Homework #1
Yiming Tong (ID: 903541553)

1. Probability

(a) A,B,C stands for the employee is from A,B,C company
W stands for the employee is female

The total employee of the three companies: 50+75+100=225
The number of female employees in the three companies:
A: 50 × 50% =25
B: 75 × 60%=45
C: 100 × 70%=70
The total number of female employees in the three companies:
25+45+70=140

$P(C|W)=\frac{P(C,W)}{P(W)} = \frac{70/225}{140/225} = 0.5$

(b) D stands for the person does have the disease;
T stands for the test result is positive;

P(T)=95% × 0.5%+ 1% × 99.5%= 1.47%
P(D, T)= P(T|D)P(D)=95%×0.5%=0.475%
$P(D|T) = \frac{P(D,T)}{P(T)} \approx 0.3231$

2. Linear algebra

(a) $\frac{\partial \alpha}{\partial z} = \frac{\partial \mathrm{y}^T}{\partial z}\mathrm{x} + \mathrm{y}^T \frac{\partial \mathrm{x}}{\partial z}$

$$\frac{\partial \mathrm{y}}{\partial z} = \begin{bmatrix} \frac{\partial y_1}{\partial z_1} & \cdots & \frac{\partial y_1}{\partial z_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial z_1} & \cdots & \frac{\partial y_n}{\partial z_p} \end{bmatrix}$$

$$\frac{\partial \mathrm{x}}{\partial z} = \begin{bmatrix} \frac{\partial x_1}{\partial z_1} & \cdots & \frac{\partial x_1}{\partial z_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_n}{\partial z_1} & \cdots & \frac{\partial x_n}{\partial z_p} \end{bmatrix}$$

$$\frac{\partial x_i}{\partial z} = \begin{bmatrix} \frac{\partial x_i}{\partial z_1} \\ \vdots \\ \frac{\partial x_i}{\partial z_p} \end{bmatrix}$$

$$\frac{\partial y_i}{\partial z} = \begin{bmatrix} \frac{\partial y_i}{\partial z_1} \\ \vdots \\ \frac{\partial y_i}{\partial z_p} \end{bmatrix}$$

$$\alpha = \sum_{i=1}^{n} y_i x_i$$

$$\frac{\partial \alpha}{\partial z} = \begin{bmatrix} \frac{\partial \sum_{i=1}^{n} y_i x_i}{\partial z_1} \\ \vdots \\ \frac{\partial \sum_{i=1}^{n} y_i x_i}{\partial z_p} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{n} x_i \frac{\partial y_i}{\partial z_1} + \sum_{i=1}^{n} y_i \frac{\partial x_i}{\partial z_1} \\ \vdots \\ \sum_{i=1}^{n} x_i \frac{\partial y_i}{\partial z_p} + \sum_{i=1}^{n} x_i \frac{\partial y_i}{\partial z_p} \end{bmatrix}$$

$$= x^T \frac{\partial y}{\partial z} + y^T \frac{\partial x}{\partial z}$$

(b) Since $AA^{-1} = I$

Take the derivative of both sides of aboving equation:

$$\frac{\partial A}{\partial \alpha} A^{-1} + A \frac{\partial A^{-1}}{\partial \alpha} = 0$$

$$A^{-1} \frac{\partial A}{\partial \alpha} A^{-1} + A^{-1} A \frac{\partial A^{-1}}{\partial \alpha} = 0$$

$$A^{-1} \frac{\partial A}{\partial \alpha} A^{-1} + \mathrm{I} \frac{\partial A^{-1}}{\partial \alpha} = 0$$

$$\frac{\partial A^{-1}}{\partial \alpha} = A^{-1} \frac{\partial A}{\partial \alpha} A^{-1}$$

3. Maximum Likelihood

(a) Poisson distribution

$P(x_i|\lambda)=\frac{\lambda^{x_i}\cdot e^{-\lambda}}{x_i!}$

$$L(\lambda)=\log\prod_{i=1}^{n} P(x_i|\lambda) = \log\prod_{i=1}^{n} \frac{\lambda^{x_i}\cdot e^{-\lambda}}{x_i!} = \frac{\lambda^{\sum_{i=1}^{n} x_i}\cdot e^{-n\lambda}}{\prod_{i=1}^{n} x_i!}$$

$$\frac{\partial L(\lambda)}{\lambda} = \frac{\sum_{i=1}^{n} x_i\cdot\lambda^{(\sum_{i=1}^{n} x_i-1)}\cdot e^{-n\lambda}}{\prod_{i=1}^{n} x_i!} - \frac{n\cdot\lambda^{\sum_{i=1}^{n} x_i}\cdot e^{-n\lambda}}{\prod_{i=1}^{n} x_i!}$$

