**Assignment -2 Solutions**

**Prasenjeet Biswal CS 7641**

**GTID - 903260510 Prof. Le Song**

1.

(a)

Given p(x) = ∑k=1:K( πkN(x|uk,∑k)) ...(1)

p(z) = ∏k=1:K(

p(x|z) = ∏k=1:K(N(x|uk, ∑k))zk

p(x) = ∑zεZ p(z)p(x|z) where Z = {z(1), z(2), .... z(k)}. .....(2)

To show that ∑zεZ p(z)p(x|z) = ∑k=1:K( πkN(x|uk,∑k))

**RHS**

The probability of a data point from kth gaussian component = πk  can be written as ∏k=1:K( because zk  is a latent variable and thus it only has values 0 if the point does not belong to kth component else it will be 1. This is only for the kth component. Summing over all components,

∑k=1:K πk = ∑zεZ ∏k=1:K(

Similarly, ∑k=1:K N(x|uk,∑k) can be written as ∑zεZ ∏k=1:K(N(x|uk, ∑k))zk.

So eq(1) can written as ∑zεZ ∏k=1:K(∏k=1:K(N(x|uk, ∑k))zk.

= ∑zεZ ∏k=1:K((N(x|uk, ∑k))zk.

= ∑zεZ p(z)p(x|z)

=RHS

**Thus Proved.**

(b) Using Bayes Rule, p(zkn|xn) = (p(xn|zkn)p(zkn)) / p(xn)

p(xn) = ∑πkN(x|uk, ∑k)

For data point xn belonging to component zk,

p(zkn) = πk

and p(xn | zkn) = N(xn | uk, ∑k)

**So, p(zkn|xn) = (πk N(xn | uk, ∑k) ) / (∑πkN(x|uk, ∑k))**

(c)

We refer to the p(zkn|xn) as .

The log likelihood is

L = ∑n=1:N∑k=1:K  (log(πk) - (xi - uk)T∑k-1(xi - uk) + log |∑|) + C

Since we have a constraint, ∑k=1:K πk = 1, we will have to use lagrangian operator,

L' = ∑n=1:N∑k=1:K (log(πk) - (xi - uk)T∑k-1(xi - uk) + log |∑|) + C + λ(1 - ∑k=1:K πk)

Differentiating L' wrt πk  setting it equal to 0, ∑n=1:N / πk - λ = 0

=> πk λ = ∑n=1:N

Summing ∑k=1:K on both sides, we get

1 = (1/ λ)∑n=1:N 1 ....because ∑k=1:K = 1.

so λ = N

replacing λ, we get

**πk = Nk / N where Nk = ∑n=1:N**

To estimate uk , we differentiate L wrt uk and set it equal to 0,we get

0 = ∑n=1:N (xn - uk) ∑k-1

∑n=1:N xn ∑k = ∑n=1:N uk ∑k-1

Multiplying both sides by ∑k, we get

∑n=1:N xn  = ∑n=1:N uk

so **uk = (∑n=1:N xn  ) / Nk**

Differentiating L wrt ∑k,  and equating to 0, we get

0 = ∑n=1:N ((1/∑k) + (xi - uk)T∑k-2(xi - uk))

∑n=1:N ∑k = ∑n=1:N (xi - uk)T(xi - uk)

**∑k = (∑n=1:N (xi - uk)T(xi - uk) ) /Nk**

(d)

Covariance matrix is given by ∈I where ∈ is variance parameter and I is the identity matrix.

So , p(x| uk, ∑k) = (1/2π∈) exp((-1/2∈)||x-uk||2), we get

Considering the EM algorithm for a mixture of K gaussians and ∈ to be fixed

**= (πk exp(-||xn - uk||2 / 2**∈**)) / (∑jπj exp(-||xn - uj||2 / 2**∈**))**

Since ∈ -> 0, we see that in the denominator, the term corresponding to smallest ||xn - uj||2 will go to zero most slowly. Hence, for point xn all go to 0 except for term j, for which it goes to 1. Thus for this limit, each point is assigned a definite cluster.

Thus maximizing the expected complete log likelihood is the same as minimizing J for the K means algorithm

2.

(a) Given the probability of a point xi lying over region i i.e. P(xi)= hi.

