STAD80: Homework #3

Harpreet Singh Sumal

Due: Sunday, March 22nd

Contents

Q1. LDA and QDA for Multivariate Gaussian (50 points)

Question 1.1: Likelihood Model (10 points)

Answer: Under the LDA framework we are given that:

$$p(x_i|y_i = 1) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)/2}$$

$$p(x_i|y_i = 2) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-(x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)/2}$$

Let the above two equations be a and b respectively

Then we have that,

$$\begin{cases} p(x_i, y_i) = a\eta, & y_i = 1 \\ p(x_i, y_i) = b(1-\eta), & y_i = 2 \end{cases}$$

where $\eta = p(y_i = 1)$ and $(1 - \eta) = p(y_i = 2)$.

Now, to get the joint log-likelihood function, we need to combine the two cases for $y_i = 1$ and $y_i = 2$

$$p(x_i, y_i) = log[(a\eta) \cdot (b(1-\eta))]$$

However, doing that calculation all at once will be... horrible, so first we will get the joint function.

$$a\eta \cdot b(1-\eta)$$

$$\left(\eta \cdot \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)/2}\right)^{n_1} \cdot \left((1-\eta) \cdot \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-(x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)/2}\right)^{n_2}$$

Note: I realize that there is supposed to n_1 and n_2 in the following steps, but I honestly don't know how to do math with those exponents there and the ridiculous amount of variables, so I'm just gonna kinda... ignore it... for now.

$$\eta^{n_1} \cdot \left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\right)^{n_1} \cdot e^{n_1(-(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)/2)}$$

$$(1-\eta)^{n_2} \cdot \left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\right)^{n_2} \cdot e^{n_2(-(x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)/2)}$$

$$\eta^{n_1}(1-\eta^{n_2})\cdot\left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\right)^{n_1+(n-n_1)}\cdot exp\left\{\frac{(n_1+(n-n_1))[(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)+(x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)]}{2}\right\}$$

$$\eta^{n_1}(1-\eta^{n_2})\cdot\left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\right)^{n}\cdot exp\left\{\frac{n[(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)+(x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)]}{2}\right\}$$

From here we will take the logarithm to get the log-likelihood,

$$log(p(x_i,y_i))=log(\eta^{n_1})+log((1-\eta)^{n_2})+nlog\left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\right)$$

$$+\sum_{i=1}^{n}log\left(exp\left\{\frac{n[(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)+(x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)]}{2}\right\}\right)$$

$$log(p(x_i,y_i))=n_1log(\eta)+n_2log((1-\eta))+nlog\left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\right)+\sum_{i=1}^{n}\frac{n[(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)+(x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)]}{2}$$

Above is my attempt at solving the log-likelihood the correct way as I had originally forgotten about the n_1 and n_2 indicator values.

Question 1.2: MLE Estimation for η and μ_1, μ_2 (10 points)

Answers:

Deriving for η:

$$\frac{\partial}{\partial\eta}log(p(x_i,y_i))=[n_1log(\eta)+n_2log((1-\eta))]'$$

$$\frac{\partial}{\partial\eta}log(p(x_i,y_i))=\frac{n_1}{\eta}-\frac{n_2}{(1-\eta)}$$

Setting the derivative to 0 and isolating for η we get,

$$\hat{\eta}=\frac{n_1}{n_1+n_2}$$

$$\hat{\eta}=\frac{n_1}{n}$$

Deriving for μ_1

$$\frac{\partial}{\partial\mu_1}log(p(x_i,y_i))=\left[\sum_{i=1}^{n}\frac{n[(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1)+(x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)]}{2}\right]'$$

$$\frac{\partial}{\partial\mu_1}log(p(x_i,y_i))=\sum_{i=1}^{n}\left[-\frac{1}{2}n_1((x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1))\right]'$$

Setting the derivative = to 0 we get,

$$0=\sum_{i=1}^{n}\left[-\frac{1}{2}n_1((x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1))\right]'$$

$$\hat{\mu_1} = \frac{1}{n_1}\sum_{i=1}^{n} x_i$$

We use $\frac{1}{n_1}$ in this case because we only want to get the average for mu_1 which is only in relation to when $y_i = 1$. Since n_1 is the indicator for $I(y_i = 1)$ this will give us the count of how many times $y_i = 1$ in our data.

Deriving for μ_2

$$\frac{\partial}{\partial \mu_2} log(p(x_i, y_i)) = \left[\sum_{i=1}^{n} \frac{n[(x_i - \mu_1)^T \Sigma^{-1}(x_i - \mu_1) + (x_i - \mu_2)^T \Sigma^{-1}(x_i - \mu_2)]}{2}\right]'$$

$$\frac{\partial}{\partial \mu_2} log(p(x_i, y_i)) = \sum_{i=1}^{n} \left[-\frac{1}{2} n_2((x_i - \mu_2)^T \Sigma^{-1}(x_i - \mu_2))\right]'$$

Setting the derivative = to 0 we get,

$$0 = \Sigma^{-1} \sum_{i=1}^{n} \left[-\frac{1}{2} n_2((x_i - \mu_2)^T (x_i - \mu_2))\right]'$$

$$\hat{\mu_2} = \frac{1}{n_2}\sum_{i=1}^{n} x_i$$

We use $\frac{1}{n_2}$ in this case because we only want to get the average for mu_2 which is only in relation to when $y_i = 2$. Since n_1 is the indicator for $y_i = 2$ since it's value is $n_2 = n - n_1$ this will give us the count of how many times $y_i = 1$ in our data.

Question 1.3: MLE Estimation for Σ (10 points)

Answers:

First let's isolate our likelihood function for what we will be deriving. Furthermore let $\Omega = \Sigma^{-1}$.

$$\frac{\partial}{\partial \Omega} log(p(x_i, y_i)) = nlog\left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\right)\left[\sum_{i=1}^{n} \frac{n[(x_i - \mu_1)^T \Sigma^{-1}(x_i - \mu_1) + (x_i - \mu_2)^T \Sigma^{-1}(x_i - \mu_2)]}{2}\right]'$$

Set the derivative to 0 and we'll get,

$$0 = n|\Sigma|^{-1}\left[\sum_{i=1}^{n} -\frac{n}{2}(x_i - \mu_1)^T \Sigma^{-1}(x_i - \mu_1) + (x_i - \mu_2)^T \Sigma^{-1}(x_i - \mu_2)\right]'$$

$$0 = n|\Sigma|^{-1}\left[\sum_{i=1}^{n} -\frac{n}{2}\Sigma^{-1}(x_i - \mu_1)(x_i - \mu_1)^T + \Sigma^{-1}(x_i - \mu_2)(x_i - \mu_2)^T\right]'$$

$$0 = n|\Sigma|^{-1}\left[\sum_{i=1}^{n} -\frac{n}{2} Tr(\Sigma^{-1}(x_i - \mu_1)(x_i - \mu_1)^T + \Sigma^{-1}(x_i - \mu_2)(x_i - \mu_2)^T)\right]'$$

Using the identities given in the hint and isolating for Σ we achieve our result,

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(A)(B)$$

Where $A = (x_i - \hat{\mu_1})(x_i - \hat{\mu_1})^T$ and $B = (x_i - \hat{\mu_2})(x_i - \hat{\mu_2})^T$

Finally, rearranging and splitting n in terms of n_1 and n_2 we get our result in terms of S_1 and S_2

$$\hat{\Sigma} = S_1 \cdot S_2$$

Question 1.4: Decision Boundary (10 points)

Answers: Using Baye's Rule we can redefine our objective function to be:

$$\frac{P(y_i = 1|x_i)}{P(y_i = 2|x_i)} = \frac{P(x_i|y_i = y) \cdot P(y_i = y)}{P(x_i = x)}$$

$$= C(p(x_i|y_i = 1)\eta \cdot p(x_i|y_i = 2)1 - \eta)$$

Where C is the value of our denominator and a constant.

