Correspondence Analysis Using Singular Value Decomposition

Shubham Pandey (MIT2020050), Shubham Sahu (MIT2020051), Pradeep Kumar(MIT2020053)
G17_SVD_A3
Indian Institute Of Information Technology, Allahabad

Abstract

In this paper we tried to discuss correspondence analysis technique With the help of an example. How correspondence analysis can be applied on multivariant data set and how inferences can be made which helps in understanding categorical data better . Paper also explains how singular value decomposition helps in dimensionality reduction in the process of correspondence analysis.

Index Terms : Correspondence analysi, Singular Value Decomposition, multivariate analysis,perceptual map

1. Introduction

Correspondence analysis is multivariant analysis and it deals with categorical data nominal data or ordeal data . Correspondence analysis is the geometric approach to multivariate descriptive data analysis .It is also dimension reduction technique and categorical variables particularly measured in nominal scale and then develops perceptual maps of extracted components. CA captures nonlinearity between variables represented in contingency tables .then based on contingency tables fit measures are computed i.e row profiles , column profiles and then relationships between column profiles and row profiles are captured through weighted y $\hat{2}$ -distances and then dimensionality reduction is possible or not will be done through singular value decomposition.

2. Related Work About Topic

A. Correspondence Analysis .

Correspondence analysis is multivariate analysis technique and it was proposed by Herman Otto Hartley and was later developed in France in early 1960s and 1970s by Jean Paul Benzecri.It is similar to principal component analysis but it is applied on categorical data rather than continuous data. Correspondence analysis creates orthogonal components CA is performed on contingency table C of size mxn where m is number of rows and n is number of columns.

B. Singular value decomposition (SVD) .

The first authors who focused on SVD were E. Beltrami (1873) and C. Jordan (1874). A further development of this method was proposed by A. Marshal and I. Olkin (1979). One of the most important works on SVD algorithm was presented by Eckart and Young in the first issue of Psychometrika (Eckart, Young, 1936). Psychometricians used this algorithm under name Eckart-Young decomposition. Other names include the basic structure (Horst, 1936, Green, Carroll, 1976), as well as the canonical form (Eckart, Young, 1936), or singular decomposition (Good, 1969, Kshirsagar, 1972). Today, it is known under the name of singular value decomposition. More information about this decomposition is given by J.M. Chambers (1977), K.R. Gabriel (1978), C.R. Rao (1980), and M. Greenacre and L.G. Underhill (1982). SVD in a correspondence analysis allows to determine the coordinates of points, which in turn allows for the application of the points representing the categories on the map perception. The problem of decomposition of A matrix by SVD in a correspondence analysis have been addressed by R.A. Fisher (1940), M.J. Greenacre (1984), E.B. Anderson (1991), and J.D. Jobson (1992). The following discussion explains how decomposition of A matrix is connected to the value of inertia. An important element of this study is a graphical presentation of the configuration of points in a two-dimensional space for all the algorithms with the use of procedures in python.

3. Algorithm

Catogorical Data are collected in terms of frequencies and the data table is known as contingency table.
Step 1. Obtain correspondence matrix Z. Where each element in Z i.e
Zij =Xij/N.

Step 2. Divide each element of matrix Z by respective row mass.

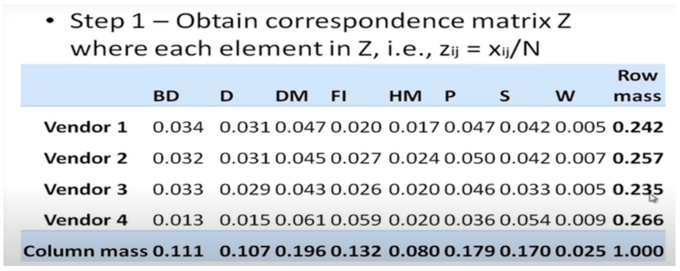
Step 3. Divide each element of matrix Z by respective column mass.

Step 4.Weighted X2 -Distance[D]
D =Dr-1/2 (Z - rcT)Dc-1/2

Step 5. SVD is applied to partion the D matrix into three matrices U, V and S where U is a p*k matrix , S is a k*k diagonal matrix with diagonal elements inthe form s1 >= s2 >= ..... >= sk >0 and k is the reduced dimensions.

