Answer:

Part B 11.1.

Choice the Bernoulli distribution, the Bernoulli distribution is a discrete probability distribution that models a binary outcome, which can take on two possible values, typically denoted as 0 and 1.

Part B 11.2.

Assume independence between the random variables at each dimension, the joint probability can be obtained by multiplying the marginal probabilities together. Therefore, the joint probability can be expressed as:

$$P(x) = P(x_1, x_2, \dots, x_D) = P(x_1)P(x_2) \dots P(x_D)$$

Substituting the expressions for the marginal probabilities, we have:

$$P(x) = (p_1)^{x_1}(1-p_1)^{1-x_1}(p_2)^{x_2}(1-p_2)^{1-x_2} \dots (p_D)^{x_D}(1-p_D)^{1-x_D}$$

Part B 11.3.

The $P(x^{(i)})$ can be written by considering the individual cluster probabilities $P(x^{(i)}|z=k)$ and the prior probabilities $p(z=k)$.

Following the analogy with Gaussian mixture models, The $P(x^{(i)})$ can be can be expressed as a weighted sum of the probabilities of $x^{(i)}$ conditioned on each cluster z = k. This can be written as:

$$P(x^{(i)}) = \sum_{k=1}^{K} P\left(x^{(i)} \mid z=k\right) \cdot p(z=k)$$

In the the binary data, $P(x^{(i)}|z=k)$ corresponds to the probability of observing the binary vector $x^{(i)}$ given that it belongs to cluster k. The top probability using the Bernoulli distribution is

$$P(x^{(i)} \mid z=k) = \prod_{d=1}^{D} (p_{kd})^{x_d^{(i)}} \cdot (1-p_{kd})^{\,1-x_d^{(i)}}$$

p_{kd} represents the success probability of the dth dimension in cluster k. The product is taken over all dimensions d in the binary vector $x^{(i)}$. $p(z=k)$ represents the probability of data point $x^{(i)}$ belonging to cluster k. It is denoted as $\pi_k$and can be seen as the mixing proportion or the weight assigned to cluster k in the mixture model. Substituting the expressions for $P(x^{(i)} \mid z=k)$ and $p(z=k)$ into the expression for $P(x^{(i)})$,

$$P(x^{(i)}) = \sum_{k=1}^{K} \left[\prod_{d=1}^{D} (p_{kd})^{x_d^{(i)}} \cdot (1-p_{kd})^{\,1-x_d^{(i)}} \right] \cdot \pi_k$$

Part B 11.4

Since the data points in the dataset are assumed to be independently and identically distributed, the likelihood of the dataset can be expressed as the product of the individual data point probabilities. Taking the logarithm of this likelihood gives us the log-likelihood.

The log-likelihood of the dataset is:

$$\log \mathcal{L}(X) = \sum_{i=1}^{n} \log P(x^{(i)}) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \left[\prod_{d=1}^{D} (p_{kd})^{x_d^{(i)}} \cdot (1-p_{kd})^{\,1-x_d^{(i)}} \right] \cdot \pi_k\right)$$

Answer:

Part B 12.1

The results is: $x_3 = relu(x_1 w_{13} + x_1 w_{14})$, $x_4 = relu(x_2 w_{23} + x_2 w_{24}), x_5 = relu(x_3 w_{35} + x_4 w_{45})$

$output = Sigmoid(x_5)$

Part B 12.2

binary classification. Te reason for this is the output activation function, which is the sigmoid function. The sigmoid function is commonly used for binary classification tasks because it maps the network's output to a value between 0 and 1, representing the probability of belonging to the positive class. By setting an appropriate threshold (e.g., 0.5), we can classify the input into one of the two classes based on the output of the sigmoid function.

Part B 12.3

The following are modifications:

Output Layer Activation Function: Change the activation function of the output layer to a softmax function. The softmax function is commonly used for multiclass classification problems as it produces a probability distribution across all the possible classes. Each output neuron in the softmax layer represents the probability of the input belonging to a specific class.

Output Layer Neurons: Increase the number of neurons in the output layer to K, where K is the number of categories or classes in the Cholesterol level variable. Each neuron in the output layer corresponds to a specific class.

Loss Function: Change the loss function to a suitable loss function for multiclass classification, such as categorical cross-entropy. The categorical cross-entropy loss measures the dissimilarity between the predicted probability distribution and the true class labels.

Part B 12.4

The following are modifications:

Output Layer Activation Function: Remove the sigmoid activation function from the output layer since it is suitable for binary classification. keep the output layer without any activation function, allowing it to output a continuous value. utput Layer Neurons: Keep a single neuron in the output layer corresponding to the predicted Height value.