Expanding our probabilities we get the equations a and b from problem 1.1 which resulted in our initial joint function,

$$C\left[\eta^{n_1}(1-\eta^{n_2}) \cdot \left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\right)^n \cdot exp\left\{\frac{n[(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1) + (x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)]}{2}\right\}\right]$$

Since C is a constant, we'll assume other constants fall into it as well

$$C\left[\eta^{n_1}(1-\eta^{n_2})\, exp\left\{\frac{n[(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1) + (x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)]}{2}\right\}\right]$$

Now take the log as done in the question,

$$log(C) + log(\eta^{n_1}) + log((1-\eta)^{n_2}) + log\left(exp\left\{\frac{n[(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1) + (x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)]}{2}\right\}\right)$$

Substituting in our MLE values and setting to 0 we get,

$$log(C) = n_1 log(\eta) + n_2 log(1-\eta) - \frac{1}{2}\left(n[(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1) + (x_i-\mu_2)^T\Sigma^{-1}(x_i-\mu_2)]\right)$$

Rearranging we get,

$$log(C) = n_1 log(\hat{\eta}) + n_2 log(1-\hat{\eta}) - \frac{1}{2}\left(n[(x_i^T\hat{\Sigma}^{-1}x_i) + (x_i^T\hat{\Sigma}^{-1}x_i) + (\hat{\mu_1}^T\hat{\Sigma}^{-1}\hat{\mu_1}) + (\hat{\mu_2}^T\hat{\Sigma}^{-1}\hat{\mu_2})]\right)$$

Let the $\hat{\mu}_i$ terms be w^T, and the $\hat{\eta}$ terms be w_0

Then we have,

$$w^T x_i + w_0 = log(C)$$

be out decision boundary.

Question 1.5: QDA (10 points)

Everything before $\hat{\Sigma}_{1,2}$ isn't in terms of Σ and so, since the only difference between QDA and LDA is that there are distinct variances for each y_i, all the other estimators will be the same.

For $\hat{\eta}$ we have:

$$\hat{\eta} = \frac{n_1}{n}$$

For $\hat{\mu_1}$ we have:

$$\hat{\mu_1} = \frac{1}{n_1}\sum_{i=1}^{n} x_i$$

For $\hat{\mu_2}$ we have:

$$\hat{\mu_2} = \frac{1}{n_2}\sum_{i=1}^{n} x_i$$

Now for our Σ estimators, we merely need to account for the fact that there will be a Σ_1 and Σ_2. Furthermore, recall that when we calculated our LDA MLE for Σ we had two equations A and B. Where $A = (x_i - \hat{\mu_1})(x_i - \hat{\mu_1})^T$ and $B = (x_i - \hat{\mu_2})(x_i - \hat{\mu_2})^T$. Therefore,

For $\hat{\Sigma}_1$ we have:

$$\hat{\Sigma}_1 = \frac{1}{n_1}\sum_{i=1}^{n_1} A$$

For $\hat{\Sigma}_2$ we have:

$$\hat{\Sigma}_2 = \frac{1}{n_2}\sum_{i=1}^{n_2} B$$

Recall that for the LDA Decision Boundary we had the following equation, before simplificaiton,

$$log(C) = n_1 log(\hat{\eta}) + n_2 log(1-\hat{\eta}) - \frac{1}{2}\left(n[(x_i^T\hat{\Sigma}^{-1}x_i) + (x_i^T\hat{\Sigma}^{-1}x_i) + (\hat{\mu_1}^T\hat{\Sigma}^{-1}\hat{\mu_1}) + (\hat{\mu_2}^T\hat{\Sigma}^{-1}\hat{\mu_2})]\right)$$

In the QDA case, two sigma, one for the x_i terms and one for the $\hat{\mu}_i$ terms ($\hat{\Sigma}_2$ for $\hat{\mu}_2$ and $\hat{\Sigma}_1$ for $\hat{\mu}_1$).

$$log(C) = n_1 log(\hat{\eta}) + n_2 log(1-\hat{\eta}) - \frac{1}{2}\left(n[(x_i^T\hat{\Sigma}_1^{-1}x_i) + (x_i^T\hat{\Sigma}_2^{-1}x_i) + (\hat{\mu_1}^T\hat{\Sigma}_1^{-1}\hat{\mu_1}) + (\hat{\mu_2}^T\hat{\Sigma}_2^{-1}\hat{\mu_2})]\right)$$

Then, when we combine the x_i terms, it will result in the values being squared since they do not share the same $\hat{\Sigma}_i$ base. Likewise for the $\hat{\mu}_i$ terms. Then we will have a polynomial which results in a quadratic curve.

Q2. Clustering Algorithms (50 points)

Question 1.1

Found the following function to assist in loading and reading the MNIST files'.

```
# Load the MNIST digit recognition dataset into R
# http://yann.lecun.com/exdb/mnist/
# assume you have all 4 files and gunzip'd them
# creates train$n, train$x, train$y  and test$n, test$x, test$y
# e.g. train$x is a 60000 x 784 matrix, each row is one digit (28x28)
# call:  show_digit(train$x[5,])   to see a digit.
```

```
# brendan o'connor - gist.github.com/39760 - anyall.org

load_image_file <- function(filename) {
  ret = list()
  f = file(filename,'rb')
  readBin(f,'integer',n=1,size=4,endian='big')
  ret$n = readBin(f,'integer',n=1,size=4,endian='big')
  nrow = readBin(f,'integer',n=1,size=4,endian='big')
  ncol = readBin(f,'integer',n=1,size=4,endian='big')
  x = readBin(f,'integer',n=ret$n*nrow*ncol,size=1,signed=F)
  ret$x = matrix(x, ncol=nrow*ncol, byrow=T)
  close(f)
  ret
}
load_label_file <- function(filename) {
  f = file(filename,'rb')
  readBin(f,'integer',n=1,size=4,endian='big')
  n = readBin(f,'integer',n=1,size=4,endian='big')
  y = readBin(f,'integer',n=n,size=1,signed=F)
  close(f)
  y
}

train <- load_image_file('./train-images.idx3-ubyte')
test <- load_image_file('./t10k-images.idx3-ubyte')

train$y = load_label_file('./train-labels.idx1-ubyte')
test$y = load_label_file('./t10k-labels.idx1-ubyte')


show_digit <- function(arr784, col=gray(12:1/12), ...) {
  image(matrix(arr784, nrow=28)[,28:1], col=col, ...)
}
```

