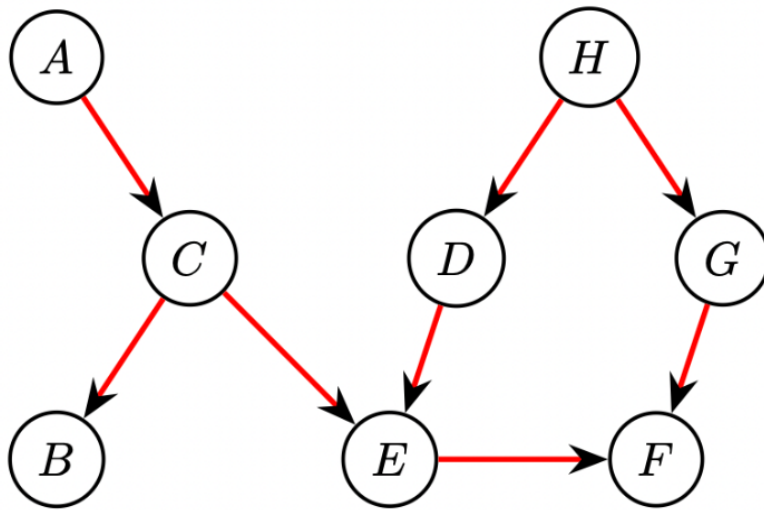


1. For the given Bayesian Network, which of the statements are correct.



A. B and E may be dependent

B. Given C, B is independent of A

C. B is independent of A

D. None of the above

A. True – not conditional and sum prob B/E must eq 1 for C

B. True – markov assumption

C. False – not conditional, and B descendent of A

D. False

2. In a hypothetical "language" consisting of 3 distinct words, someone has written a 100-word essay with each word associated with one of 5 possible moods: Angry, Happy, Sad, Excited, and Neutral. Using a Hidden Markov Model, what would be the dimension of the observation probability matrix, which shows the probability of each mood emitting each word in the vocabulary?

- A. 5 x 3
- B. 3 x 5
- C. 100 x 5
- D. 5 x 100

angry
happy
sad
excited
neutral
5 hidden states

word1
word2
word3
3 emission states

observation prob matrix =
5(hidden states) x 3(emission states)
since $Obs(i,j) = p(o_i | s_j)$

Solution:

Answer: A. 5 x 3

The size of the observation probability matrix in this case would be 5x3, since there are 5 possible hidden states (moods) and each state can emit one of 3 possible words from the vocabulary.

3. In a hypothetical "language" consisting of 3 distinct words, someone has written a 100-word essay with each word associated with one of 5 possible moods: Angry, Happy, Sad, Excited, and Neutral. Using a Hidden Markov Model, What is the size of the state transition probability matrix in our HMM model?

- A. 5 x 3
- B. 3 x 5
- C. 5 x 5
- D. 5 x 100

state transition matrix =
5(hidden states at time t) x 5(hidden states at time t+1)

4. In a hypothetical "language" consisting of 3 distinct words, someone has written a 100-word essay with each word associated with one of 5 possible moods: Angry, Happy, Sad, Excited, and Neutral. Using a Hidden Markov Model, What is the length of path of hidden state

- A. 5
- B. 3
- C. 50
- D. 100

length of path of hidden state = 100



5. Which of the following statements about k-means clustering is true?

- A. K-means is a supervised learning algorithm.
- B. K-means is guaranteed to find the globally optimal clustering solution.
- C. K-means requires the number of clusters to be specified in advance.
- D. K-means is not sensitive to the initial choice of cluster centers.

Solution:

Answer: C. K-means requires the number of clusters to be specified in advance.

K-means is an unsupervised learning algorithm, meaning that it does not rely on any labeled data to find patterns in the data.

K-means is an iterative algorithm that can converge to a local minimum, which may not be the globally optimal solution.

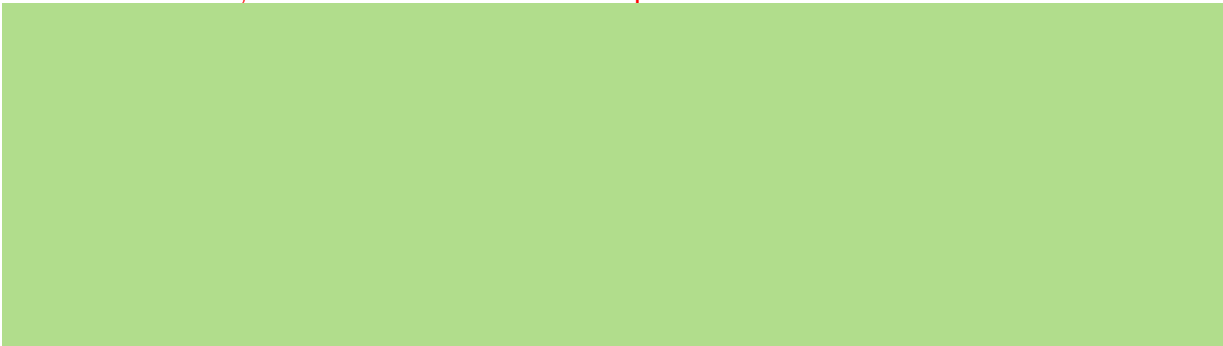
K-means is sensitive to the initial choice of cluster centers. If the initial centers are chosen poorly, the algorithm may converge to a suboptimal solution.

