MATHEMATICAL INTELLIGENCE SYSTEM

GROUP-ASSIGNMENT -1

CONTRIBUTED BY:-

BATCH-A	TEAM-7
GAJULA SRI VATSANKA	CB.EN.U4AIE.21010
GUNNAM HIMAMSH	CB.EN.U4AIE.21014
M.PRASANNA TEJA	CB.EN.U4AIE.21035
VIKHYAT BANSAL	CB.EN.U4AIE.21076

LINEAR ALGEBRA

Dr. NAVANEETH HARIDHARSHAN

1

Write a code in a programming language to do the $PA = LU$ decomposition of a $m \times n$ matrix A.

VIKHYAT BANSAL

LU Decomposition

- In numerical analysis and linear algebra, **lower – upper (LU) decomposition** or **factorization** as it factors a matrix as the product of a lower triangular matrix and an upper triangular matrix. The product sometimes includes a permutation matrix(P) as well. LU decomposition can be viewed as the matrix form of Gaussian elimination. Computers usually solve square systems of linear equations using LU decomposition, and it is also a key step when inverting a matrix or computing the determinant of a matrix.

Let A be a square matrix. An **LU factorization** refers to the factorization of *A*, with proper row and/or column orderings or permutations, into two factors – a lower triangular matrix *L* and an upper triangular matrix *U*

A = LU

In the lower triangular matrix all elements above the diagonal are zero, in the upper triangular matrix, all the elements below the diagonal are zero. For example, for a 3 × 3 matrix *A*, its LU decomposition looks like this:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}.$$

Without a proper ordering or permutations in the matrix, the factorization may fail to materialize. For example, it is easy to verify (by expanding the matrix multiplication) that $a_{11} = \ell_{11} u_{11}$.

If $a_{11} = 0$, then atleast one of l_{11} and u_{11} has to be zero, which implies that either L or U is singular. This is impossible if A is an invertible matrix. This is a common problem and can be removed by simply reordering the rows of A so that the first element of the matrix is non-zero.

LU factorization with partial pivoting

- It turns out that a proper permutation in rows (or columns) is sufficient for LU factorization. **LU factorization with partial pivoting** (LUP) refers often to LU factorization with row permutations only.

$$PA = LU$$

where L and U are again lower and upper triangular matrices, and P is a permutation matrix, which, when left-multiplied to A, reorders the rows of A. It turns out that all square matrices can be factorized in this form, and the factorization is numerically stable in practice. This makes LUP decomposition a useful technique in practice.

Solving of NxN matrix for LU Decomposition

There are several algorithms for calculating L and U. To derive *Crout's algorithm* for a 3x3 example, we have to solve the following system:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = LU$$

We now would have to solve 9 equations with 12 unknowns. To make the system uniquely solvable, usually the diagonal elements of L are set to 1

$$\begin{aligned} l_{11} &= 1 \\ l_{22} &= 1 \\ l_{33} &= 1 \end{aligned}$$

so we get a solvable system of 9 unknowns and 9 equations.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{11}l_{21} & u_{12}l_{21} + u_{22} & u_{13}l_{21} + u_{23} \\ u_{11}l_{31} & u_{12}l_{31} + u_{22}l_{32} & u_{13}l_{31} + u_{23}l_{32} + u_{33} \end{pmatrix} = LU$$

Solving for the other l and u, we get the following equations:

$$\begin{aligned} u_{11} &= a_{11} \\ u_{12} &= a_{12} \\ u_{13} &= a_{13} \end{aligned}$$

$$\begin{aligned} u_{22} &= a_{22} - u_{12}l_{21} \\ u_{23} &= a_{23} - u_{13}l_{21} \end{aligned}$$

$$u_{33} = a_{33} - (u_{13}l_{31} + u_{23}l_{32})$$

We see that there is a calculation pattern, which can be expressed as the following formulas, first for U

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} u_{kj} l_{ik}$$

and then for L

$$l_{ij} = \frac{1}{u_{jj}} (a_{ij} - \sum_{k=1}^{j-1} u_{kj} l_{ik})$$

We see in the second formula that to get the l_{ij} below the diagonal, we have to divide by the diagonal element (pivot) u_{jj}, so we get problems when u_{jj} is either 0 or very small, which leads to numerical instability.

