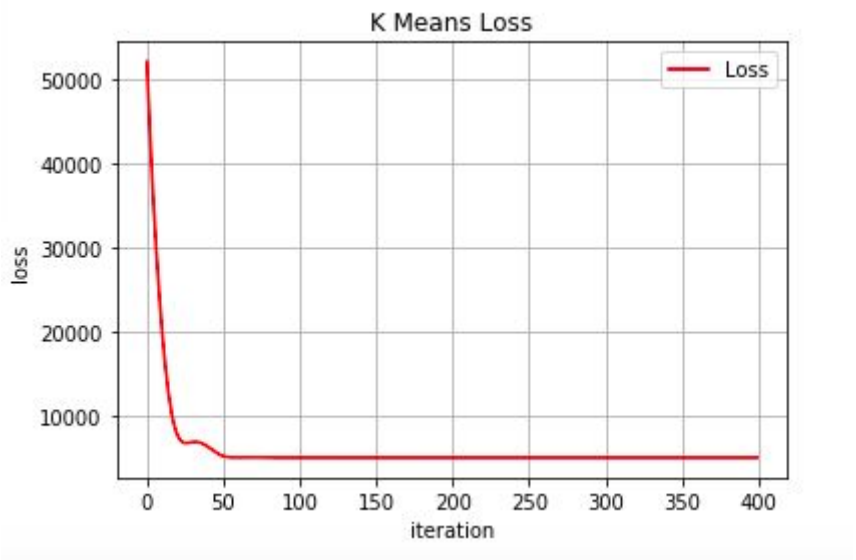
1. K-means [9 pt.]

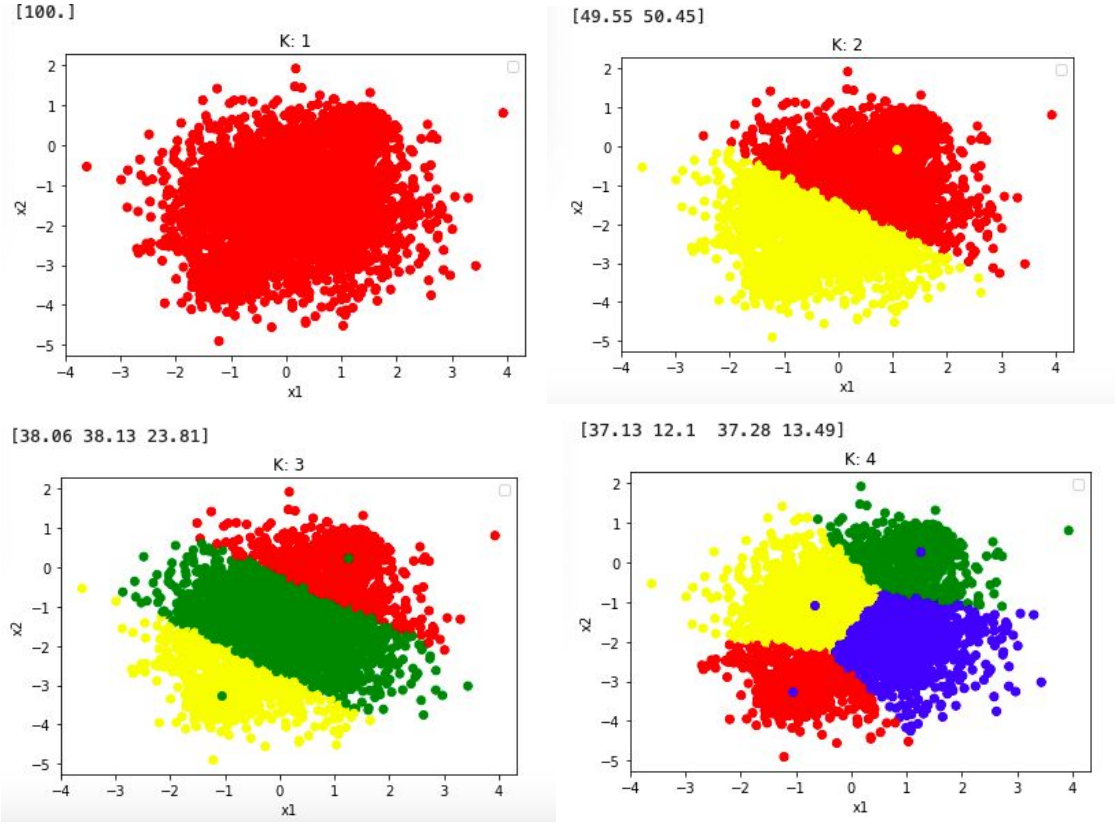
1.1 Learning K-means [4 pt.]