6. Which of the following is a goal of unsupervised learning?

- A. To learn a function that maps input to output.
- B. To find patterns and structure in the data.
- C. To classify new data into predefined categories.
- D. To minimize the prediction error between the predicted and actual outputs.

Solution:

Answer: B. The goal of unsupervised learning is to find patterns and structure in the data, without the use of labeled output data.



7. For a K means algorithm with N distinct points, if there are K clusters, then the loss function J is given by:

$$J = \sum_{i=1}^k \sum_{x \in D_i} ||x - \mu_i||^2$$

Which of the following statements are wrong about J.

- A. TRUE since N clusters can have centroids at each N distinct points
- A. The value of loss function J is 0, when number of clusters is N
- B. The least possible value for J can be $-\infty$ (Negative infinity)
- B. FALSE since it is squared error (distance) so least possible value is 0
- C. Both options are wrong.

8. In the context of clustering, which of the following statements is true about hard class membership and soft class membership?:

- ☒ A. Hard class membership assigns each data point to a single cluster, while soft class membership assigns each data point a probability of belonging to each cluster.
- ☐ B. Hard class membership assigns each data point a probability of belonging to each cluster, while soft class membership assigns each data point to a single cluster.
- ☐ C. Hard class membership assigns each data point to multiple clusters, while soft class membership assigns each data point to a single cluster.
- ☐ D. None of the above

Solution:

9. MinCut, RatioCut, NCut

Solution:

Mincut, RatioCut, and NCut are three commonly used clustering algorithms that operate on graph representations of data.

Mincut: In graph theory, mincut is a way to find a cut in a graph that minimizes the number of edges between the two partitions. In other words, it is a partition of the graph into two disjoint subsets that minimizes the number of edges connecting the two subsets. Mincut can be used for clustering by treating the graph as a set of points and the edges between them as similarities.

RatioCut: RatioCut is similar to mincut, but it is a more balanced clustering algorithm. It works by minimizing the ratio of the number of edges between the two partitions to the total number of edges in the graph. This ensures that the two partitions are of roughly equal size, rather than one partition being much smaller than the other.

NCut: NCut is a clustering algorithm that takes into account the normalized cut size of a partition. Normalized cut size is a measure of how well the graph has been partitioned, taking into account both the number of edges between the partitions and the size of the partitions. NCut is designed to produce balanced clusters and has been shown to work well on a variety of data sets.

10. Few points on Back Propagation -

Solution:

It is a supervised learning algorithm: The backpropagation algorithm requires a labeled dataset to train the neural network. The inputs and outputs of the network must be known in advance for each training example.

It is an iterative algorithm: The backpropagation algorithm requires multiple passes through the training dataset. During each pass, the weights and biases of the network are adjusted to reduce the error between the predicted outputs and the true labels.

It uses gradient descent optimization: The backpropagation algorithm adjusts the weights and biases of the network using the gradient of the error function with respect to the weights and biases.

It involves two phases: The forward phase, where the input is propagated through the network to produce an output, and the backward phase, where the error is propagated back

through the network to update the weights and biases.

It requires a differentiable activation function: The backpropagation algorithm requires the activation function of the neurons to be differentiable, so that the gradients can be calculated and propagated back through the network.

It can suffer from the vanishing gradient problem: In deep neural networks, the gradients can become very small as they are propagated back through the network, leading to slow convergence or no convergence at all. Techniques such as weight initialization, batch normalization, and residual connections can help mitigate this problem.

11. When can we achieve 100% training accuracy for Multi-layer perceptron ?

Solution:

In general, it is not always possible to achieve 100% training accuracy for a Multi-Layer Perceptron (MLP) on all datasets, particularly when dealing with complex, real-world problems.

However, for simple and linearly separable datasets, it may be possible to achieve 100% training accuracy with an MLP. This is because the decision boundary between classes is a straight line or a hyperplane, which can be learned by a single-layer MLP. This is because the decision boundary between classes is a straight line or a hyperplane, which can be learned by a single-layer MLP.

12. Gaussian Mixture Model -

$$p(x) = \sum_{k=1}^K \pi_k N(x|\mu_k, \Sigma_k)$$

Solution:

Here, $p(x)$ is the probability density function of the GMM, x is the input data, K is the number of mixture components, π_k is the prior probability of the k -th component, $N(x|\mu_k, \Sigma_k)$ is the Gaussian probability density function of the k -th component with mean μ_k and covariance matrix Σ_k .

In other words, GMM models the input data as a mixture of K Gaussian distributions, where each component is weighted by its prior probability π_k . The model is estimated by optimizing the parameters π_k , μ_k , and Σ_k using the Expectation-Maximization (EM) algorithm. Once the model is trained, it can be used to compute the probability density function of new input data.

The prior of the k th component, denoted by π_k , represents the prior probability that a data

point belongs to the k th component of the mixture. It is a scalar value between 0 and 1 and satisfies the constraint that the sum of all priors equals 1.

13. Principal Component Analysis -

Solution:

PCA (Principal Component Analysis) is a linear dimensionality reduction technique used to find a low-dimensional representation of high-dimensional data. It is commonly used to reduce the number of variables or features of a dataset and to better understand the underlying structure of the data. PCA works by identifying the directions in which the data varies the most and projecting the data onto those directions.