The solution to this problem is *pivoting* A, which means rearranging the rows of A, prior to the LU decomposition, in a way that the largest element of each column gets onto the diagonal of A. Rearranging the rows means to multiply A by a permutation matrix P:

$$PA \Rightarrow A'$$

Example:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 3 \end{pmatrix} \Rightarrow \begin{pmatrix} 2 & 3 \\ 1 & 4 \end{pmatrix}$$

The decomposition algorithm is then applied on the rearranged matrix so that

$$PA = LU$$

Solving of MxN matrix for LU Decomposition

Understanding LU decomposition of 3x4 Matrix will give us a overview of how LU decomposition of rectangular matrix. The method works just as well for other sizes since the LU-decomposition arises naturally from the study of Gaussian elimination via multiplication by elementary matrices.

$$A = \begin{bmatrix} 1 & 2 & -3 & 1 \\ 2 & 4 & 0 & 7 \\ -1 & 3 & 2 & 0 \end{bmatrix} \xrightarrow{r_2 - 2r_1 \to r_2} \begin{bmatrix} 1 & 2 & -3 & 1 \\ 0 & 0 & 6 & 5 \\ -1 & 3 & 2 & 0 \end{bmatrix} \xrightarrow{r_3 + r_1 \to r_3}$$

$$\begin{bmatrix} 1 & 2 & -3 & 1 \\ 0 & 0 & 6 & 5 \\ 0 & 5 & -1 & 1 \end{bmatrix} \xrightarrow{r_2 \leftrightarrow r_3} \begin{bmatrix} 1 & 2 & -3 & 1 \\ 0 & 5 & -1 & 1 \\ 0 & 0 & 6 & 5 \end{bmatrix} = U$$

We have $U = E_3 E_2 E_1 A$ hence $A = E_1^{-1} E_2^{-1} E_3^{-1} U$ and we can calculate the product $E_1^{-1} E_2^{-1} E_3^{-1}$ as follows:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{r_2 \leftrightarrow r_3} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \xrightarrow{r_3 - r_1 \to r_3}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix} \xrightarrow{r_2 + 2r_1 \to r_2} \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix} = PL$$

A"P“ is inserted in front of the L since the matrix above is not lower triangular. However, if we go one step further and let $r_2 \leftrightarrow r_3$ then we will obtain a lower triangular matrix:

$$PL = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix} \xrightarrow{r_2 \leftrightarrow r_3} \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} = L$$

Therefore, we find that $E_1^{-1}E_2^{-1}E_3^{-1} = PL$ where L is as above and $P = E_{2\leftrightarrow 3}$. This means that A has a modified LU-decomposition. Some mathemticians call it a PLU-decomposition,

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 1 & 2 & -3 & 1 \\ 0 & 5 & -1 & 1 \\ 0 & 0 & 6 & 5 \end{bmatrix}}_{U} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix}}_{PL} \underbrace{\begin{bmatrix} 1 & 2 & -3 & 1 \\ 0 & 5 & -1 & 1 \\ 0 & 0 & 6 & 5 \end{bmatrix}}_{U}.$$

Since permutation matrices all satisfy the condition $P^k = I$ (for some k) the existence of a PLU-decomposition for A naturally suggests that $P^{k-1}A = LU$. Therefore, even when a LU decomposition is not available we can just flip a few rows to find a LU-decomposable matrix. This is a useful observation because it means that the slick algorithms developed for LU-decompositions apply to all matrices with just a little extra fine print.

