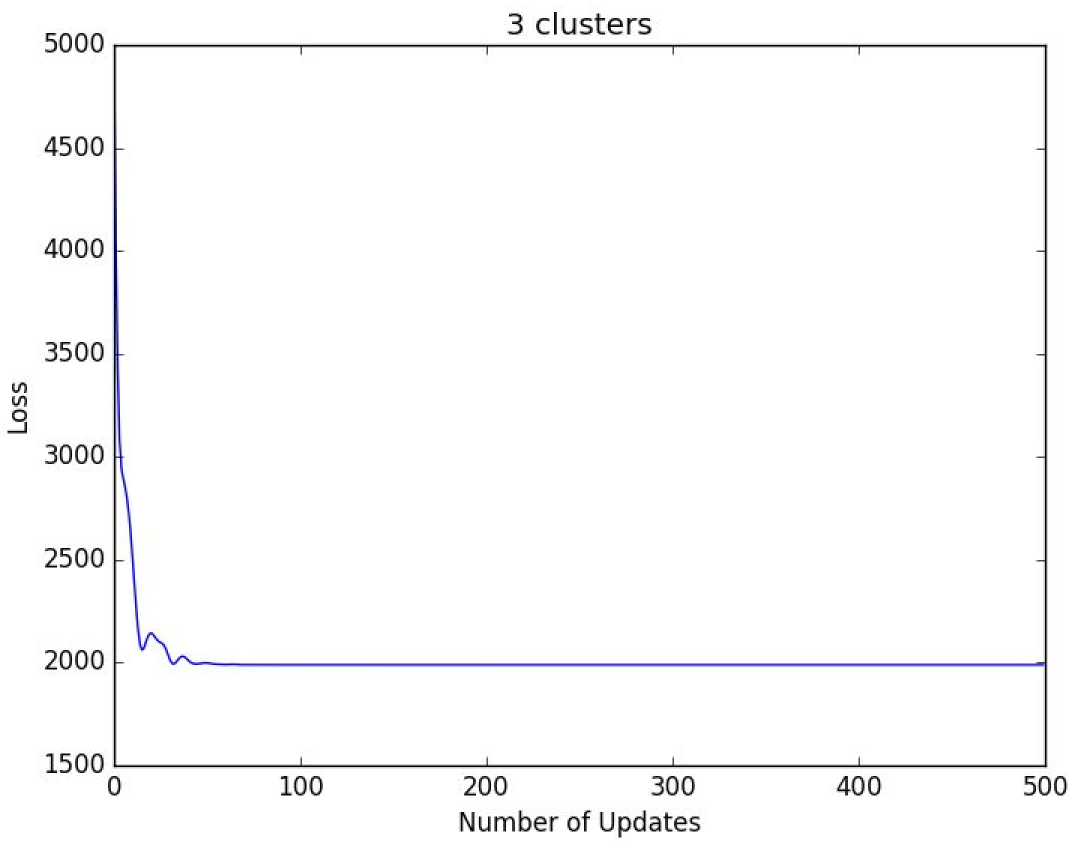
ECE521 Assignment 3 Report

Xiaoyang Guo 999578513
Xin Ge Gai 1000151131

Part 1.1.1
The loss function is not convex. If we cluster a data set to two optimum clusters K1 with C1 as the center and K2 with C2 as the center, other clustering with the same cost is possible such as assigning K1 to C2 and assigning K2 to C1. As the number of clusters goes up, there are more ways of clustering, but they all give the same loss. The k mean clustering algorithm will eventually converge to either a global minimum or local minimum.

