Matrix and table structure

Hyerim Bae
Department of Industrial Engineering, Pusan National University
hrbae@pusan.ac.kr

Outline

Matrix Overview

- How is data saved in 2D array?

Matrix A

	0	1	2	3	4
0	0.1	1.2	2.3	3.7	4.5
1	2.1	4.2	5.2	1.1	6.1
2	4.1	7.9	3.3	3.2	3.4
3	0.3	0.9	1.3	4.2	3.5
4	3.3	2.9	2.2	7.9	8.9

- What is dimension of Matrix A ?
- A[0][3] = 3.7
- A[2][1] = 7.9
- A[5][2] = ?

Matrix Overview

- The layout of a two-dimensional Java arrays

Array elements that refers to another array creates a multidimensional array.

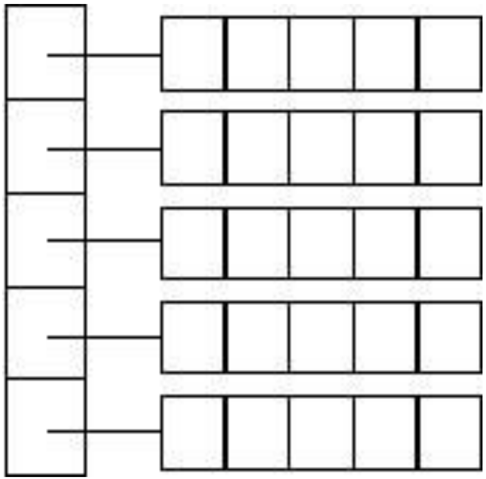


Image source : G. Gundersen & T. Steihaug, Data Structures in Java for Matrix Computation

Matrix Overview

- Create a Matrix in Java

Creating a matrix (object of array 2D)

```
double[][] a = new double[m][n];
```

Creating and directly initializing matrix value

```
double[][] a = {
    { 99.0, 85.0, 98.0, 0.0 },
    { 98.0, 57.0, 79.0, 0.0 },
    { 92.0, 77.0, 74.0, 0.0 },
    { 94.0, 62.0, 81.0, 0.0 },
    { 99.0, 94.0, 92.0, 0.0 },
    { 80.0, 76.5, 67.0, 0.0 },
    { 76.0, 58.5, 90.5, 0.0 },
    { 92.0, 66.0, 91.0, 0.0 },
    { 97.0, 70.5, 66.5, 0.0 },
    { 89.0, 89.5, 81.0, 0.0 },
    {  0.0,  0.0,  0.0, 0.0 }
};
```

Setting matrix value

```
double[][] a;
a = new double[m][n];
for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
        a[i][j] = 0;
```

Accessing matrix value

```
for (int i = 0; i < a.length; i++) {
    for (int j = 0; j < a[i].length; j++) {
        System.out.print(a[i][j] + " ");
    }
    System.out.println();
}
```

Matrix in Python

- Python 자체 리스트로 생성

```
row, column = 2, 3
arr_2d = [[None] * column for i in range(row)]
print(arr_2d)
# [[None, None, None], [None, None, None]]
R=[[1, 2, 3],[4, 5, 6]]
print(R)
# [[1, 2, 3], [4, 5, 6]]
```

- Numpy 라이브러리 사용

```
import numpy as np
# 0으로 초기화된 2차원 배열 생성
arr_2d = np.zeros((2, 3)) print(arr_2d)
# [[0. 0. 0.] # [0. 0. 0.]]
R = np.array([[0, 1, 2], [3, 4, 5]])
# [[0 1 2], [3 4 5]]
```

Matrix Operation

- Matrix addition

```
double[][] c = new double[n][n];
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        c[i][j] = a[i][j] + b[i][j];
    }
}
```

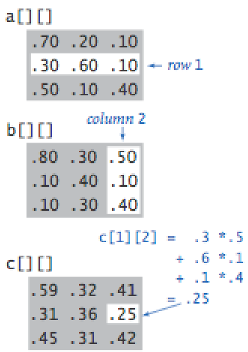
- Matrix Multiplication

```
double[][] c = new double[n][n];
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++)  {
        for (int k = 0; k < n; k++)  {
            c[i][j] += a[i][k]*b[k][j];
        }
    }
}
```

- Matrix Transpose

```
int[][] trasposedMatrix = new int[n][m];

for(int x = 0; x < n; x++)
{
    for(int y = 0; y < m; y++)
    {
        trasposedMatrix[x][y] = matrix[y][x];
    }
}
```



Matrix multiplication

Matrix calculation in python

```
def matMulti(A, B):
    answer = [len(B[0]) * [0] for i in range(len(A))]
    if len(A[0]) == len(B):
        for i in range(0, len(A)):
            for j in range(0,len(B[0])):
                for k in range(0, len(A[0])):
                    answer[i][j] += A[i][k]*B[k][j]
        return answer
```

Matrix Operation

- Frobenius Norm example

$$s = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A_{ij}$$

Loop-order (i,j):

```
double s = 0;
double[] array = new double[m][n];
for(int i = 0; i<m; i++){
	for(int j = 0; j<n; j++){
		s += array[i][j];
	}
}
```

Loop-order (j,i):

```
double s = 0;
double[] array = new double[m][n];
for(int j = 0;j<n;j++){
	for(int i = 0;i<m; i++){
		s+=array[i][j];
	}
}
```

What is the difference ?