In PCA, the first principal component is the direction in which the data varies the most. The second principal component is the direction that is orthogonal to the first principal component and captures the next highest amount of variance in the data. Subsequent principal components are defined in a similar way. The principal components are sorted in order of the amount of variance they capture in the data.

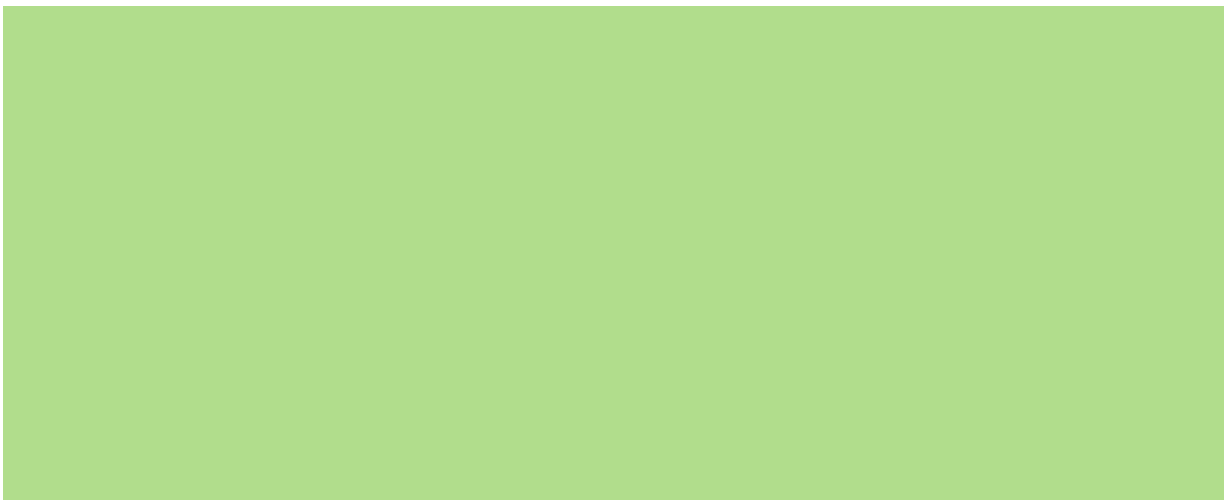
PCA can be used for a variety of purposes, including data compression, feature extraction, and data visualization. It is widely used in machine learning, statistics, and other fields.

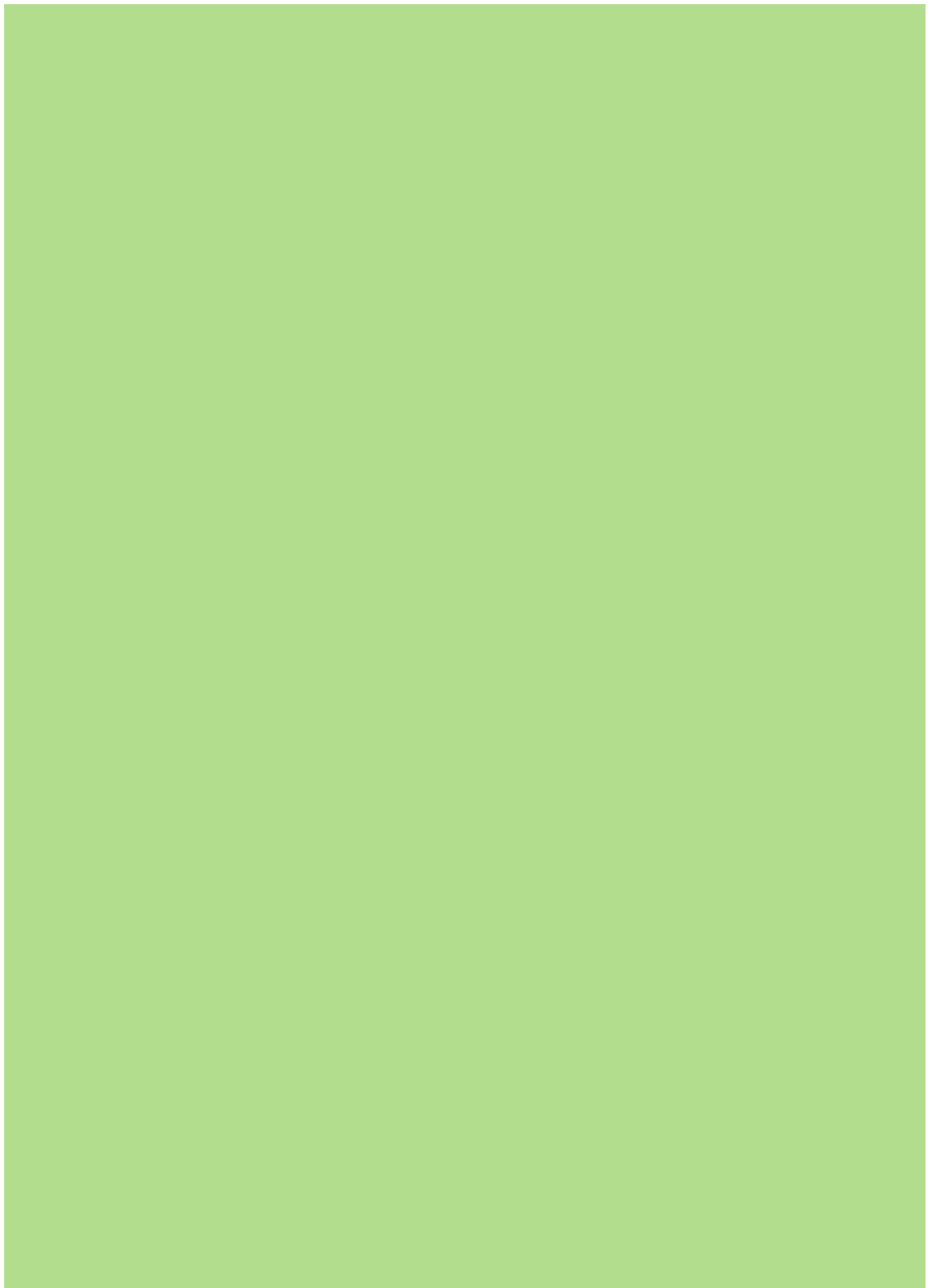
14. Suppose we have the following data points in 2-d space $(0, 0)$, $(-1, 2)$, $(-3, 6)$, $(1, -2)$, $(3, -6)$.