Much of the writing above can be spared if we adopt the notational scheme illustrated below.

$$A = \begin{bmatrix} 1 & 2 & -3 & 1 \\ 2 & 4 & 0 & 7 \\ -1 & 3 & 2 & 0 \end{bmatrix} \xrightarrow{r_2 - 2r_1 \to r_2} \begin{bmatrix} 1 & 2 & -3 & 1 \\ (2) & 0 & 6 & 5 \\ -1 & 3 & 2 & 0 \end{bmatrix} \xrightarrow{r_3 + r_1 \to r_3}$$

$$\begin{bmatrix} 1 & 2 & -3 & 1 \\ (2) & 0 & 6 & 5 \\ (-1) & 5 & -1 & 1 \end{bmatrix} \xrightarrow{r_2 \leftrightarrow r_3} \begin{bmatrix} 1 & 2 & -3 & 1 \\ (-1) & 5 & -1 & 1 \\ (2) & 0 & 6 & 5 \end{bmatrix} = U$$

We find if we remove the parenthetical entries from U and ajoing them to I then it gives back the matrix L we found previously:

$$U = \begin{bmatrix} 1 & 2 & -3 & 1 \\ 0 & 5 & -1 & 1 \\ 0 & 0 & 6 & 5 \end{bmatrix} \qquad L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix}.$$

Java Code for LU decomposition Matrices

MIS.java

MIS.java > MIS > main(String[])

```java
import static java.util.Arrays.stream;
import java.util.Locale;
import static java.util.stream.IntStream.range;

public class MIS {

    static double dotProduct(double[] a, double[] b) {
        return range(0, a.length).mapToDouble(i -> a[i] * b[i]).sum();
    }

    static double[][] matrixMul(double[][] A, double[][] B) {
        double[][] result = new double[A.length][B[0].length];      //initialization of 2-D input array
        double[] aux = new double[B.length];

        for (int j = 0; j < B[0].length; j++) {

            for (int k = 0; k < B.length; k++)
                aux[k] = B[k][j];

            for (int i = 0; i < A.length; i++)
                result[i][j] = dotProduct(A[i], aux);
        }
        return result;
    }

    static double[][] pivotize(double[][] m) {                     // To calculate pivot matrix P
        int n = m.length;
        double[][] id = range(0, n).mapToObj(j -> range(0, n)
                .mapToDouble(i -> i == j ? 1 : 0).toArray())
                .toArray(double[][]::new);
```

```java
static double[][] pivotize(double[][] m) {                    // To calculate pivot matrix P
    int n = m.length;
    double[][] id = range(0, n).mapToObj(j -> range(0, n)
            .mapToDouble(i -> i == j ? 1 : 0).toArray())
            .toArray(double[][]::new);

    for (int i = 0; i < n; i++) {
        double maxm = m[i][i];
        int row = i;
        for (int j = i; j < n; j++)
            if (m[j][i] > maxm) {
                maxm = m[j][i];
                row = j;
            }

        if (i != row) {
            double[] tmp = id[i];
            id[i] = id[row];
            id[row] = tmp;
        }
    }
    return id;
}
```

MIS.java

MIS.java > MIS > main(String[])

```java
    static double[][][] lu(double[][] A) {                          // Matrix multiplication which gives LU
        int n = A.length;
        double[][] L = new double[n][n];                            // Initializing 2-D matrix for output in form of L,U and P
        double[][] U = new double[n][n];
        double[][] P = pivotize(A);
        double[][] A2 = matrixMul(P, A);

        for (int j = 0; j < n; j++) {
            L[j][j] = 1;
            for (int i = 0; i < j + 1; i++) {
                double s1 = 0;
                for (int k = 0; k < i; k++)
                    s1 += U[k][j] * L[i][k];
                U[i][j] = A2[i][j] - s1;
            }
            for (int i = j; i < n; i++) {
                double s2 = 0;
                for (int k = 0; k < j; k++)
                    s2 += U[k][j] * L[i][k];
                L[i][j] = (A2[i][j] - s2) / U[j][j];
            }
        }
        return new double[][][]{L, U, P};
    }

    static void print(double[][] m) {
        stream(m).forEach(a -> {
            stream(a).forEach(n -> System.out.printf(Locale.US, "%5.1f ", n));
            System.out.println();
        });
        System.out.println();
    }
```