Now, having loaded in the mnist data-set, let us do the pre-processing,

```
# I have written some pseudo-code for how I believe
# To pre-process the images in MNIST and compress
# them to be 14x14 in size.

# I was going to attempt this with our actual data-set,
# but the way we read it in makes it different than
# how I initially imagined so now I'm not quite sure
# how to tackle the problem. Maybe this can net me some
# part marks.


# ~~~~~~~ MNIST Compression Pseudo Code ~~~~~~~~~~~~~~~~


# 14x14M = matrix(0, nrow = 14, ncol = 14)

# for(i in 1:# of rows in MNIST){
    # for(j in 1:# of cols in MNIST){
```

```
        # mean_val = (mnist[i,j] + mnist[i,j+1] + mnist[i+1,j] + mnist[i+1,j+1]) / 4

        # 14x14M[i,j] = mean_val

        # j+1
    # }
    # i+1
# }



# I did attempt doing this with my data-set, and assuming that perhaps each
# subset of 4 consecutive numbers in train$x, test$x
# would be the numbers in our 2x2 block,
# however that was not the case so I just left that code out
```

I will implement K-Means, and leave it commented out so that hopefully you can give me some part-marks if the implementation seems correct to you. I really need all the marks I can get.

```
# We know each row of train$x is a drawing
# Train$y contains the correct labelings for each of our rows, i.e. row 1 is a 5, row 2 is a 0, etc.

# Since we're only considered with the images for numbers {0,1,2,3,4}, let's filter the data

train_list = matrix(nrow = 60000, ncol = 784)
j = 0
for(i in 1:length(train$y)){
  if(train$y[i] < 5){
    train_list[j,] = train$x[i,]
    j = j+1
  }
}


df_data = as.data.frame(train_list)

clean_data = na.omit(df_data)

clustering = kmeans(clean_data, centers = 5, nstart = 25)

# I'm not sure where to go from here, because none of the values
# in either of the 3 variables contained in 'clustering'
# are of length 784. (Or 196 had I been able to compress images)
# so I'm not sure what to use to draw the digits.
# I did do kmeans tho technically, on the subset you asked for,
# part marks maybe? hehe :)
```

Question 1.2

Now I will attempt the GMM version of things, by attempting the preliminary work,

PART A:

Given that each of our x_i varibales are simply data-points, the hidden variable Z_i must be what makes these Gaussian Models. Therefore the $P(Z_i = j) = N(\mu_j, \Sigma_j)$. This results in the PDF for the Normal Distribution

that we see,

$$\Pi_{i=1}^{n} \sum_{j=1}^{k} \frac{\eta_j}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} exp\left(-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j)\right)$$

Everything besides the x_i data points that are being multiplied over (With the Π at the beginning) belong to our marginal Z_i distribution.

PART B:

Now, to find the $P(Z_i = j|x_i)$ seems a little confusing to me since the hint says to use Bayes Rule. Assuming that $A = P(Z_i = j)$ and $B = P(x_i)$, we have that $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$. We know what the numerator is, from the given in the question and what we solved for above, however, since x_i is simply a datapoint and not a random variable, how do we associate a probability for it? I'm assuming it's something similar to question 1 with η and the indicator variables n_1 and n_2 but... it doesn't really add up. However, referencing the notes we get that...

$$P(Z_i = j|X_i = x_i) = \frac{p_{\theta_j^{(t)}}(x_i|Z_i = j)\eta_j}{\sum_{i=1}^{n} p_{\theta_j^{(t)}}(x_i|Z_i = j)\eta_j}$$

We get the value for η_j in our initialization, and then it gets iterated on in the M-Step, as well as the θ values.

PART C:

From the given equation in (0.8), we can take the expecation value without incurring any change in the result.

$$\sum_{i=1}^{n} log\left[\sum_{i=1}^{n} E_{z_i|x_i}\gamma_{ij}\frac{p_\theta(x_i, Z_i = j)}{\gamma_{ij}}\right]$$

Using Jensen's Inequality we get,

$$\sum_{i=1}^{n} E_{z_i|x_i}\left[\sum_{i=1}^{n} log\,\gamma_{ij}\frac{p_\theta(x_i, Z_i = j)}{\gamma_{ij}}\right]$$

Since this our expecation referenced Z_i and x_i which are within the summation, this equates to just one, since we're already summing over all the values. And with that we have our result,

$$\sum_{i=1}^{n}\sum_{i=1}^{n} log\,\gamma_{ij}\frac{p_\theta(x_i, Z_i = j)}{\gamma_{ij}}$$

PART D:

Simply plugging into $F(\gamma, \theta)$ with θ^{old} we will get to our result.

$$F(\gamma, \theta^{old}) = \sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}\,log\left[\frac{p_\theta(x_i, Z_i = j)}{\gamma_{ij}}\right]$$

$$F(\gamma, \theta^{old}) = \sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}\,log\,p_{\theta^{old}}(x_i)$$

This is possible because we know that $\gamma_{ij} = p_{\theta^{old}}(Z_i = j|x_i)$

$$F(\gamma, \theta^{old}) = \sum_{i=1}^{n} log\,p_{\theta^{old}}(x_i)$$

$$F(\gamma, \theta^{old}) = \ell(\theta^{old})$$

PART E and F:

For Spherical Models:

For η_j:

$$\eta_j^{t+1} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} p_\theta^{old}(Z_i = j | x_i)$$

For μ_j:

$$\mu_j^{t+1} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} p_\theta(x_i, Z_i = j)$$

For μ_j:

$$\mu_j^{t+1} = \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{p_\theta(x_i, Z_i = j)}{p_\theta^{old}(Z_i = j | x_i)}$$

Not sure about the derivation for mixture of diagonal Gaussians.

```
# As for implementing either of these GMMs... Yeah I have no clue how that is supposed to look, so that
```

Question 1.3:

Since I wasn't able to actually implement the code, I still want to try and do this question with comparisons I have found online, as well as my theoretical understanding of GMMs and K-Means clustering.

This question I'll attempt to give an informal proof with mainly English as I'm not sure on the specifics. Our goal is to make our equation similar to the K-Means objective function of $|x_i - \mu|$. This is in a sense a sort of normalization. If we let $\Sigma_j = \sigma_j$, and let $\gamma = \frac{1}{N}$ where \$N = \$ our total number data points x_i, we will have transformed our EM algorith with the same parameters of K-Means. Now to get the objective function to be more similar to K-Means, since our new γ is a constant term, they can both be pulled out of the summations resulting in 1. Now all that's left is our probability functions. Since our Σ is in terms of σ_j and if we invoke the hint of letting σ_j go to 0, our objective function will eventually converge to be identical to that of K-Means.

Hopefully that can get me some marks, if I was on the right track

Question 1.4:

The mixture of diagonal Gaussians worked best, as expected, since it was malleable and able to form cluster circles that could find more rounded numbers like 2 and 0, but also sharper ones like 1 and 4. However, as expected it did take the longest because of this, needing more time to optimally create the decision boundary for each cluster circle. While this did work the best, in terms of run-time to efficiency of implementation, I would say K-Means wins by a mile. The only cases that it wasn't correct one were cases that even the human-eye could have some trouble deciphering between. But when the writing was clean and distinct enough, it was able to cluster our values correctly with good enough accuracy. If we only cared about results and not run-time, we would choose mixture of diagonal Gaussians anytime for a case like this. One thing I noticed however is, the errors in the spherical mixture of Gaussians, while it did have almost equal amount of errors to K-Means, most of it's errors occured early on in the clustering. As each update made the parameters tighter and it learned more, less errors occurred. Compared to the more randomness of K-Means, this makes me believe that spherical mixture of Gaussians would be better in a larger data-set than K-Means would be.