Matrix Operation

Basic observation:

Accessing the consecutive elements in a row will be faster then accessing consecutive elements in a column.

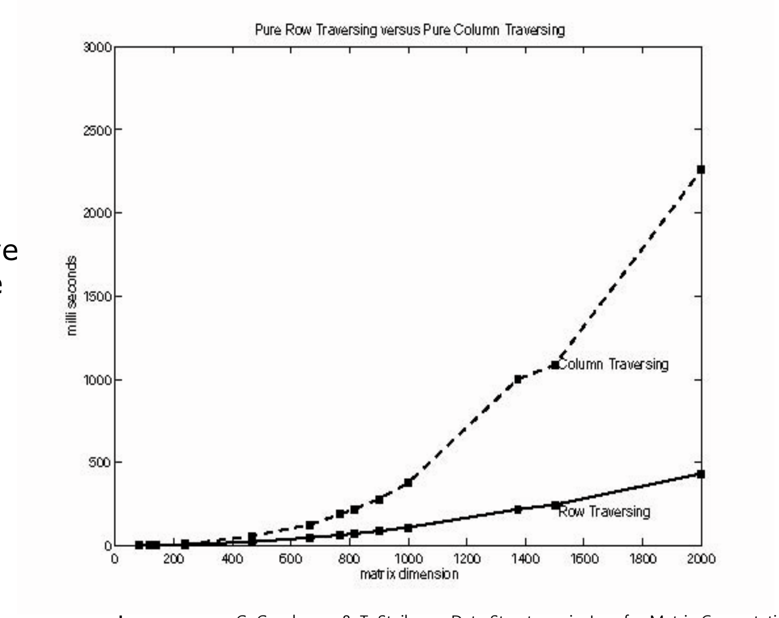


Image source : G. Gundersen & T. Steihaug, Data Structures in Java for Matrix Computation

JAMA Packages

- A basic linear algebra package implemented in Java.
- It provides user-level classes for constructing and manipulating real dense matrices.
- It is intended to serve as the standard matrix class for Java.
- JAMA packages is already optimized.

Elementary Operations	addition subtraction multiplication scalar multiplication element-wise multiplication element-wise division unary minus transpose norm
Decompositions	Cholesky LU QR SVD symmetric eigenvalue nonsymmetric eigenvalue
Equation Solution	nonsingular systems least squares
Derived Quantities	condition number determinant rank inverse pseudoinverse

Table Structure

How to make table structure in Java

1. Matrix Representation (Array 2D)
2. Create your own table object
3. Using Java API, JTable

(1) Create your own table object

- Read data from CSV File

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class CSVReader {

    public static void main(String[] args) {
        String fileName= "src/country.csv";
        File file= new File(fileName);

        // this gives you a 2-dimensional array of strings
        List<List<String>> lines = new ArrayList<>();
        Scanner inputStream;
        try{
            inputStream = new Scanner(file);

            while(inputStream.hasNext()){
                String line= inputStream.next();
                String[] values = line.split(",");
                // this adds the currently parsed line to the 2-dimensional string array
                lines.add(Arrays.asList(values));
            }
            inputStream.close();
        }catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        // the following code lets you iterate through the 2-dimensional array
        int rowNo = 1;
        for(List<String> line: lines) {
            int columnNo = 1;
            for (String value: line) {
                System.out.println("Row " + rowNo + " Column " + columnNo + ": " + value);
                columnNo++;
            }
            rowNo++;
        }
    }
}
```

country.csv

country.csv

```
"1.0.0.0","1.0.0.255","16777216","16777471","AU","Australia"
"1.0.1.0","1.0.3.255","16777472","16778239","CN","China"
"1.0.4.0","1.0.7.255","16778240","16779263","AU","Australia"
"1.0.8.0","1.0.15.255","16779264","16781311","CN","China"
"1.0.16.0","1.0.31.255","16781312","16785407","JP","Japan"
"1.0.32.0","1.0.63.255","16785408","16793599","CN","China"
"1.0.64.0","1.0.127.255","16793600","16809983","JP","Japan"
"1.0.128.0","1.0.255.255","16809984","16842751","TH","Thailand"
```

output

```
run:
Row 1 Column 1: "1.0.0.0"
Row 1 Column 2: "1.0.0.255"
Row 1 Column 3: "16777216"
Row 1 Column 4: "16777471"
Row 1 Column 5: "AU"
Row 1 Column 6: "Australia"
Row 2 Column 1: "1.0.1.0"
Row 2 Column 2: "1.0.3.255"
Row 2 Column 3: "16777472"
Row 2 Column 4: "16778239"
Row 2 Column 5: "CN"
Row 2 Column 6: "China"
Row 3 Column 1: "1.0.4.0"
Row 3 Column 2: "1.0.7.255"
Row 3 Column 3: "16778240"
Row 3 Column 4: "16779263"
Row 3 Column 5: "AU"
Row 3 Column 6: "Australia"
```

(3) JTable

import javax.swing.JTable