To maximize, we will multiply over all regions.

so P(x1)\*P(x2)..... = hj(1)\*hj(2)..... - the likelihood of all points xi falling into region hj.

so ∏i P(xi) = ∏ihj(i)

Taking log on both sides, we get

log(∏i P(xi)) = log(∏ihj(i))

so , **the log likelihood ∑ilogP(xi) = ∑i log(hj(i))**

(b) Since there is a constraint of ∑ihiΔi =1

we use the LaGrange operator.

∑ilogP(xi) = ∑i log(hj) - λ(∑ihiΔi - 1)

Derivating wrt hj and equating to 0, we get

(∑1:N 1)/ hj + λΔj = 0 .... (1)

As we see we are summing from 1 to n, but only nj fall in region j so ,

∑i=1:N 1 = nj

Changing the equation (1), we get

(nj / hj) + λΔj = 0

nj = - λΔjhj

To get λ, we sum over all regions j,

∑j nj = - λ∑j Δjhj

λ = -N

Putting the value of λ in equation 1 we get

maximum likelihood estimator **hj = (nj / NΔj)**

(c)

**Non-parametric density estimation usually does not have parameters. -** False, Non parametric density estimators have parameters that depends on the size of data. The non-parametric density estimators have hyper-parameters that has to be changed to fit the model.

**The Epanechnikov kernel is the optimal kernel function for all data. -** True, the Epanechnikov kernel minimizes the mean integrated square error, and other kernels are measured relative to Epanechnikov kernel. So Epanechnikov kernel is the optimal kernel function for all data.

**Histogram is an efficient way to estimate density for high-dimensional data. -** False, because when the dimensions of data increase to d, the memory required increases to nd, which makes the histogram quite inefficient.

**Parametric density estimation assumes the shape of probability density. -** True, the parametric density estimation makes assumptions about the shape of the data, and thus finds the parameters involved. For e.g. the Gaussian distribution assumes a uni-modal distribution.

3(a)

To Prove - H(X,Y) ≤ H(X) + H(Y)

LHS -

-∑xεX∑yεY p(x,y) log(p(x)p(y|x))

= -∑xεX∑yεY p(x,y) log(p(x)) - -∑xεX∑yεY p(x,y) log(p(y|x))

**= H(x) + H(Y|X)**

In RHS, we have H(X) + H(Y),

We need to show that H(Y|X) ≤H(Y),

***By the property, I(X,Y) ≥0 where I(X,Y) is information gain***

***I(X,Y) can written as H(Y) - H(Y|X) (See part b of this question).***

so H(Y) - H(Y|X) ≥0

so H(Y) ≥ H(Y|X)

so **H(X,Y) ≤ H(X) + H(Y). Thus Proved.**

(b) To Prove I(X,Y) = H(Y) +H(X) - H(X,Y)

I(X,Y) = ∑xεX∑yεY p(x,y) log(p(x,y)/p(x)p(y))

= ∑xεX∑yεY p(x,y) log(p(x|y)/p(x))

= ∑xεX∑yεY p(x,y) log(p(y|x)/p(y))

= - ∑xεX∑yεY p(x,y) log(p(y)) + ∑xεX∑yεY p(x,y) log(p(y|x))

**so I(X,Y) = H(Y) - H(Y|X) ......(1)**

From previous part, H(X,Y) = H(X) + H(Y|X)

replacing H(Y|X) in equation (1), we get,

**I(X,Y) = H(Y) +H(X) - H(X,Y)**

(c) Find under what conditions does H(Z) = H(X) + H(Y).

Since Z is a function (X,Y), Z can be represented as f(X,Y).

**We know that H(f(X)) ≤ H(X) --- property of entropy.**

Thus H(Z) ≤ H(X,Y). .... From above

and H(X,Y) ≤ H(X) + H(Y) .... from part (a)

The equality will only be satisfied if and only if H(X,Y) = H(X) + H(Y) which is only true **when X and Y are independent.** (when X and Y are independent, H(Y|X) = H(Y)).

4. I ran mycluster.m for 200 iterations and the statistics for accuracy and running time are as follows -

**mean(accuracy) - 77.4412**

**standard deviation(accuracy) - 8.0292**

**Median (accuracy) - 79.1042**

**Mean running time - 1.3632736 s**

**Median Running Time - 1.2034 s**

**The converging condition for our algorithm is when ujc and πc do not change or the number of iterations reaches 1000. I saw only two observations when the running time went to around 6s when the algorithm stopped after 1000 iterations.**