Run | Debug

```java
public static void main(String[] args) {
    double[][] a = {{1.0, 3, 5}, {2.0, 4, 7}, {1.0, 1, 0}};

    double[][] b = {{6.0, 9, 24,1}, {1.0, 5, 2,7}, {3.0, 17, 18,9},
    {9.0, 5, 7,5}};

    for (double[][] m : lu(a))
        print(m);

    System.out.println();

    for (double[][] m : lu(b))
        print(m);
}
}
```

Output for LU decomposition Matrices

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[Running] cd "d:\JAVA\" && javac MIS.java && java MIS
 1.0   0.0   0.0
 0.5   1.0   0.0
 0.5  -1.0   1.0

 2.0   4.0   7.0
 0.0   1.0   1.5
 0.0   0.0  -2.0

 0.0   1.0   0.0
 1.0   0.0   0.0
 0.0   0.0   1.0


 1.0   0.0   0.0   0.0
 0.3   1.0   0.0   0.0
 0.1   0.3   1.0   0.0
 0.7   0.4  -4.1   1.0

 9.0   5.0   7.0   5.0
 0.0  15.3  15.7   7.3
 0.0   0.0  -3.3   4.3
 0.0   0.0   0.0  12.6

 0.0   0.0   0.0   1.0
 0.0   0.0   1.0   0.0
 0.0   1.0   0.0   0.0
 1.0   0.0   0.0   0.0


[Done] exited with code=0 in 0.63 seconds
```

Python Code for LU decomposition Matrices

MIS.py

D: > MIS.py > ...

```python
# Python3 Program to decompose
# a matrix into lower and
# upper triangular matrix
MAX = 100


def luDecomposition(mat, n):

    lower = [[0 for x in range(n)]
             for y in range(n)]
    upper = [[0 for x in range(n)]
             for y in range(n)]

    # Decomposing matrix into Upper
    # and Lower triangular matrix
    for i in range(n):

        # Upper Triangular
        for k in range(i, n):

            # Summation of L(i, j) * U(j, k)
            sum = 0
            for j in range(i):
                sum += (lower[i][j] * upper[j][k])

            # Evaluating U(i, k)
            upper[i][k] = mat[i][k] - sum

```

```python
        # Lower Triangular
        for k in range(i, n):
            if (i == k):
                lower[i][i] = 1  # Diagonal as 1
            else:

                # Summation of L(k, j) * U(j, i)
                sum = 0
                for j in range(i):
                    sum += (lower[k][j] * upper[j][i])

                # Evaluating L(k, i)
                lower[k][i] = int((mat[k][i] - sum) /
                                  upper[i][i])

    # setw is for displaying nicely
    print("Lower Triangular\t\tUpper Triangular")
```

```python
    # Displaying the result :
    for i in range(n):

        # Lower
        for j in range(n):
            print(lower[i][j], end="\t")
        print("", end="\t")

        # Upper
        for j in range(n):
            print(upper[i][j], end="\t")
        print("")


# Driver code
mat = [[5, 1, 6],
       [-4, 6, 3],
       [7, 4, 3]]

luDecomposition(mat, 3)
```

Output for LU decomposition Matrices

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
[Running] python -u "d:\MIS.py"
Lower Triangular        Upper Triangular
1    0    0        5    1    6
0    1    0        0    6    3
1    0    1        0    0    -3

[Done] exited with code=0 in 0.117 seconds
```

2

Write a code in a programming language to do A = QR decomposition for a given m x n matrix A. Also do the least square fitting of the problem in the medium article using the decomposition.

HIMAMSH & M.PRASANNA TEJA

QR DECOMPOSITION

QR decomposition is a decomposition of a matrix A into a product A = QR of an orthogonal matrix Q and an upper triangular matrix R.