```java
//headers for the table
String[] columns = new String[] {
    "Id", "Name", "Hourly Rate", "Part Time"
};

//actual data for the table in a 2d array
Object[][] data = new Object[][] {
    {1, "John", 40.0, false },
    {2, "Rambo", 70.0, false },
    {3, "Zorro", 60.0, true },
};
//create table with data
JTable table = new JTable(data, columns);

//add the table to the frame
this.add(new JScrollPane(table));

this.setTitle("Table Example");
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.pack();
this.setVisible(true);
```

Setting table header

Setting data table in a 2D array

Create table with data

Setting table display

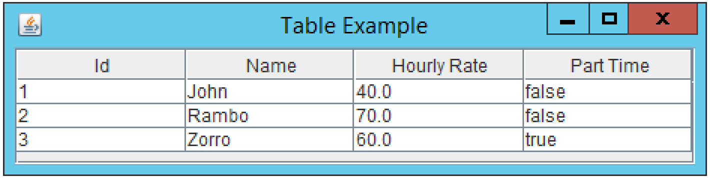


Table Example

Id	Name	Hourly Rate	Part Time
1	John	40.0	false
2	Rambo	70.0	false
3	Zorro	60.0	true

Analytic Hierarchy Process (AHP)

- Analytic Hierarchy Process (AHP), introduced by **Thomas Saaty** (1980) is a method on multi criteria decision making (MCDM).

- The AHP had been applied for Decision Support System (DSS).

- This algorithm involves both subjective **human judgments** and **objective evaluation**

- Important term in AHP: **criteria or factors**, **weight of important**, **alternatives or choice** and **goal**.

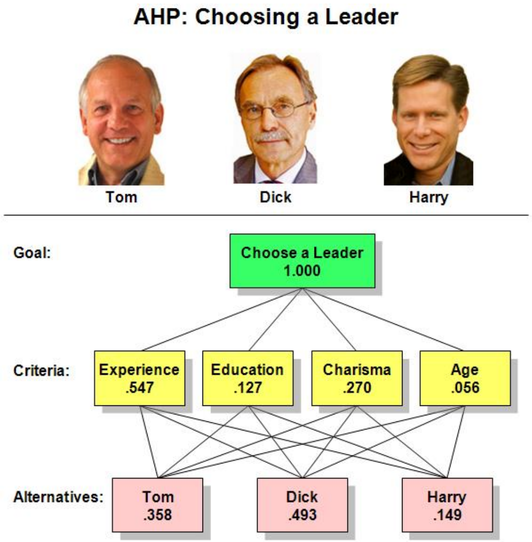


Image source : https://en.wikipedia.org/wiki/Analytic_hierarchy_process

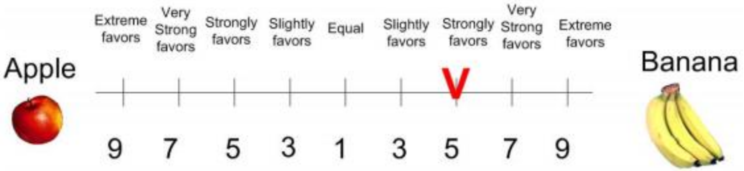
Some AHP Applications

Goal	Criteria	Alternatives
Decide best school	• Distance, • Reputation, • Cost, • Teacher kindliness	Name of schools under consideration
Finding best apartment	• Price, • Down payment, • Distance from shops, • Distance from work/school • Neighbor's Friendliness	List of apartments under consideration
Select best politician	• Charm • Good working program • Benefit for our organization • Attention to our need	List of candidates
Determine thesis topic	• Fast to finish, • Research Cost , • Level of Attractiveness,	List of thesis topics
Buy car	• Initial Price • Operating & Maintenance cost, • Service and comfort, • Status	Car's trade mark (Honda, GM, Ford, Nissan etc.)
Decide whether to buy or to rent a machine	• Total cost (capital, maintenance, operational) • Service • Time to operate • Interconnection with other machines	Rent or Buy

How AHP works

- It is a method to derive ratio scales from **paired comparisons.** The input can be obtaine d from **actual measurement** such as price, weight etc., or from **subjective opinion** suc h as satisfaction feelings and preference.
- AHP allows some **small inconsistency** in judgment because human is not always consi stent.
- Pair-wise Comparison

E.g. John try to choose his the most favourite fruit

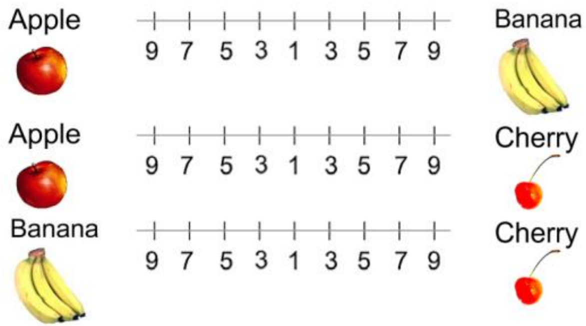


How AHP works

- Pair-wise comparison

Now suppose you have three choices of fruits.

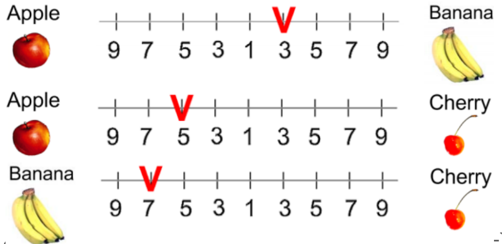
Then the pair wise comparison goes as the following



Number of things	1	2	3	4	5	6	7	n
number of comparisons	0	1	3	6	10	15	21	$\frac{n(n-1)}{2}$

How AHP works