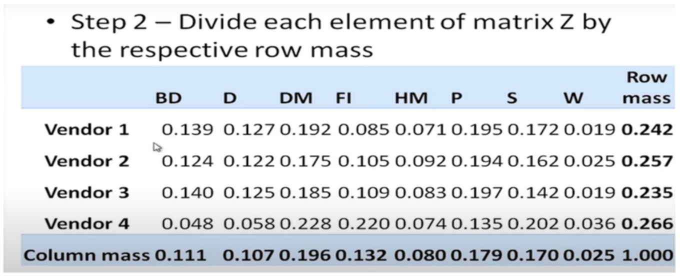
Step 6- Obtain Perputual map

4. Experiment And Result



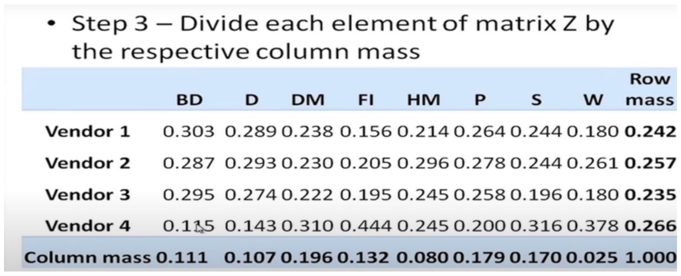
• Step 1 – Obtain correspondence matrix Z where each element in Z, i.e., $z_{ij} = x_{ij}/N$

	BD	D	DM	FI	HM	P	S	W	Row mass
Vendor 1	0.034	0.031	0.047	0.020	0.017	0.047	0.042	0.005	**0.242**
Vendor 2	0.032	0.031	0.045	0.027	0.024	0.050	0.042	0.007	**0.257**
Vendor 3	0.033	0.029	0.043	0.026	0.020	0.046	0.033	0.005	**0.235**
Vendor 4	0.013	0.015	0.061	0.059	0.020	0.036	0.054	0.009	**0.266**
Column mass	**0.111**	**0.107**	**0.196**	**0.132**	**0.080**	**0.179**	**0.170**	**0.025**	**1.000**



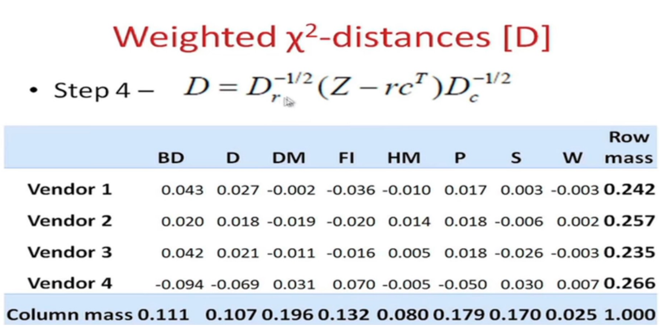
• Step 2 – Divide each element of matrix Z by the respective row mass

	BD	D	DM	FI	HM	P	S	W	Row mass
Vendor 1	0.139	0.127	0.192	0.085	0.071	0.195	0.172	0.019	**0.242**
Vendor 2	0.124	0.122	0.175	0.105	0.092	0.194	0.162	0.025	**0.257**
Vendor 3	0.140	0.125	0.185	0.109	0.083	0.197	0.142	0.019	**0.235**
Vendor 4	0.048	0.058	0.228	0.220	0.074	0.135	0.202	0.036	**0.266**
Column mass	**0.111**	**0.107**	**0.196**	**0.132**	**0.080**	**0.179**	**0.170**	**0.025**	**1.000**



• Step 3 – Divide each element of matrix Z by the respective column mass

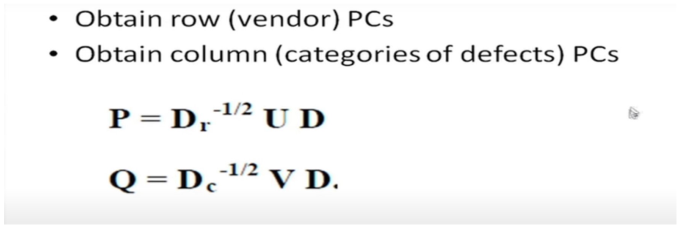
	BD	D	DM	FI	HM	P	S	W	Row mass
Vendor 1	0.303	0.289	0.238	0.156	0.214	0.264	0.244	0.180	**0.242**
Vendor 2	0.287	0.293	0.230	0.205	0.296	0.278	0.244	0.261	**0.257**
Vendor 3	0.295	0.274	0.222	0.195	0.245	0.258	0.196	0.180	**0.235**
Vendor 4	0.115	0.143	0.310	0.444	0.245	0.200	0.316	0.378	**0.266**
Column mass	**0.111**	**0.107**	**0.196**	**0.132**	**0.080**	**0.179**	**0.170**	**0.025**	**1.000**