QR decomposition is often used to solve the linear least squares problem and is the basis for a particular eigenvalue algorithm

$$A = QR$$

A is a square matrix
Q is an orthogonal matrix
R is an upper triangular matrix

- For writing A=QR decomposition we have to assume a matrix for A
- Suppose A is $\begin{bmatrix} 1 & 2 & 0 \\ 4 & 8 & 7 \\ 3 & 9 & 5 \end{bmatrix}$
- So, we have to divide it into 3 coloum matrices to find Q.we divide it as a,b,c.
- $a = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}$ $\quad b = \begin{bmatrix} 2 \\ 8 \\ 9 \end{bmatrix}$ $\quad c = \begin{bmatrix} 0 \\ 7 \\ 5 \end{bmatrix}$

- So, we know Q is a Orthogonal matrix. Considering it we have to write it q1,q2,q3 as the coloumn vector of Q which are ortho-normal.
- $q_1 = \frac{a}{||a||}$ → $|a| = \sqrt{(1)^2 + (4)^7 + (3)^2} = \sqrt{26}$

$$\sqrt{26}\begin{bmatrix}1\\4\\3\end{bmatrix} = \begin{bmatrix}0.1961\\0.7845\\0.5883\end{bmatrix}$$

- B = b - a.$\frac{(a^T \cdot b)}{a^T \cdot a}$ and q2 $= \frac{B}{||B||}$

- Since q2 is orthogonal to q1 we have to subtract from b the projection of b onto q1.

- $$B = b - a.\frac{(a^T \cdot b)}{a^T \cdot a} = \begin{bmatrix} 2 \\ 8 \\ 9 \end{bmatrix} - \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} (61/26) = \begin{bmatrix} -0.3462 \\ -1.3846 \\ 1.9615 \end{bmatrix}$$

$$q2 = \frac{B}{||B||} = \begin{bmatrix} -0.1427 \\ -0.5708 \\ 0.8086 \end{bmatrix}$$

- $$C = c - a.\frac{(a^T \cdot c)}{a^T \cdot a} - B.\frac{(B^T \cdot c)}{B^T \cdot B} \quad \text{and} \quad q3 = \frac{C}{||C||}$$

- Since q3 is orthogonal to both q1,q2.we have to subtract from c the projection of c onto q1and projection of c onto q2.

$$C = c - a.\frac{(a^T \cdot c)}{a^T \cdot a} - B.\frac{(B^T \cdot c)}{B^T \cdot B} = \begin{bmatrix} -1.6471 \\ 0.4118 \\ 0 \end{bmatrix}$$

$$\text{So , } q3 = \frac{C}{||C||} = \begin{bmatrix} -0.9701 \\ 0.2425 \\ 0 \end{bmatrix}$$

$$Q = [q1, q2, q3] = \begin{bmatrix} 0.1961 & -0.1427 & -0.9701 \\ 0.7845 & -0.5708 & 0.2425 \\ 0.5883 & 0.8086 & 0 \end{bmatrix}$$

- Now R is a upper triangular matrix So,we have to find transpose for q1,q2,q3. let R be p1,p2,p3
- P1= [0.1961   0.7845   0.5883]
- P2= [−0.1427   −0.5708   0.8086]
- P3=[−0.9701   0.2425   0]
- $$R = \begin{bmatrix} p1.a & p1.b & p1.c \\ 0 & p2.b & p2.c \\ 0 & 0 & p3.c \end{bmatrix} = \begin{bmatrix} 5.099 & 11.9631 & 8.433 \\ 0 & 2.4258 & 0.0476 \\ 0 & 0 & 1.6977 \end{bmatrix}$$

- Now to find least square fitting

we need 'x'