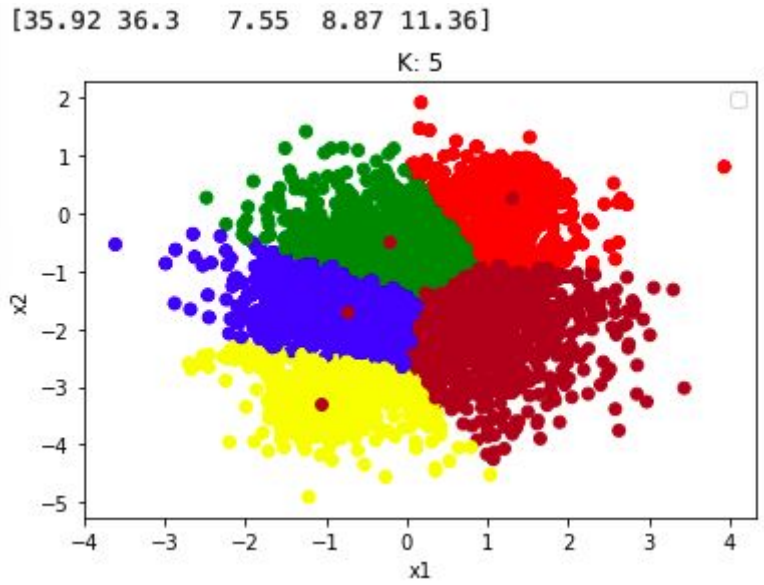
1.



Observation: This is a graph of K means loss vs the number of updates. With 400 updates, the loss decreases shapely within 50 updates and becomes stable after 50 updates.

2.



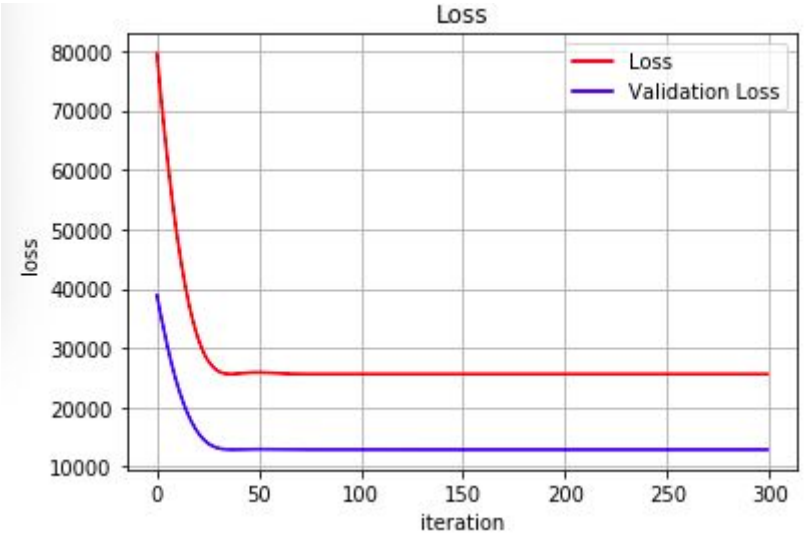


Observation and discussion:

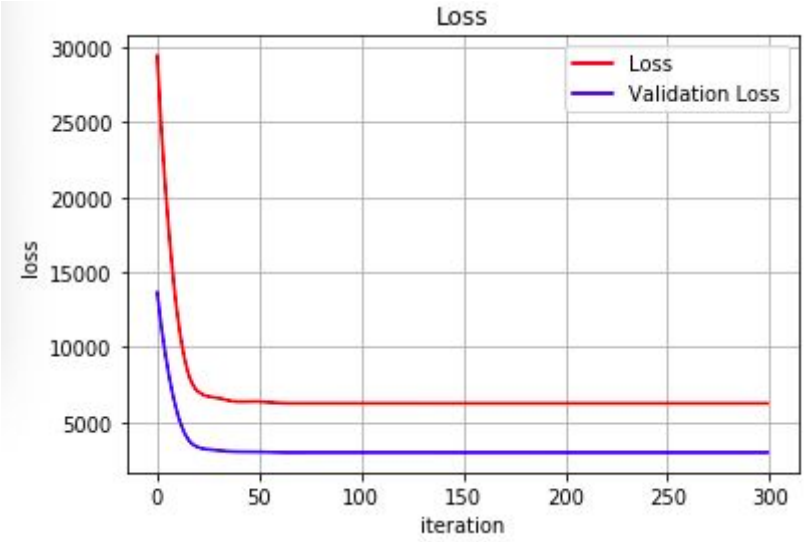
After getting five 2D plots from different clusters, we found that both K = 4 and K = 5 have good cohesion and separation for K mean algorithm. From these two graphs, each data was centered at the cluster center from the graph. They dont have the data point far away from the cluster center. Therefore, we thought that the 4 or 5 cluster were the best.

3. Hold ⅓ of the data our for validation.

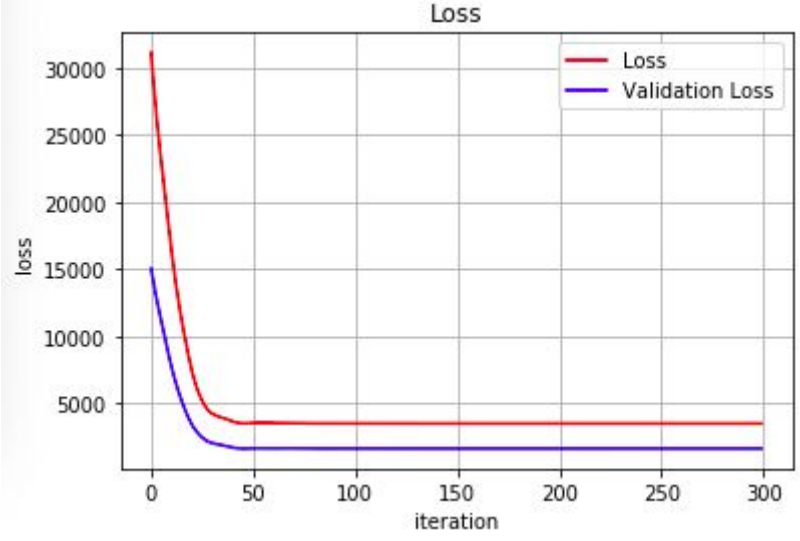
K = 1:



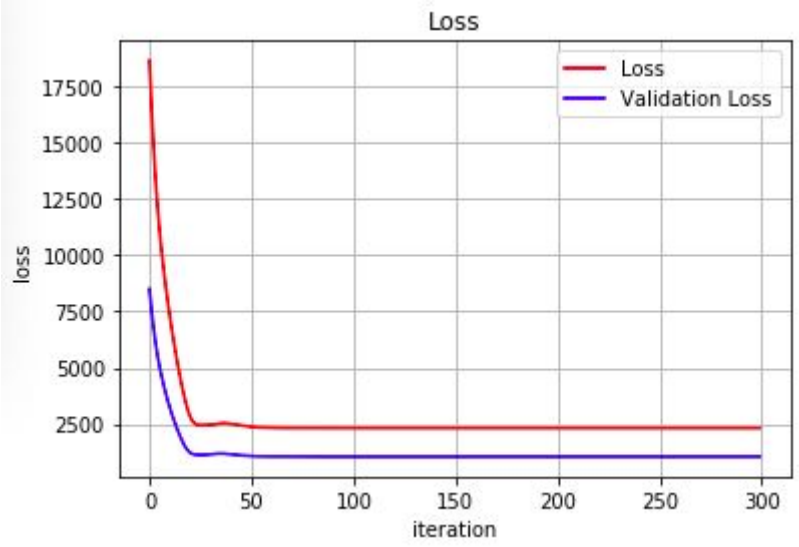
K = 2:



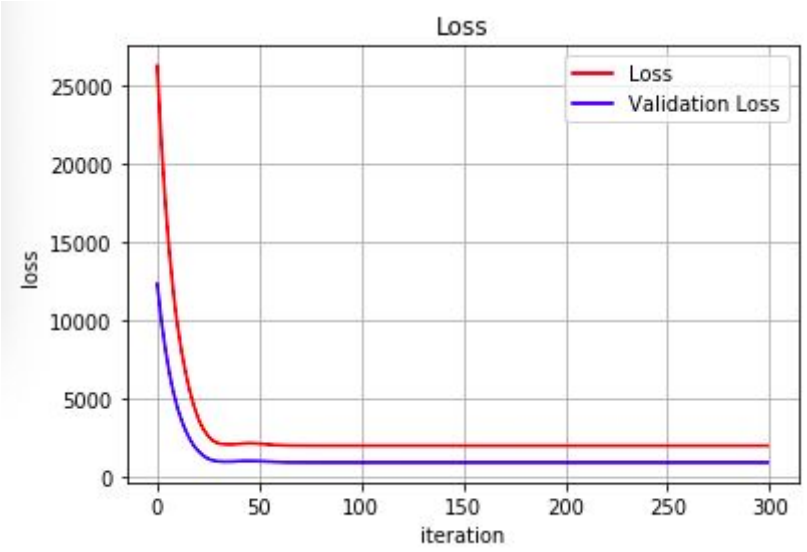
K = 3:



K = 4:



K = 5:



Observation and discussion:

From the above 5 graphs, the first three graphs has more validation loss than the last two and the last two has the similar validation loss for K means algorithm. Therefore, K = 4 is the turning point for K means algorithm. So I think that 4 clusters is the best.

2. Mixtures of Gaussians [16 pt.]

2.1 The Gaussian cluster mode [7 pt.]

1. Modify the K-means distance function.

```
# Distance function for GMM
def distanceFunc(X, MU):
    t1 = -2*tf.matmul(X, MU, transpose_a=False, transpose_b=True)
    sqX = tf.reshape(tf.reduce_sum(tf.square(X), axis=1), [-1, 1])
    sqMU = tf.reshape(tf.reduce_sum(tf.square(MU), axis=1), [1, -1])
    ret = t1 + sqX + sqMU
    return ret

def log_GaussPDF(X, mu, sigma):
    dist_pr = distanceFunc(X, mu)
    dist_sigma = -1*tf.div(dist_pr,tf.transpose(2*sigma))
    e = -1*tf.log((2*math.pi)**(dim/2)*sigma)
    ret = tf.transpose(e) + dist_sigma
    return ret

def log_posterior(log_PDF, log_pi):
    p_ = tf.add(log_PDF,log_pi)
    return hlp.logsoftmax(p_)
```

2. TF log Gasussian pdf function.

```
def Loss(K):
    N, D = num_pts, dim
    X = tf.placeholder(tf.float32, shape=(N,D),name="X_v")
    MU = tf.get_variable(name="MU_v",initializer=tf.random_normal(shape=[K,D])
    sigma = tf.get_variable(shape=(K,1),name="sigma_v")
    pi = tf.get_variable(shape(1,K),name="pi_v")
    sigma_e = tf.exp(sigma)
    log_PDF = log_GaussPDF(X,MU,sigma_e)
    add = hlp.reduce_logsumexp(hlp.logsoftmax(pi)+log_PDF)
    loss = tf.reduce_sum(-1*add)
    summ = hlp.logsoftmax(log_PDF)
    return MU, X, loss, sigma, pi, summ
```

Discussion:

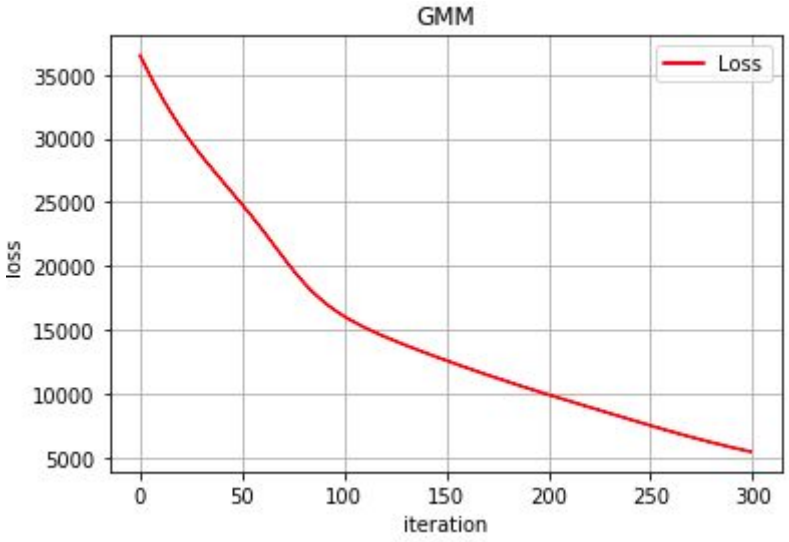
$$log \sum_{n=1}^{N} exp\{x_n\} = a + log \sum_{n=1}^{N} exp\{x_n - a\}$$

$$a = \{max\}_n x_n$$

Therefore, this ensures that the maximum value for the exponent is 0. This technique never overflows, and even the rest of the underflows get a reasonable value.

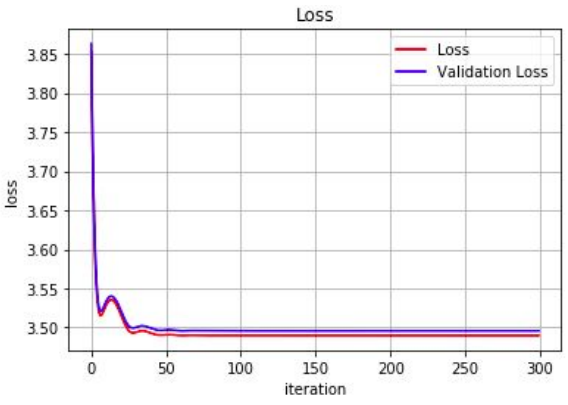
2.2 Learning the MoG [7 pt.]

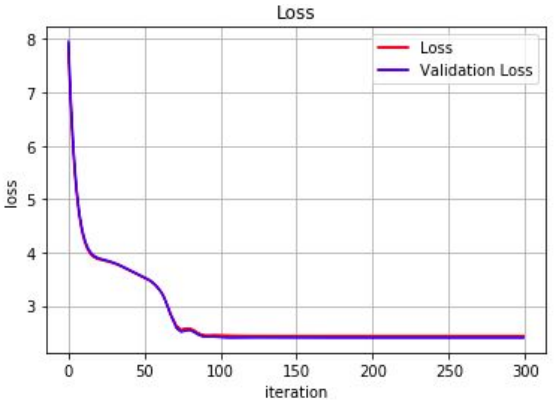
1. K = 3 loss vs updates from learning MoG

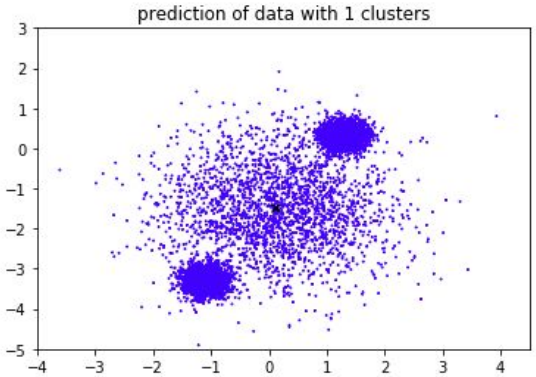


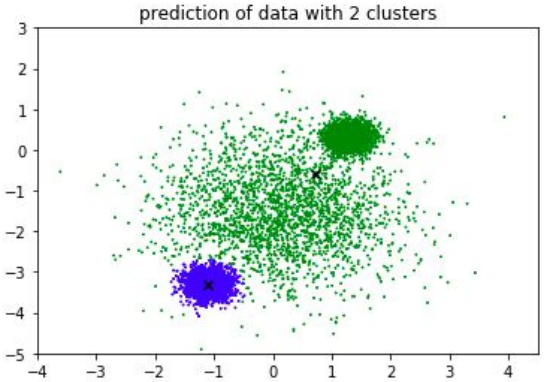
Observation and Discussion: This is a graph of MoG loss vs the number of updates. With 300 updates, the loss decreases persistently and it has an elbow turning point at around 100 updates.