Weighted χ^2-distances [D]

• Step 4 – $D = D_r^{-1/2}(Z - rc^T)D_c^{-1/2}$

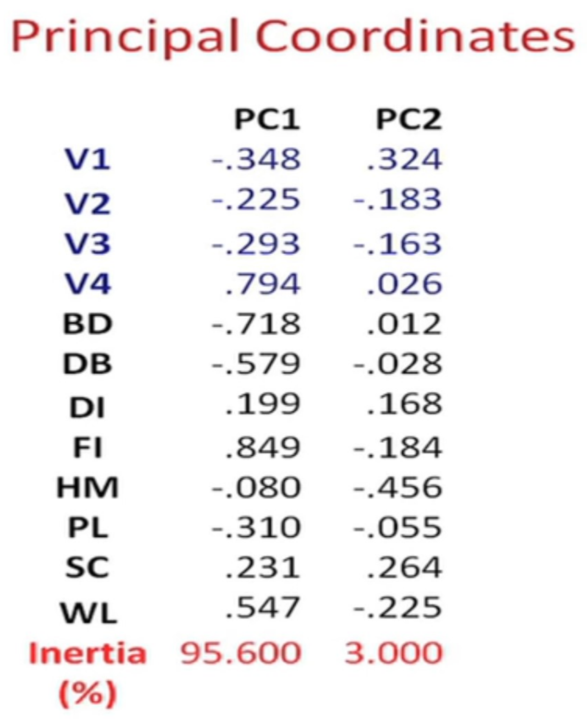
	BD	D	DM	FI	HM	P	S	W	Row mass
Vendor 1	0.043	0.027	-0.002	-0.036	-0.010	0.017	0.003	-0.003	**0.242**
Vendor 2	0.020	0.018	-0.019	-0.020	0.014	0.018	-0.006	0.002	**0.257**
Vendor 3	0.042	0.021	-0.011	-0.016	0.005	0.018	-0.026	-0.003	**0.235**
Vendor 4	-0.094	-0.069	0.031	0.070	-0.005	-0.050	0.030	0.007	**0.266**
Column mass	**0.111**	**0.107**	**0.196**	**0.132**	**0.080**	**0.179**	**0.170**	**0.025**	**1.000**



• Obtain row (vendor) PCs
• Obtain column (categories of defects) PCs

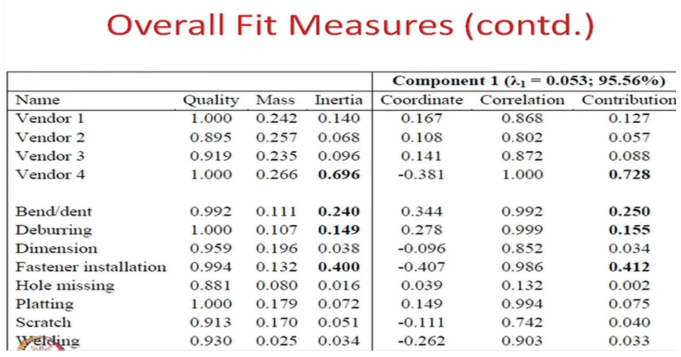
$$\mathbf{P} = \mathbf{D_r}^{-1/2}\,\mathbf{U}\,\mathbf{D}$$