M =R^-1. Q^1. K [# K is a variable]

$$K = q1.||a|| + q2.||B|| + q3.||C||$$

$$= a+B+C$$

$$K = \begin{bmatrix} -0.9932 \\ 3.0271 \\ 4.9615 \end{bmatrix}$$

- M=R^-1. Q^1. K

$$M = \begin{bmatrix} -2.9540 \\ 0.9240 \\ 1 \end{bmatrix}$$

- We will see the code of this in next slide.

```
A=[8 9 3; 9 6 5; 2 1 9];
a=[8;9;2];
b=[9;6;1];
c=[3;5;9];
q1=a/norm(a);
x=transpose(a);
B=b-(a*((x*b)/(x*a)));
y=transpose(B);
C=c-(a*((x*c)/(x*a)))-(B*((y*c)/(y*B)));
q2=B/norm(B);
q3=C/norm(C);
Q=[q1 q2 q3];
disp(Q);
p1=transpose(q1);
p2=transpose(q2);
p3=transpose(q3);
R=[p1*a,p1*b,p1*c;0,p2*b,p2*c;0,0,p3*c];
D=Q*R;
disp(R);
disp(D)
K=a+B+C;
M=inv(R)*transpose(Q)*K;
disp(M)
```

```
>> Mis_assignment_Q2
    0.6554    0.7503    0.0867
    0.7373   -0.6107   -0.2889
    0.1638   -0.2533    0.9534

   12.2066   10.4862    7.1273
         0    2.8355   -3.0817
         0         0    7.3962

    8.0000    9.0000    3.0000
    9.0000    6.0000    5.0000
    2.0000    1.0000    9.0000

   -1.3766
    2.0868
    1.0000
```

Solving the same by rectangular matrix

untitled2.m | mis.m | Mis_assignment_Q2.m | rect.m

```
A=[8 9 ; 9 6 ];
a=[8;9];
b=[9;6];
q1=a/norm(a);
x=transpose(a);
B=b-(a*((x*b)/(x*a)));
y=transpose(B);

q2=B/norm(B);
Q=[q1 q2];
disp(Q);
p1=transpose(q1);
p2=transpose(q2);
R=[p1*a,p1*b,;0,p2*b];
D=Q*R;
disp(R);
disp(D)
K=a+B;
M=inv(R)*transpose(Q)*K;
disp(M)
```

Output

```
>> rect
    0.6644    0.7474
    0.7474   -0.6644

   12.0416   10.4637
         0    2.7405

    8.0000    9.0000
    9.0000    6.0000

    0.1310
    1.0000
```

- As the process has been explained in the previous slides we used the same concept for doing rectangular too but we have used 3*2 matrix instead of 3*3 matrix as we require rectangular matrix

PROBABILITY

Dr. NIMAL MADHU

1

1. Given that 42% of high school students would admit to lying at least once to a teacher during the past year and that 25% of students are male and would admit to lying at least once to a teacher during the past year. Assume that 50% of the students are male.
   i. What is the probability that a randomly selected student is either male or would admit to lying to a teacher, during the past year?
   ii. A student is selected from the subpopulation of those who would admit to lying to a teacher during the past year. What is the probability that the student is female?

GUNNAM HIMAMSH and VIKHYAT BANSAL

SOLUTION

Event A = student admit lying to teacher at least once during past year

Event M = student who is selected is a male

Event$(A \cap M)$ = student who admit lying to teacher during past year is a male

Event F = student who is selected is a female

So as given in the question students who admit lying is 42%, student selected is a male is 50% and student who admit lying is a boy is 25%

Then the probability of occurring event A will be 42%

Then the probability of occurring event M will be 50%

Then the probability of occurring event (A∩M) will be 25%

$$P(A) = 0.42$$

$$P(M) = 0.50 \qquad P(F) = 0.50$$

$$P(A \cap M) = 0.25$$

i) The probability that a randomly selected student is either male or would admit lying to a teacher during the past year is given by **$P(A \cup M)$** .

