STAD80 Analysis of Big Data **Assignment: 3**

Unsupervised Learning

Lecturer: Qiang Sun *Email: qiang.sun@utoronto.ca*

Due Date: This assignment is due on Monday, March 16th by 5:00PM.

Submission: Please submit the hard copies to Yicheng Zeng in the tutorial and the electronic versions on Quercus.

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

We do not have time for linear discriminant analysis and quadratic discriminant analysis in class. In this question, we'll derive the linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) for multivariate Gaussians. We'll provide subquestions that guide you through the derivation.

Let our training set be $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, 2\}$. Under the LDA framework, we assume

$$p(\boldsymbol{x}_i|y_i = 1) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-(\boldsymbol{x}_i-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_1)/2}$$
$$p(\boldsymbol{x}_i|y_i = 2) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-(\boldsymbol{x}_i-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_2)/2}$$

where $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}^{-1}$ denotes the inverse of $\boldsymbol{\Sigma}$. Under the QDA framework, we instead assume

$$p(\boldsymbol{x}_i|y_i = 1) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_1|^{1/2}} e^{-(\boldsymbol{x}_i-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_1)/2}$$
$$p(\boldsymbol{x}_i|y_i = 2) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_2|^{1/2}} e^{-(\boldsymbol{x}_i-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_2)/2}$$

These are multivariate Gaussians with means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and different covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$. We assume that each data point $(\boldsymbol{x}_i, y_i)$ is independent of each other.

We denote $\eta = \mathbb{P}(y_i = 1)$ (the prior probability of class 1), $n_1 = \sum_{i=1}^n I(y_i = 1)$ and $n_2 = n - n_1$.

Question 1.1: Likelihood Model (10 points)

For LDA, write down the joint log-likelihood of a parameter configuration $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \eta$ with respect to the observed data

$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$.

That is, write down the factorization of

$$\sum_{i=1}^n \log p(\boldsymbol{x}_i, y_i).$$

Use the term η in your answer. [This question reviews log-likelihood.]

Hint: (Swapping Determinant and Inverse) Use $\log\big(|\boldsymbol{\Sigma}|^{-1}\big) = \log|\boldsymbol{\Sigma}^{-1}|$. This will be useful in Q4.3.

Question 1.2: MLE Estimation for η and $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ (10 points)

For LDA, using the log-likelihood equation developed in Q4.1, determine the maximum likelihood estimator $\widehat{\eta}$, the estimator for η. Likewise, determine the maximum likelihood estimators $\widehat{\boldsymbol{\mu}}_1$ and $\widehat{\boldsymbol{\mu}}_2$ for $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$. Use the terms n_1, n_2 in your answer. [This question reviews MLE estimation.]

(For the rest of this problem, you may assume that the log-likelihood equation is concave.)

Question 1.3: MLE Estimation for $\boldsymbol{\Sigma}$ (10 points)

For LDA, using the log-likelihood equation developed in Q4.1, determine the maximum likelihood estimator $\widehat{\boldsymbol{\Sigma}}$, the estimator for $\boldsymbol{\Sigma}$. Let

$$S_1 = \frac{1}{n_1} \sum_{i:y_i=1}^{n} (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_1)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_1)^T,$$

$$S_2 = \frac{1}{n_2} \sum_{i:y_i=2}^{n} (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_2)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_2)^T,$$

where $\widehat{\boldsymbol{\mu}}_1$ and $\widehat{\boldsymbol{\mu}}_2$ are the MLEs obtained in Question 4.2. Use the terms S_1 and S_2 in your answer. (Taking the derivative with respect to $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ might make your life easier.) [This question shows you how to use basic facts in linear algebra to derive MLE.]

Hint 1: Somewhere in your calculation, you will need the following facts. For a symmetric matrix $\boldsymbol{A}$,

$$\frac{\partial \log |\mathbf{A}|}{\partial \mathbf{A}} = \mathbf{A}^{-1}.$$

Hint 2: If c is some scalar computed by $\boldsymbol{b}^T \mathbf{A} \boldsymbol{b}$ for a vector $\boldsymbol{b}$ and matrix $\mathbf{A}$, we can write this term as $c = \text{Tr}(\boldsymbol{b}^T \mathbf{A} \boldsymbol{b})$ where $\text{Tr}(\cdot)$ is the trace of a matrix.

Hint 3: (Commutative Property of Trace): For matrices $\mathbf{A} \in \mathbb{R}^{d_1 \times d_2}$ and $\mathbf{B} \in \mathbb{R}^{d_2 \times d_1}$, $\text{Tr}(\mathbf{BA}) = \text{Tr}(\mathbf{AB})$.

Hint 4: For a matrix $\mathbf{A}$ and $\mathbf{B}$,

$$\frac{\partial \text{Tr}(\mathbf{B}^T \mathbf{A})}{\partial \mathbf{A}} = \mathbf{B}.$$

Question 1.4: Decision Boundary (10 points)

Recall that the LDA framework utilizes a ratio

$$\frac{P(y_i = 1 \,|\, \boldsymbol{x}_i)}{P(y_i = 2 \,|\, \boldsymbol{x}_i)}$$

to predict whether a data point $\boldsymbol{x}_i$ has unknown label $y_i = 1$ or $y_i = 2$. If this ratio is larger than 1, we predict $y_i = 1$. Else, we predict $y_i = 2$. Clearly, we are interested in knowing when $\frac{P(y_i=1 \,|\, \boldsymbol{x}_i)}{P(y_i=2 \,|\, \boldsymbol{x}_i)} = 1$ since this will represent our decision boundary. In the previous questions Q4.2 and Q4.3, we trained our LDA classifier to determine $\widehat{\eta}$, $\widehat{\boldsymbol{\mu}}_1$, $\widehat{\boldsymbol{\mu}}_2$ and $\widehat{\boldsymbol{\Sigma}}$. Now we are concerned with how to use these fitted parameters to derive the classifier.

We can rewrite the decision boundary as

$$\log \frac{P(y_i = 1 \,|\, \boldsymbol{x}_i)}{P(y_i = 2 \,|\, \boldsymbol{x}_i)} = 0. \tag{0.1}$$

Show that this decision boundary can be written as $\boldsymbol{w}^T x_i + w_0$ for a vector $\boldsymbol{w}$ and a scalar w_0. Use the terms $\widehat{\eta}$, $\widehat{\boldsymbol{\mu}}_1$, $\widehat{\boldsymbol{\mu}}_2$ and $\widehat{\boldsymbol{\Sigma}}$ in your answer. [This question shows why the decision boundary of LDA is linear, hence why it is called a linear discriminant analysis.]