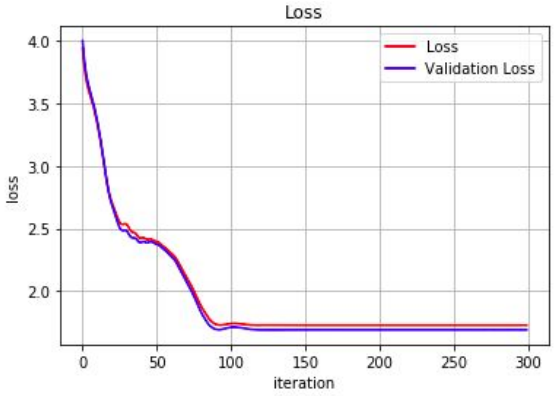
2.

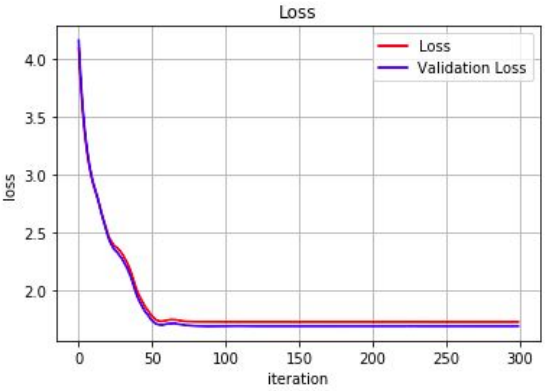


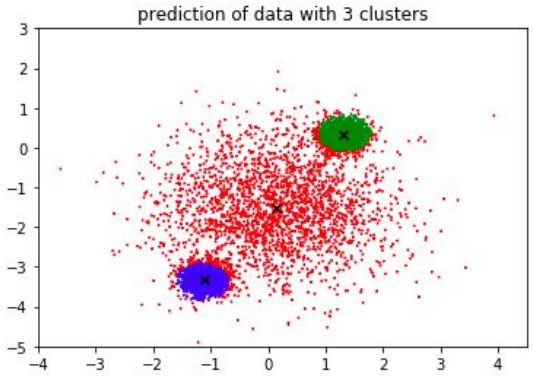


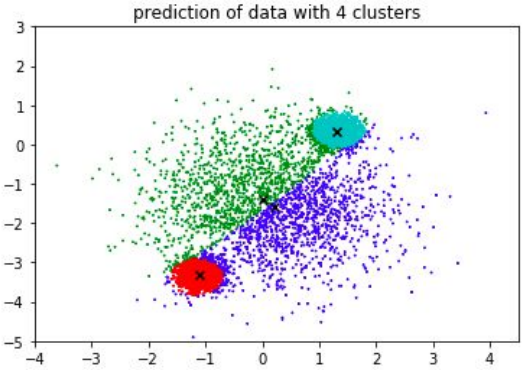


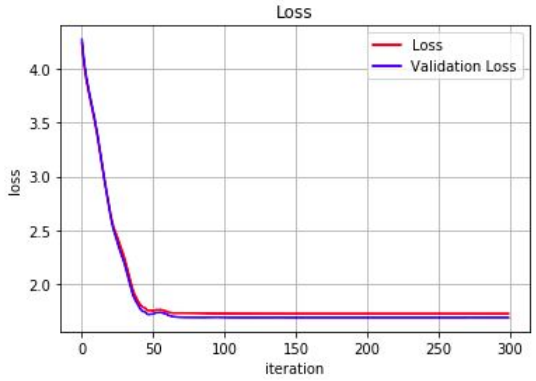


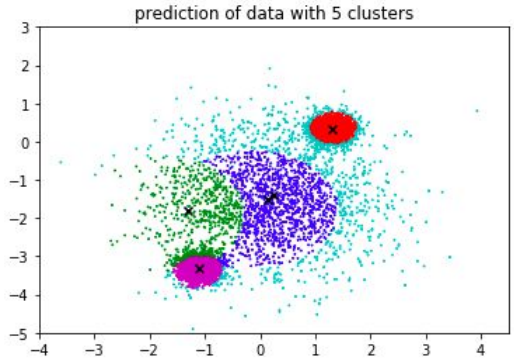








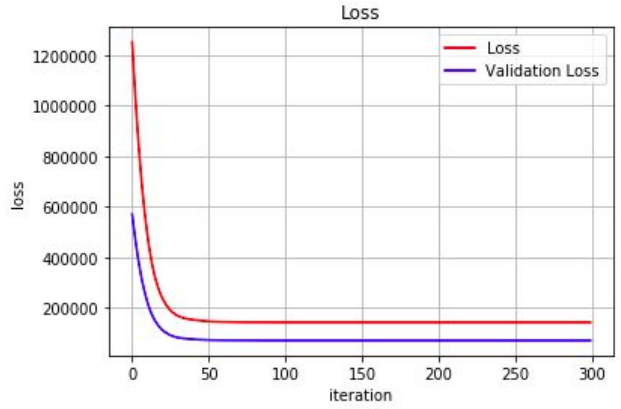


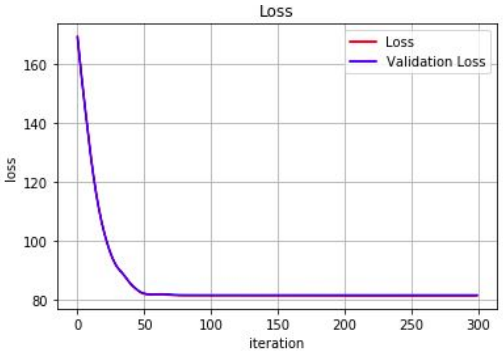


Observation and Discussion: As MoG algorithm was more cohesion and separation, the