$$P(A \cup M) = P(A) + P(M) - P(A \cap M)$$

$$= 0.42 + 0.5 - 0.25$$

$$= 0.67$$

$$P(A \cup M) = \mathbf{0.67}$$

ii) Now the probability that the selected student is a female given that student would admit to lying to a teacher during the past exam is given by P(F/A) and we can write

$$P(F/A) = 1 - P(M/A)$$

$$P(M/A) = P(A \cap M)/P(A) \qquad \text{(Conditional probability formula)}$$

$$= 0.25/0.42$$

$P(M/A) = 0.5952$

Hence,

$$P(F/A) = 1 – P(M/A)$$

$$= 1 – 0.5952$$

$$P(F/A)= 0.4048$$

i) **P**(A∪M) $= \mathbf{0.67}$

ii) $P(F/A)= = 0.4048$

2

A household is categorized as 'Prosperous' if its income exceeds $80,000. Similarly, a household is categorized as 'Educated' if at least one of the members has completed college. The current Population Survey says that of all the households 15.2% are prosperous, 34.1% are educated, and 9% are both prosperous and educated. From this information, estimate

i. The probability that a household selected is either prosperous or educated?

ii. The probability that at least one person in a household is educated, given it is not prosperous.

M.PRASANNA TEJA

Answer

Probability of a family being NOT prosperous is

$P(\text{not pros}) = 1 - 0.152 = 0.848$

Probability of a family being prosperous and educated

$P(\text{not pros} \cap \text{edu}) = P(\text{edu}) - P(\text{pros} \cap \text{edu})$

$= 0.341 - 0.09 = 0.251$

Let the probability of the event, atleast one person in a household is educated given it is not prosperous be P(B)

$P(B) = P(\text{pros} \cup \text{edu}) = P(\text{not pros} \cap \text{edu}) / P(\text{not pros})$

$= 0.251 / 0.848$

$= 0.296$

3

3. People with albinism have little pigment in their skin, hair, and eyes. The gene that governs albinism has two forms (called alleles), which we denote by *a* and *A*. Each person has a pair of these genes, one inherited from each parent. A child inherits one of each parent's two alleles independently with probability 0.5. Albinism is a recessive trait, so a person is albino only if the inherited pair is *aa*.

   i. Alan's parents and his sister Beth are not albino, but he is. What can you infer about the gene type present in Alan's parent?

   ii. Which of the types *aa, Aa, AA* could a child of Alan's parents have? What is the probability of each type?

   iii. Given Beth is not an albino. What are the probabilities for Beth's possible genetic types, given this fact?

GAJULA SRI VATSANKA

i) According to the given question ,there are 4 different gene types: aa,aA,AA,Aa

- Now, we have been given that Alan is an albino ,so he must be of the gene type of "aa".
- As Alan parents are not albinos, there gene type probably should be "Aa“, “Aa” or “AA” and cannot be “aa”.
- Since Alan is an albino ,if anyone of his parent's gene is AA ,he won't be an Albino.
- **INFERENCE:**

  So, it's mentioned that his(Alan’s) genes make him Albino ,which means their(parent’s) gene type is either "Aa" or "aA“ but neither “AA” nor “aa”.

- As we know that Alan's parent can't have a gene type of "aa“.
- So, the probability of Alan's parent's child having the gene type

1) aa, P(aa) = 1/4
2) AA, P(AA) =1/4
3) Aa, P(Aa) =1/4
4) aA, P(aA) =1/4

- A child of Alan parent's can have all the 4 possible combination of gene types.
- Also if Aa and aA are same gene types P(Aa)=P(aA)=1/2

iii)

- Alan's parent can have gene type of "Aa" and "aA". As Beth is not an Albino, Probability of her having the gene type is:

1) aA, P(aA)=1/3
2) Aa, P(Aa)=1/3
3) AA, P(AA)=1/3

if Aa and aA are same gene types

P(aA)=2/3 : P(Aa)=2/3

- P(Aa)=P(aA)=P(AA)=1/3 ,since she can contain only 3 gene types.
- As this selection of any one type of gene among the possible 3 gene types will be equal to 1/3

THANK YOU