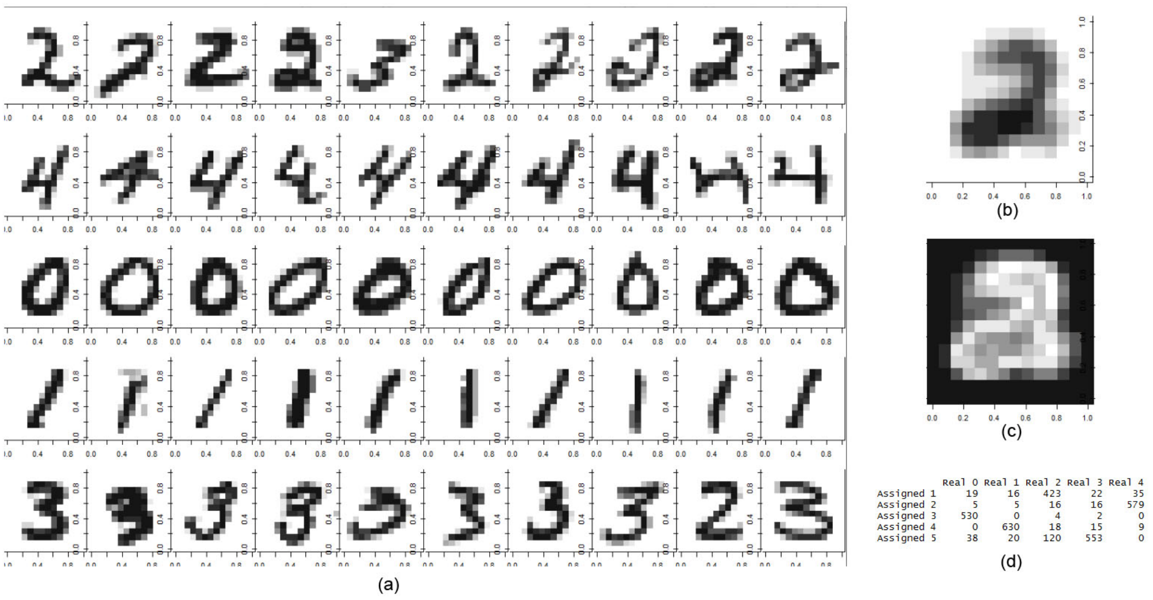
Question 1.5: QDA (10 points)

Using similar strategies, derive the MLE estimators $\widehat{\eta}$, $\widehat{\mu}_1$, $\widehat{\mu}_2$, $\widehat{\mathbf{\Sigma}}_1$ and $\widehat{\mathbf{\Sigma}}_2$ for QDA. You do not need to show the derivations; instead, directly give the solutions. In addition, show that the decision boundary (0.1) for QDA is a quadratic curve. Use the terms $\widehat{\eta}$, $\widehat{\boldsymbol{\mu}}_1$, $\widehat{\boldsymbol{\mu}}_2$, $\widehat{\mathbf{\Sigma}}_1$ and $\widehat{\mathbf{\Sigma}}_2$ in your answer.

Q2. Clustering Algorithms (50 points)

Please download the MNIST handwritten digit dataset from `http://yann.lecun.com/exdb/mnist/`. It contains 28×28-pixel images for the hand-written digits {0,1,. . . ,9} by storing each pixel value ranging between 0 and 255, and their corresponding true labels. Load the data into R and preprocess the data by compressing each image to $1/4$ of the original size in the following way: Divide each 28×28 image into 2×2 non-overlapping blocks. Calculate the mean pixel value of each 2×2 block, and create a new 14×14 image. This preprocessing step will drastically help your computation. We will be clustering the digits $\{0, 1, \ldots, 4\}$ in this homework.

You will not be graded on the performance of your code but rather on the correctness of your algorithm.



```
           Real 0 Real 1 Real 2 Real 3 Real 4
Assigned 1     19     16    423     22     35
Assigned 2      5      5     16     16    579
Assigned 3    530      0      4      2      0
Assigned 4      0    630     18     15      9
Assigned 5     38     20    120    553      0
```

Figure 1: (a): Example of Plot 1. Notice that we do not have perfect clusters. (b): Example of Plot 2 for cluster "2". The mean digit looks like a 2, so this is good. (c): Example of Plot 3 for cluster 2. The area of high variance is in the shape of a 2, so this means our cluster did indeed cluster mostly 2's. If our plot was amorphous, this would be bad. (d): Example of Table 1 on only a subset of the data. Notice that each column has one unique row-element that dominates all the others. We can see here we did not prefectly cluster.

For Questions 1.1 and 1.2, you should attach the code and also generate the following plots/table for each algorithm. (When doing these for the EM algorithm, you may assign each image to its most likely cluster.)

- Plot 1: (Visualization) For each cluster, plot 12 random data points in that cluster (as illustrated in Figure 1(a) for clustering digits {0,1,2,3,4}).
- Plot 2: (Cluster Center) For each cluster j, visualize the cluster center $\boldsymbol{\mu}_j$. (An example is shown in Figure 1(b)

for clustering digits {0,1,2,3,4}.)

- Plot 3: (Variance Visualization) For each cluster, visualize the variance among all the data points in that cluster as a heat map. That is, for each cluster, for each pixel position in the 14×14 image, calculate the variance of the values for that pixel position among all images in the cluster. After you have done this, scale all 196 values and load them into a new 14×14 image in such a way that black pixels in this image denote very low variance in that pixel position among all the digits in the cluster, white pixels denote very high variance. (You should rescale all the 196 pixel values in the heat map into the range [0, 255]. An example is shown in Figure 1(c) for clustering digits {0,1,2,3,4}.)
- Table 1: (Distribution of Labels and Error Rates) For each cluster, using the assigned true labels (provided in the dataset), denote which cluster represents which digit. Count how many images in that cluster had the actual label (as illustrated in Figure 1(d) for clustering digits {0,1,2,3,4}) and report the error rate (the percentage of misclassified digits) within each cluster.