I think if we did greater amounts of pre-processing, and were able to differentiate between the more sharper numbers and rounded numbers, we could efficiently run two different algorithms, one spherical and one diagonal mixture of Gaussians so that the more complex algorithm has the smallest data-set as possible. Furthermore, if anything a mixture of diagonal Gaussians would likely take longer on more rounded numbers because the number of edges each thing, diagonal ellipses can cover is reduced. For example, it would take much longer to cluster a '0' compared to a '7', theoretically.

I can't speak on the initalization strategy as I wasn't able to do it, however in my mind I would use around 4 clusters for K-Means for the $\{0, 1, 2, 3, 4\}$ data-set as all of these numbers only really have "4 turns" or "4 hot points" where points would change direction and need a separate clustering. For the EM algorithm, I would initalize around the same mark as well, but run it several times to see which one causes the smoothest run through the M-Step recalculation as well.

Q3. K-Means Wikipedia Documents Clustering (30 points)

Question 1:

```
load("./Wikipedia.RData")

head(dat)
```

```
##                 name
## 1         Michel Che
## 2 Hossein Modarressi
## 3      Xiao-Gang Wen
## 4  Nicholas Doumanis
## 5         Don Cahoon
## 6        Lance Davis
##
## 1
## 2
## 3
## 4
## 5 don toot cahoon is a retired american ice hockey coach he was the head coach of the princeton tige
## 6
```

The first three individuals are Michel Che, Hossein Modarressi, and Xiao-Gang Wen. Their respective professions seem to be; Professor, Muslim Jurist, and American Physicist.

```
####### SCRIPT1_HW3 ########

library(tm)
```

```
## Warning: package 'tm' was built under R version 3.6.2
```

```
## Loading required package: NLP
```

```
library(XML)
```

```
## Warning: package 'XML' was built under R version 3.6.2
```

```
library(Matrix)
library(slam)
```

```
## Warning: package 'slam' was built under R version 3.6.2
```

```
library(SnowballC)
library(akmeans)
library(lsa)
```

```
## Warning: package 'lsa' was built under R version 3.6.3
```

```
load("Wikipedia.RData")
text = dat[,"text"]
text = iconv(text, to = "utf-8") #some conversion on SMILE needed
corpus = Corpus(VectorSource(text))
dtm.control.raw <- list(tolower = TRUE, removePunctuation = TRUE, removeNumbers = TRUE,
                    removestopWords = TRUE, stemming = TRUE, wordLengths = c(3, 15), bounds = list(globa

dtm.raw = DocumentTermMatrix(corpus, control = dtm.control.raw)
dtm.mat.raw = as.matrix(dtm.raw) # our term-document matrix
```

```
num_individuals = NROW(dat)

num_individuals
```

```
## [1] 812
```

```
# first number = 812 = number of documents

# View(head(dtm.mat.raw)) to get an understanding of the structure of our matrix

num_words = colSums(dtm.mat.raw)

length(num_words)
```

```
## [1] 6888
```

```
# second number = 6888 = number of words

sorted_num_words = data.frame(V1=sort(num_words, decreasing=TRUE))

top_10 = head(sorted_num_words, 10)

top_10
```

```
##            V1
## the     17316
## and     10729
## univers  3169
## for      3165
## was      2817
## his      2353
## from     2126
## has      1708
## with     1674
## new      1174
```

```
top_count = sorted_num_words

# Top 10 words are the, and, univers, for, was, his, from, has, with, new

quantile(num_words, probs = c(0, .25, .50, .75, 1))
```

```
##    0%   25%   50%   75%  100%
##     2     3     5    14 17316
```

The number of individuals is equivalent to the number of documents in our data-set which is 812. The number

of words in our dataset can be retrieved by adding up all the columns of our term-document matrix, which results in 6888. The top 10 most used/frequent words in the dataset are the following; "the, and, univers, for, was, his, from, has, with, new". Taking the quantiles, we can see that the number of words in each quantile are doubling each time, until the 100% quantile which has many words. This does indeed follow Zipf's Law.

Question 2:

```
####### SCRIPT2_HW3 ########

dtm.control <- list(tolower = TRUE, removePunctuation = TRUE, removeNumbers = TRUE,
                    removestopWords = TRUE, stemming = TRUE, wordLengths = c(3, 15), bounds = list(globa
                    weighting = function(x){weightTfIdf(x, normalize = FALSE)})

dtm = DocumentTermMatrix(corpus, control = dtm.control)
dtm.mat = as.matrix(dtm)
```

```
num_words2 = colSums(dtm.mat)

sorted_num_words2 = data.frame(V1=sort(num_words2, decreasing=TRUE))

top_10_weighted = head(sorted_num_words2, 10)
top_10_weighted
```

```
##                 V1
## she      2414.6500
## her      1611.8679
## music    1179.0219
## econom   1160.7760
## law      1132.1644
## scienc    989.8247
## mathemat  975.4508
## new       908.1503
## histori   905.1748
## research  903.4017
```

```
top_weighted = sorted_num_words2
```

The top 10 words in our re-paramterized matrix are the following; "she, her, music, econom, law, scienc, mathemat, new, histori, research".

Question 3:

```
####### SCRIPT3_HW3 ########
dtm.mat.indicator = dtm.mat.raw
dtm.mat.indicator[dtm.mat.indicator!=0] = 1

word.presence = apply(dtm.mat.indicator, 2, sum)
idx = which(word.presence >= quantile(word.presence, prob = 0.99))
dtm.mat.raw = dtm.mat.raw[,-idx]

common.words = read.csv("google-10000-english.txt", header = F)
idx = which(colnames(dtm.mat) %in% common.words[1:300,1])
dtm.mat.raw = dtm.mat.raw[,-idx]
```

```
num_words3 = colSums(dtm.mat.raw)

sorted_num_words3 = data.frame(V1=sort(num_words3, decreasing=TRUE))
```

```r
top_10_cleaned = head(sorted_num_words3, 10)

top_10_cleaned
```

```
##           V1
## she      956
## her      633
## histori  418
## econom   397
## polit    390
## law      389
## music    364
## mathemat 329
## theori   323
## physic   310
```

The top 10 words with the special cleaning of Script 3 results in top words similar to those in dat.mat from Question 2 as wanted. These include; "she, her, histori, econom, polit, law, music, mathemat, theori, physic".