$$\mathbf{Q} = \mathbf{D_c}^{-1/2}\,\mathbf{V}\,\mathbf{D}.$$



Principal Coordinates

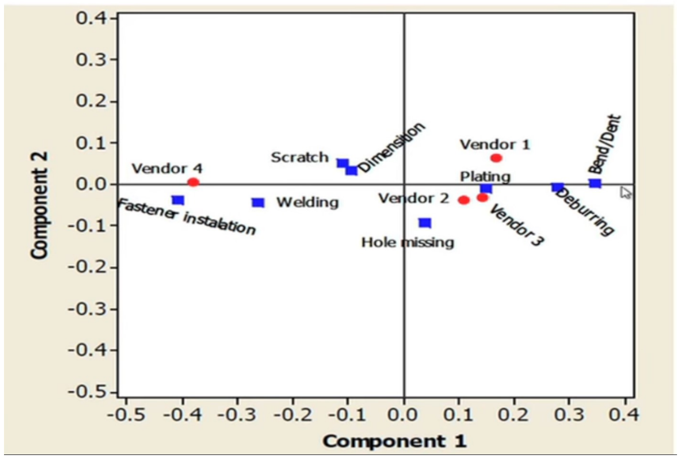
	PC1	PC2
V1	-.348	.324
V2	-.225	-.183
V3	-.293	-.163
V4	.794	.026
BD	-.718	.012
DB	-.579	-.028
DI	.199	.168
FI	.849	-.184
HM	-.080	-.456
PL	-.310	-.055
SC	.231	.264
WL	.547	-.225
Inertia (%)	95.600	3.000



Overall Fit Measures (contd.)

				Component 1 ($\lambda_1 = 0.053; 95.56\%$)		
Name	Quality	Mass	Inertia	Coordinate	Correlation	Contribution
Vendor 1	1.000	0.242	0.140	0.167	0.868	0.127
Vendor 2	0.895	0.257	0.068	0.108	0.802	0.057
Vendor 3	0.919	0.235	0.096	0.141	0.872	0.088
Vendor 4	1.000	0.266	**0.696**	-0.381	1.000	**0.728**
Bend/dent	0.992	0.111	**0.240**	0.344	0.992	**0.250**
Deburring	1.000	0.107	**0.149**	0.278	0.999	**0.155**
Dimension	0.959	0.196	0.038	-0.096	0.852	0.034
Fastener installation	0.994	0.132	**0.400**	-0.407	0.986	**0.412**
Hole missing	0.881	0.080	0.016	0.039	0.132	0.002
Platting	1.000	0.179	0.072	0.149	0.994	0.075
Scratch	0.913	0.170	0.051	-0.111	0.742	0.040
Welding	0.930	0.025	0.034	-0.262	0.903	0.033

Perceptual Map (contd.)



5. Conclusion

1)Correspondence Analysis is an exploratory method designed to account for association (Pearson 2) in a small number of dimensions

- Row and column scores provide an optimal scaling of the category levels

- Plots of these can suggest an explanation for association

2) Correspondence Analysis uses the singular value decomposition to approximate the matrix of residuals from independence

3) Standard and principal coordinates have different geometric properties, but are essentially re-scalings of each other.

4)Multi-way tables can be handled by:

- Stacking approach— collapse some dimensions interactively to a 2-way table

- Each way of stacking $\rightarrow$ a loglinear model

- MCA analyzes the full n way table using an indicator matrix or the Burtnmatrix

6. References

[1] International Encyclopedia of Education (Third Edition), 2010

[2] https://www.researchgate.net/publication/330252187

[3] Understanding the Math of Correspondence Analysis by Tim Bock

[4] Rao, C.R. (1980). Matrix approximation and reduction of dimensional-ity in multivariate statis-tical analysis. In: P.R. Krishnaiah (ed.), Multivariate analysis V (pp. 3–22). North Holland, Amsterdam.