Question 1: K-Means

Program (in R) the K-means algorithm and use it to cluster the digit dataset . We set $K = 5$. If a tie happens, we assign the data to the lowest-indexed cluster. Try 3 random initializations and pick the one that minimizes the objective function

$$\sum_{i=1}^{n} \|\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}_i}\|_2^2 \tag{0.2}$$

where $\boldsymbol{\mu}_{\mathbf{x}_i}$ is the cluster center that the data point $\mathbf{x}_i$ is closest to.

Question 2: Expectation-Maximization Algorithm

Use the Gaussian mixture model as described in lecture to cluster the digit dataset.

Closely following the derivation in class, we now derive the EM algorithm for Gaussian mixture models. More specifically, we will use 2 models, the "mixture of spherical Gaussians" and the "mixture of diagonal Gaussians". By the end of this question, you should have derived two EM algorithms, one for each models.

Let $\mathbf{x}_1, \dots, \mathbf{x}_n$ be n i.i.d. data points, where $\mathbf{x}_i \in \mathbb{R}^d$. The likelihood of the data for k clusters is:

$$\begin{aligned} L(\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \eta_j\}_{j=1}^k) &= \prod_{i=1}^n p(\mathbf{x}_i; \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \eta_j\}_{j=1}^k) \\ &= \prod_{i=1}^n \sum_{j=1}^k p(\mathbf{x}_i | Z_i = j; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) p(Z_i = j; \eta_j) \\ &= \prod_{i=1}^n \sum_{j=1}^k \frac{\eta_j}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_j|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)\right) \end{aligned} \tag{0.3}$$

The following denote the meaning of each symbol:

- each $\boldsymbol{\mu}_j$ is a d-dimensional vector representing the cluster center for cluster j
- each $\boldsymbol{\Sigma}_j$ is a $d \times d$ matrix representing the covariance matrix for cluster j
- $\sum_{j=1}^k \eta_j = 1$ and $\forall j, \eta_j \geq 0$ where all the η_j are the mixing coefficients.
- Z_i represents the hidden variable associated with x_i for $i \in \{1, \dots, n\}$. It takes integer values $1, \dots, k$.

In a **mixture of diagonal Gaussians** model, $\mathbf{\Sigma}_j = \mathrm{diag}(\sigma^2_{j_1}, \dots, \sigma^2_{j_d})$ is a diagonal matrix for all $j = 1, \dots, k$.

In a **mixture of spherical Gaussians** model, $\mathbf{\Sigma}_j = \sigma_j^2 \mathbf{I}_d$ for all $j = 1, \dots, k$. Here $\sigma_j > 0$ is a scalar and $\mathbf{I}_d$ is the $d \times d$ identity matrix.

(i) First, we do some preliminary work that will be useful later on.

- **Part a:** Write down the marginal distribution of the Z_i (Hint: What is the probability $p(Z_i = j)$).
- **Part b:** Calculate $p(Z_i = j \mid \mathbf{x}_i)$. (Hint: Bayes Rule)

(ii). We now derive the E- and M- steps. We denote $\boldsymbol{\theta} := \{\boldsymbol{\mu}_j, \mathbf{\Sigma}_j, \eta_j\}_{j=1}^k$. From Equation (0.3), we can write down the log-likelihood and derive a lower bound:

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \sum_{i=1}^n \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) && (0.4)\\ &= \sum_{i=1}^n \log \left[\sum_{j=1}^k p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j) \right] && (0.5)\\ &= \sum_{i=1}^n \log \left[\sum_{j=1}^k \gamma_{ij} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{\gamma_{ij}} \right], && (0.6) \end{aligned}$$

where $\gamma_{ij} > 0$ for all i, j and satisfies

$$\sum_{j=1}^k \gamma_{ij} = 1 \ \text{ for } i = 1, \dots, n. \tag{0.7}$$

- **Part c:** Prove the following lower bound of the log-likelihood function:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^n \log \left[\sum_{j=1}^k \gamma_{ij} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{\gamma_{ij}} \right] \geq \sum_{i=1}^n \sum_{j=1}^k \gamma_{ij} \log \left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{\gamma_{ij}} \right]. \tag{0.8}$$

  Hint: you can use the Jensen's inequality $\log \mathbb{E} X \geq \mathbb{E} \log X$ (You are not required to prove the Jensen's inequality).

- **Part d:** (E-Step) We define

$$F(\boldsymbol{\gamma}, \boldsymbol{\theta}) := \sum_{i=1}^n \sum_{j=1}^k \gamma_{ij} \log \left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{\gamma_{ij}} \right] \tag{0.9}$$

  to be the lower bound function of $\ell(\boldsymbol{\theta})$. Let $\boldsymbol{\theta}^{\text{old}}$ be a fixed value of $\boldsymbol{\theta}$ (e.g., $\boldsymbol{\theta}^{\text{old}}$ could be the parameter value of the current iteration). Recall from Part c that we designed $F(\boldsymbol{\gamma}, \boldsymbol{\theta}^{\text{old}})$ to be a lower bound for $\ell(\boldsymbol{\theta}^{\text{old}})$. We want to make this lower bound as tight as possible.

  Prove that $\ell(\boldsymbol{\theta}^{\text{old}}) = F(\boldsymbol{\gamma}, \boldsymbol{\theta}^{\text{old}})$ when

$$\gamma_{ij} = p_{\boldsymbol{\theta}^{\text{old}}}(Z_i = j \mid \mathbf{x}_i). \tag{0.10}$$

  This result implies that, at the current parameter value $\boldsymbol{\theta}^{\text{old}}$, the function $F(\gamma, \boldsymbol{\theta}^{\text{old}})$ is maximized when γ_{ij} takes the form as in Equation (0.10). This justifies the E-step of the EM algorithm.