1. Draw them on a 2-d plot.
2. What is the first principle component?
3. What is the second principle component?

(Question from Exam 2 Practice questions)

Solution:









15. CNN:

Solution:

A convolutional neural network (CNN) is a type of neural network that is commonly used for image and video processing applications. In a CNN, the input image is passed through a series of convolutional layers, each of which applies a set of filters to the input to extract certain features. These filters are learned during training and are used to identify patterns in the image that are important for the given task. The output of the convolutional layers is then passed through one or more fully connected layers, which perform the final classification.

The number of kernels in a convolutional layer defines the number of filters that are applied to the input image in that layer. Each kernel extracts a specific feature from the input image, and the output of the layer is a set of feature maps, one for each kernel. The number of kernels in a convolutional layer is a hyperparameter that is typically determined through trial and error, and a larger number of kernels can allow the network to learn more complex features in the input image. However, increasing the number of kernels also increases the computational cost and can lead to overfitting if not properly regularized.

16. GAN

Solution:

Generative Adversarial Networks (GANs) are a type of deep learning model that involve two networks: a generator and a discriminator. The goal of a GAN is to generate new data that is similar to a set of training data. The generator network takes a random input and produces a sample that is intended to be similar to the training data, while the discriminator network tries to distinguish between the generated samples and the real training data.

During training, the generator and discriminator are trained together in a minimax game. The generator tries to produce samples that the discriminator cannot distinguish from the real training data, while the discriminator tries to correctly identify the generated samples as fake. As the training progresses, the generator learns to produce more realistic samples, while the discriminator becomes better at identifying fake samples. The training process continues until the generator produces samples that are difficult to distinguish from real data.

GANs have become popular for generating images, videos, and other types of data. They have been used for applications such as image synthesis, style transfer, and video prediction.

Question 1:

Given a graph with 6 nodes (i.e., data points) in Figure where the weight for each edge/link is 1. We want to run MinCut algorithm to find two clusters.



Figure 1 : The Input Graph

1. Write down the adjacency matrix W of this graph.
2. Write down the graph Laplacian matrix L of the adjacency matrix of this graph.
3. Suppose we have the following partition/clustering result: $\{1, 2, 3\}$ as partition-1 and $\{4, 5, 6\}$ as partition-2. What is the corresponding clustering membership vector q ? What is the cutsize for this partition result?
4. Now, suppose we have the following partition/clustering result: $\{1\}$ as partition-1 and $\{2, 3, 4, 5, 6\}$ as partition-2. What is the corresponding clustering membership vector q ? What is the cutsize for this partition result?

Question 2:

Suppose we have the following data points in 2-d space $(0, 0)$, $(-1, 2)$, $(-3, 6)$, $(1, -2)$, $(3, -6)$.

1. Draw them on a 2-d plot.
2. What is the first principle component?
3. What is the second principle component?

Question 3:

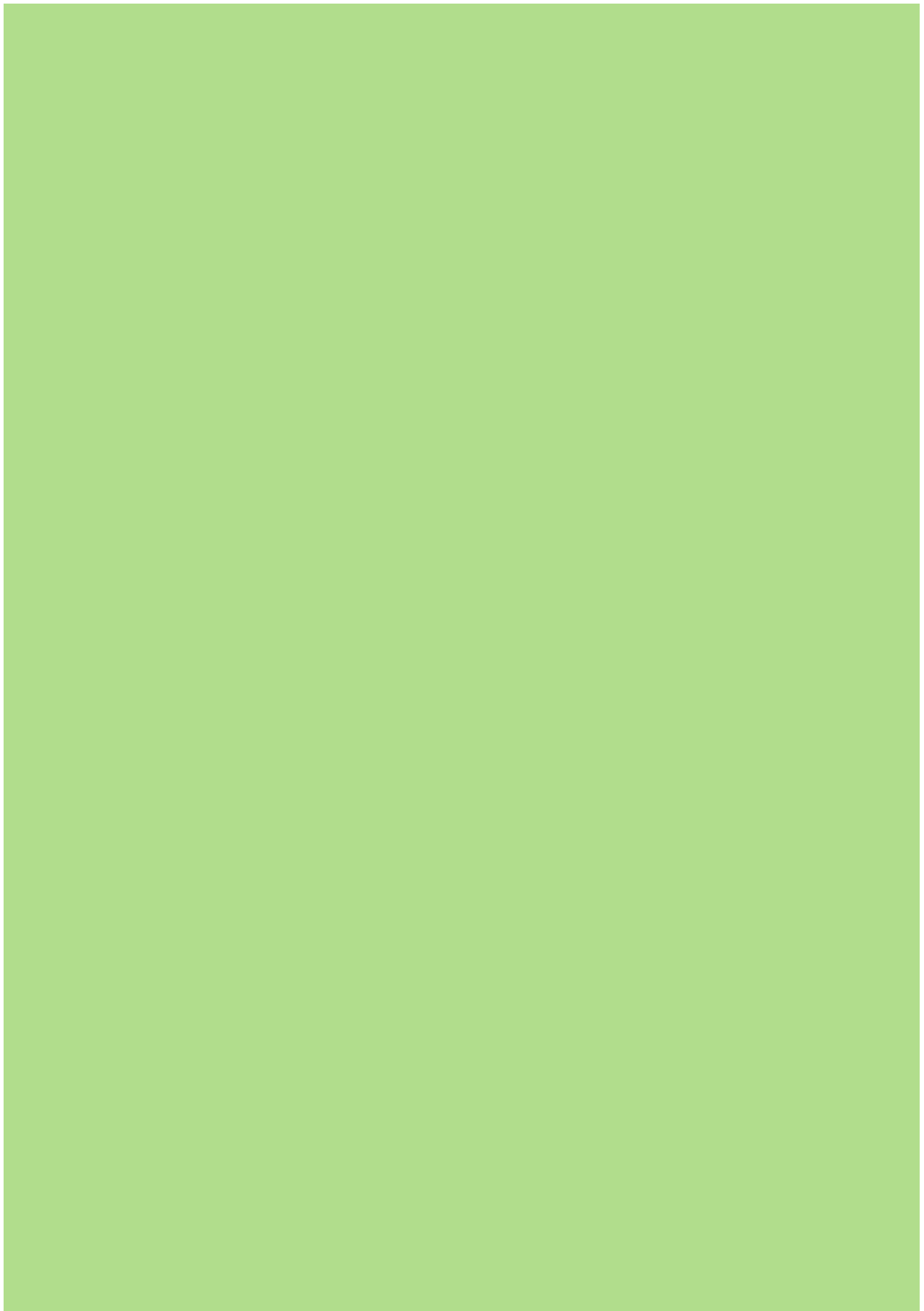
Suppose that we use four distinct words to write a paragraph with 100 segments, and we treat each word in the paragraph as a segment. We want to infer three possible class labels of all the segments in this paragraph, including (a) location (b) person name and (c) background by HMM (Hidden Markov Models).

1. What is the size of the state transition probability matrix in our HMM model?
2. What is the size of the state-observation probability matrix?
3. In a particular trial, how many observations do you see? What is the length of the path of states?
4. Suppose that the first state is about 'background', how many different possible state paths are there in total?

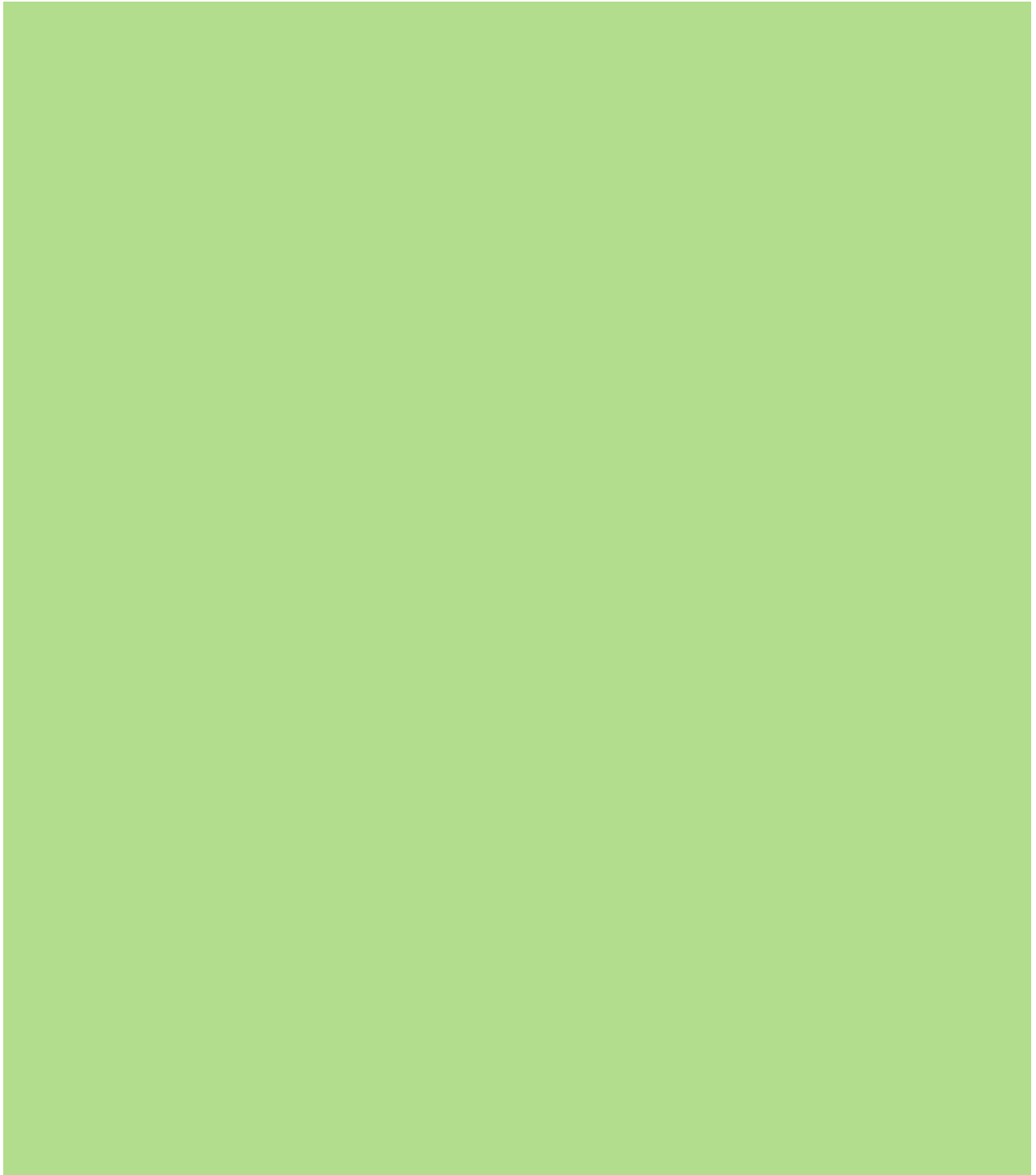
Question 4:

Given N data points x_i ($i = 1, \dots, N$), Kmeans will group them into K clusters by minimizing the loss/distortion function $J = \sum_{n=1}^N \sum_{k=1}^K r_{n,k} \|x_n - \mu_k\|^2$, where μ_k is the center of the k^{th} cluster; and $r_{n,k} = 1$ if x_n belongs to the k^{th} cluster and $r_{n,k} = 0$ otherwise. In this question, we will use the following iterative procedure.

- Initialize the cluster center μ_k ($k = 1, \dots, K$);
 - Iterate until convergence
 - Step 1: Update the cluster assignments $r_{n,k}$ for each data point x_n .
 - Step 2: Update the center μ_k for each cluster k .
1. Given 8 data points in 1-d space: $x_1 = -3$, $x_2 = -1$, $x_3 = 0.5$, $x_4 = 2$, $x_5 = 3$, $x_6 = 4$, $x_7 = 7$ and $x_8 = -5$. Plot these eight data points in 1-d space.
 2. Suppose the initial cluster centers are $\mu_1 = -1$ and $\mu_2 = 3$ (i.e., $K = 2$). If we only run Kmeans for one iteration on the above data set, what is the cluster assignment for each data point after Step 1 ? What are the updated cluster centers after Step 2 ?
 3. Suppose the initial cluster centers are $\mu_1 = -4$, $\mu_2 = 0$ and $\mu_3 = 3$ on the above data set (i.e., $K = 3$). If we only run Kmeans for one iteration, what is the cluster assignment for each data point after Step 1? What are the updated cluster centers after Step 2 ? What is the loss function J after this iteration ?
 4. If we run Kmeans on the above dataset to find eight clusters (i.e., $K = 8$). What is the optimal cluster assignment for each data point? What is the corresponding loss function J for the optimal clustering assignment?