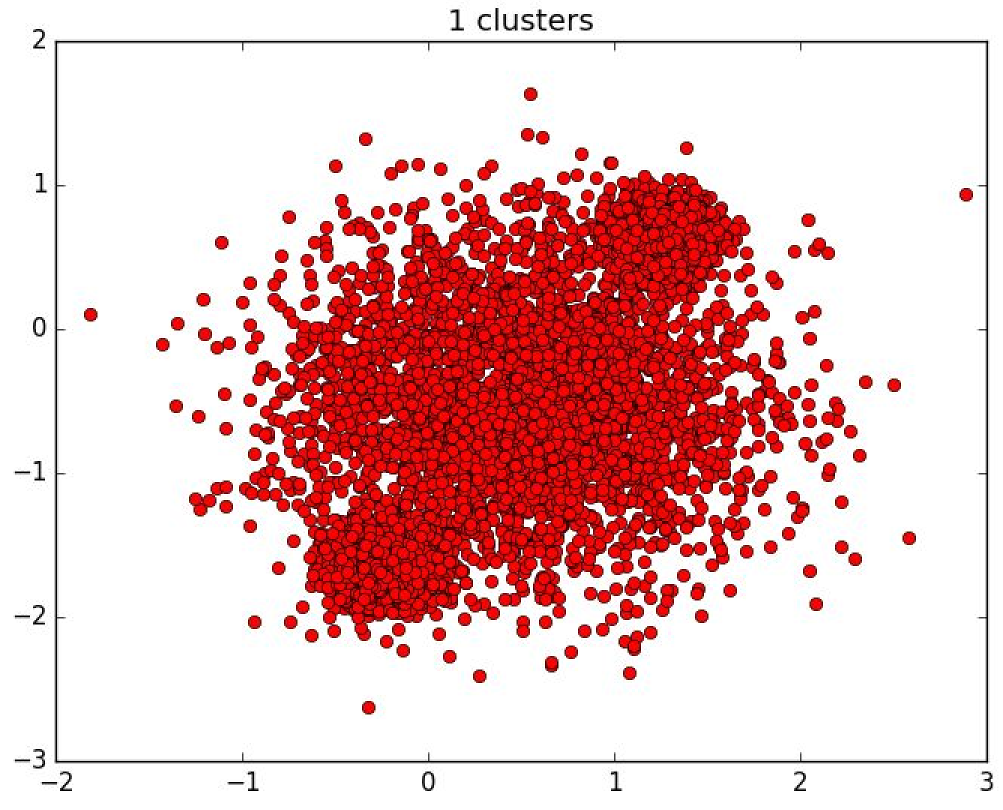
Part 1.1.2

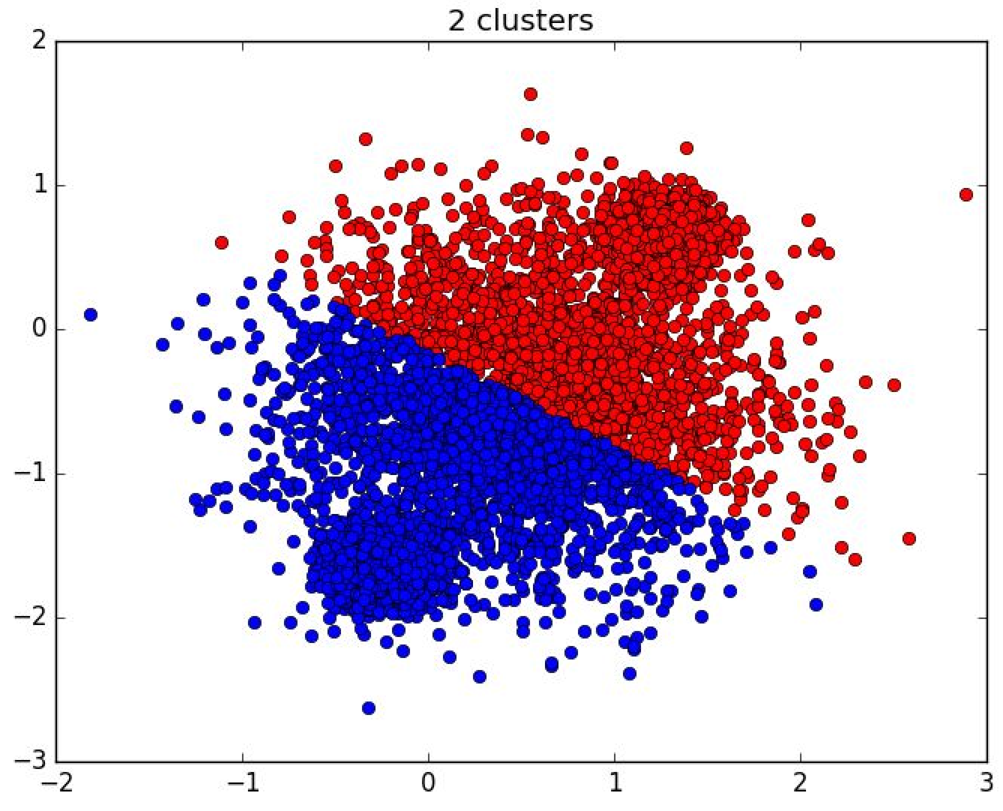


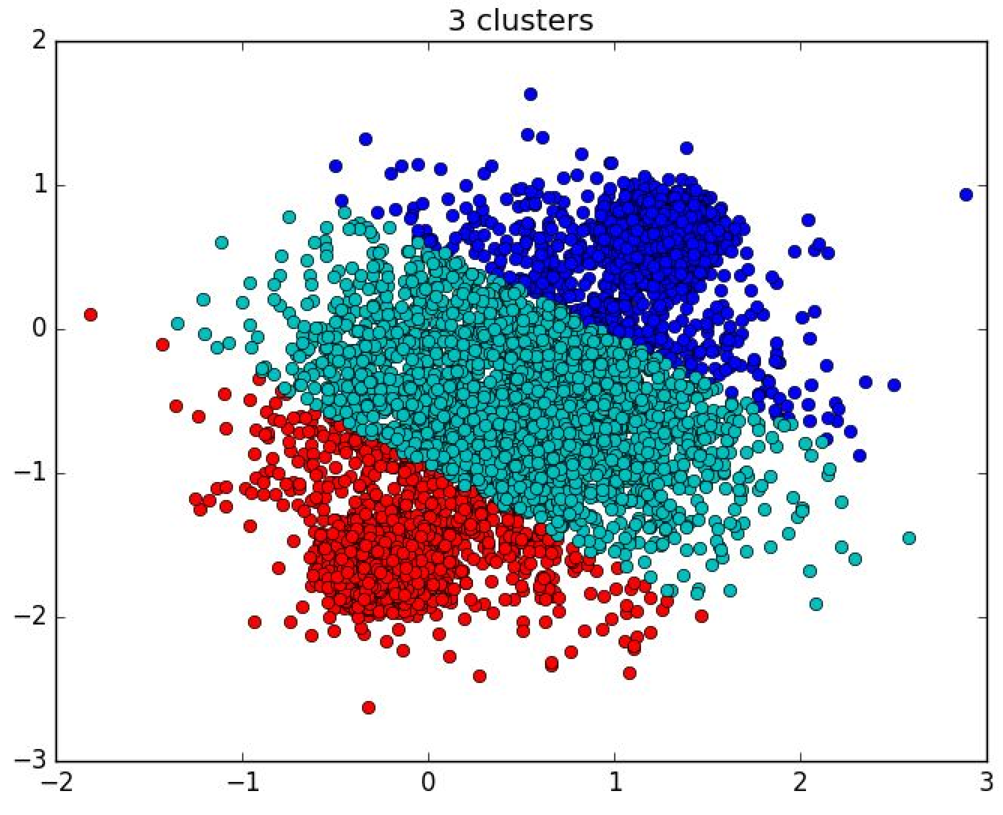
The model converges after 100 updates which conforms with the expectation.

