ECE521 Assignment #3 Example solutions

Due: March 2017

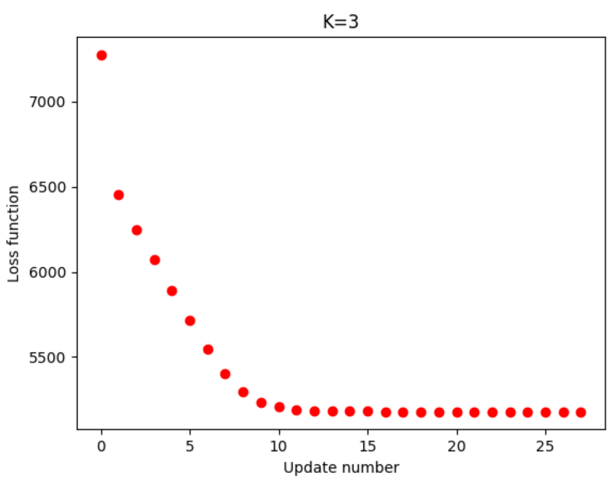
Question 1

(a) $\mathcal{L}(\boldsymbol{\mu})$ is not convex, and this can most simply be seen through its multi-modality. Suppose $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are two of many cluster centres positioned such that $\mathcal{L}$ is at the global minimum. Switch any two cluster centres and the loss function is identical. When each of the cluster centres is in between switching, at point $\mathbf{x} = \lambda\boldsymbol{\mu}_1 + (1-\lambda)\boldsymbol{\mu}_2$, the loss function is higher. Thus Jensen's inequality does not hold, i.e. we cannot say that:

$$\mathcal{L}\left(\lambda\boldsymbol{\mu}_1 + (1-\lambda)\boldsymbol{\mu}_2\right) \leq \lambda\mathcal{L}(\boldsymbol{\mu}_1) + (1-\lambda)\mathcal{L}(\boldsymbol{\mu}_2).$$

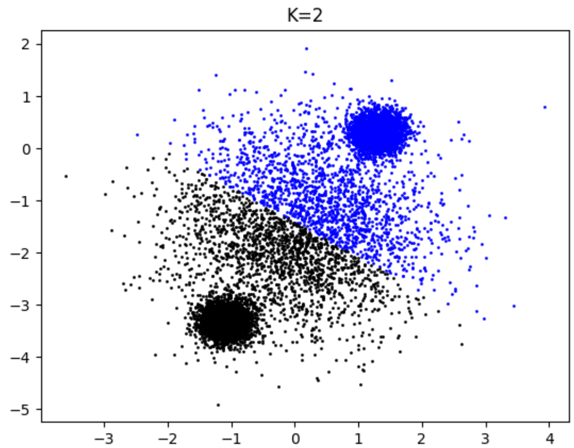
(b) The estimated cluster centres were:

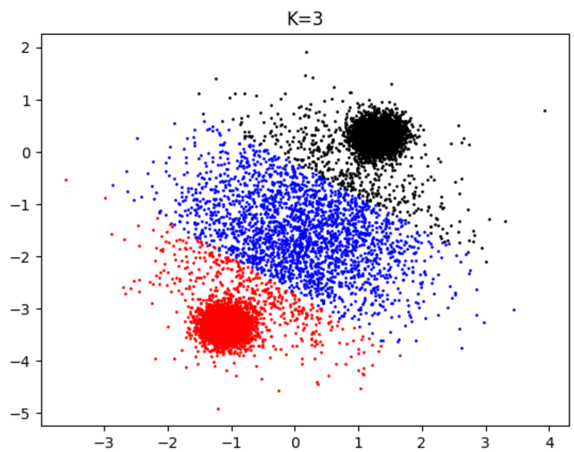
```
 1.253296  0.246562
 0.121239 -1.51951
-1.055331 -3.242612
```