- Making a Comparison Matrix



Because we have three comparisons, thus we have 3 by 3 matrix

- The diagonal elements of the matrix are always 1 and we only need to **fill up the upper triangular matrix**.

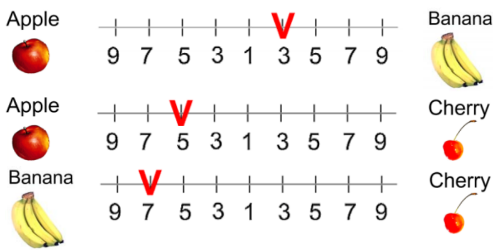
$$\mathbf{A} = \begin{array}{c} \\ apple \\ banana \\ cerry \end{array} \begin{array}{c} \begin{array}{ccc} apple & banana & cerry \end{array} \\ \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \end{array}$$

How the AHP works

- Making a Comparison Matrix

  Because we have three comparisons, thus we have 3 by 3 matrix
- The diagonal elements of the matrix are always 1 and we only need to **fill up the upper triangular matrix**.
- How to fill up the upper triangular matrix is using the following rules

1. If the judgment value is on the left side of 1, we put the actual judgment valu



$$\mathbf{A} = \begin{array}{c} \\ apple \\ banana \\ cerry \end{array} \begin{array}{c} \begin{array}{ccc} apple & banana & cerry \end{array} \\ \begin{bmatrix} 1 & \frac{1}{3} & 5 \\ & 1 & 7 \\ & & 1 \end{bmatrix} \end{array}$$

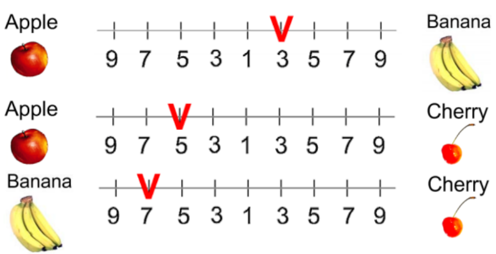
How AHP works

- Making Comparison Matrix

  Because we have three comparisons, thus we have 3 by 3 matrix
- The diagonal elements of the matrix are always 1 and we only need to fill up the upper t riangular matrix.
- How to fill up the upper triangular matrix is using the following rules

1. If the judgment value is on the left side of 1, we put the actual judgment valu
2. If the judgment value is on the right side of 1, we put the reciprocal value

Reciprocal values, $a_{ji} = \dfrac{1}{a_{ij}}$



$$\mathbf{A} = \begin{array}{c} \\ apple \\ banana \\ cerry \end{array} \begin{array}{c} \begin{array}{ccc} apple & banana & cerry \end{array} \\ \begin{bmatrix} 1 & \frac{1}{3} & 5 \\ 3 & 1 & 7 \\ \frac{1}{5} & \frac{1}{7} & 1 \end{bmatrix} \end{array}$$

How AHP works

- How to compute Eigen Vector and Eigen Value

Computing Eigen Vector by using **approximation**

1. Suppose we have 3 by 3 reciprocal matrix from paired comparison

$$\mathbf{A} = \begin{array}{c} \\ apple \\ banana \\ cerry \end{array} \begin{array}{c} \begin{array}{ccc} apple & banana & cerry \end{array} \\ \begin{bmatrix} 1 & \frac{1}{3} & 5 \\ 3 & 1 & 7 \\ \frac{1}{5} & \frac{1}{7} & 1 \end{bmatrix} \end{array}$$

2. We sum each column of the reciprocal matrix to get

$$\mathbf{A} = \begin{array}{c} \\ apple \\ banana \\ cerry \\ \text{sum} \end{array} \begin{array}{c} \begin{array}{ccc} apple & banana & cerry \end{array} \\ \begin{bmatrix} 1 & \frac{1}{3} & 5 \\ 3 & 1 & 7 \\ \frac{1}{5} & \frac{1}{7} & 1 \end{bmatrix} \\ \begin{array}{ccc} \frac{21}{5} & \frac{31}{21} & 13 \end{array} \end{array}$$

3. Then we divide each element of the matrix with the sum of its column

$$\mathbf{A} = \begin{array}{c} \\ apple \\ banana \\ cerry \\ \text{sum} \end{array} \begin{array}{c} \begin{array}{ccc} apple & banana & cerry \end{array} \\ \begin{bmatrix} \frac{5}{21} & \frac{7}{31} & \frac{5}{13} \\ \frac{15}{21} & \frac{21}{31} & \frac{7}{13} \\ \frac{1}{21} & \frac{3}{31} & \frac{1}{13} \end{bmatrix} \\ \begin{array}{ccc} 1 & 1 & 1 \end{array} \end{array}$$