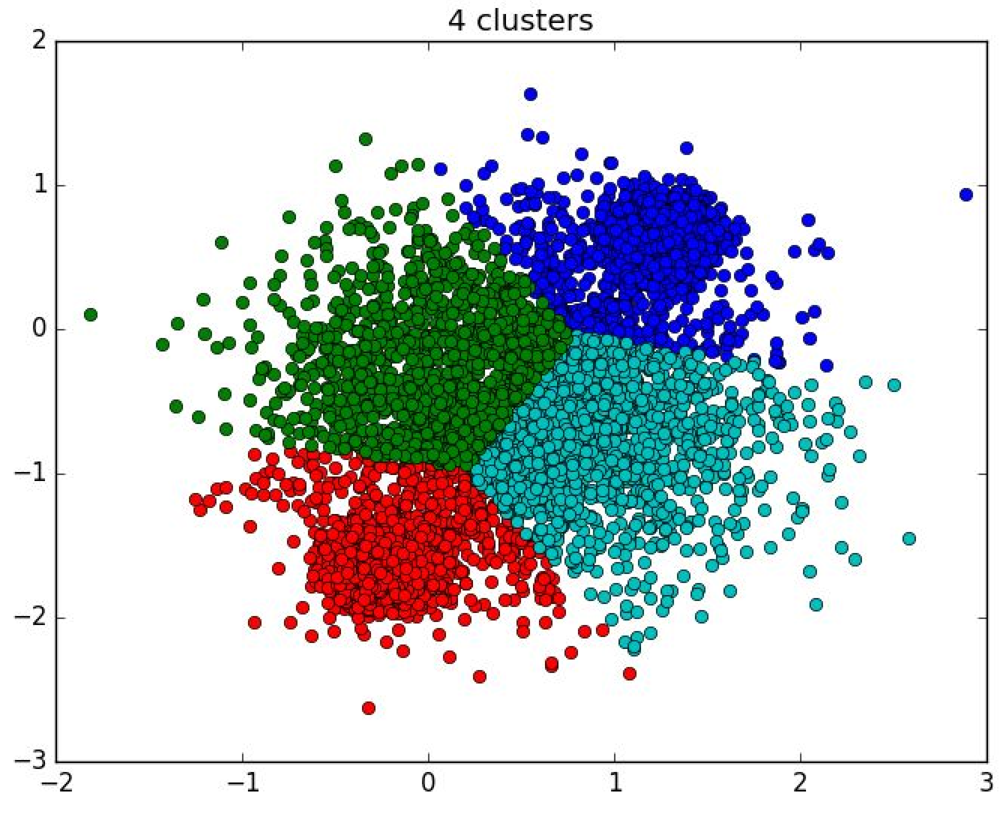
Part 1.1.3

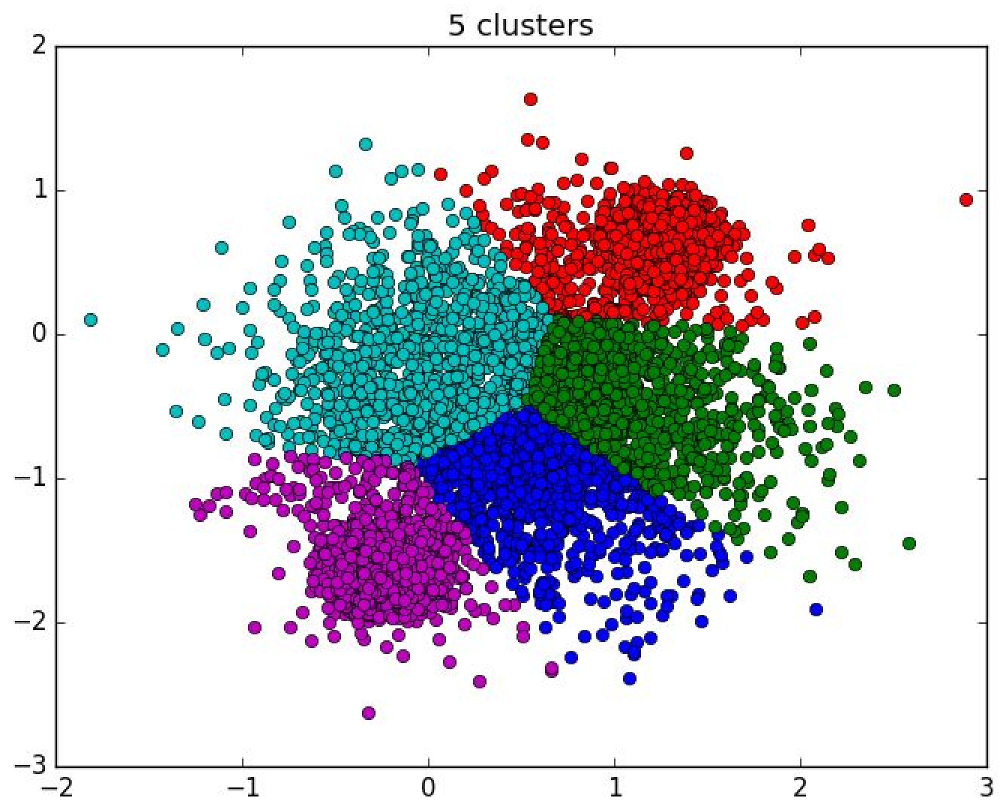
K	Percentage of data points in each cluster					Loss
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster5	
1	100	0	0	0	0	14967
2	49	51	0	0	0	3582
3	38	38	24	0	0	1989
4	12	14	37	37	0	1313
5	10	10	36	36	8	1108


1 clusters
2
1
0
−1
−2
−3
−2
−1
0
1
2
3


2 clusters
2
1
0
−1
−2
−3
−2
−1
0
1
2
3


3 clusters
2
1
0
−1
−2
−3
−2
−1
0
1
2
3


4 clusters
2
1
0
−1
−2
−3
−2
−1
0
1
2
3



Based on the experimental result, 5 clusters is be best choice since the cost is the lowest. From the graph it is easy to tell that with lower number of clusters, there are more points lying far away from the center.

Part 1.1.4

K	Validation Cost
1	5048
2	1193
3	660
4	428
5	365

From the experimental result, it is not hard to tell that 5 clusters gives be lowest validation cost, thus is the best number of clusters.