(iii). Once the E-step is derived, we now derive the M-step. First, we plug $\gamma_{ij} = p_{\boldsymbol{\theta}^{\text{old}}}(Z_i = j \,|\, \mathbf{x}_i)$ into Equation (0.9) and define the lower bound function at $\boldsymbol{\theta}^{\text{old}}$ to be

$$F_{\boldsymbol{\theta}^{\text{old}}}(\boldsymbol{\theta}) := \sum_{i=1}^{n} \sum_{j=1}^{k} p_{\boldsymbol{\theta}^{\text{old}}}(Z_i = j \,|\, \mathbf{x}_i) \log \left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, Z_i = j)}{p_{\boldsymbol{\theta}^{\text{old}}}(Z_i = j \,|\, \mathbf{x}_i)} \right]. \tag{0.11}$$

- **Part e** (M-step for mixture of spherical Gaussians) In the M-step, we aim to find $\boldsymbol{\theta}$ that maximize the lower bound function $F_{\boldsymbol{\theta}^{\text{old}}}(\boldsymbol{\theta})$. Under the **mixture of spherical Gaussians** model, derive the M-step updating equations for $\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ and η_j.
- **Part f** (M-step for mixture of diagonal Gaussians) Under the **mixture of diagonal Gaussians** model, derive the M-step updating equations for $\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ and η_j.

(iv) Program (in R) the EM algorithm you derived for mixture of spherical Gaussians. Assume 5 clusters. Terminate the algorithm when the fractional change of the log-likelihood goes under 0.0001. (Try 3 random initializations and present the best one in terms of maximizing the likelihood function).

(v) Program (in R) the EM algorithm you derived for mixture of diagonal Gaussians. Assume 5 clusters. Terminate the algorithm when the fractional change in the log-likelihood goes under 0.0001. (Try 3 random initializations and present the best one in terms of maximizing the likelihood function).

Hint: For these implementations, you will run into three different problems. Apply these following hints to solve each problem.

- 1) Use the log-sum-exp trick to avoid underflow on the computer. You will run into this problem when computing the log-likelihood. That is, when you calculate $\log \sum_j \exp^{a_j}$ for some sequence of variables a_j, calculate instead $A + \log \sum_j \exp^{a_j - A}$ where $A = \max_j a_j$.
- 2) Some pixels in the images do not change throughout the entire dataset. (For example, the top-left pixel of each image is always 0, pure white.) To solve this, after updating the covariance matrix $\boldsymbol{\Sigma}_j$ for the mixture of diagonal Gaussians, add $0.05\mathbf{I}_d$ to $\boldsymbol{\Sigma}_j$ (ie: add 0.05 to all the diagonal elements).
- 3) Be mindful of how you initialize $\boldsymbol{\Sigma}_j$. Note that for a diagonal matrix $\boldsymbol{\Sigma}_j$, the determinant $|\boldsymbol{\Sigma}_j|$ is the product of all the diagonal elements. Setting each diagonal element to a number too big at initialization will result in overflow on the computer.

Question 3: Relationship between K-Means and EM

The K-Means algorithm can be derived from as a limiting case of the EM algorithm for mixture of spherical Gaussians. Prove this. That is, start from the EM algorithm and find the appropriate settings for the E- and M- steps to reconstruct the K-Means algorithm. Justify. (Hint: Calculate the limit $\lim_{\sigma_j^2 \to 0} \gamma_{ij}$)

Question 4: Comparison of K-Means and EM Instructions: Answer each of the following questions with a few sentences.

- Compare the outputs of all your three algorithms. In your opinion, which one worked the best? Were any of the outputs (in your opinion) satisfactory? Which ones, and why or why not? What aspect of the K-Means or EM algorithm made these good traits discoverable, and what traits might have the algorithms overlooked?
- Note the time it took to run each algorithm. Were the quality of the outputs worth the amount of time needed to run the algorithm?

- In your opinion, were mixture models for Gaussian distributions good for modelling the data? Could you think of some other ways to model the data?

- What was your initialization strategy for K-Means and EM? Informally describe if you think your strategy was "successful."

Question 4. K-Means Wikipedia Documents Clustering (30 points)

In this problem set, we will be using K-means clustering on Wikipedia introductions of famous individuals who are associated with Princeton University. Through our analysis, we will determine broad categories that capture the different academic and cultural backgrounds of these individuals. This analysis is part of the field called "topic modeling".

Question 1 Load in the dataset `Wikipedia.RData`. Print out the head of the dataset. Who are the first 3 individuals in the dataset and their corresponding professions? (It's okay to give a broad category). Then use the script `script1_HW3.R` to convert the matrix into a document-term matrix. How many individuals are there? How many words are there? What are the top 10 most frequent words in the entire dataset? List the 0%, 25%, 50%, 75% and 100% quantiles of the word counts across the entire dataset. The phenomenon you will see can be attributed to "Zipf's Law", saying the distribution of word counts in any text data follows an exponential-like curve. In the later questions, we will like to handle these "over-represented" words.

Question 2 We notice that the top ten words are not very interesting with "the" and "and" being among the most frequent words. One way to combat this is to re-parameterize the document-term matrix using the TF-IDF (Term Frequency, Inverse Document Frequency) parameterization. The essential idea of TF-IDF is that if a word appears frequently in a document, it's important. However, if a word appears in many documents, it's not a unique identifier. In this way, common words like "the" and "for", which appear in many documents, will be scaled down. Run the script `script2_HW3.R` to implement TF-IDF parameterization. Based on this new parameterization, what are the 10 words with highest total weight across the entire dataset?

Question 3 Great! Hopefully we are starting to see meaningful topics appear, but the TF-IDF parameterization might not be highly interpretable as our `dtm.mat` matrix contain weights instead of word counts. We would still like to know which words are common in the dataset (for our downstream analysis) but we'll like to remove "common" words. Run the `script3_HW3.R`, which does the following two things: First, it removes words that appear too frequently across all the documents. For example, the word "Princeton" is in every document. Second, it removes the top 300 most frequent words according to Google.

After running the following script, how many words are left in our analysis? What are the top ten words in our remaining matrix `dtm.mat.raw`?

Question 4 We see that the most common words in `dtm.mat.raw` are now similar to the top-weighted words in `dtm.mat`. This is great for our future interpretability. Now, let's zoom in one particular person and make sure our data was processed correctly. Search for "Ben Bernanke" in our dataset. Print out the raw text for him according to `dat` (our original dataset). Find in this text the 10 top-weighted words according to `dtm.mat` and the 10 words with the highest word counts according to `dtm.mat.raw`.