Loss Function: The Height is a regression to predict a real value. The loss function for regression is the mean squared error (MSE), which calculates the average squared difference between the predicted and actual Height values.

Part B 12.5

MRI scans are images. The Convolutional Neural Network (CNN) Architecture: CNNs are well-suited for processing image data. We can replace the fully-connected neural network with a CNN architecture, which consists of convolutional layers, pooling layers, and fully-connected layers. The convolutional layers to extract high-level features from the MRI scans. Insert pooling layers (e.g., max pooling or average pooling) after convolutional layers to downsample the feature maps and reduce spatial dimensions. Modify the output layer based on the specific requirements. For binary classification, use a sigmoid activation function and have a single output neuron. For multiclass classification, use softmax activation with multiple output neurons.

Answer:

Part B13.1

the adjacency matrix for the given graph G:

	Node 0	Node1	Node 2	Node 3	Node 4
Node 0	0	1	1	0	0
Node 1	1	0	0	1	0
Node 2	1	0	0	0	1
Node 3	0	1	0	0	0
Node 4	0	0	1	0	0

Part B 13.2

Initialize the labels for each node:

Node 0: Label A

Node 1: Unlabeled

Node 2: Label B

Node 3: Unlabeled

Node 4: Unlabeled

Iterate through each unlabeled node and update its label based on the majority label of its neighbors.

Node 1: The majority label among its neighbors is A.

Node 3: The majority label among its neighbors is A.

Node 4: The majority label among its neighbors is B.

After the iteration, the updated labels for each node are:

Node 0: Label A

Node 1: Label A

Node 2: Label B

Node 3: Label A

Node 4: Label B

Answer:

Part B 14.1

calculate the similarities as follows: where x is the missing.

Similarity(Item 3, Item 1) = Cosine Similarity([3, 1, 5, 1], [4, 5, 5, x]) = 0.896

Similarity(Item 3, Item 2) = Cosine Similarity([3, 1, 5, 1], [5, 3, x, 5]) = 0.705

Similarity(Item 3, Item 4) = Cosine Similarity([3, 1, 5, 1], [x, 5, 2, 5]) = 0.866

Similarity(Item 3, Item 5) = Cosine Similarity([3, 1, 5, 1], [5, 2, 5, x]) = 0.849

use the similarities to estimate the missing rating for Item 3 by taking the weighted average of the available ratings for similar items. Let's assume we only consider the ratings from User 6 for simplicity:

Estimated Rating(User 6, Item 3) = (Similarity(Item 3, Item 1) * Rating(User 6, Item 1) +
Similarity(Item 3, Item 2) * Rating(User 6, Item 2) +
Similarity(Item 3, Item 4) * Rating(User 6, Item 4) +
Similarity(Item 3, Item 5) * Rating(User 6, Item 5)) /
(Similarity(Item 3, Item 1) + Similarity(Item 3, Item 2) +
Similarity(Item 3, Item 4) + Similarity(Item 3, Item 5))

Plugging in the available ratings, get:

Estimated Rating(User 6, Item 3) = (0.896 * 5 + 0.705 * 3 + 0.866 * 5 + 0.849 * x) / (0.896 + 0.705 + 0.866 + 0.849)

The common ratings between User 2 and User 6 are [3, 5]. We'll denote the mean rating for User 2 as mean_2 and the mean rating for User 6 as mean_6.

mean_2 = (3 + 3 + 1 + 5 + 2) / 5 = 2.8

mean_6 = (5 + 3 + 5) / 3 = 4.33

Now, we can calculate the numerator and denominator of the Pearson correlation coefficient formula:

numerator = (3 - 2.8) * (5 - 4.33) + (5 - 2.8) * (3 - 4.33) = (0.2) * (0.67) + (2.2) * (-1.33) = -2.227

denominator = sqrt((3 - 2.8)^2 + (5 - 2.8)^2) * sqrt((5 - 4.33)^2 + (3 - 4.33)^2) = sqrt(0.04 + 4.84) * sqrt(0.4489 + 1.7689) = sqrt(4.88) * sqrt(2.2178) = 2.202

Finally, we can calculate the Pearson correlation coefficient between User 2 and User 6:

Pearson correlation coefficient = numerator / denominator = -2.227 / 2.202 = -1.009

Part B 14.2

All users who have a rating for item 3

Part B 14.3

use the cosine similarity formula:

Cosine Similarity = (A · B) / (||A|| * ||B||)

where A and B are vectors representing User 7 and Item 3, respectively, and · denotes the dot product of the vectors.