4. The normalized principal Eigen vector can be obtained by averaging across the rows

$$\mathbf{w} = \frac{1}{3}\begin{bmatrix} \frac{5}{21}+\frac{7}{31}+\frac{5}{13} \\ \frac{15}{21}+\frac{21}{31}+\frac{7}{13} \\ \frac{1}{21}+\frac{3}{31}+\frac{1}{13} \end{bmatrix} = \begin{bmatrix} 0.2828 \\ 0.6434 \\ 0.0738 \end{bmatrix}$$

How AHP works

- Priority Vector

$$\mathbf{w} = \frac{1}{3}\begin{bmatrix} \frac{5}{21}+\frac{7}{31}+\frac{5}{13} \\ \frac{15}{21}+\frac{21}{31}+\frac{7}{13} \\ \frac{1}{21}+\frac{3}{31}+\frac{1}{13} \end{bmatrix} = \begin{bmatrix} 0.2828 \\ 0.6434 \\ 0.0738 \end{bmatrix}$$

The priority vector shows relative weights among the things that we compare. In our ex ample above, Apple is 28.28%, Banana is 64.34% and Cherry is 7.38%. So, John most pre ferable fruit is Banana, followed by Apple and Cheery

We know not only about ranking but also ratio, we can say that John likes banana 2.27 (=64.34/28.28) times more than apple and he also like banana so much 8.72 (=64.34/7.38) times more than cheery.

- Principal Eigen Value

we can also check the consistency.
Principal Eigen value is obtained from the summation of products between each element of Eigen vector and the sum of columns of the reciprocal matrix

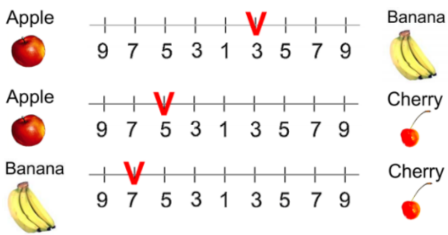
$$\lambda_{\max} = \tfrac{21}{5}(0.2828) + \tfrac{31}{21}(0.6434) + 13(0.0738) = 3.0967$$

How AHP works

- Consistency Index and Consistency Ratio
1. What is the meaning that our opinion is consistent?
2. How do we measure the consistency of subjective judgment?

- Let us look again on John's judgment that we discussed in the previous section.

Is John judgment consistent or not?



First he prefers Banana to Apple. Thus we say that for John, Banana has greater value than Apple. We write it as B > A .

Next, he prefers Apple to Cherry. For him, Apple has greater value than Cherry. We write it as A > C

Since B > A and A > C , logically, we hope that B > C or Banana must be preferable than Cherry. This logic of preference is called transitive property. If John answers in the last comparison is transitive (that he like Banana more than Cherry), then his judgment is consistent.
On the contrary, if John prefers Cherry to Banana then his answer is inconsistent.
Thus consistency is closely related to the transitive property

How AHP works

- Then he gave a measure of consistency, called Consistency Index (CI) as deviation or deg ree of consistency using the following formula

$$CI = \frac{\lambda_{\max} - n}{n-1}$$

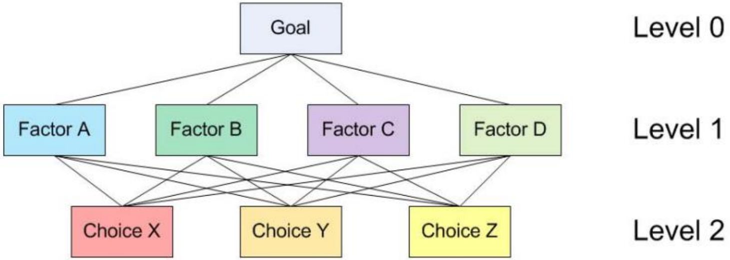
- $$CI = \frac{\lambda_{\max} - n}{n-1} = \frac{3.0967 - 3}{2} = 0.0484$$

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

$$CR = \frac{CI}{RI} = \frac{0.0484}{0.58} = 8.3\% < 10\%$$

Thus, John's subjective evaluation about his fruit preference is consistent