Question 5 We will now implement K-Means analysis. One of the most important ingredients for K-Means is a good distance function. Typically, the Euclidean distance is used between two vectors, but in this question, we will assume that the cosine-distance is more appropriate (`https://en.wikipedia.org/wiki/Cosine_similarity`). We will work with the `dtm.mat` (the TF-IDF parameterized) matrix when we use K-Means (as it includes all the words), but we will use `dtm.mat.raw` to interpret our results (as it includes only the important words we want to see).

Renormalize all the columns of `dtm.mat` to have norm 1. Then using `norm.sim.ksc` in the `akmeans` package, set the seed to 10 (for reproducibility) and find 8 clusters according to `dtm.mat`. How big are each of resulting clusters? Based on the cluster assignments, print out the top 25 words in each cluster according to `dtm.mat.raw`. Also print out the quantiles (0%, 25%, 50%, 75%, 100%) of the word counts with non-zero word counts within each cluster to get a sense of how significant the top words are.

Q4. The Graphical Model (50 points)

To the lecture! And beyond!
In the lectures, we have learned two very power methods for high dimensional regression and classification: the Lasso (or ℓ_1-penalized least square regression) and sparse logistic regression (or ℓ_1-penalized logistic regression). In this problem, we will teach you how these two methods are related to a very powerful data analysis tool–"Probabilistic graphical models!" In a typical machine learning course, the topics on graphical models could cost half a semester to cover, but here we will use one question (though it's a big question) to illustrate to you this amazing technique! In particular, we are going to introduce the concept of undirected graphical model (also named Markov random field)!

Given d random variables $X_1, \ldots, X_d$, an undirected graphical model is a statistical model (thus, by definition it is a set of probability distributions) associated with an undirected graph (Each node of the graph corresponds to one of these variables). As will be explained in the later part of this question, this graph specifies the conditional dependence structures between these random variables. By applying the Lasso and logistic regression, we can exploit purely observational data to recover and visualize the conditional dependence structures of random variables.

Before digging into the formal definition and theory of the graphical model, let's first get a taste of how the graphical model can be used as a powerful tool for data analysis.

Part I. Nationwide GDP Growth Correlation (15 points)

Big Data! Big World!

Background:
In this modern era of globalization, we live in a world where the economic growth of any nation may depend on the prosperity of many other nations–with whom the nation share political ties, geographical dependence, or trade partnerships. These economic relationships are often complex and difficulty to summarize, but can be very useful information for global investors and policy makers. One powerful approach is to visualize these relationships using a graph that connects countries whose economic growths are correlated. We will do just that!
In particular, we use Gross Domestic Product (GDP) as the measure for economic growth of a country. The GDP is the monetary values of all the goods and services produced in a country during a year, and it is commonly used to access a country's economic growth and health. In this problem, we will use this GDP data to estimate a graphical model that represents the economic dependence between different countries.

Load `gdp.Rdata`, you will get a numerical matrix `gdp` where each row has GDP annual growth rates of certain country from 2001 to 2014. Our goal is to conduct a graph analytics of the GDP growth among countries using the graphical model tool. Install and load the packages `glmnet` and `igraph`. You can also find a function called `graphplot` in `q4.r` for visualizing the estimated graph.

Question 1 (5 points): Note that the matrix `gdp` has missing values. Directly delete countries that have no GDP rates reported. Then, for a given country, replace `NA`'s with the mean value of the reported growth rates across the years.

(Hint: The function `is.na` will be useful.)

Question 2 (10 points):

Given a Country A, we use the Lasso regression to find out the countries whose GDP growth rates are significantly associated (in the linear regression sense) with Country A. In particular, do the following analysis:

1. Create a $d \times d$ matrix of zeros, $\boldsymbol{M}$, where d is the number of different countries in the dataset. This will be your adjacency matrix.

   In an adjacency matrix, $\boldsymbol{M}_{jk} = 1$, if there is an edge between the j-th and k-th nodes of the graph. $\boldsymbol{M}_{jk} = 0$, otherwise. (The diagonal entries do not matter.)

   In this problem, the j-th and k-th nodes represent the j-th and k-th countries, respectively.

   You will populate this matrix in the next few steps. (Observe that $\boldsymbol{M}$ must be symmetric since we are working with undirected graphs.)

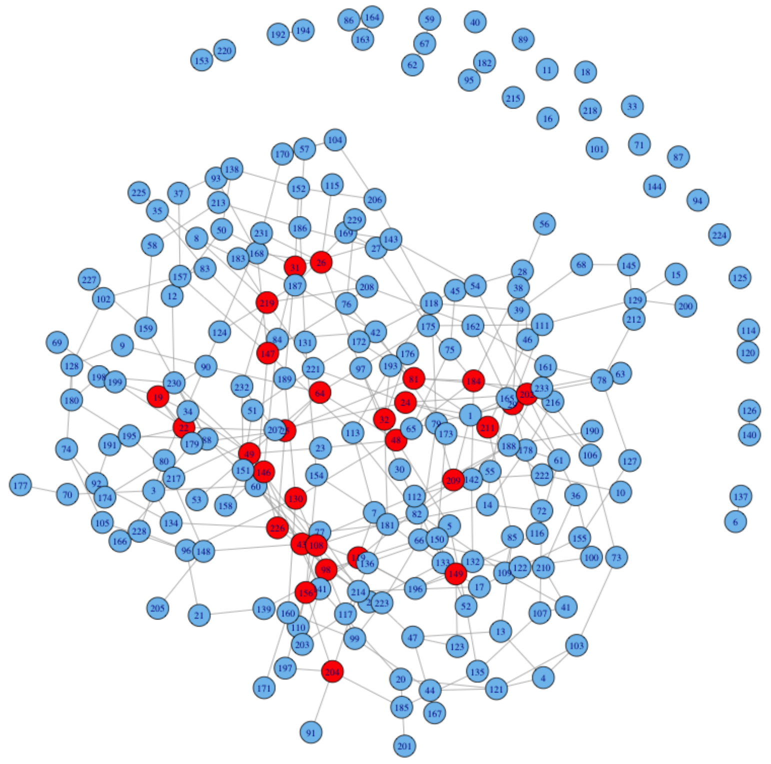
2. For each country, apply Lasso so that we can find the neighborhood of every node and thus recover the whole graph. In particular, call the function `glmnet` in `R` to regress the GDP growth rate of the country (call it Country A) over the GDP growth rates of all other countries with the tuning parameter $\lambda = 1$.