Question 4.

```r
library(stringr)


my_data = subset(dat, name == "Ben Bernanke")
my_data$text
```

```
## [1] ben shalom bernanke brnki brnangkee born december 13 1953 is an american economist at the brooki
## 59071 Levels: 108 born 1978 is an italian artist in the field of street art and contemporary art fro
```

```r
ben_dex = which(dat$name == "Ben Bernanke")

# Get words at this index from dtm.mat.raw that are >0

ben_words = dtm.mat.raw[735,]

ben_words_cleaned = data.frame(ben_words)

# Printed one extra to make sure that it was the last word
head(ben_words_cleaned[order(-ben_words_cleaned$ben_words),,drop=FALSE],75)
```

```
##             ben_words
## chairman            6
## bernank             5
## feder               5
## reserv              5
## term                4
## econom              3
## bush                2
## februari            2
## second              2
## succeed             2
## tenur               2
## when                2
## advis               1
## alan                1
```

```
## bank              1
## barack            1
## becom             1
## befor             1
## began             1
## ben               1
## board             1
## brook             1
## busi              1
## central           1
## chair             1
## chang             1
## confirm           1
## council           1
## cycl              1
## decad             1
## decemb            1
## declin            1
## develop           1
## diminish          1
## discuss           1
## doctrin           1
## dure              1
## economi           1
## economist         1
## end               1
## financi           1
## fiscal            1
## georg             1
## governor          1
## great             1
## him               1
## increas           1
## influenc          1
## janet             1
## januari           1
## late              1
## macroeconom       1
## moder             1
## monetari          1
## nomin             1
## obama             1
## occur             1
## oversaw           1
## particular        1
## propos            1
## recent            1
## respons           1
## septemb           1
## servic            1
## stabil            1
## structur          1
## system            1
## then              1
```

```
## theori              1
## there               1
## tradit              1
## until               1
## volatil             1
## went                1
## aaa                 0
```

```
ben_words_used = rownames(head(ben_words_cleaned[order(-ben_words_cleaned$ben_words),,drop=FALSE],74))
# ben_words_used

text2 = rownames(top_weighted)

text3 = rownames(top_count)

r2 = str_detect(my_data$text, text2)

r3 = str_detect(my_data$text, text3)

top_weight_ben = which(r2 == TRUE)

top_count_ben = which(r3 == TRUE)


print("~~~~~ Top Weighted Words ~~~~~")
```

```
## [1] "~~~~~ Top Weighted Words ~~~~~"
```

```
for(i in 1:10){
  print(text2[top_weight_ben[i]])
}
```

```
## [1] "her"
## [1] "econom"
## [1] "art"
## [1] "that"
## [1] "intern"
## [1] "state"
## [1] "serv"
## [1] "nation"
## [1] "his"
## [1] "american"
```

```
print("~~~~~ Top Counted Words ~~~~~")
```

```
## [1] "~~~~~ Top Counted Words ~~~~~"
```

```
for(i in 1:10){
  print(text3[top_count_ben[i]])
}
```

```
## [1] "the"
## [1] "and"
## [1] "univers"
## [1] "for"
## [1] "was"
## [1] "his"
```

```
## [1] "from"
## [1] "princeton"
## [1] "american"
## [1] "professor"
```

This shows that the top 10 most counted words in our text are; 'the', 'and', 'univers', 'for', 'was', 'his', 'from', 'princeton', 'american', 'professor'. Furthermore it also shows that the top 10 weighted words in our text are; and, for, american, she, also, school, scienc, serv, institut, her.

Question 5:

```
library(akmeans)
library(BBmisc)
```

```
## Warning: package 'BBmisc' was built under R version 3.6.3
```

```
##
## Attaching package: 'BBmisc'
```

```
## The following object is masked from 'package:base':
##
##     isFALSE
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v forcats 0.4.0
## v readr   1.3.1
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
## Warning: package 'tibble' was built under R version 3.6.2
```

```
## Warning: package 'tidyr' was built under R version 3.6.2
```

```
## Warning: package 'readr' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.2
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
## Warning: package 'forcats' was built under R version 3.6.2
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x ggplot2::annotate() masks NLP::annotate()
## x dplyr::coalesce()   masks BBmisc::coalesce()
## x dplyr::collapse()   masks BBmisc::collapse()
## x tidyr::expand()     masks Matrix::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x tidyr::pack()       masks Matrix::pack()
## x tidyr::unpack()     masks Matrix::unpack()
```

```
# margin=2 so that it does it by column instead of by row
dtm.mat2 = normalize(dtm.mat, method="standardize", margin=2)

set.seed(10)
```

```r
clustering = norm.sim.ksc(dtm.mat2, 8)

cf = data.frame(clustering$centers)

colnames(cf) = colnames(dtm.mat)

rownames(cf) = 1 : 8

cf_transpose = t(cf)

# transpose data so we can have the columns be clusters
cf_t_df = data.frame(cf_transpose)

# Orders words in clusters and shows top 25
c1 = cf_t_df[order(-cf_t_df$X1),,drop=FALSE]
c2 = cf_t_df[order(-cf_t_df$X2),,drop=FALSE]
c3 = cf_t_df[order(-cf_t_df$X3),,drop=FALSE]
c4 = cf_t_df[order(-cf_t_df$X4),,drop=FALSE]
c5 = cf_t_df[order(-cf_t_df$X5),,drop=FALSE]
c6 = cf_t_df[order(-cf_t_df$X6),,drop=FALSE]
c7 = cf_t_df[order(-cf_t_df$X7),,drop=FALSE]
c8 = cf_t_df[order(-cf_t_df$X8),,drop=FALSE]


# Top 25 words for each cluster

rownames(head(c1,25))
```

```
##  [1] "labor"         "cole"          "costa"         "panama"
##  [5] "border"        "propon"        "rica"          "span"
##  [9] "sage"          "iran"          "development"   "encyclopaedia"
## [13] "leed"          "lift"          "mead"          "anthropolog"
## [17] "foucault"      "cnn"           "ethnic"        "contin"
## [21] "michel"        "suprieur"      "surviv"        "todd"
## [25] "worker"
```

```r
rownames(head(c2,25))
```

```
##  [1] "nonfict"  "pulitz"   "prizewin" "darwin"   "prize"    "maxwel"
##  [7] "jonathan" "runnerup" "report"   "gray"     "stop"     "plasma"
## [13] "writer"   "newspap"  "deborah"  "won"      "fusion"   "hart"
## [19] "blood"    "rosemari" "albanes"  "alcott"   "alicia"   "avalanch"
## [25] "eckert"
```

```r
rownames(head(c3,25))
```

```
##  [1] "analys"        "algorithm"     "function"      "mathemat"
##  [5] "combinatori"   "analysi"       "wellknown"     "disciplin"
##  [9] "reli"          "analyt"        "combinator"    "thesi"
## [13] "supervis"      "robert"        "mathematician" "machineri"
## [17] "explicit"      "ture"          "goldman"       "bell"
## [21] "ali"           "drop"          "mathematica"   "olympiad"
## [25] "adob"
```

```r
rownames(head(c4,25))
```

```
##  [1] "editori"     "council"     "hong"        "advisori"    "technic"
##  [6] "transact"    "chair"       "kong"        "member"      "electr"
## [11] "commerc"     "professor"   "journal"     "committe"    "polish"
## [16] "optic"       "editor"      "joint"       "poland"      "board"
## [21] "specialist"  "distinguish" "process"     "deliv"       "presidenti"
```

```
rownames(head(c5,25))
```

```
##  [1] "festschrift" "literatur"   "languag"     "russian"     "present"
##  [6] "moscow"      "modern"      "claud"       "avantgard"   "light"
## [11] "russia"      "civil"       "protocol"    "serious"     "vladimir"
## [16] "awardsh"     "expir"       "garrison"    "semiot"      "sergei"
## [21] "cultur"      "postwar"     "broad"       "polic"       "lend"
```

```
rownames(head(c6,25))
```

```
##  [1] "had"       "mechan"    "caltech"   "low"       "dynam"     "space"
##  [7] "amount"    "lack"      "suffer"    "put"       "aerospac"  "astronom"
## [13] "largescal" "flight"    "fuel"      "matter"    "observ"    "cosmolog"
## [19] "fluid"     "courant"   "captur"    "chatham"   "astronomi" "mitchel"
## [25] "prove"
```

```
rownames(head(c7,25))
```

```
##  [1] "ivi"           "player"         "undef"          "philanthropist"
##  [5] "leagu"         "championship"   "team"           "entrepreneur"
##  [9] "basketbal"     "wellb"          "champion"       "final"
## [13] "firstteam"     "flourish"       "philanthrop"    "vibrant"
## [17] "all"           "coach"          "profession"     "allivi"
## [21] "men"           "shot"           "skill"          "path"
## [25] "ncaa"
```

```
rownames(head(c8,25))
```

```
##  [1] "poet"         "wale"         "australia"    "metropolitan" "poetri"
##  [6] "getti"        "icon"         "poem"         "sister"       "downtown"
## [11] "carolyn"      "darci"        "sydney"       "australian"   "york"
## [16] "resid"        "museum"       "san"          "photographi"  "jose"
## [21] "burk"         "lee"          "oreilli"      "club"         "angelesin"
```

```
# Find only words > 0 in the clusters
c11 = subset(cf_t_df, X1 > 0)
c21 = subset(cf_t_df, X2 > 0)
c31 = subset(cf_t_df, X3 > 0)
c41 = subset(cf_t_df, X4 > 0)
c51 = subset(cf_t_df, X5 > 0)
c61 = subset(cf_t_df, X6 > 0)
c71 = subset(cf_t_df, X7 > 0)
c81 = subset(cf_t_df, X8 > 0)

q1 = quantile(c11$X1, probs = c(0, .25, .50, .75, 1))
q2 = quantile(c21$X2, probs = c(0, .25, .50, .75, 1))
q3 = quantile(c31$X3, probs = c(0, .25, .50, .75, 1))
q4 = quantile(c41$X4, probs = c(0, .25, .50, .75, 1))
q5 = quantile(c51$X5, probs = c(0, .25, .50, .75, 1))
q6 = quantile(c61$X6, probs = c(0, .25, .50, .75, 1))
q7 = quantile(c71$X7, probs = c(0, .25, .50, .75, 1))
q8 = quantile(c81$X8, probs = c(0, .25, .50, .75, 1))
```

```
q1
```

```
##           0%          25%          50%          75%         100%
## 3.197356e-05 5.503459e-03 1.215094e-02 1.796611e-02 6.290604e-02
```

```
q2
```

```
##           0%          25%          50%          75%         100%
## 1.447471e-05 5.967294e-03 1.258497e-02 2.261522e-02 1.036576e-01
```

```
q3
```

```
##           0%          25%          50%          75%         100%
## 2.608571e-05 5.912201e-03 1.210510e-02 2.047241e-02 7.390178e-02
```

```
q4
```

```
##           0%          25%          50%          75%         100%
## 5.240915e-05 5.235510e-03 1.163364e-02 1.614885e-02 7.433143e-02
```

```
q5
```

```
##           0%          25%          50%          75%         100%
## 3.286962e-05 5.729329e-03 1.116293e-02 1.680170e-02 6.568419e-02
```

```
q6
```

```
##           0%          25%          50%          75%         100%
## 4.055549e-05 6.212738e-03 1.200023e-02 1.694789e-02 6.970646e-02
```

```
q7
```

```
##           0%          25%          50%          75%         100%
## 3.017091e-05 6.021709e-03 1.273238e-02 1.834829e-02 7.968314e-02
```

```
q8
```

```
##           0%          25%          50%          75%         100%
## 0.0000488962 0.0062407061 0.0128871160 0.0186635184 0.0830287666
```