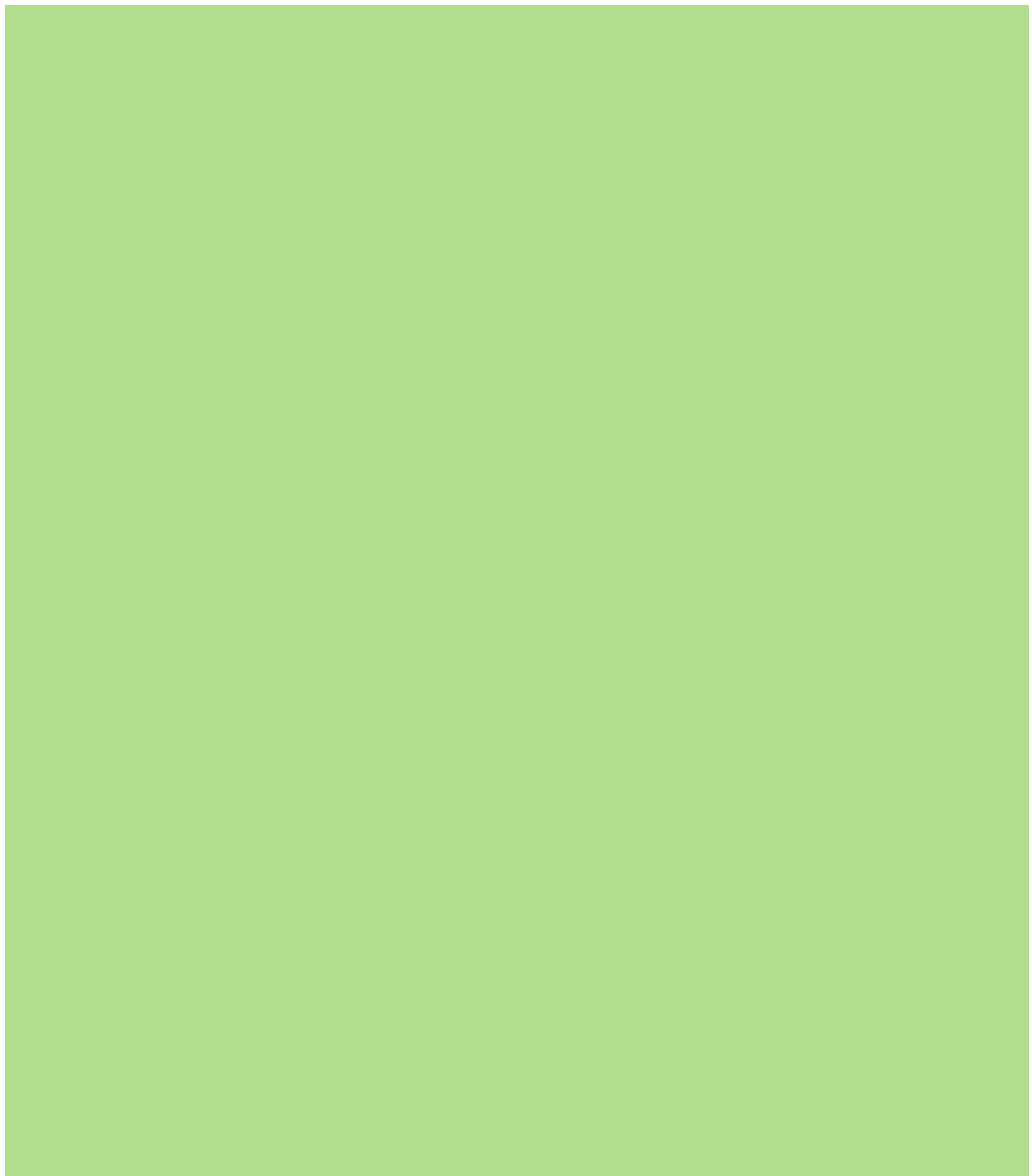


Question 2: SOLUTION



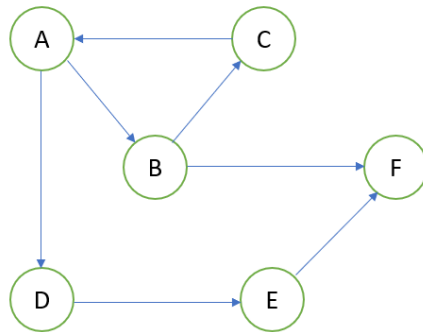
Question 3: SOLUTION



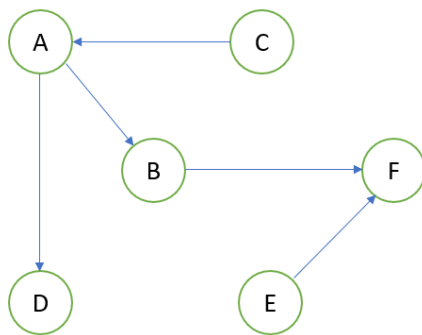


Question 1

- 1) Is the following graph a valid Bayesian Network? Why or why not?



- 2) Using the following graph of a Bayesian Network, which of the statements below can be inferred? Mark as either true or false.



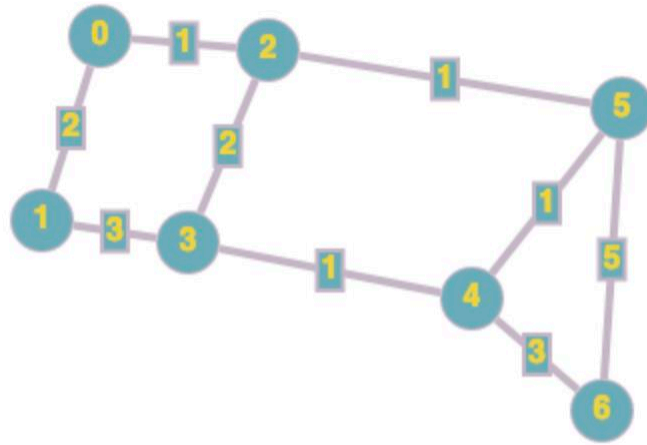
- a. A is independent of C _____
- b. D is independent of C _____
- c. B and D may be dependent _____
- d. Given A , B is independent of C _____
- e. Given F , B is independent of E _____

Question 2

Consider a set of 2D points $\{(-3,-3), (-1,-1), (1,1), (3,3)\}$. We want to reduce the dimensionality of these points (to 1D) by using PCA. Perform the required computations to answer the following questions:

- 1) What is the covariance matrix of the data?
- 2) What is the largest eigenvalue and corresponding eigenvector?
- 3) What is the data after its dimensionality has been reduced to 1D?