3. Country A's GDP growth is associated with Country B's growth if Lasso chooses a non-zero coefficient for the variable associated with Country B. Use this fact to fill in the correct entries in $\boldsymbol{M}$.

   (Note 1: It is possible that we select country B as the neighborhood of country A but do not select country A as the neighborhood as country B. In this case, we adopt the "**AND**" rule to address the contradiction here, meaning that country A and B are connected in the graph if and only if both A and B are in the neighborhood of each other.)

   (Note 2: As a result, the countries selected by Lasso in this approach are considered the neighborhood of country A in the GDP growth interaction graph. We will provide theoretical justifications later in Part III.)

4. Visualize the estimated graph by the function `graphplot` provided in `q4.r` (which takes an adjacency matrix as input). This function will print the graph in a file called `gdp_nodewise.png` with nodes colored red if they have degrees greater than or equal to five and colored blue if they have degrees less than five. Your graph should be similar to the following one.



5. List the country names that correspond to the red nodes in the graph.

Part II. Preliminary Theories (10 points)

In this part, we rigorously establish the theory of the graphical model. First, we provide a brief lesson on **independence** and **conditional independence** and their application to graphical models.

Time to Learn!

Independence and Conditional Independence:
Suppose we have d random variables $\{X_j\}_{j=1}^d$, and for an index set A we let $\boldsymbol{X}_A = \{X_j : j \in A\}$. For three index sets A, B and C such that $A \cap B = \emptyset$, $B \cap C = \emptyset$ and $A \cap C = \emptyset$, we say $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ are globally independent if

$$p_{\boldsymbol{X}_A, \boldsymbol{X}_B}(\mathbf{x}_A, \mathbf{x}_B) = p_{\boldsymbol{X}_A}(\mathbf{x}_A) p_{\boldsymbol{X}_B}(\mathbf{x}_B)$$

for arbitrary realizations $\mathbf{x}_A, \mathbf{x}_B \in \mathbb{R}^d$. And we say $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ are conditionally independent given $\boldsymbol{X}_C$ if

$$p_{\boldsymbol{X}_A, \boldsymbol{X}_B \mid \boldsymbol{X}_C}(\mathbf{x}_A, \mathbf{x}_B \mid \mathbf{x}_c) = p_{\boldsymbol{X}_A \mid \boldsymbol{X}_C}(\mathbf{x}_A \mid \mathbf{x}_C) p_{\boldsymbol{X}_B \mid \boldsymbol{X}_C}(\mathbf{x}_B \mid \mathbf{x}_C)$$

for arbitrary realizations $\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C \in \mathbb{R}^d$, where $p(\cdot)$ could be a probability density or mass function depending on whether the random variable is continuous or discrete.

Application to Graphical Models:
The key idea of the graphical model is to represent conditional independence relationship by a graph. More specifically, we define a graph $G = (V, E)$. Here $V = \{1, \dots, d\}$ is the node set corresponding the random variables $X_1, \dots, X_d$. $E \subset V \times V$ is the edge set. For each pair of nodes $j, k \in V$, an edge (j, k) is not in the edge set E if and only if the corresponding variables X_j and X_k are conditionally independent given the rest of the variables $\boldsymbol{X}_{\backslash j}$ (where $\backslash j := \{\ell : \ell \neq j, k\}$). More rigorously we say

$$\forall j, k \in V, (j, k) \notin E \quad \text{if and only if} \quad X_j \perp X_k \mid X_{\backslash\{j,k\}}. \tag{0.12}$$

Therefore, this graph actually visualizes the the conditional dependence relationship between any pair of variables.

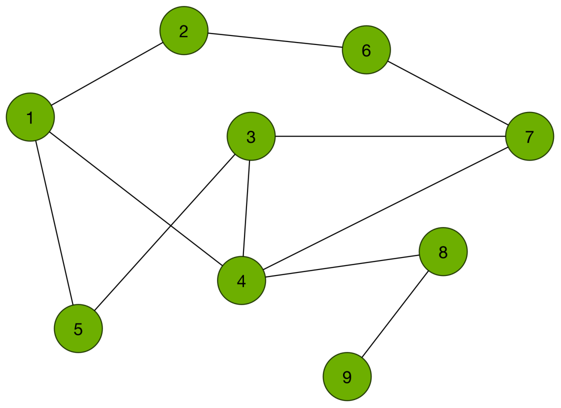
Separation:
In the graph theory, there is an important concept called "separation": Let $A, B, C \subset V$ such that every two of them have no intersection. We say a node set C separates A and B if any path connecting the nodes in A and B must contain at least one node in C (In another word, the removal of C renders A and B disconnected).
Even though the graph G is defined only based on pairwise conditional independence (i.e., each time, each edge (j, k) only corresponds to the conditional independence relationship between two variables X_j and X_k). However, an amazing result shows that it actually encodes a more powerful relationship! More specifically, under very weak assumptions (which always holds in our class), for the graph G we develop in (0.12), it holds that if C separates A and B, then $\boldsymbol{X}_A \perp \boldsymbol{X}_B \mid \boldsymbol{X}_C$. So here we establish the correspondence between separation and conditional independence.

Now, we apply this knowledge in some exercises.

Question 3 (5 points): Construct concrete examples to show that global independence cannot imply conditional independence and conditional independence cannot imply global independence either.

(Hint 1: For the example of global independence not leading to conditional independence, consider the independent coin tosses. Construct some event conditional on which the coin tosses are not independent any more. For the example of conditional independence not leading to global independence, consider two responses in the linear regression model that share the same design covariate.)

Question 4 (5 points): Consider the graphical model as illustrated below. Denote the random variable corresponding to node i by X_i.



Answer the following questions:

- Does the node set $\{3\}$ separate node set $\{5\}$ and node set $\{7\}$?
- Does the node set $\{8\}$ separate node set $\{7\}$ and node set $\{9\}$?
- Is it true that $X_1 \perp X_8 \mid \boldsymbol{X}_{\setminus\{1,8\}}$?
- Is it true that $X_1 \perp X_8 \mid X_4$?
- Is it true that $X_1 \perp X_3 \mid X_5$?

Part III. The Gaussian Graphical Model (15 points)

Having introduced the general concept of undirected graphical models, we now focus on a specialized case: the Gaussian graphical model. We aim to provide theoretical justification on why we can use node-wise Lasso in Part I to estimate the graph.