Top 25 Words were printed in the R-output above, please don't require me to type them all out again. Thank you.

Q4. The Graphical Model (50 points)

Part I. Nationwide GDP Growth Correlation (15 points)

Question 1 (5 points):

```
load("gdp.RData")

library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.2
```

```
## Loaded glmnet 3.0-2
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 3.6.3
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:purrr':
##
##     compose, simplify

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following object is masked from 'package:tibble':
##
##     as_data_frame

## The following object is masked from 'package:BBmisc':
##
##     normalize

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
gdp=gdp[c(which((rowSums(is.na(gdp)) == ncol(gdp)) == FALSE)),]

# Now we need to find the average for every country that has any na values at all

indicies = which(is.na(gdp), arr.ind=TRUE)

for(i in 1:310){
  sum = 0
  count = 0
  for(j in 1:14){
    if(is.na(gdp[indicies[i,1],j]) == TRUE){
      sum = sum + 0
      count = count + 0
    } else {
      sum = sum + gdp[indicies[i,1],j]
      count = count + 1
    }
  }
  mean = sum/count
  gdp[indicies[i,1], indicies[i,2]] = mean
}

# Fully cleaned gdp dataset
```

Question 2 (10 points):

```
# to get number of countries = NROW

d = NROW(gdp)
```