Examples



Paired comparison matrix level 1 with respect to the goal

Criteria	A	B	C	D	Priority Vector
A	1.00	3.00	7.00	9.00	**57.39%**
B	0.33	1.00	5.00	7.00	**29.13%**
C	0.14	0.20	1.00	3.00	**9.03%**
D	0.11	0.14	0.33	1.00	**4.45%**
Sum	1.59	4.34	13.33	20.00	100.00%

λ_{max} =4.2692, CI = 0.0897, CR = 9.97% < 10% (acceptable)

Examples

Paired comparison matrix level 1 with respect to the goal

Criteria	A	B	C	D	Priority Vector
A	1.00	3.00	7.00	9.00	57.39%
B	0.33	1.00	5.00	7.00	29.13%
C	0.14	0.20	1.00	3.00	9.03%
D	0.11	0.14	0.33	1.00	4.45%
Sum	1.59	4.34	13.33	20.00	100.00%

$\lambda_{\max}$ =4.2692, CI = 0.0897, CR = 9.97% < 10% (acceptable)

$$\lambda_{\max} = (0.5739)(1.59) + (0.2913)(4.34) + (0.0903)(13.33) + (0.0445)(20) = 4.2692$$

$$CI = \frac{\lambda_{\max} - n}{n-1} = \frac{4.2692 - 4}{3} = 0.0897$$

$$CR = \frac{CI}{RI} = \frac{0.0897}{0.90} = 9.97\% < 10\%$$ (Thus, OK because quite consistent)

Examples

Paired comparison matrix level 2 with respect to Factor A

Choice	X	Y	Z	Priority Vector
X	1.00	1.00	7.00	**51.05%**
Y	1.00	1.00	3.00	**38.93%**
Z	0.14	0.33	1.00	**10.01%**
Sum	2.14	2.33	11.00	100.00%

λ_{max} =3.104, CI = 0.05, CR = 8.97% < 10% (acceptable)

Paired comparison matrix level 2 with respect to Factor B

Choice	**X**	**Y**	**Z**	**Priority Vector**
X	1.00	0.20	0.50	**11.49%**
Y	5.00	1.00	5.00	**70.28%**
Z	2.00	0.20	1.00	**18.22%**
Sum	8.00	1.40	6.50	100.00%

λ_{max} =3.088, CI = 0.04, CR = 7.58% < 10% (acceptable)

We can do the same for paired comparison with respect to Factor C and D. However, the w eight of
factor C and D are very small (look at Table 9 again, they are only about 9% and 5% respect ively),
therefore we can assume the effect of leaving them out from further consideration is negligi ble.
We ignore these two weights as set them as zero.

Examples

Adjusted weight for factor A = $\dfrac{57.39\%}{57.39\% + 29.13\%} = 0.663$

Adjusted weight for factor B = $\dfrac{29.13\%}{57.39\% + 29.13\%} = 0.337$

$$X = (0.663)(51.05\%) + (0.337)(11.49\%) = 37.72\%$$
$$Y = (0.663)(38.93\%) + (0.337)(70.28\%) = 49.49\%$$
$$Z = (0.663)(10.01\%) + (0.337)(18.22\%) = 12.78\%$$

Overall composite weight of the alternatives

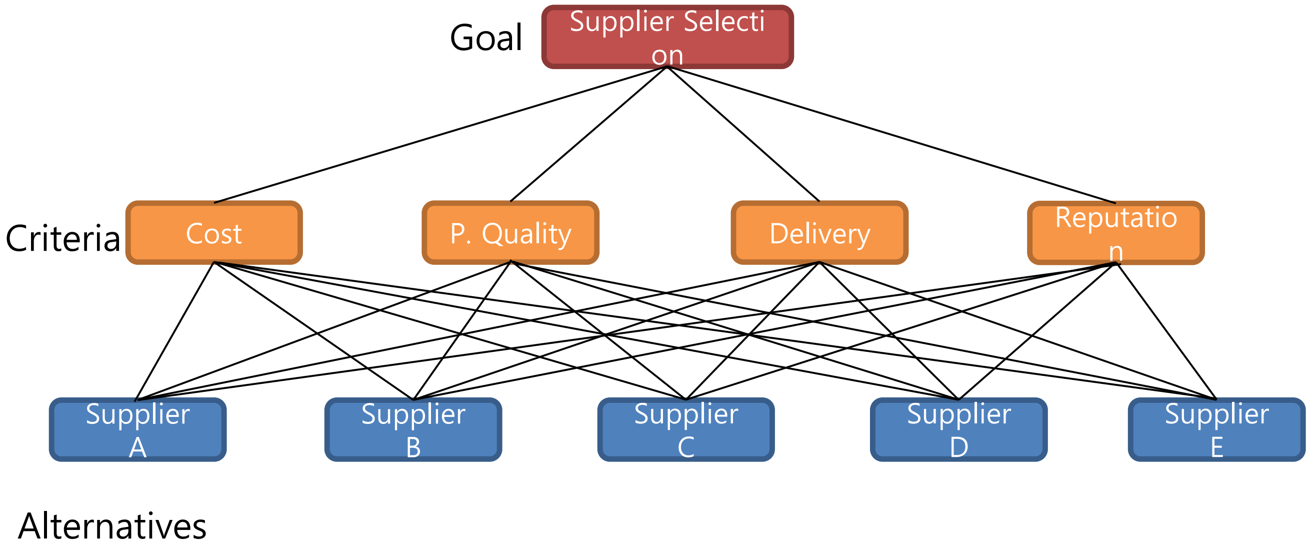
	Factor A	Factor B	Composite Weight
(Adjusted) Weight	*0.663*	*0.337*	
Choice X	51.05%	11.49%	**37.72%**
Choice Y	38.93%	70.28%	**49.49%**
Choice Z	10.01%	18.22%	**12.78%**

$$\overline{CR} = \frac{\sum_i w_i \, CI_i}{\sum_i w_i \, RI_i} = \frac{0.0897(1)+0.05(0.663)+0.04(0.337)}{0.90\,(1)+0.58(0.663)+0.58(0.337)} = 0.092 < 10\% \text{ (Acceptable)}$$

Case Study 1

- Vendor Selection for Service Sector Industry

  Choose the best supplier among the suppliers



Markov Matrix

- A *nxn* matrix is called a Markov matrix if all entries are non-negative and the sum of eac h column vector is equal to 1.

$$A = \begin{bmatrix} 1/2 & 1/3 \\ 1/2 & 2/3 \end{bmatrix}$$

- Markov matrices are also called **stochastic matrices**
- Java code to check whether it is Markov matrix or not