Time to Learn!
Gaussian Graphical Models
Suppose $\boldsymbol{X} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{X}$ is a d-dimensional random vector and $\boldsymbol{\Sigma}$ is a d-by-d covariance matrix. It turns out that the precision matrix $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$ encodes the information of the conditional dependence structure of $\boldsymbol{X}$, as indicated by the following theorem.

Theorem 0.1. *Suppose $\boldsymbol{X} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{X}$ is a d-dimensional random vector and $\boldsymbol{\Sigma}$ is a d-by-d matrix. Then for any different $j, k \in \{1, ..., d\}$, we have*

$$X_j \perp X_k \mid X_{\setminus\{j,k\}} \Longleftrightarrow \boldsymbol{\Theta}_{jk} = 0$$

where $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix, $\perp$ means independence, and $\mid X_{\setminus\{j,k\}}$ means conditional on all the variables expect X_j and X_k. Therefore estimating the edge set E is equivalent to estimating the support of $\boldsymbol{\Theta}$.

To prove the the theorem above, we use two steps that correspond to two small questions below. Without loss of generality, we can always assume that $j = 1$ and $k = 2$. We will use the following result directly without proof. For those who are interested in deriving the result, please see the blog posts. For any $A \subseteq \{1, ..., d\}$, we have

$$\boldsymbol{X}_A \mid \boldsymbol{X}_{A^c} \sim N(\boldsymbol{\Sigma}_{AA^c}\boldsymbol{\Sigma}_{A^cA^c}^{-1}\boldsymbol{X}_{A^c}, \boldsymbol{\Sigma}_{AA} - \boldsymbol{\Sigma}_{AA^c}\boldsymbol{\Sigma}_{A^cA^c}^{-1}\boldsymbol{\Sigma}_{A^cA}), \tag{0.13}$$

where A^c is the complement set of A.

Question 5 (5 points):

Suppose $A = \{1, 2\}$. We partition the covariance matrix $\boldsymbol{\Sigma}$ and $\boldsymbol{\Theta}$ in the following pattern:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{AA} & \boldsymbol{\Sigma}_{AA^c} \\ \boldsymbol{\Sigma}_{A^cA} & \boldsymbol{\Sigma}_{A^cA^c} \end{bmatrix}, and\ \boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Theta}_{AA} & \boldsymbol{\Theta}_{AA^c} \\ \boldsymbol{\Theta}_{A^cA} & \boldsymbol{\Theta}_{A^cA^c} \end{bmatrix}.$$

Show that $\boldsymbol{\Theta}_{AA}^{-1} = \boldsymbol{\Sigma}_{AA} - \boldsymbol{\Sigma}_{AA^c}\boldsymbol{\Sigma}_{A^cA^c}^{-1}\boldsymbol{\Sigma}_{A^cA}$. Combined with the results (0.13), we know that $\boldsymbol{X}_A \mid \boldsymbol{X}_{A^c} \sim N(\boldsymbol{\Sigma}_{AA^c}\boldsymbol{\Sigma}_{A^cA^c}^{-1}\boldsymbol{X}_{A^c}, \boldsymbol{\Theta}_{AA}^{-1})$.

(Hint: Start from the fact that $\boldsymbol{\Theta}\boldsymbol{\Sigma} = \mathbf{I}$)

Question 6 (5 points): Use the result derived from Question 3.5 to reach the final conclusion.

(Hint: Two Gaussian variables are independent to each other if and only if their covariance is zero.)

Now let's discuss how to recover the support of $\boldsymbol{\Theta}$ under the Gaussian graphical model, or equivalently speaking, the edge set E in the graph G. In Part I, we run nodewise Lasso to find significant association between covariates. This is actually called the method of *neighborhood pursuit*, which means that we estimate $\boldsymbol{\Theta}$ by investigating the neighborhood of each node in the graph. Here we justify this method theoretically under the Gaussian graphical model. For X_j, define its neighbor $\mathcal{N}(X_j) := \{k : \Theta_{jk} \neq 0\}$. Without loss of generality, we look at the neighborhood of X_1, and we denote the other variables by $\boldsymbol{X}_{\setminus 1}$ for notational convenience. By (0.13), we can express X_1 as

$$X_1 = \underbrace{\boldsymbol{\Sigma}_{1,\setminus 1}\boldsymbol{\Sigma}_{\setminus 1,\setminus 1}^{-1}}_{\boldsymbol{\beta}^T}\boldsymbol{X}_{\setminus 1} + \epsilon, \tag{0.14}$$

where $\boldsymbol{\beta} = \boldsymbol{\Sigma}_{\setminus 1,\setminus 1}^{-1}\boldsymbol{\Sigma}_{\setminus 1,1}$ and $\mathrm{Var}(\epsilon) = \Sigma_{11} - \boldsymbol{\Sigma}_{1,\setminus 1}\boldsymbol{\Sigma}_{\setminus 1,\setminus 1}^{-1}\boldsymbol{\Sigma}_{\setminus 1,1}$.

Question 7 (5 points): Show that $\epsilon \perp \boldsymbol{X}_{\setminus 1}$ and $\boldsymbol{\beta} = \Theta_{11}^{-1}\boldsymbol{\Theta}_{\setminus 1,1}$, which implies that the support of $\boldsymbol{\beta}$ is the same as $\boldsymbol{\Theta}_{\setminus 1,1}$. In this way, recovering the support of $\boldsymbol{\beta}$ is equivalent to recovering the support of $\boldsymbol{\Theta}_{\setminus 1,1}$, i.e., the neighborhood of X_1 on the graph. This question justifies the method of neighborhood pursuit.