resulting graphs were likely circle distribution. With more circular distribution, we thought that K = 5 is the best.

3. Left K-Means / Right MoG

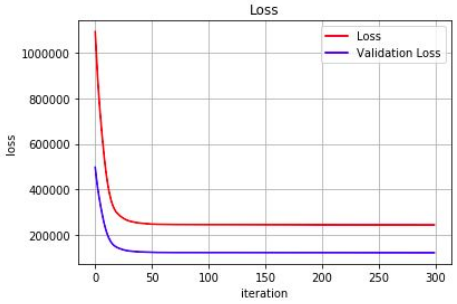
K = 5

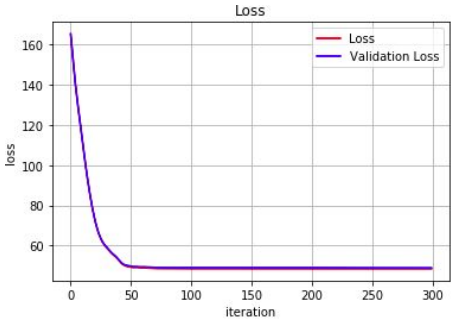




K = 10

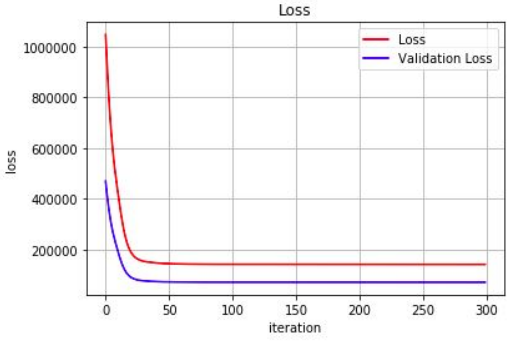
```
[ 0.          6.50967452 20.03899805 14.87925604  0.119994    14.42927854
 30.37348133  6.52467377  0.          7.12464377]
```