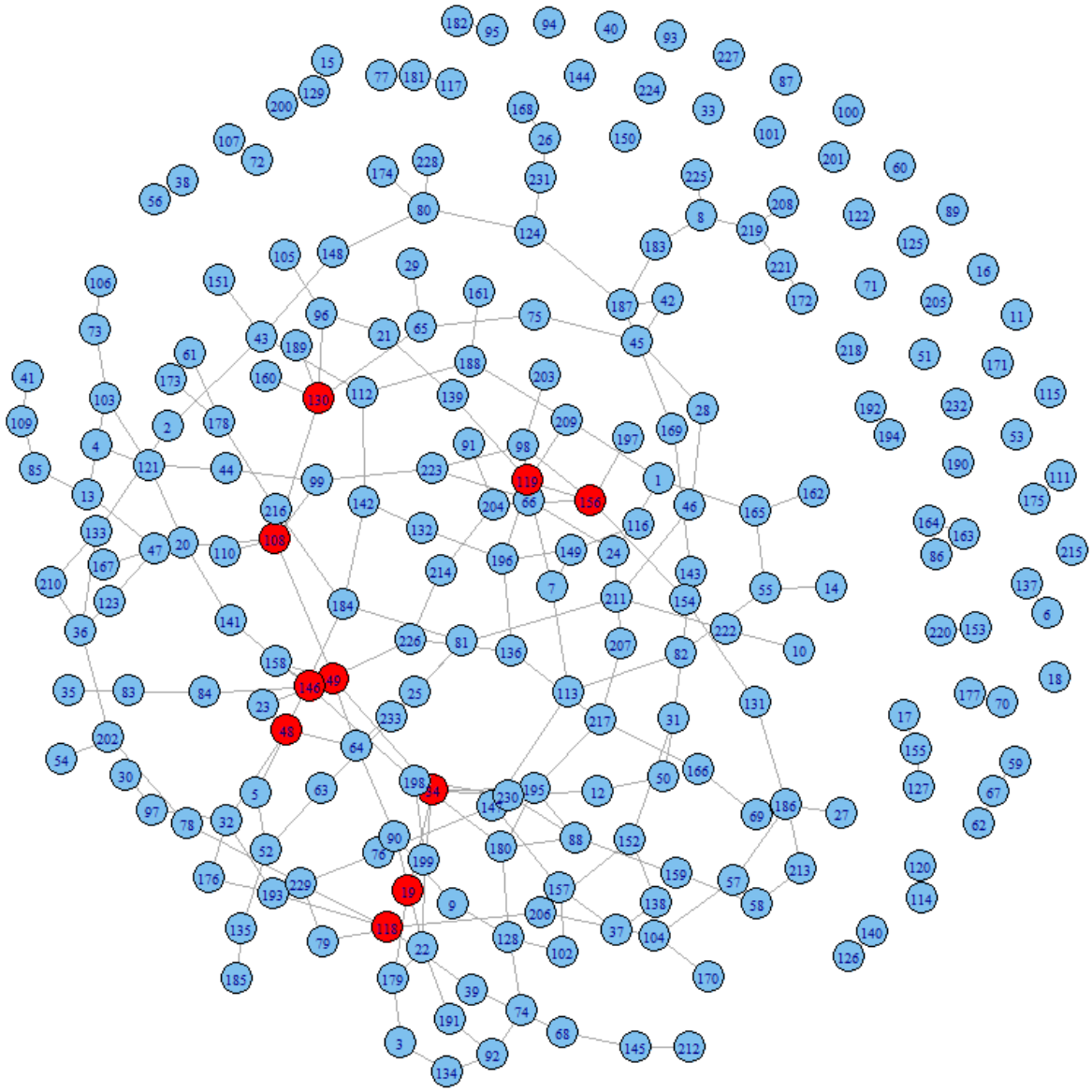
```r
M = matrix(0, nrow = d, ncol = d)

# We need to transpose our dataset now however, since all of our countries are rows instead of columns

gdpt = t(gdp)

gdpt = as.data.frame(gdpt)

gdp_matrix = data.matrix(gdpt)


# Performing Lasso regression on each country
for(i in 1:d){
  model = glmnet(x=gdp_matrix[,-i], y=gdp_matrix[,i], family = "gaussian", lambda=1, alpha=1)

  # Get the coefficients
  coefs = coef(model)[-1]

  coefs1 = vector(length = d)
  coefs1[1:i-1] = coefs[1:i-1]
  coefs1[i] = 0

  if(i != d){
    coefs1[(i+1):d] = coefs[i:(d-1)]
  }

  M[i,] = as.numeric(coefs1 != 0)
}


# And Rule

# Loop through rows of M
for (i in 1:d)
{
  # Loop through each column in row i
  for (j in i:d)
  {
    # M[i,j] checks country i's relationship with country j
    # Checking j relationship on i tells us if they're both factors
    # If they are both 0, they aren't factors, nothing to be done
    # If one is non-zero at least,
    if (M[i,j] != 0 || M[j,i] != 0)
    {
      # Check if one is zero, if so make both 0
      # else we leave as is if both are non-zero
      if (M[i,j] == 0 || M[j,i] == 0)
      {
        M[i,j] = 0
        M[j,i] = 0
      }
    }
  }
```

```
}


#### q4 SCRIPT ####
ag=graph.adjacency(M, mode="undirected")
V(ag)$colors=ifelse(degree(ag)<5, 'SkyBlue2', 'red')
png(filename="gdp_nodewise.png", height=800, width=800)
par(mai=c(0,0,0,0))
plot.igraph(ag, vertex.color=V(ag)$colors, vertex.size=6, vertex.label.cex=0.8, layout=layout_nicely(ag
dev.off()
```

```
## pdf
##   2
```

```
# We get red nodes for 3 and 48 so getting those countries

colnames(gdp_matrix)[c(19,34,48,49,108,118,119,130,146,156)]
```

```
##  [1] "Bulgaria"                      "Central Europe and the Baltics"
##  [3] "Cyprus"                        "Czech Republic"
##  [5] "Cambodia"                      "Libya"
##  [7] "St. Lucia"                     "Latvia"
##  [9] "Montenegro"                    "Nigeria"
```



Bulgaria, Central Europe and the Baltics, Cyprus, Czech Republic, Cambodia, Libya, St. Lucia, Latvia, Montenegro, Nigeria are the ten countries with degrees greater than or equal to 5 on our node graph.

Part 2: Preliminary Theories (10 points)

Question 3 (technically 1) (5 points):

Global Independence not leading to Conditional Independence:

Consider three events, all related to coin tosses, A, B, and C. Let A be the event our first coin flip lands 'Heads'. let B be the event our second coin flip lands 'Heads'. Now, let C be the event that both of we land ('Heads', 'Heads') in two coinflips. Now event A and B are independent but they are not conditionally independent to C. If we know that event C and A have occurred, that guarantees event B will also occur. Hence, global independence will not lead to conditional independence.

Conditional Independence not leading to Global Independence:

Again we will go with a coin toss example. Assume you have two un-fair coins that have a greater than 50% chance to either roll 'Heads' or 'Tails'. Let A and B be the same events from above, of rolling 'Heads' on the first and second coin toss respectively. Now, if A occurs, we know that B has a very high chance to occur as well as we have likely chosed the 'Heads' rigged coin. Clearly these two events are now not independent. Now, let event C be the probability we got the 'Heads' rigged coin out of a hat. This implies $P(C) = \frac{1}{2}$. Now A and B are conditionally independent, given C. If you know that C occurred, then you are just doing an experiment where you take an unfair, 'Heads' rigged coin, and flip it twice. Whether the first flip was 'Heads' or not, we don't care because we know we are using this rigged coin, and testing its probability. Thus our Conditional Independence does not resullt in Global Independence.

Question 4 (technically 2) (5 points):

a) Does the note set $\{3\}$ separate node set $\{5\}$ and node set $\{7\}$

Answer: No it does not because we can connect node set $\{5\}$ and node set $\{7\}$ through the node set $\{1, 2, 6\}$

b) Does the node set $\{8\}$ separate node set $\{7\}$ and node set $\{9\}$

Answer: Yes it does, if we remove node set $\{8\}$ there is no way to connect node set $\{7\}$ and $\{9\}$.

c) Answer: Yes this is true.

D) Answer: Yes this is true.

E) Answer: No, since $\{5\}$ does not separate $\{1\}$ and $\{3\}$.

Part 3: The Gaussian Graphical Model (15 points):

Question 5 (5 points):

We know that X always follows a multivariate normal distribution in a Gaussian Graphical Model:

$$X \sim N_d(0, \Sigma)$$

Here we have $A \subseteq \{1, \ldots, d\}$ with the A in our question being equal to $A = \{1, 2\}$. Therefore, let $A^c = 3, \ldots, d$

Using the conditional distribution from (0.13) of X_A given X_{A^c} we get the following:

$$X_A | X_{A^c} \sim N(\Sigma_{AA^c}\Sigma^{-1}_{A^cA^c}\Sigma_{A^cA})$$

Using (0.14) we know we can put X_A in terms of Σ and an additional ϵ variable such that,

$$X_A = X_{A^c} \cdot \beta_A + \epsilon_A$$

Where β is a $(d-1)$ dimensional-vector equal to,

$$\beta_A = \Sigma^{-1}_{A^cA^c}\Sigma_{A^cA}$$

And,

$$\epsilon_A \sim N(0, \Sigma_{AA^c}\Sigma^{-1}_{A^cA^c}\Sigma_{A^cA})$$

Hence, we know that $Var(\epsilon_A) = \sigma_j^2 = \Sigma_{AA} - \Sigma_{AA^c}\Sigma^{-1}_{A^cA^c}\Sigma_{A^cA}$

Therefore we know that,

$$\Theta_{AA} = (\Sigma_{AA} - \Sigma_{AA^c}\Sigma^{-1}_{A^cA^c}\Sigma_{A^cA})^{-1}$$

which is equivalent to,

$$\Theta_{AA}^{-1} = \Sigma_{AA} - \Sigma_{AA^c}\Sigma_{A^cA^c}^{-1}\Sigma_{A^cA}$$

Question 6 (5 points):

We know that the variance for our conditional distribution is Θ_{AA}^{-1} as shown from question 5 and (0.13).