To maximize $L(\lambda)$: $\frac{\partial L(\lambda)}{\lambda}=0$

That is, $$\frac{\sum_{i=1}^{n} x_i\cdot\lambda^{(\sum_{i=1}^{n} x_i-1)}\cdot e^{-n\lambda}}{\prod_{i=1}^{n} x_i!} - \frac{n\cdot\lambda^{\sum_{i=1}^{n} x_i}\cdot e^{-n\lambda}}{\prod_{i=1}^{n} x_i!} = 0$$

$$\frac{\sum_{i=1}^{n} x_i\cdot\lambda^{(\sum_{i=1}^{n} x_i-1)}}{\prod_{i=1}^{n} x_i!}=\frac{n\cdot\lambda^{\sum_{i=1}^{n} x_i}}{\prod_{i=1}^{n} x_i!}$$

$$\sum_{i=1}^{n} x_i\cdot\lambda^{\sum_{i=1}^{n} x_i}/\lambda = n\cdot\lambda^{\sum_{i=1}^{n} x_i}$$

$$\lambda = (\sum_{i=1}^{n} x_i)/n$$

(b) Multinomial distribution

$f(x_1, x_2 \dots x_z; n, \theta_1, \theta_2 \dots \theta_z)=\frac{n!}{\prod_{i=1}^{z} x_i!}\prod_{i=1}^{z}\theta_i^{x_i}$

$L(\theta_1, \theta_2 \dots \theta_z)=\log\prod_{j=1}^{n} f(x_1, x_2 \dots x_z; n, \theta_1, \theta_2 \dots \theta_z)$

$=\sum_{j=1}^{n}\left(\ln(n!) - \sum_{i=1}^{z}\ln(x_{ji}!)\right) + \sum_{j=1}^{n}(\sum_{i=1}^{z} x_{ji}\ln(\theta_i))$

$$\boldsymbol{\theta} = \begin{bmatrix}\theta_1\\ \vdots \\ \theta_z\end{bmatrix}$$

Since the first part of $L(\theta_1, \theta_2 \dots \theta_z)$ dose not have parameter θ_i

$$\frac{\partial L(\theta_1,\theta_2 \dots \theta_z)}{\boldsymbol{\theta}} = \begin{bmatrix}\frac{\partial \sum_{j=1}^{n}(\sum_{i=1}^{z} x_{ji}\ln(\theta_i))}{\partial\theta_1}\\ \vdots \\ \frac{\partial \sum_{j=1}^{n}(\sum_{i=1}^{z} x_{ji}\ln(\theta_i))}{\partial\theta_z}\end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial \sum_{j=1}^{n} x_{j1} \ln(\theta_1)}{\partial \theta_1} \\ \vdots \\ \frac{\partial \sum_{j=1}^{n} x_{jz} \ln(\theta_z)}{\partial \theta_z} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sum_{j=1}^{n} x_{j1}}{\theta_1} \\ \vdots \\ \frac{\sum_{j=1}^{n} x_{jz}}{\theta_z} \end{bmatrix}$$

To maximize $L(\theta_1, \theta_2 \ldots \theta_z)$ within the restriction:

$$\sum_{i=1}^{z} \theta_i = 1$$

$$\sum_{i=1}^{z} x_{ji} = n \quad , for\ each\ j$$

So, $\theta_i = \frac{\sum_{j=1}^{n} x_{ji}}{\sum_{i=1}^{z} \sum_{j=1}^{n} x_{ji}}$

(c) Gaussian normal distribution

$p(x_i|\mu,\sigma) = \frac{1}{(2\pi)^{0.5}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right)$

$L(\mu,\sigma;D) = \ln \prod_{i=1}^{n} \frac{1}{(2\pi)^{0.5}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right)$

$$= -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \sum_{i=1}^{n} \frac{1}{2\sigma^2}(x_i-\mu)^2$$

Take derivatives w.r.t. μ, σ^2

$\frac{\partial L(\mu,\sigma)}{\mu} = \sum_{i=1}^{n} \frac{1}{\sigma^2}(x_i-\mu) = 0$

So $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$

$\frac{\partial L(\mu,\sigma)}{\sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n} (x_i-\mu)^2 = 0$

So $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i-\mu)^2$

4. Clustering

(a) Take derivatives with μ

$\frac{\partial J}{\partial \mu^k}$ = $2\sum_{n=1}^{N} r^{nk}(x^n - \mu^k)$=0

$$\mu^k = \frac{\sum_n r^{nk} x^n}{\sum_n r^{nk}}$$

(b)

First, there are at most k^N ways to partition N data points into k clusters; each such partition can be called a "clustering". This is a large but finite number. For each iteration of the algorithm, we produce a new clustering based *only* on the old clustering. Notice that

1. if the old clustering is the same as the new, then the next clustering will again be the same.
2. If the new clustering is different from the old then the newer one has a lower cost

Since the algorithm iterates a function whose domain is a finite set, the iteration must eventually enter a cycle. The cycle can not have length greater than 1. because otherwise by 2. you would have some clustering which has a lower cost than itself which is impossible. Hence the cycle must have length exactly 1. Hence k-means converges in a finite number of iterations.