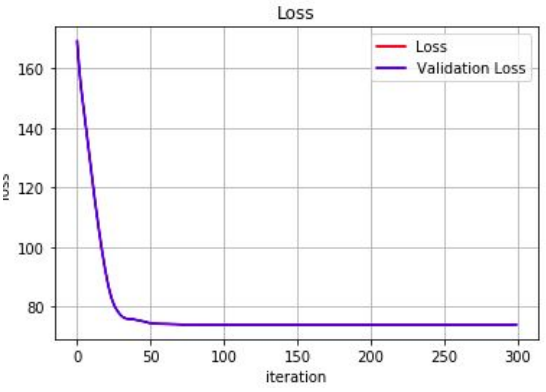




K = 15

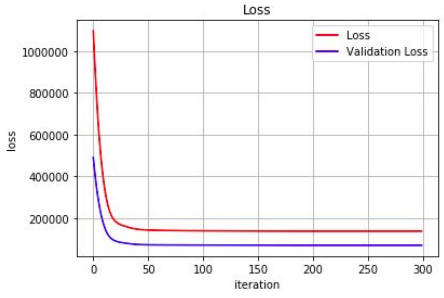
```
[2.02789861e+01 0.00000000e+00 0.00000000e+00 2.03539823e+01
 2.00389981e+01 7.84460777e+00 6.68966552e+00 0.00000000e+00
 7.28963552e+00 0.00000000e+00 7.46962652e+00 1.49992500e-02
 0.00000000e+00 0.00000000e+00 1.00194990e+01]
```

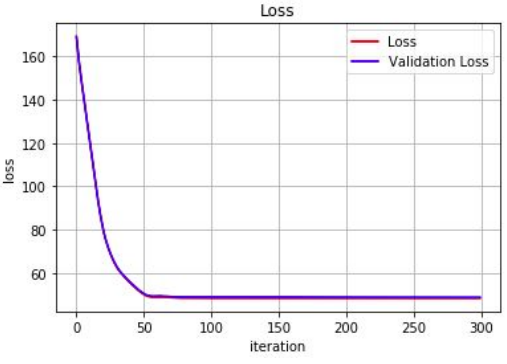




K = 20

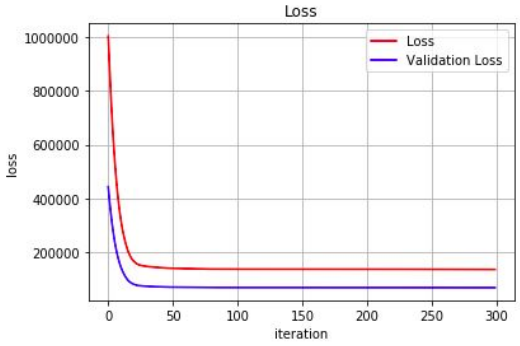
```
[ 3.52482376  6.86965652  0.          0.          2.89485526  3.70481476
 20.03899805  8.00959952  0.         20.3539823   3.56982151  3.05984701
  0.          0.          8.08459577  0.          0.          6.34468277
 10.01949903  3.52482376]
```

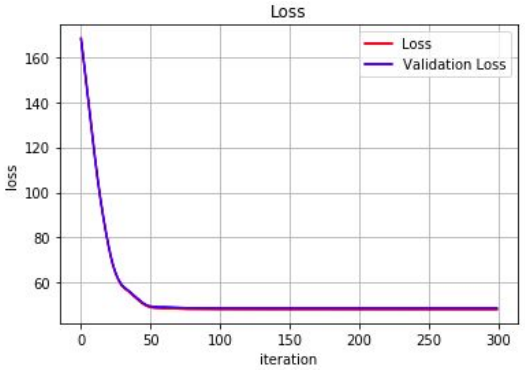




K = 30

```
[0.00000000e+00 1.49992500e-02 9.29953502e+00 1.49992500e-02
 0.00000000e+00 2.68486576e+00 0.00000000e+00 3.05984701e+00
 4.49977501e-02 0.00000000e+00 2.03539823e+01 2.03989801e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 9.64451777e+00
 1.03494825e+01 0.00000000e+00 0.00000000e+00 0.00000000e+00
 2.00389981e+01 0.00000000e+00 2.99985001e-02 0.00000000e+00
 3.76481176e+00 2.83485826e+00 1.00194990e+01 2.83485826e+00
 0.00000000e+00 2.96985151e+00]
```





Observation and Discussion: From the graphs above, it clearly shows that K means has a larger different in train and validation loss than the MoG algorithm. Meanwhile, the above two graphs have similar turning point while the last three are similar turning point. Thus, we thought the best cluster number is between 10 to 15.