Part 2.1.1

$$P(z=k \mid x) = \frac{P(x, z=k)}{\sum_{z^*=1}^{K} p(x \mid z^*)} = \frac{P(x \mid z=k)\, P(z=k)}{\sum_{z^*=1}^{K} P(x \mid z^*)}$$

$$P(z) = \prod_{k=1}^{K} \pi_k^{z_k} \qquad P(x \mid z) = \prod_{k=1}^{K} \mathcal{N}(x \mid \mu_k, \sigma_k^2)^{z_k}$$

$$P(x, z) = \prod_{k=1}^{K} \left[\pi_k \mathcal{N}(x \mid \mu_k, \sigma_k^2)\right]^{z_k}$$

$$P(z=k \mid x) = \frac{\pi_k \mathcal{N}(x \mid \mu_k, \sigma_k^2)}{\sum_{k^*=1}^{K} \pi_{k^*} \mathcal{N}(x \mid \mu_{k^*}, \sigma_{k^*}^2)}$$

Part 2.1.2

$$\log \mathcal{N}(x \mid \mu_k, \sigma_k^2) = \log P(x \mid \mu_k, \sigma_k^2)$$

$$= \log \frac{1}{\sqrt{2\pi\sigma_k}} + \left(\frac{-(x-\mu_k)^2}{2\sigma_k^2}\right)$$

```
def log_prob_density(x, u, sigma):
  density = tf.sqrt(pi*sigma*2)
  log_prob = -(tf.to_float(tf.rank(x)))*tf.log(density)
  sqr_dist = Distance(x, u)
  log_prob -= sqr_dist / (2*sigma)
  return log_prob


def Distance(X, Y):
  Y_T = tf.transpose(Y)
  XX = tf.mul(X, X)
  YY = tf.mul(Y_T, Y_T)
  XX = tf.reshape(tf.reduce_sum(XX, 1), [-1, 1])
  YY = tf.reshape(tf.reduce_sum(YY, 0), [1, -1])
  XY = tf.mul(2.0, tf.matmul(X, Y_T))
  return XX - XY + YY
```

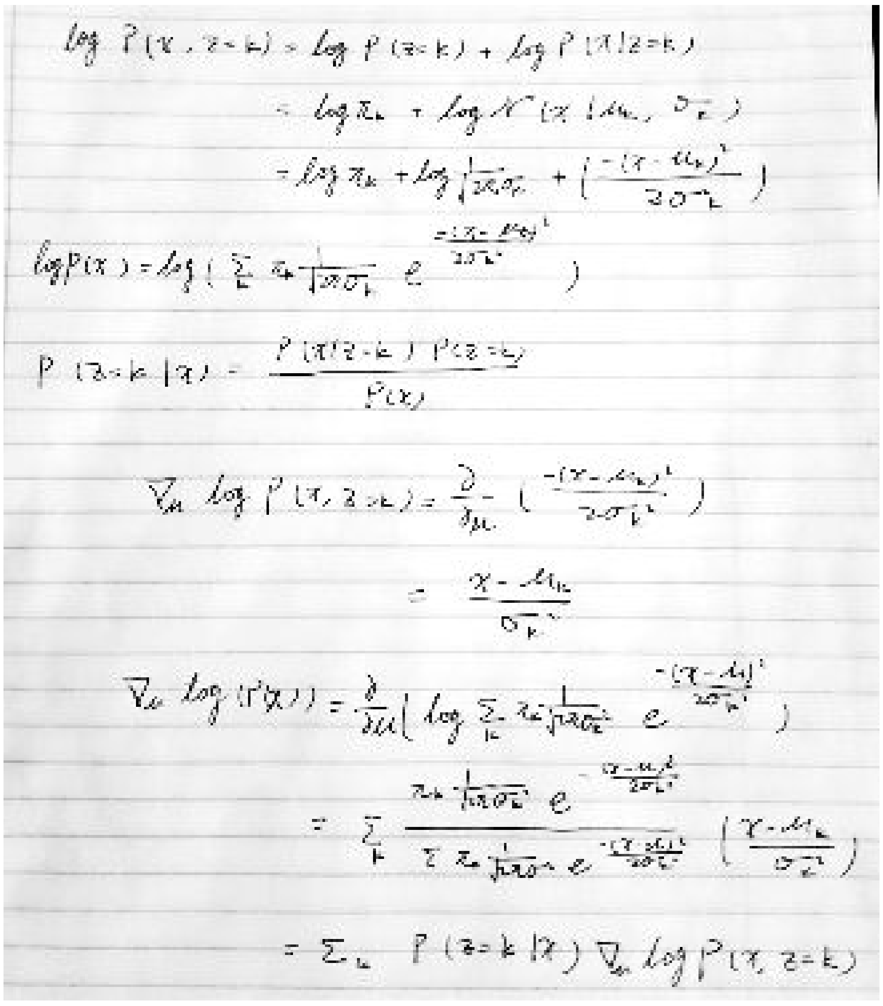
Part 2.1.3

Even though tf.reduce_sum can be used, but it lacks precision since the calculation would involve floating point numbers which has limited precision. The calculation of log probability is very sensitive to errors, thus the reduce_logsumexp function is a better choice since it provides high precision.

```
def log_cluster_probability (x, log_pz, u, sigma):
  log_p_x_given_z = log_prob_density(x, u, sigma)
  sum = log_pz + tf.transpose (log_p_x_given_z)
  density = reduce_logsumexp(sum, 0)
  log_p_z_given_x = sum / density

  return log_p_z_given_x
```

Part 2.2.1



$$\log P(x, z=k) = \log P(z=k) + \log P(x|z=k)$$

$$= \log \pi_k + \log \mathcal{N}(x | \mu_k, \sigma_k)$$

$$= \log \pi_k + \log \frac{1}{\sqrt{2\pi\sigma_k}} + \left(\frac{-(x-\mu_k)^2}{2\sigma_k}\right)$$

$$\log P(x) = \log\left(\sum_k \pi_k \frac{1}{\sqrt{2\pi\sigma_k}} e^{\frac{-(x-\mu_k)^2}{2\sigma_k^2}}\right)$$

