IIT ROORKEE

NPTEL ONLINE CERTIFICATION COURSE

Cluster analysis: Part - V

Dr. A. Ramesh

DEPARTMENT OF MANAGEMENT STUDIES

Agenda

- Dissimilarity matrix for mixed type variables
- Python demo for computing different types of distances
- Python demo for computing distance matrix for interval scaled data

Example

Consider the data given in the following table and compute a dissimilarity matrix for the objects of the table
Now we will consider all of the variables, which are of different types

object identifier	*test-1 (categorical)*	*test-2 (ordinal)*	*test-3 (ratio-scaled)*
1	code-A	excellent	445
2	code-B	fair	22
3	code-C	good	164
4	code-A	excellent	1,210

Example

- The procedures we followed for test-1 (which is categorical) and test-2 (which is ordinal) are the same as outlined above for processing variables of mixed types
- For categorical variable - $d(i, j) = \frac{p - m}{p}$,
- For ordinal variable - $z_{if} = \frac{r_{if} - 1}{M_f - 1}$
- For interval scale variable - $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}}$,

Normalizing the interval scale data

- First, however, we need to complete some work for test-3 (which is ratio-scaled)
- We have already applied a logarithmic transformation to its values
- Based on the transformed values of 2.65, 1.34, 2.21, and 3.08 obtained for the objects 1 to 4, respectively, we let $\max_h x_h = 3.08$ and $\min_h x_h = 1.34$
- We then normalize the values in the dissimilarity matrix obtained in Example solve for ratio data by dividing each one by $(3.08 - 1.34) = 1.74$

Dissimilarity matrix for test-3

- This results in the following dissimilarity matrix for test-3:

Object Identifier	Ratio scaled Data (x)	Log (x)
1	445	2.65
2	22	1.34
3	164	2.21
4	1210	3.08

(2,1)

$$\begin{bmatrix} 0 & & & \\ 0.75 & 0 & & \\ 0.25 & 0.50 & 0 & \\ 0.25 & 1.00 & 0.50 & 0 \end{bmatrix}$$

- For 1 and 2 = (2.65- 1.34)/(3.08–1.34) = 0.75

dissimilarity matrices for the three variables

- We can now use the dissimilarity matrices for the three variables in our computation of Equation $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}}$,
- For example, we get d(2,1)= (1(1)+1(1)+1(0.75))/ 3 =0.92

$$\begin{bmatrix} 0 & & & \\ \boxed{1} & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Dissimilarity matrix for categorical

$$\begin{bmatrix} 0 & & & \\ \boxed{1} & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}$$

Dissimilarity matrix for ordinal

$$\begin{bmatrix} 0 & & & \\ \boxed{0.75} & 0 & & \\ 0.25 & 0.50 & 0 & \\ 0.25 & 1.00 & 0.50 & 0 \end{bmatrix}$$

normalize the values in the dissimilarity matrix for ratio data

Example

- The resulting dissimilarity matrix obtained for the data described by the three variables of mixed types is:

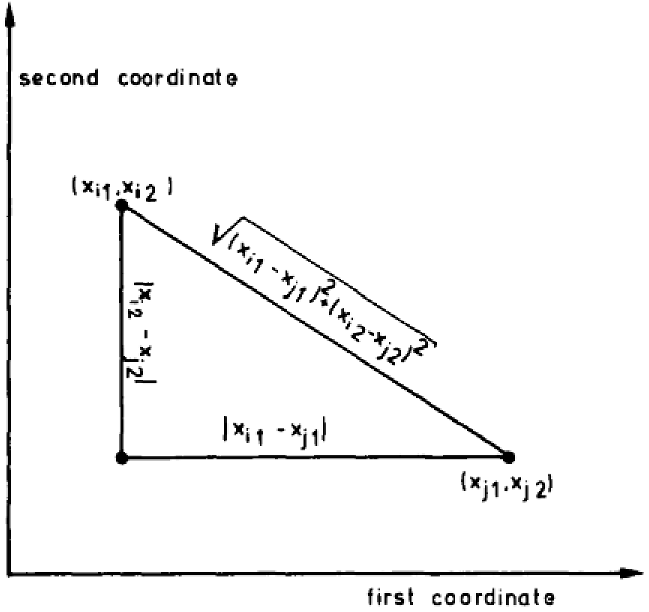
$$
\begin{array}{cc}
 & \\
(2,1) & \\
 & \\
(4,1) &
\end{array}
\begin{bmatrix}
0 & & & \\
\boxed{0.92} & 0 & & \\
\boxed{0.58} & 0.67 & 0 & \\
0.08 & \boxed{1.00} & 0.67 & 0
\end{bmatrix}
$$

Interpretation

- If we go back and look at Table of given data, we can intuitively guess that objects 1 and 4 are the most similar, based on their values for test-1 and test-2
- This is confirmed by the dissimilarity matrix, where d(4,1) is the lowest value for any pair of different objects
- Similarly, the matrix indicates that objects 2 and 4 are the least similar