(c)

I think average linkage method would most likely result in clusters most similar to those given by K-means. As for the case when k=2, under both methods the centroids will be the average of the points in the certain cluster.

(d)

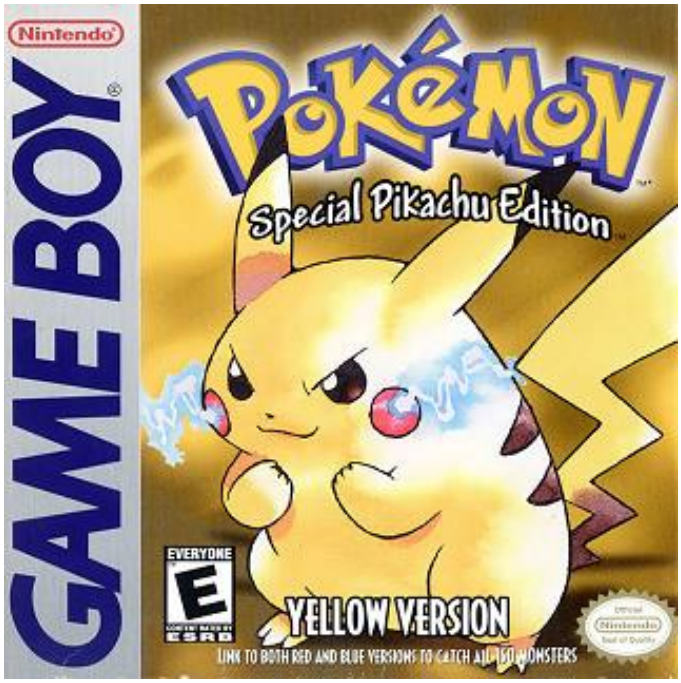
I think single linkage would successfully separate the two moons. Since under single linkage , bottom up hierarchical clustering merge clusters based on the minimum distance between any pairs of points from the two clusters. Since in a certain moon, there will always be two points next to each other, the merging will occur along in the moon and separate the two moons.

5. Programming: Image compression

1.

The uploaded K-Metroids implementation realizes:
1) initialization, randomly choose K unique samples from the data set.
2) assign samples to Metroid, according to the Euclidean norm.
3) update representative of clusters. in this step, a strict k-medroid would choose the sample which minimize the cost function (sum of norm to every other points in the cluster), which would involve an evaluation among all samples, I.e. calculate the distance matrix, making a n^2 time complexity. In my implementation, the cost functions is evaluated only within a random subset of each cluster, which is selected at a certain rate of like 0.001. This is actually inspired by the Matlab source code of k-medoids. This is a compromise due to an optimization problem, which is choosing between infinite running time using double loop and tens GB of memory consumption using matrix calculation. (parfor won't help at this amount of calculation). To further speed up, once a lower cost function is found the current loop will be terminated.
4) repeat 2) and 3) until reaching maximum loops declared or the centroids cease to change.

2.



3.,4.

It will. especially in my implementation of k-medoids, random choice of evaluation set and random break from loop increase the randomness. Due to the random breaking from the loop, some of the results are obvious far from equilibrium. The results are especially instable with this game cover which is mainly mono-colored. See below for details:

K=2 (result from two runnings)

With an larger K the k-medoids result will be worse off (Not converging in a reasonable number of iterations/ falls into several mixed clusters), falling in to local minimum nearly every time. (It will take more iterations to converge.) It will also take more time in my implementation since the time is linear to K approximately.

K=4 K=16

The result will improve if switching to the given football picture, which is a more reasonable mixing of color.

K=2 K=4

K=16

It's clearly that some of the clusters are degenerated, they are mixed together and falls into local minimum, which make the picture less "colorful".

Conclusion:

Overall, I made a bad implementation of k-medoids, which fails to handle certain kind of pictures and takes lots of running time. Plus, the result and the time consumption have internal instability from the implementation. I'm amazed how the original implementation of MATLAB achieves such performance. The k-means, on the other hand, converges much more quickly and takes less time per iteration. The representative samples of these two algorithms will differ even if both seems converge (K=2), since k-medoids only choose representative from the original data set. The speed may be enhanced if Hamiltonian distance is chosen.

6. Spectral clustering

Given m components, the graph Laplacian has m blocks.

$$L=\begin{bmatrix} L_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & L_{\mathrm{m}} \end{bmatrix}$$

So, The graph Laplancian has m eigenvectors with zero eigenvalues. And eigenvector ith is constant in block ith, but 0 in other blocks.
So, **first** the m eigenvectors are independent.
Second, according to graph Laplacian's structure, it has at most m independent eigenvectors.
Thus, according to the definition of zero eigenspace, the indicator vectors of these components span the zero eigenspace.