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

$$\nabla_\mu \log P(x, z=k) = \frac{\partial}{\partial\mu}\left(\frac{-(x-\mu_k)^2}{2\sigma_k^2}\right)$$

$$= \frac{x-\mu_k}{\sigma_k^2}$$

$$\nabla_\mu \log(P(x)) = \frac{\partial}{\partial\mu}\left(\log \sum_k \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{\frac{-(x-\mu_k)^2}{2\sigma_k^2}}\right)$$

$$= \sum_k \frac{\pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}}{\sum \pi_k \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu_k)^2}{2\sigma_k^2}}} \left(\frac{x-\mu_k}{\sigma_k^2}\right)$$

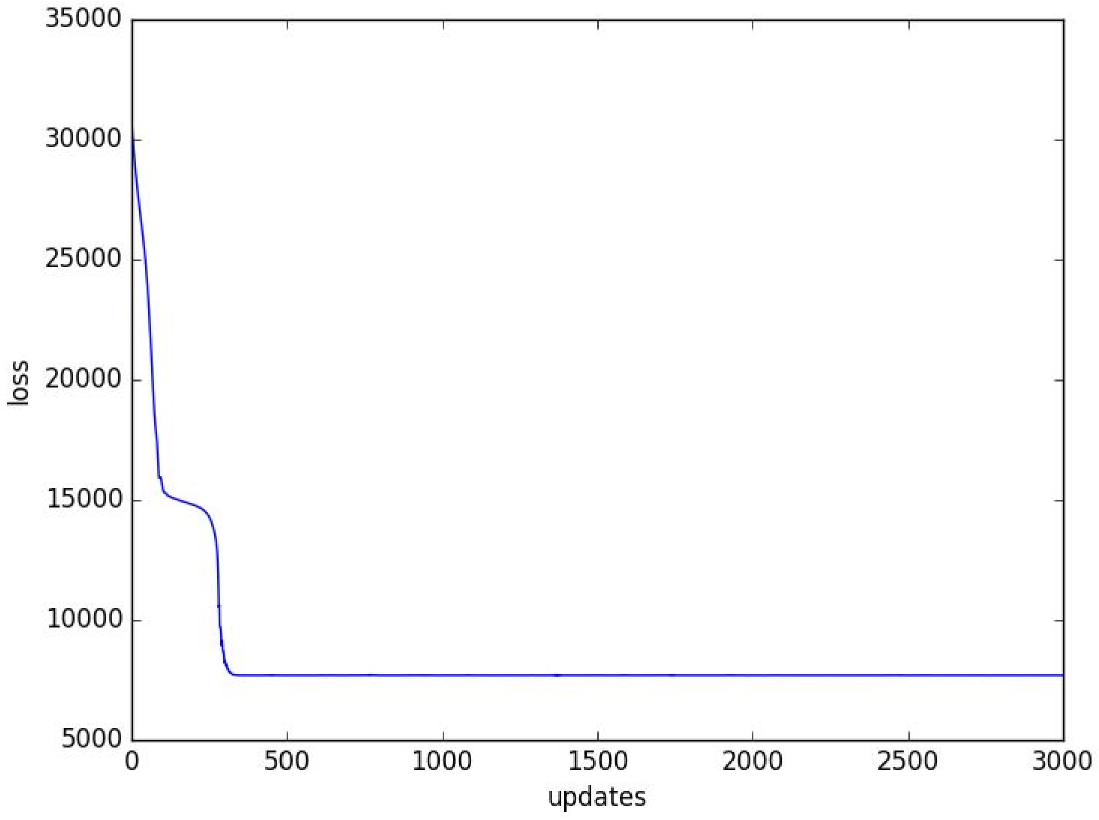
$$= \sum_k P(z=k|x)\,\nabla_\mu \log P(x, z=k)$$

Part 2.2.2

Best parameters found:

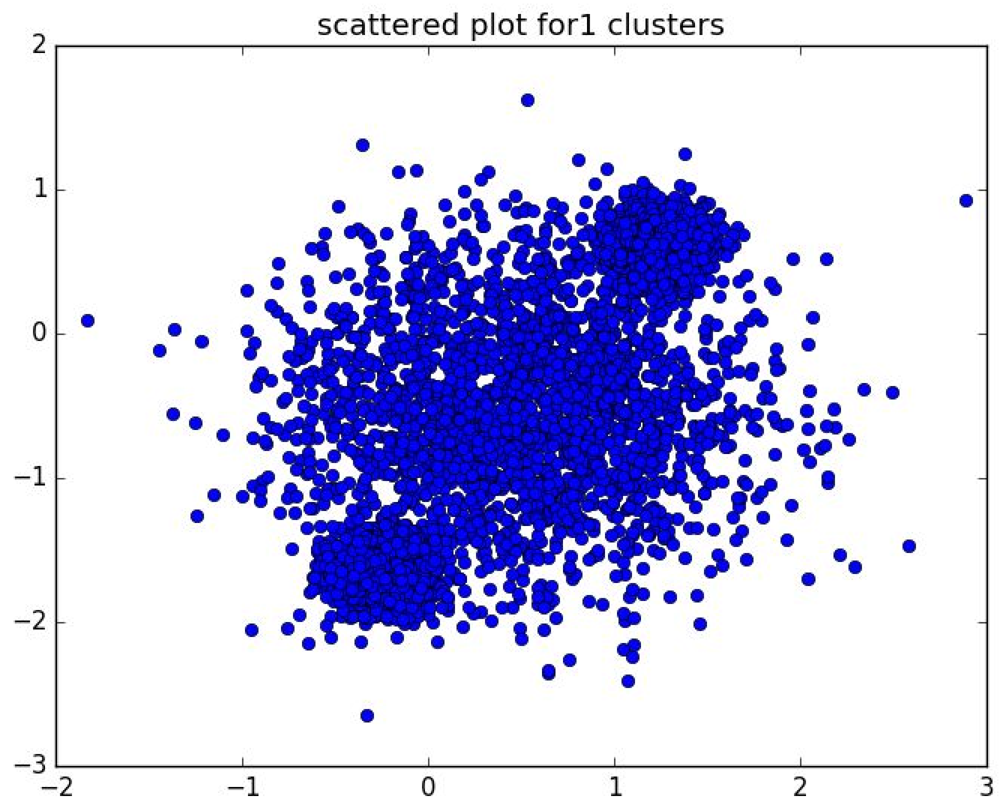
learning rate: 0.01

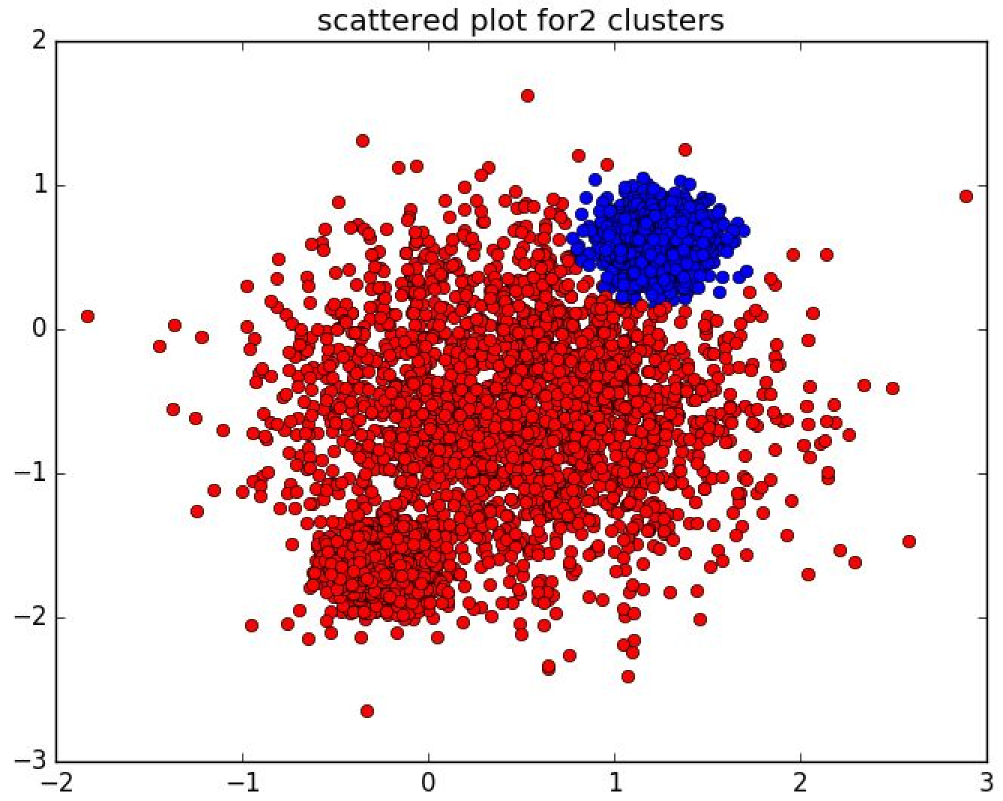
	Cluster 1	Cluster 2	Cluster 3
Mean	0.50371116 -0.51532578	1.24755263 0.63023138	-0.24950066 -1.62478387
Covariance	0.38389751	0.01496186	0.0149125
Log pi	-1.09444439	-1.09838033	-1.10303056