Distance Measurement using python - Euclidean Distance :

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}$$



Python Demo for Euclidean Distance

In [1]:
```
import scipy
from scipy.spatial import distance
```

#Euclidean Distance

In [2]:
```
import numpy as np
a = [1,2,3]
b = [4,5,6]
dst = distance.euclidean(a,b)
```

In [3]:
```
dst
```

Out[3]: 5.196152422706632

Distance Measurement using python – Minkowski Distance :

$$d(i, j) = (|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \cdots + |x_{in} - x_{jn}|^p)^{1/p},$$

- P =1 Manhattan distance
- P = 2 Euclidean distance

Python Demo for Minkowski Distance

#Minkowski Distance

```
In [4]: distance.minkowski([1, 0, 0], [0, 1, 0], 1) #Manhattan distance
Out[4]: 2.0

In [5]: distance.minkowski([1, 0, 0], [0, 1, 0], 2) #Euclidean distance
Out[5]: 1.4142135623730951

In [6]: distance.minkowski([1, 2, 3], [4, 5, 6], 2)
Out[6]: 5.196152422706632

In [7]: distance.minkowski([1, 2, 3], [4, 5, 6], 3)
Out[7]: 4.3267487109222245
```

Dissimilarity matrix

#dissimilarity or distance matrix

In [9]:
```
import pandas as pd
from scipy.spatial import distance_matrix

data = [[1, 4], [2, 5], [3, 6]]
df = pd.DataFrame(data,columns=['a', 'b'])
df
```

Out[9]:

	a	b
0	1	4
1	2	5
2	3	6

In [10]:
```
pd.DataFrame(distance_matrix(df.values, df.values))
```

Out[10]:

	0	1	2
0	0.000000	1.414214	2.828427
1	1.414214	0.000000	1.414214
2	2.828427	1.414214	0.000000

Distance matrix calculation for Interval-Scaled Variables

- For example :
- Take eight people, the weight (in kilograms) and the height (in centimetres
- In this situation, n = 8 and p = 2.

Person	Weight(Kg)	Height(cm)
A	15	95
B	49	156
C	13	95
D	45	160
E	85	178
F	66	176
G	12	90
H	10	78

#data matrix

In [5]:
```
import pandas as pd
from scipy.spatial import distance_matrix

data = [[15, 95], [49, 156], [13, 95], [45, 160], [85, 178], [66, 176], [12, 90], [10, 78]]
ctys = ['A', 'B','C','D','E','F','G','H']
df = pd.DataFrame(data, columns=['Weight', 'Height'], index=ctys)
```

In [6]:
```
df
```

Out[6]:

	Weight	Height
A	15	95
B	49	156
C	13	95
D	45	160
E	85	178
F	66	176
G	12	90
H	10	78

In [7]:

```
Distance_matrix = pd.DataFrame(distance_matrix(df.values, df.values), index=df.index, columns=df.index)
Distance_matrix
```

Out[7]:

	A	B	C	D	E	F	G	H
A	0.000000	69.835521	2.000000	71.589105	108.577162	95.718337	5.830952	17.720045
B	69.835521	0.000000	70.830784	5.656854	42.190046	26.248809	75.663730	87.206651
C	2.000000	70.830784	0.000000	72.449983	109.877204	96.798760	5.099020	17.262677
D	71.589105	5.656854	72.449983	0.000000	43.863424	26.400758	77.388630	89.157165
E	108.577162	42.190046	109.877204	43.863424	0.000000	19.104973	114.337221	125.000000
F	95.718337	26.248809	96.798760	26.400758	19.104973	0.000000	101.548018	112.871608
G	5.830952	75.663730	5.099020	77.388630	114.337221	101.548018	0.000000	12.165525
H	17.720045	87.206651	17.262677	89.157165	125.000000	112.871608	12.165525	0.000000

Distance matrix calculation using Python

In [8]:
```
Distance_matrix.round(decimals=1, out=None)
```

Out[8]:

	A	B	C	D	E	F	G	H
A	0.0	69.8	2.0	71.6	108.6	95.7	5.8	17.7
B	69.8	0.0	70.8	5.7	42.2	26.2	75.7	87.2
C	2.0	70.8	0.0	72.4	109.9	96.8	5.1	17.3
D	71.6	5.7	72.4	0.0	43.9	26.4	77.4	89.2
E	108.6	42.2	109.9	43.9	0.0	19.1	114.3	125.0
F	95.7	26.2	96.8	26.4	19.1	0.0	101.5	112.9
G	5.8	75.7	5.1	77.4	114.3	101.5	0.0	12.2
H	17.7	87.2	17.3	89.2	125.0	112.9	12.2	0.0

Thank You