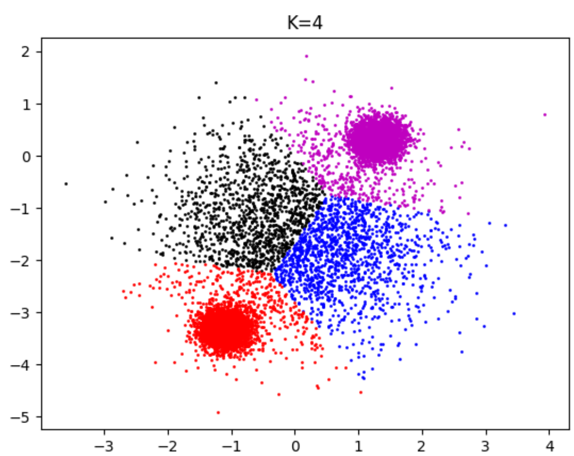


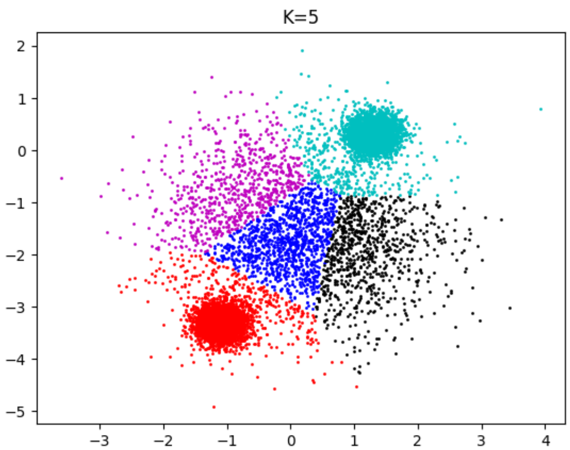
(c) The percentage breakdown was:

K	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
1	100%				
2	50.5%	49.5%			
3	38.2%	38.0%	23.8%		
4	37.3%	37.1%	13.5%	12.0%	
5	36.3%	35.9%	11.3%	9.0%	7.5%


K=2
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1
2
3
4


K=3
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1
2
3
4


K=4
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1
2
3
4



From the figures, the dataset is comprised of a roughly Gaussian component with high variance, plus two dense circular areas (lower left and upper right). A good value for K is 3. Two is the minimum to capture the two dense areas, three is the minimum to model these plus the region in between separately, and going above 3 is mainly a matter of subdividing the latter.

(d) The loss function values were:

K	Validation loss
1	6724
2	2639
3	1705
4	1465
5	1370

Creating a third component drops the validation loss by about 35% which is significant. Adding a fourth lowers the validation loss by about 14%, which may be important in some applications. Adding a fifth lowers the loss by only about 6%. The best K is 3 or 4.

Python code for Question 1:

```
import numpy as np
import math
import random

# Initializations
K = 2
Xall = np.load('data2D.npy')
D = int(Xall.shape[1])
B = int(Xall.shape[0])
#B=int(round(B*2/3)) # For part 4
X = Xall[:B,:]
mu = np.random.randn(K,D)

# Distance function from Assignment 1:
def getD(X,Z): #KxD - BxD -> BxK
```

```
    diff=X[np.newaxis,:,:]-Z[:,np.newaxis,:]
    return np.sqrt(np.sum(diff**2,axis=-1))

def gradL(mu,X):
    K = int(mu.shape[0])
    D = int(X.shape[1])
    B = int(X.shape[0])
    myD = getD(mu,X) # BxK
    minMu = np.argmin(myD,axis=1)
    gradMu = np.zeros((K,D))
    for n in range(B):
        gradMu[minMu[n],:] -= 2*(X[n,:]-mu[minMu[n],:])
    for k in range(K):
        if not any(minMu==k):
            gradMu[k,0] = -99
    return gradMu

def L(mu,X):
    K = int(mu.shape[0])
    D = int(X.shape[1])
    B = int(X.shape[0])
    myD = getD(mu,X)
    minMu = np.argmin(myD,axis=1)
    Lmu = myD[range(B),minMu]
    return np.sum(Lmu)

# Optimize
V = 1000
t = 0
losses = np.zeros(V)
if K==1:
    alpha = .1/B
else:
    alpha = 1/B
while t < V:
   t += 1
   oldMu = mu
   g = gradL(mu,X)
   for k in range(K):
       if (g[k,0] == -99): # lost sheep
           mu[k,:] = np.random.randn(D)
   mu = mu - alpha*g # Gradient descent
   losses[t-1] = L(mu,X)
   theChange = np.sum(np.sum(np.power(oldMu-mu,2)))
   if theChange < 1e-8:
       V = t
print(t, theChange, L(mu,X))

# Part 2: The centres
print(mu)
```

```
# Part 3: Plotting and percentages
import matplotlib.pyplot as plt
col = ['bo','ko','ro','mo','co']
myD = getD(mu,X)
minMu = np.argmin(myD,axis=1)
plt.clf()
for k in range(K):
    u = np.where(minMu==k)
    ul = len(np.transpose(u))/B*100
    print('Points in cluster %d: %.1f'%(k,ul))
    plt.plot (X[u,0],X[u,1],col[k],markersize=1)
    plt.plot (mu[k,0],mu[k,1],'gx')
plt.title('K=%d'%(K))
plt.show()

if (K==3 and B==10000):
    plt.clf()
    plt.plot (range(V),losses[:V],'ro')
    plt.xlabel ('Update number')
    plt.ylabel ('Loss function')
    plt.title('K=3')
    plt.show()

# Part 4: Loss on validation data
if (B==6667):
    s1 = int(X.shape[0])
    X = Xall[B:,:]
    s2 = int(X.shape[0])
    print ('Training on %d points, the loss on %d validation points was %.1f'
           %(s1,s2,L(mu,X)))
```