Question 3



Given the graph above, the nodes 0, 1, 2 & 3 belong to cluster *A* and the nodes 4, 5 & 6 belong to cluster *B*. The line separating these clusters is cutting the edge connecting vertices 2,5 and 3,4. Compute the following properties of the graph:

1. $\text{Cut}(A,B)$
2. $\text{Cut}(A,A)$
3. $\text{Cut}(B,B)$
4. $\text{Vol}(A)$
5. $\text{Vol}(B)$
6. $|A|$
7. $|B|$
8. $\text{MinMaxCut}(A,B)$
9. $\text{NormalizedCut}(A,B)$

Question 4

- 1) Apply the average-pooling operation on the input below using a 2x2 kernel and:
 - a. Stride = 1x1
 - b. Stride = 2x2

1	3	3	6
2	1	2	7
6	6	8	2
8	7	5	1

- 2) Perform the max-pooling operation on the input matrix using:
Kernel = 3x3
Stride = 1x1
Padding = 1

2	4	6	7	8	9
7	8	6	2	1	7
2	5	8	1	3	0
1	3	4	7	0	1
7	8	5	5	1	7
6	5	3	1	7	1

Question 4 SOLUTION

- 1) Using a 2x2 Kernel with our 4x4 input matrix:
- a. With Stride = 1x1, we shift our 2x2 kernel one step at a time in either the horizontal or vertical direction. This means that the resulting output matrix will be of size 3x3. So, starting at the top-left:

1	3	3	6
2	1	2	7
6	6	8	2
8	7	5	1

1.75		

The kernel covers the squares that are shaded green in the image on the left above. The average of these values is taken:

$$\mu = \frac{1+3+2+1}{4} = \frac{7}{4} = 1.75$$

And placed in the output matrix (shown in green in the image on the right above) in its top-left position. Now, the kernel shift to the right by one step:

1	3	3	6
2	1	2	7
6	6	8	2
8	7	5	1

1.75	2.25	

Again, the average value under the kernel (top-left image) is calculated and placed into the output matrix (top-right image) at the next spot. The kernel again shifts to the right by one step:

1	3	3	6
2	1	2	7
6	6	8	2
8	7	5	1

1.75	2.25	4.5

Again, the average value under the kernel (top-left image) is calculated and placed into the output matrix (top-right image) at the next spot. At this point, the kernel cannot shift to the right anymore so it needs to return to the start of this row and then shift down by one step:

1	3	3	6
2	1	2	7
6	6	8	2
8	7	5	1

1.75	2.25	4.5
3.75		

Again, the average value under the kernel (top-left image) is calculated and placed into the output matrix (top-right image) in the corresponding spot. This process continues until the entire input matrix has been processed and the corresponding output matrix has been populated. The final output matrix is given below:

1.75	2.25	4.5
3.75	4.25	4.75
6.75	6.5	4

- b. Using a stride of 2x2, the process is the same as in part (a), except that we need to shift two steps in the horizontal or vertical direction when moving the kernel. The result is that the output matrix will be of size 2x2. Illustrating the process by starting from the top-left:

1	3	3	6
2	1	2	7
6	6	8	2
8	7	5	1

1.75	

Stepping to the right two steps puts the kernel in the top-right of the input matrix:

1	3	3	6
2	1	2	7
6	6	8	2
8	7	5	1

1.75	4.5

Now that the first row is complete, we need to return to the start of the row, shift down two steps, and we get:

1	3	3	6
2	1	2	7
6	6	8	2
8	7	5	1

1.75	4.5
6.75	

And finally, shifting to the right two steps puts us at the bottom-right of the input matrix and calculating the average and placing it in the output matrix gives us the final output matrix of:

1.75	4.5
6.75	4

- 2) For the max-pooling operation, we take the values under the kernel and select the single largest value and place it into the corresponding spot in the output matrix. The original input matrix is 6x6, however we are instructed to add a single layer of padding around it which yields:

0	0	0	0	0	0	0	0
0	2	4	6	7	8	9	0
0	7	8	6	2	1	7	0
0	2	5	8	1	3	0	0
0	1	3	4	7	0	1	0
0	7	8	5	5	1	7	0
0	6	5	3	1	7	1	0
0	0	0	0	0	0	0	0

In the image above, the padding layer has been highlighted with a gray-colored fill and, when performing the max-pooling operation, we will include these new rows and columns in our calculations. Therefore, we can now state that the size of our input matrix has grown to 8x8. And, with a 3x3 kernel and a stride of 1x1, this will result in an out matrix of size 6x6.

So, for example, when we start our max-pooling operation at the top-left of the input matrix, we will include the top and left layers of padding when placing our kernel and get:

0	0	0	0	0	0	0	0
0	2	4	6	7	8	9	0
0	7	8	6	2	1	7	0
0	2	5	8	1	3	0	0
0	1	3	4	7	0	1	0
0	7	8	5	5	1	7	0
0	6	5	3	1	7	1	0
0	0	0	0	0	0	0	0

8					

The left-most image above represents the input matrix and the right-most image represents the output matrix. Based on the values under the kernel in the input matrix, we select '8' because it has the largest value. Then, we place '8' in the corresponding position in the output matrix and perform the shift. Considering the shifting was shown in detail in Part 1, we will not go over it again here.

After completing the max-pooling operation on the entire input matrix, the resulting output matrix is as follows:

8	8	8	8	9	9
8	8	8	8	9	9
8	8	8	8	7	7
8	8	8	8	7	7
8	8	8	7	7	7
8	8	8	7	7	7