```java
public static boolean isMarkovMatrix(double[][] m) {
    for (int i = 0; i < m.length; i++) {
        double sum = 0;
        for (int j = 0; j < m.length; j++) {
            if (m[j][i] < 0) {
                return false;
            }
            sum += m[j][i];
        }
        if (sum != 1.0) {
            return false;
        }
    }
    return true;
}
```

Markov Matrix Example

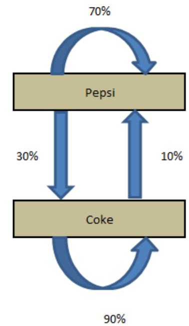
Currently, Pepsi owns 55% and Coke owns 45% of market share. Following are the conclusions drawn out by the market research co mpany:

P(P->P) : Probability of a customer staying with the brand Pepsi over a month = 0.7

P(P->C) : Probability of a customer switching from Pepsi to Coke over a month = 0.3

P(C->C) : Probability of a customer staying with the brand Coke over a month = 0.9

P(C->P) : Probability of a customer switching from Coke to Pepsi over a month = 0.1



The diagram simply shows the transitions and the current market share. Now, if we want to calculate the ma
we need to do following calculations :

Market share (t+1) of Pepsi = Current market Share of Pepsi * P(P->P) + Current market Share of Coke * P(C-

Market share (t+1) of Coke = Current market Share of Coke * P(C->C) + Current market Share of Pepsi * P(P-

These calculations can be simply done by looking at the following matrix multiplication :

Current State X Transition Matrix = Final State

Final state

$$\begin{array}{cc}\text{Pepsi} & \text{Coke}\end{array} \qquad \begin{array}{ccc} & \text{Pepsi} & \text{Coke}\end{array} \qquad \begin{array}{cc}\text{Pepsi} & \text{Coke}\end{array}$$

$$\begin{pmatrix}55\% & 45\%\end{pmatrix} \times \begin{array}{c}\text{Pepsi}\\ \text{Coke}\end{array}\begin{pmatrix}70\% & 30\%\\ 10\% & 90\%\end{pmatrix} = \begin{pmatrix}43\% & 57\%\end{pmatrix}$$

Initial state

As we can see clearly see that Pepsi, although has ı. This si
mple calculation is called Markov chain. If the tra ıt any fut
ure time point. Let's make the same calculation fo

Final state

$$\begin{array}{cc}\text{Pepsi} & \text{Coke}\end{array} \qquad \begin{array}{ccc} & \text{Pepsi} & \text{Coke}\end{array} \qquad \begin{array}{cc}\text{Pepsi} & \text{Coke}\end{array}$$

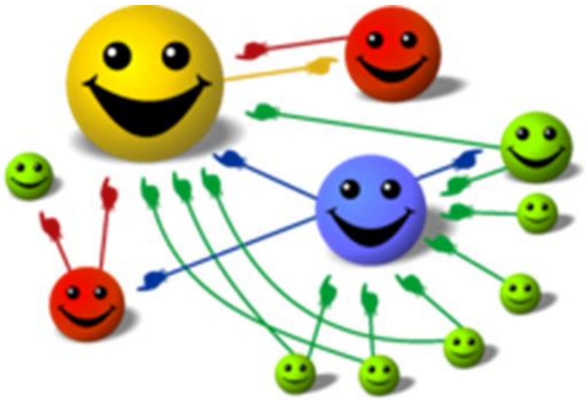
$$\begin{pmatrix}43\% & 57\%\end{pmatrix} \times \begin{array}{c}\text{Pepsi}\\ \text{Coke}\end{array}\begin{pmatrix}70\% & 30\%\\ 10\% & 90\%\end{pmatrix} = \begin{pmatrix}36\% & 64\%\end{pmatrix}$$