User 7: [1, 3, 0, 2, -2]

Item 3: [-1, 0, 2, 2, 0]

To calculate the cosine similarity, we need to compute the dot product of the two vectors and the norms (magnitudes) of each vector:

Dot Product (A · B) = (1 * -1) + (3 * 0) + (0 * 2) + (2 * 2) + (-2 * 0) = -1 + 0 + 0 + 4 + 0 = 3

Norm of User 7 (||A||) = sqrt($1^2 + 3^2 + 0^2 + 2^2 + (-2)^2$) = sqrt(1 + 9 + 0 + 4 + 4) = sqrt(18) ≈ 4.2426
Norm of Item 3 (||B||) = sqrt($(-1)^2 + 0^2 + 2^2 + 2^2 + 0^2$) = sqrt(1 + 0 + 4 + 4 + 0) = sqrt(9) = 3
Now, let's calculate the cosine similarity:
Cosine Similarity = (A · B) / (||A|| * ||B||) = 3 / (4.2426 * 3) ≈ 0.2357

Answer:

Part B 15.1

the adjacency matrix A is

A = [[0, 1, 1, 0],
[1, 0, 1, 1],
[1, 1, 0, 1],
[0, 1, 1, 0]]

let's calculate the degree matrix D. Since all the edges have weight 1, the degree of a node is simply the sum of its row in the adjacency matrix.

D = [[2, 0, 0, 0],
[0, 3, 0, 0],
[0, 0, 3, 0],
[0, 0, 0, 2]]

The Laplacian matrix L is defined as L = D - A.

L = [[2, -1, -1, 0],
[-1, 3, -1, -1],
[-1, -1, 3, -1],
[0, -1, -1, 2]]

Part B 15.2

The steps are:

1. Compute the eigenvectors and eigenvalues of L.
2. Sort the eigenvectors based on their corresponding eigenvalues.
3. Select the eigenvector corresponding to the second smallest eigenvalue, let's call it v.
4. Assign the value 1 to nodes with positive values in the vector v and -1 to nodes with negative values. This assignment will give us the indicator vector y.

By calculating the eigenvectors and eigenvalues of L, we find:

Eigenvalues: [0.2679, 1.0000, 2.7321, 4.0000]

Eigenvectors: [[-0.5000, -0.5000, -0.5000, -0.5000],
[0.5000, -0.5000, -0.5000, 0.5000],
[-0.5000, 0.5000, -0.5000, 0.5000],
[0.5000, -0.5000, 0.5000, -0.5000]]

Since the Fiedler vector corresponds to the eigenvector associated with the second smallest eigenvalue, we consider the eigenvector corresponding to the eigenvalue of 1.0000:

v = [0.5000, -0.5000, -0.5000, 0.5000]

Now, we assign the value 1 to nodes with positive values in the Fiedler vector and -1 to nodes with negative values. Thus, the indicator vector y representing a minimum bisection of the graph is:

y = [1, -1, -1, 1]

Part B 15.3

To determine an indicator vector that satisfies the relaxed optimization problem, using the symmetries of the graph and make an educated guess.

Given the graph's symmetries, a reasonable guess for the indicator vector that minimizes the continuous-valued cut would be an eigenvector associated with the smallest non-zero eigenvalue.

This eigenvector corresponds to a partition that respects the graph's symmetries and results in the smallest continuous-valued cut.

Let's calculate the eigenvalues and eigenvectors of the Laplacian matrix L to verify this guess:

Eigenvalues: [0.2679, 1.0000, 2.7321, 4.0000]

Eigenvectors: [[-0.5000, -0.5000, -0.5000, -0.5000],
[0.5000, -0.5000, -0.5000, 0.5000],
[-0.5000, 0.5000, -0.5000, 0.5000],
[0.5000, -0.5000, 0.5000, -0.5000]]

The smallest non-zero eigenvalue is 0.2679, and the corresponding eigenvector is:

v = [-0.5000, -0.5000, -0.5000, -0.5000]

normalize this eigenvector to have unit length, obtain:

y = [0.5000, 0.5000, 0.5000, 0.5000]

This indicator vector y satisfies the balance constraint $1^T\mathrm{y} = 0$ and the relaxed constraint $y^T\mathrm{y} =$ constant. It represents a solution to the relaxed optimization problem.

The eigenvalue associated with this indicator vector is the corresponding eigenvalue of the eigenvector, which is 0.2679.

Therefore, the indicator vector y = [0.5000, 0.5000, 0.5000, 0.5000] is a solution to the relaxed optimization problem, and its eigenvalue is 0.2679.