Appendex

```
import pandas as pd
import numpy as np
import itertools from scipy.stats
import chi2_contingency
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt
%matplotlib inline
```

```
authors = ["Charles Darwin", "Rene Descartes","Thomas Hobbes", "Mary Shelley", "Mark Twain"]

initials=['CD1','CD2','CD3','RD1','RD2','RD3','TB1','TB2','TB3','MS1','MS2','MS3','MT1','MT2','MT3']

authorSamples = list(itertools.chain.from_iterable([[a+": "+str(i) for i in [1,2,3]] for a in authors]))

authorSamples chars=["B", "C", "D", "F", "G", "H", "I", "L", "M", "N","P", "R", "S", "U", "W", "Y"]
```

```
sampleCrosstab= [[34, 37, 44, 27, 19, 39, 74, 44, 27, 61, 12, 65, 69,22,
                14, 21],
                [18, 33, 47, 24, 14, 38, 66, 41, 36,72, 15, 62, 63, 31, 12, 18],
                [32, 43, 36, 12, 21, 51, 75, 33, 23, 60, 24, 68, 85,18, 13, 14],
                [13, 31, 55, 29, 15, 62, 74, 43, 28,73, 8, 59, 54, 32, 19, 20],
                [8, 28, 34, 24, 17, 68, 75, 34, 25, 70, 16, 56, 72,31, 14, 11],
                [9, 34, 43, 25, 18, 68, 84, 25, 32, 76,14, 69, 64, 27, 11, 18],
                [15, 20, 28, 18, 19, 65, 82, 34, 29, 89, 11, 47, 74,18, 22, 17],
                [18, 14, 40, 25, 21, 60, 70, 15, 37,80, 15, 65, 68, 21, 25, 9],
                [19, 18, 41, 26, 19, 58, 64, 18, 38, 78, 15, 65, 72,20, 20, 11],
                [13, 29, 49, 31, 16, 61, 73, 36, 29,69, 13, 63, 58, 18, 20, 25],
                [17, 34, 43, 29, 14, 62, 64, 26, 26, 71, 26, 78, 64, 21, 18, 12],
                [13, 22, 43, 16, 11, 70, 68, 46, 35,57, 30, 71, 57, 19, 22, 20],
                [16, 18, 56, 13, 27, 67, 61, 43, 20, 63, 14, 43, 67,34, 41, 23],
                [15, 21, 66, 21, 19, 50, 62, 50, 24, 68, 14, 40, 58, 31, 36, 26],
                [19, 17, 70, 12, 28, 53, 72, 39, 22, 71, 11, 40, 67,25, 41, 17]]

dfTableForm = pd.DataFrame(data=np.transpose(sampleCrosstab),

columns=authorSamples)

dfTableForm.head()

grandTotal = np.sum(sampleCrosstab)

correspondenceMatrix = np.divide(sampleCrosstab,grandTotal)

rowTotals = np.sum(correspondenceMatrix, axis=1)

columnTotals = np.sum(correspondenceMatrix, axis=0)

independenceModel = np.outer(rowTotals, columnTotals)

chiSquaredStatistic = grandTotal * np.sum (np.square
(correspondenceMatrix -independenceModel)/independenceModel)

print(chiSquaredStatistic)

# Quick check - compare to scipy Chi-Squared test statistic, prob, dof,
ex = chi2_contingency(sampleCrosstab) print(statistic)
print(np.round(prob, decimals=2))

# pre-calculate normalised rows
norm_correspondenceMatrix =
np.divide(correspondenceMatrix,rowTotals[:, None])
chiSquaredDistances =
np.zeros((correspondenceMatrix.shape[0],correspondenceMatrix.shape[
0]))

norm_columnTotals = np.sum(norm_correspondenceMatrix, axis=0)
for row in range(correspondenceMatrix.shape[0]):

chiSquaredDistances[row]=np.sqrt(np.sum(np.square(norm_correspond
enceMatrix
                                                            -
norm_correspondenceMatrix[row])/columnTotals, axis=1))
```

```
dfchiSquaredDistances =
pd.DataFrame(data=np.round(chiSquaredDistances*100).astype(int),
columns=authorSamples)
dfchiSquaredDistances
```

```
standardizedResiduals = np.divide((correspondenceMatrix-
independenceModel),np.sqrt(independenceModel))
u,s,vh = np.linalg.svd(standardizedResiduals, full_matrices=False)
deltaR = np.diag(np.divide(1.0,np.sqrt(rowTotals)))
rowScores=np.dot(np.dot(deltaR,u),np.diag(s))
dfFirstTwoComponents = pd.DataFrame(data=[l[0:2] for l in rowScores],
columns=['X', 'Y'], index=initials)
dfFirstTwoComponents
```

```
ax = sns.scatterplot(data=dfFirstTwoComponents,x='X', y='Y',
hue=initials) ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')
ax.get_legend().set_visible(False)
for label in initials: plt.annotate(label,
(dfFirstTwoComponents.loc[label,:]['X'],
dfFirstTwoComponents.loc[label,:]['Y']), horizontalalignment='center',
verticalalignment='center',size=11)
```