Initial state

Case Study 2

- **Idea: Links as votes**
- **Page is more important if it has more links**
- **Think of in-links as votes**
- **Are all in-links are equal?**

  Links from important pages count more

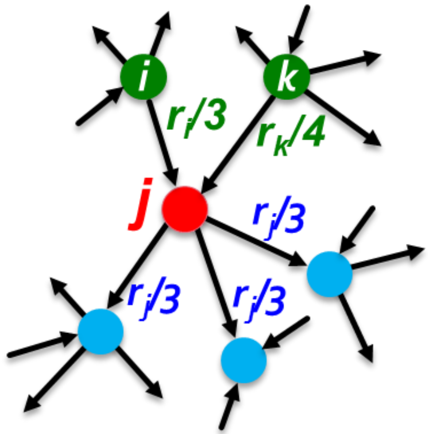


Case Study

- Google PageRank

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

r_i = importance score of page i
d_i = number of out-links



$r_j = r_i/3 + r_k/4$

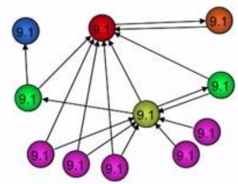
$$r_j = \beta \cdot \sum_{i \to j} \frac{r_i}{d_i} + (1 - \beta) \cdot \frac{1}{n}$$

$0 \le \beta \le 1$, in practice the best value of $\beta = 0.8 - 0.9$

n *is number of node*

Initialize PR(X) = 100%/N

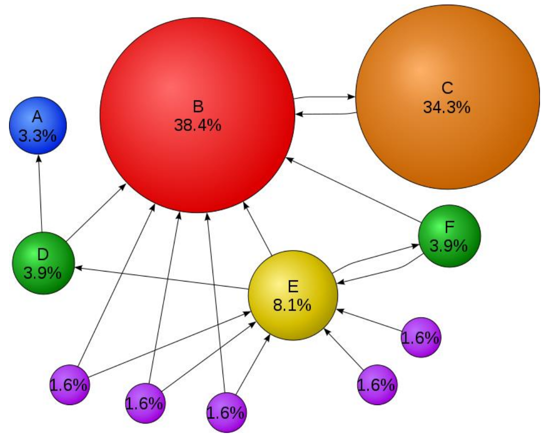
- total number of pages in our collection



t=1

PR(B) = 0.18 * 9.1 + 0.82 *
[PR(C) + ⅓ PR(E)
+ ½ PR(D) + ½ PR(F) + ½ PR(G)
+ ½ PR(H) + ½ PR(I)] ≈ 31

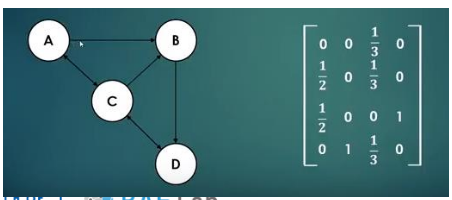
PR(C) = 0.18 * 9.1 + 0.82 * 9.1 = 9.1



t=2

PR(C) = 0.18*9.1 + 0.82*31 = 27.058

연결관계를 Matrix로 나타내기



Case Study

- Identifying key resource in business process using f-PageRank

  Finding important resource (machine or human) in business process

case	activity	Performer
case 1	activity A	John
case 1	activity B	Mike
case 1	activity C	John
case 1	activity D	Pete
case 2	activity A	John
case 2	activity C	Mike
case 2	activity B	John
case 2	activity D	Pete
case 3	activity A	Sue
case 3	activity B	Carol
case 3	activity C	Sue
case 3	activity D	Pete
case 4	activity A	Sue
case 4	activity C	Carol
case 4	activity B	Sue
case 4	activity D	Pete
case 5	activity A	Sue
case 5	activity E	Clare
case 5	activity D	Clare

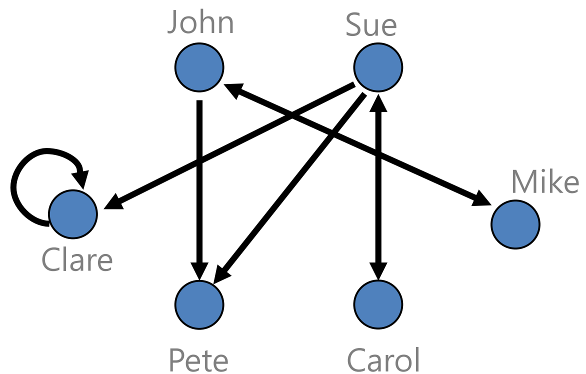
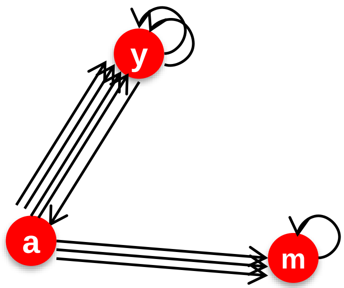


Fig. 1. Sociogram based on transfer of work

Case Study

- F-PageRank Example



Markov matrix

$$0.8 \begin{array}{c} \\ y \\ a \\ m \end{array} \begin{array}{c} \begin{array}{ccc} y & a & m \end{array} \\ \begin{bmatrix} 2/3 & 4/7 & 0 \\ 1/3 & 0 & 0 \\ 0 & 3/7 & 1 \end{bmatrix} \end{array} + 0.2 \underbrace{\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}}_{[1/N]_{NxN}} = \underbrace{\begin{array}{c} y \\ a \\ m \end{array} \begin{bmatrix} 0.60 & 0.52 & 0.07 \\ 0.33 & 0.07 & 0.07 \\ 0.07 & 0.41 & 0.87 \end{bmatrix}}_{A}$$

$$r^{(t+1)} = A \cdot r^{t}$$

		r^0	r^1	r^2	r^3		r^n
y		1/3	0.39	0.34	0.33		0.31
a	=	1/3	0.15	0.17	0.16	. . .	0.15
m		1/3	0.45	0.47	0.51		0.53