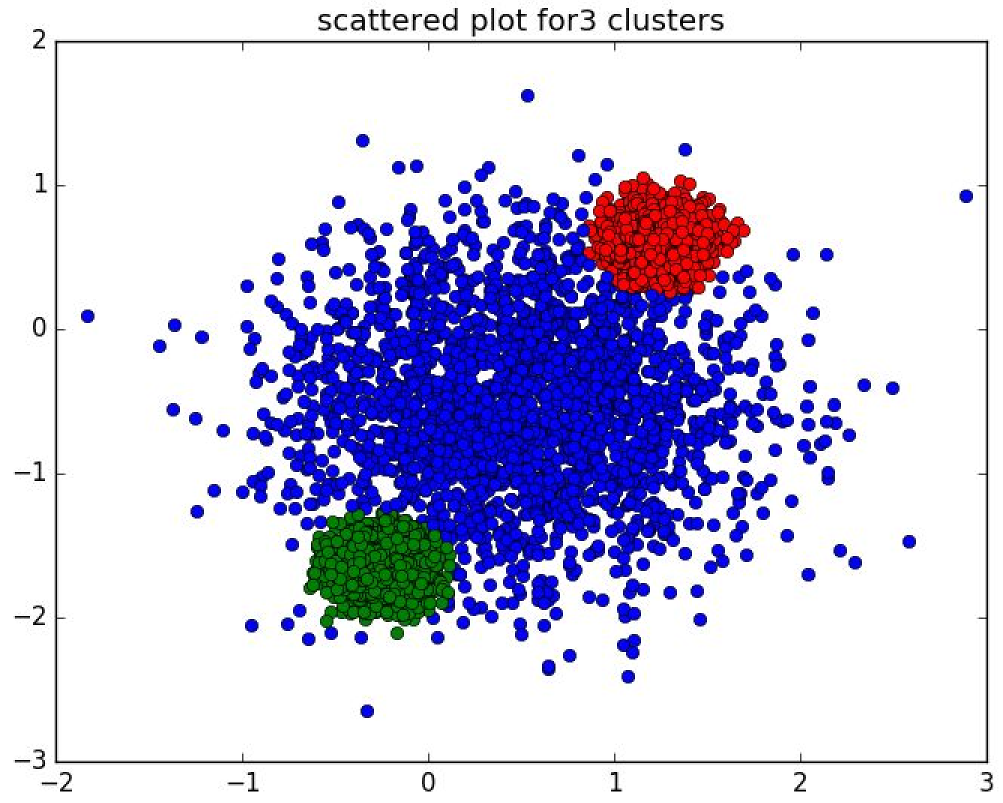


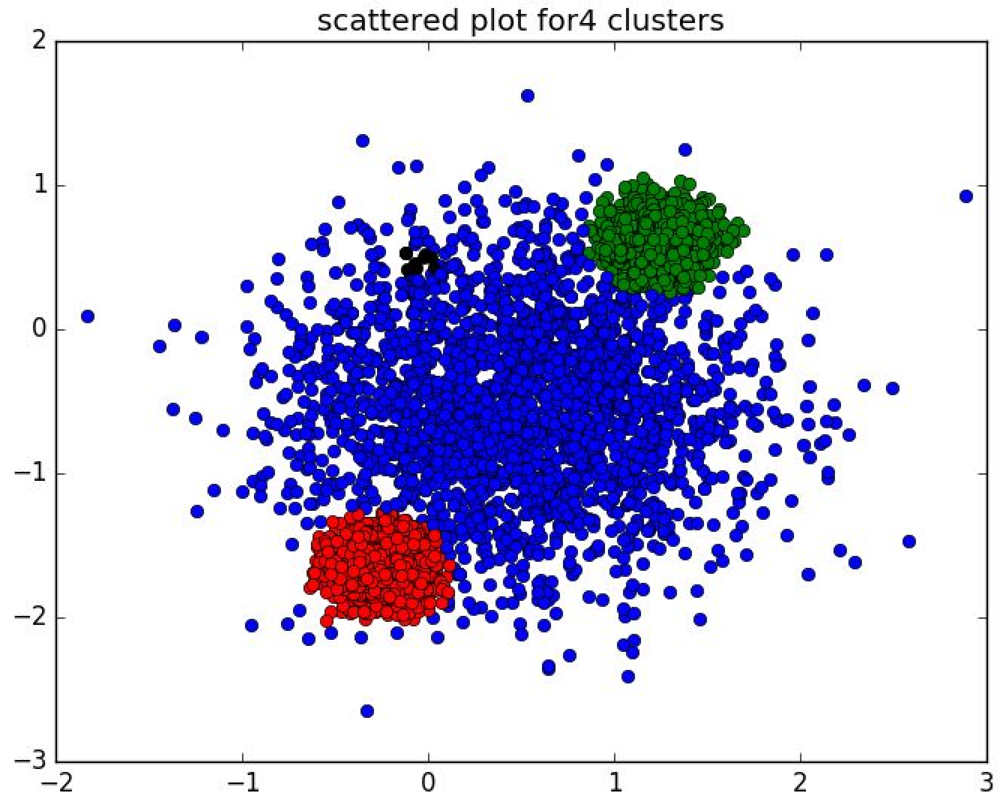
Part 2.2.3

K	Validation Loss
1	2556
2	2558
3	3129
4	4928
5	5130

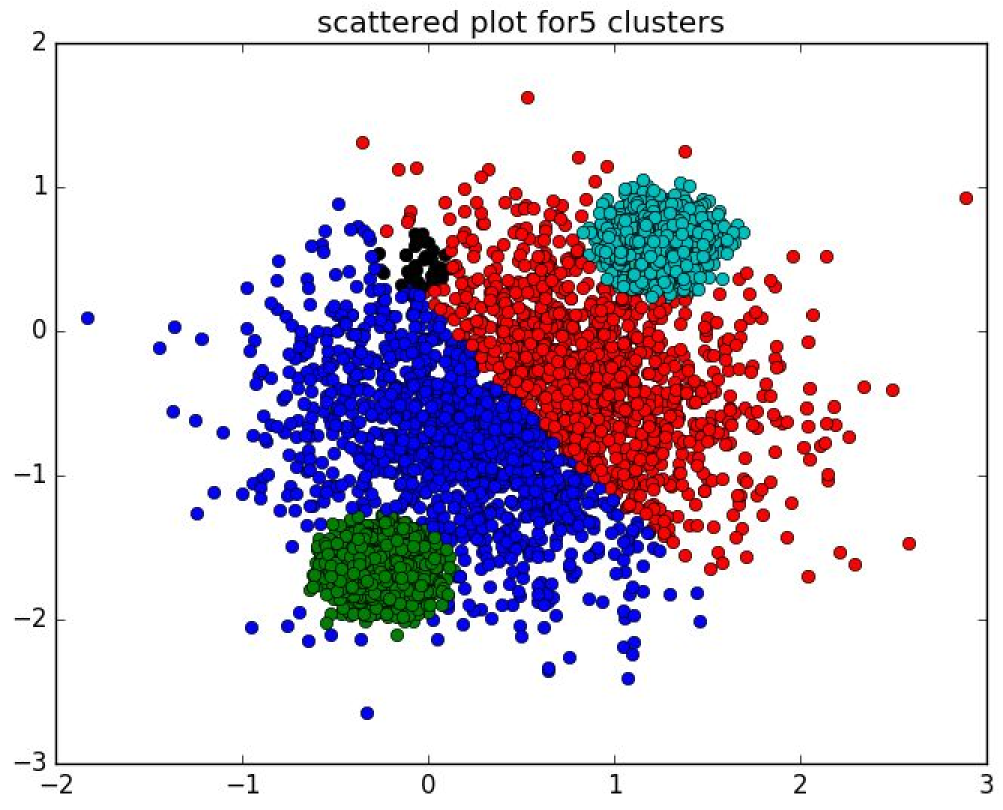



scattered plot for2 clusters
2
1
0
−1
−2
−3
−2
−1
0
1
2
3


scattered plot for3 clusters
2
1
0
−1
−2
−3
−2
−1
0
1
2
3



scattered plot for4 clusters
2
1
0
−1
−2
−3
−2
−1
0
1
2
3



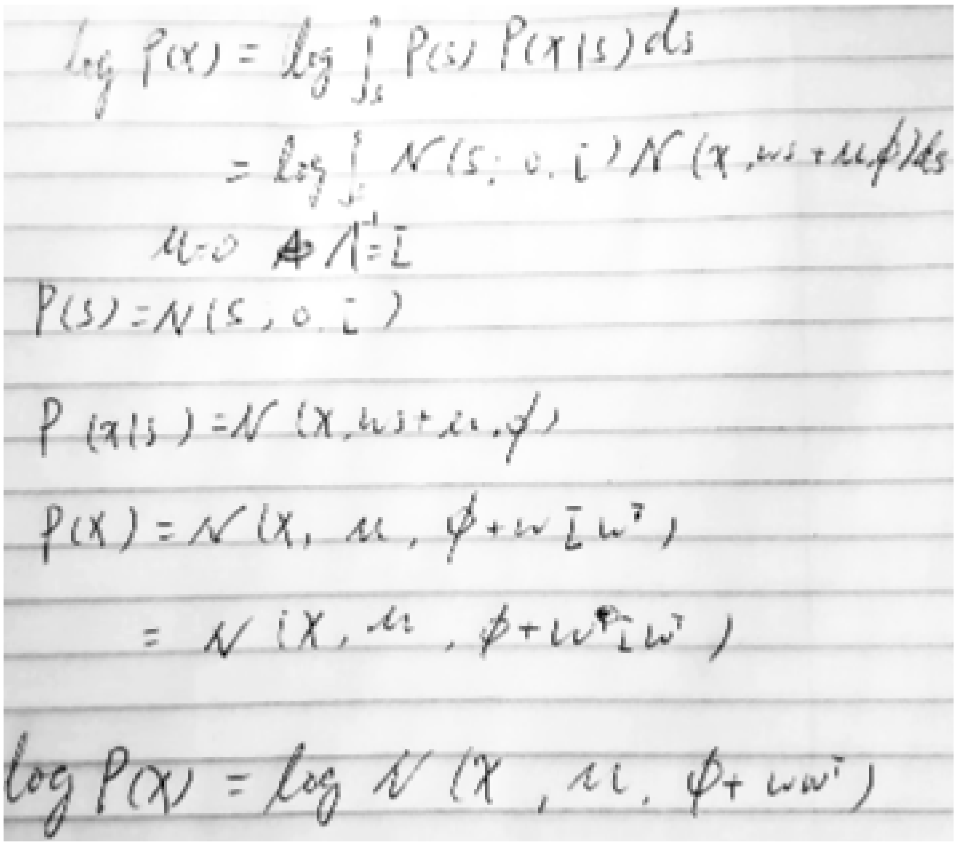
scattered plot for5 clusters
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

Based on the experimental result, 5 clusters is the best since it has the lowest validation loss.

Part 2.2.4

The k-mean model has the lowest loss when k is 4, and the MOG model has the highest probability when k is 5. Since MOG model gives far less fluctuation for choosing the k value, 5 clusters is more probable.

Part 3.1.1



$$\log P(x) = \log \int_s P(s)\, P(x|s)\, ds$$