In our theorem, we have that $X_j \perp X_k \sim X_{/jk}$ (where the '/' indicates we are taking the conditional distribution between 'j' and 'k').

Since our variables are perpendicular, hence independant, this would imply that for our Variance $\Theta_{AA}^{-1} = \Sigma_{AA} - \Sigma_{AA^c}\Sigma_{A^cA^c}^{-1}\Sigma_{A^cA}$, both Σ_{AA^c} and Σ_{A^cA} are 0 by definition. Therefore, the product must also be 0. Now we're left with $\Theta_{AA}^{-1} = \Sigma_{AA}$ as wanted. This maintains the definition of our precision matrix, $\Theta = \Sigma^{-1}$, as well as showing that if any values in our precision matrix are zero values, we have conditional independence.

In general, if $\Theta_{AA}^{-1} = \Sigma_{AA}$ then, $\Theta_{jk} = \Sigma_{jk}^{-1}$ and since earlier in the question we showed that $\Sigma_{jk} = \Sigma_{kj} = 0$. Therefore $\Theta_{jk} = 0$ as wanted.

Question 7 (5 points):

Showing $\epsilon \perp X_{/1}$ (or in the previous question X_{A^c}), that would be the same as showing $\Theta_{11} \perp X_{/1}$. However, one of these values is a constant, while the other is a random variable. Knowing this information, we have to show that $X_{/1}$ can be the negative reciprocal of Θ_{11}. And that is sadly something I have no idea how to do...

For part 2 of this question we want to show that $\beta = \Theta_{11}^{-1}\Theta_{/1,1}$

Continuing from Question 5,

$$\Theta_{AA} = (\Sigma_{AA} - \Sigma_{AA^c}\Sigma_{A^cA^c}^{-1}\Sigma_{A^cA})^{-1}$$
$$\Theta_{AA} = (\Sigma_{AA} - (2\beta_A)^T\Sigma_{A^cA} + \beta_j^T\Sigma_{A^cA^c}\beta_j)^{-1}$$
$$\Theta_{A^cA} = -\Theta_{AA}\beta_A$$
$$-\Theta_{AA}^{-1}\Theta_{A^cA} = \beta_A$$

as wanted.

Part 4. The Ising model (10 points)

Question 8 (5 points):

Honestly man, I don't know how to do this question, I'll give a brief explanation with words, to see if I can get some part marks.

The Ising model seems similar to GGMs, except instead of $X \sim N(0, \Sigma)$ our X seems to follow more of a Bernouilli Distribution. Knowing this fact, and how conditional independence holds for bernouilli random variables, I can extrapolate that Z is independent of X but it is also in part X. Due to this Z existing in our equation and causing a potential conditional (in)dependence, we can show that each individual X_i, X_j where $i \neq j$, these two things are conditionally independent as well, thus causing our $\beta_{jk} = 0$.

Question 9 (5 points):

Similarly to above, all I can give is an explanation via. words as I don't fully understand the concept and there's not enough time left for me to continue working on this assignment.

What we're asking for is the probability we label as 1 if our covariates equate to a particular set of values. We know that for all of our β_{jk} they equal 0 for all $j \neq k$. With this information, this means that whenever our covariates equate to their particular $x_{/j}$ we will get the value we need, since that won't be 0. This makes our given hold true, and proves that it will occur. Thus, since this probability function just assigns a class label, using covariates, it is by definition a form of regression.

Q5. Develop Your Own AlphaGO (20 points)

Since I'm confused on how to submit, I'll put the code for AlphaGo into the main file and edit it here.

```
#install.packages("IsingSampler")
library(IsingSampler)
```

```
## Warning: package 'IsingSampler' was built under R version 3.6.3
```

```
## Loading required package: Rcpp
```

```
# w[1]: w_chains,   w[2]: w_inter-chain
# w[3]: w_chain-empty,   w[4]: w_empty
# w[5]: h_stone
w <- c(2.47, 0.521, 0.442, 0.427, 0.265)

# Go board for game 2
C2 = as.matrix(read.table("AlphaGo-vs-Lee-game2_80.txt", header=FALSE, sep = ",") )
# Go board for game 4
C4 = as.matrix(read.table("AlphaGo-vs-Lee-game4_80.txt", header=FALSE, sep = ",") )

construct.ising.graph <- function(weight, c) { # Complete the following function
    g=matrix(0, 19^2, 19^2)
    for (i in 1:19){
        for (j in 1:19){ # Enumerate every point on the board. i is the row index. j is the column inde
            i0=(j-1)*19+i
            i1=j*19+i # right neighbor
            i2=(j-1)*19+i+1 # lower neighbor
            if (j<19){
                if (c[i,j]*c[i,j+1]==1){
                    g[i0, i1]=weight[1]
                    g[i1, i0]=weight[1]
                }
                else if (c[i,j]*c[i,j+1]==-1){
                    g[i0, i1]=weight[2]
                    g[i1, i0]=weight[2]
                }
                else if ((c[i,j]==0)&(c[i, j+1]==0)){
                    g[i0, i1]=weight[4]
                    g[i1, i0]=weight[4]
                }
                else {
                    g[i0, i1]=weight[3]
                    g[i1, i0]=weight[3]
                }
            }
            if (i<19){
                if (c[i,j]*c[i+1,j]==1){
                    g[i0, i2]=weight[1]
                    g[i2, i0]=weight[1]
                }
                else if (c[i,j]*c[i+1,j]==-1){
                    g[i0, i2]=weight[2]
                    g[i2, i0]=weight[2]
                }
                else if ((c[i,j]==0)&(c[i+1, j]==0)){
```

```
                    g[i0, i2]=weight[4]
                    g[i2, i0]=weight[4]
                }
                else {
                    g[i0, i2]=weight[3]
                    g[i2, i0]=weight[3]
                }
            }
        }
    }
    return(g)
}

predict_go <- function(w, C, nsample){
  # C is a matrix, obtained from the current go board
  # do sampling
  W <- construct.ising.graph(w, C); # Coefficient matrix for the coupling term.

  # Complete the line below with your code (just one line of code)
  h <-  w[5] * as.vector(C)# Coeffcient vector for the external field term.

  S_mat <- IsingSampler(nsample, graph = W,
                        thresholds = h,
                        nIter = 100,
                        response = c(-1L,1L)) # Each row of S_mat is a simulated realization of the fin

  s <- colMeans(S_mat); # s is the empirical expectation of the final territory outcome.
  val = sum(s) - 3.75; # Scoring adjustment
  if(val >0){
    result = TRUE
  }
  else{
    result = FALSE
  }
  return(list(mean = s, result = result))
}# return the expectation

set.seed(2016)

pred_game2 = predict_go(w, C2, nsample = 500)
expectation = pred_game2$mean;
result = pred_game2$result
if(result){
  cat(sprintf("Alpha-go wins game 2"))
}else{
  cat(sprintf("Lee Sedol wins game 2"))
}
```

```
## Alpha-go wins game 2
```

```
pred_game4 = predict_go(w, C4, nsample = 500)
expectation = pred_game4$mean;
result = pred_game4$result
if(result){
```

```
  cat(sprintf("Alpha-go wins game 4"))
}else{
  cat(sprintf("Lee Sedol wins game 4"))
}
```

```
## Lee Sedol wins game 4
```

Despite the extensions you gave, that doesn't change the fact that the difficulty of this assignment was honestly quite hard. Hopefully I can pass this assignment and course.