Part IV. The Ising model (10 points)

In this part, we introduce another kind of probabilistic graphical model called the Ising graphical model, which is very important in statistical physics. Under the Ising model, $X_1, ..., X_d \in \{-1, 1\}$ and for $\{\beta_j\}_{j=1}^d \in \mathbb{R}$ and $\{\beta_{jk}\}_{j \neq k} \in \mathbb{R}$ such that $\beta_{jk} = \beta_{kj}$, we have

$$P(X_1 = x_1, X_2 = x_2, ..., X_d = x_d) = \frac{1}{Z}\exp(\sum_{j=1}^d \beta_j x_j + \sum_{j<k} \beta_{jk} x_j x_k), \tag{0.15}$$

where $Z = \sum_{x_1, x_2, ..., x_d \in \{-1,1\}} \exp(\sum_{j=1}^d \beta_j x_j + \sum_{j<k} \beta_{jk} x_j x_k)$. Similarly, we will show that under this Ising model, $\{\beta_{jk}\}_{j,k=1}^d$ capture the conditional dependence structure of $\{X_j\}_{j=1}^d$ and that the neighborhood pursuit method is still effective in recovering the support of $\{\beta_{jk}\}_{j,k=1}^d$.

Question 8 (5 points):

Analogous to Theorem 0.1, show that $X_j \perp X_k \mid X_{\setminus\{j,k\}} \iff \beta_{jk} = 0$ for different j and k.

(Hint: There will be many exponential quantities in your equations. Denoting them by short letters like A_1 and A_2 will very much simplify the equations you need to verify.)

Question 9 (5 points):

Show that

$$P(X_j = 1 | \boldsymbol{X}_{\setminus j} = \mathbf{x}_{\setminus j}) = \frac{1}{1 + \exp(-2(\beta_j + \sum_{k \neq j} \beta_{jk} x_k))},$$

which is a logistic regression model if we treat X_j as the class label and $\boldsymbol{X}_{\setminus j}$ as the covariates.

(Therefore from the perspective of the neighborhood pursuit, we can run logistic regression of X_j over $\boldsymbol{X}_{\setminus j}$ to estimate $\{\beta_{jk}\}_{k=1}^d$. Then by the result from Question 8, we can recover the correspondent graphical model.)

Q5. Develop Your Own AlphaGo (20 points)

We all know the news that AlphaGo developed by Google DeepMind defeated the human being champion Lee Sedol. This is a huge improvement in the development of artificial intelligence because computers were thought to be incapable of winning professional human GO players before.

In this problem, we will develop our own AlphaGo to play the Go game. Consider the Go board as an undirected Graph $G = (V, E)$ with $V = \{1, \dots, 19\} \times \{1, \dots, 19\}$ and $N = |V| = 19 \times 19$. A node $v \in V$ represents a vertice on the board and an edge $e \in E$ connects vertically and horizontally neighboring points.

Now we introduce how to use a vector $\boldsymbol{c}$ to encode a GO board. Suppose we have a simple 3×3 board:

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{0.16}$$

where 1 represents a Black stone, -1 represents a White stone and 0 represents an empty spot. Then $\boldsymbol{c}$ is just the vectorized version of the matrix $\boldsymbol{C}$ by columns. In this example, $\boldsymbol{c} = (1, 1, 0, 0, -1, 0, 1, 0, 1)^T$. This is how we use the vecotor $\boldsymbol{c}$ to encode the board $\boldsymbol{C}$.

Next we define the neighborhood of a node on the board. Any node on the board has four neighboring nodes at most: the left, right, upper and lower neighbors. Nodes on the boundaries of the board have fewer neighbors. For example, let's look at $\boldsymbol{C}$ defined in (0.16). $\boldsymbol{C}[2, 2]$, the center node, has four neighbors: $\boldsymbol{C}[1, 2], \boldsymbol{C}[2, 1], \boldsymbol{C}[2, 3]$ and $\boldsymbol{C}[3, 2]$. However, $\boldsymbol{C}[1, 1]$ has only a right neighbor $\boldsymbol{C}[1, 2]$ and a lower neighbor $\boldsymbol{C}[2, 1]$, and $\boldsymbol{C}[2, 1]$ has only three neighbors, i.e., $\boldsymbol{C}[1, 1], \boldsymbol{C}[2, 2]$ and $\boldsymbol{C}[3, 1]$.

The distribution we wish to model is $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{s} \mid \boldsymbol{c})$, that is, the distribution over $\boldsymbol{s}$, the final territory outcome, given $\boldsymbol{c}$, the board at the 80-th step of the game. Here $\boldsymbol{c}$ and $\boldsymbol{s}$ are both $19 \times 19-$dimensional vectors representing the Go board. In order to predict the final territory outcome $\boldsymbol{s}$, we consider the following Ising graphical model:

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{s} \mid \boldsymbol{c}) = \frac{1}{Z(\boldsymbol{c}, \boldsymbol{\theta})} \exp\left(\sum_{(j,k) \in \mathcal{F}} w(c_j, c_k) s_j s_k + h(c_j) s_j + h(c_k) s_k \right), \tag{0.17}$$

where $\mathcal{F}$ is the set of all the edges that connect the neighboring nodes, $w(\cdot, \cdot)$ and $h(\cdot)$ are coefficient functions and $\boldsymbol{\theta}$ denotes all the unknown parameters that will be specified later. We call the term $w(c_j, c_k) s_j s_k$ the **coupling** term and $h(c_j) s_j + h(c_k) s_k$ the **external field** term. We assume the coefficients $w(c_j, c_k), h(c_j)$ are symmetric with respect to Black and White, so that we only have five unknown parameters in $\boldsymbol{\theta}$ for this model, i.e.,

- w_{chains} = w(Black, Black) = w(White, White),
- $w_{\text{inter-chain}}$ = w(Black, White) = w(White, Black),
- $w_{\text{chain-empty}}$ = w(Empty, White) = w(Empty, Black),
- w_{empty} = w(Empty, Empty),
- h_{stones} = h(Black) = $-h$(White),

and h(Empty) is set to zero by symmetry.

From a large number of Go games, we have found maximum likelihood estimators for $\boldsymbol{\theta}$, i.e.,

$$\widehat{w}_{\text{chains}} = 2.74,\ \widehat{w}_{\text{inter-chain}} = 0.251,\ \widehat{w}_{\text{chain-empty}} = 0.442,\ \widehat{w}_{\text{empty}} = 0.427,\ \widehat{h}_{\text{stones}} = 0.265.$$

Let's predict the final outcomes of the AlphaGo-vs-Lee Sedol games (game 2 and game 4) using the game boards at the 80-th step. In the script file `predictGo.R`, fill in the line 68, and complete the function `construction.ising.graph`, which is for calculating the matrix w in (0.17). Now you can predict the winner of the Go game for any possible Go board position!