$$= \log \int_s \mathcal{N}(s; 0, I)\, \mathcal{N}(x, ws + \mu, \phi)\, ds$$

$\mu = 0$ ~~A~~ $\Lambda^{-1} = I$

$P(s) = \mathcal{N}(s, 0, I)$

$P(x|s) = \mathcal{N}(x, ws + \mu, \phi)$

$P(x) = \mathcal{N}(x, \mu, \phi + w I w^T)$

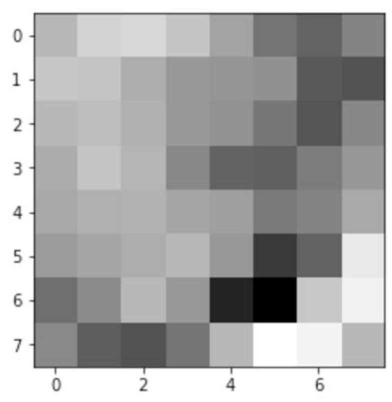
$= \mathcal{N}(x, \mu, \phi + w^T I w^T)$

$\log P(x) = \log \mathcal{N}(x, \mu, \phi + w w^T)$

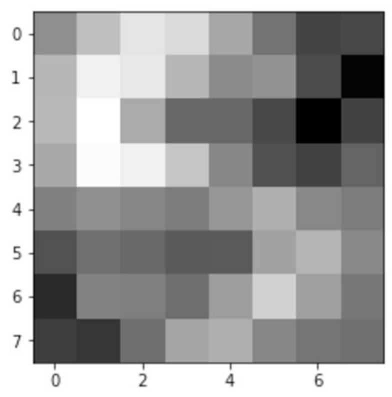
Part 3.1.2

```
pz = tf.Variable(tf.zeros([1,k]))
log_P = logsoftmax(pz)
sigma = tf.Variable(tf.ones([k, 1])*(-3))
weight = tf.Variable(tf.random_normal([k, k], mean=0, stddev=0.01))
log_dtm = 2.0 * tf.reduce_sum(tf.log(tf.diag(tf.cholesky(weight))))
var = tf.matrix_diag(np.random.rand(k, k))
sigma = var + log_dtm
exp_sigma = tf.exp(sigma)
```

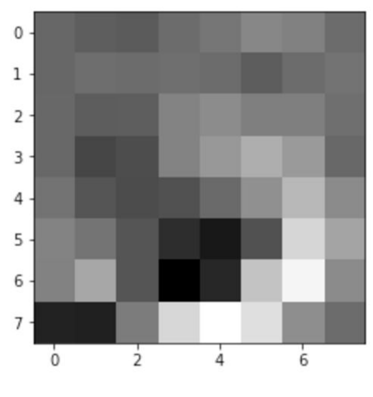
Train:



Validation:



Test:



Darker block implies big K while lighter block implies smaller k.