Unsupervised Learning

by

AINDRILA GARAI

MSC STATISTICS, IIT KANPUR

aindrilag22@iitk.ac.in

Challenge:

In unsupervised learning, there is no way to check our work because we don't know the true answer of the problem.

Principal Components Analysis(PCA):

Principal components analysis refers to the process by which principal components are computed and the subsequent use of these components in understanding the data.

- PCA is an **unsupervised** approach since it involves only a set of features and no associated response.
- PCA serves as a tool for data visualization and data imputation (to fill in missing values in a data matrix).

What Are Principal Components?

PCA finds a low-dimensional representation of a data set that contains as much as possible of the variation. The first principal component of a set of features $X_1, X_2, \ldots, X_p$ is the normalized linear combination of the features that has the largest variance and each of the variables in X has been centered to have mean zero. The second principal component is the linear combination of $X_1, \ldots, X_p$ that has maximal variance out of all linear combinations that are uncorrelated with Z_1.

- A **biplot** is a single display that represents both the principal component scores and the loading vectors.
- Principal components provide low-dimensional linear surfaces that are closest to the observations. Together the first M principal component score vectors and the first M principal component loading vectors provide the best M-dimensional approximation (in terms of Euclidean distance) to the ith observation.

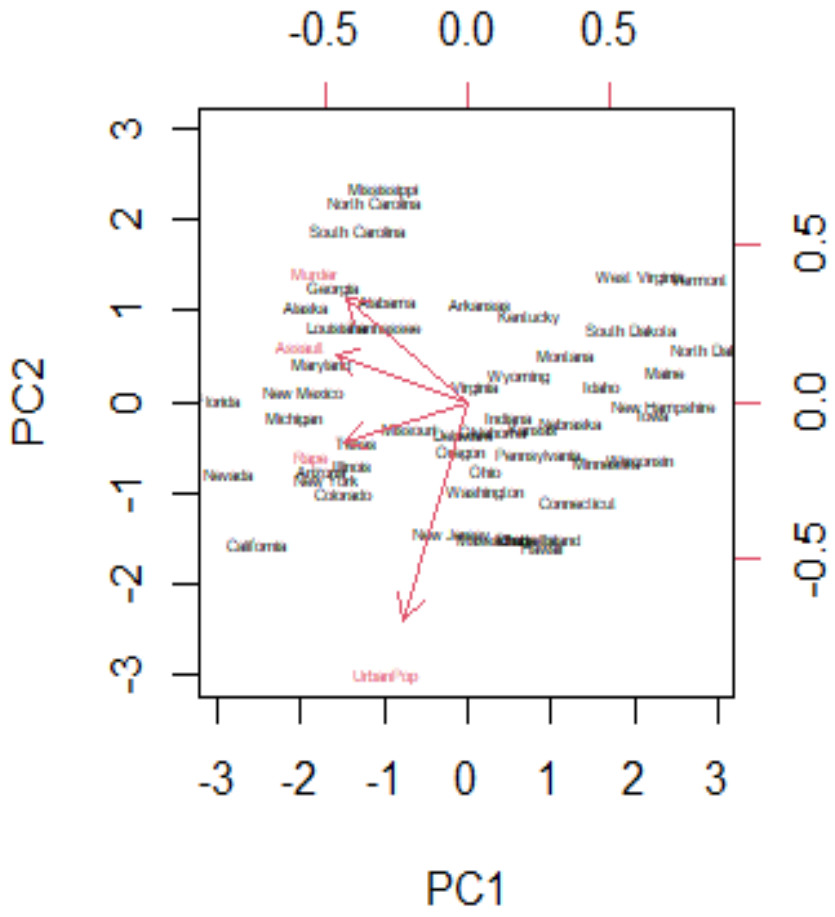
```
# we perform PCA on the USArrests data se
states <- row.names (USArrests)
pr.out <- prcomp (USArrests , scale = TRUE) # centers the variables to have m
ean zero
names (pr.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"

pr.out$rotation # loading vector

##                  PC1        PC2        PC3         PC4
## Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
## Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
## UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
## Rape     -0.5434321 -0.1673186  0.8177779  0.08902432

biplot (pr.out , scale = 0 ,cex = 0.4 )
```



The Proportion of Variance Explained:

A natural question is how much of the information in a given data set is lost by projecting the observations onto the first few principal components? Therefore, the PVE of the mth principal component is given by -

$$\underbrace{\sum_{j=1}^{p}\frac{1}{n}\sum_{i=1}^{n}x_{ij}^2}_{\text{Var. of data}} = \underbrace{\sum_{m=1}^{M}\frac{1}{n}\sum_{i=1}^{n}z_{im}^2}_{\text{Var. of first } M \text{ PCs}} + \underbrace{\frac{1}{n}\sum_{j=1}^{p}\sum_{i=1}^{n}\left(x_{ij}-\sum_{m=1}^{M}z_{im}\,\phi_{jm}\right)^2}_{\text{MSE of } M\text{-dimensional approximation}}$$

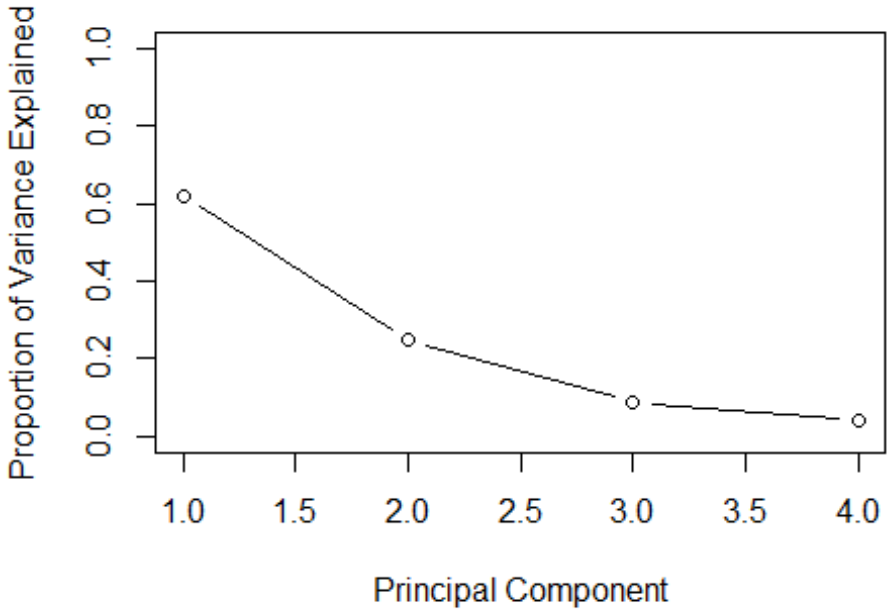
$$\text{PVE} = 1 - \frac{\text{RSS}}{TSS}$$

- Each principal component loading vector is unique up to a sign flip. - In general, a $n \times p$ data matrix X has $\min(n-1, p)$ distinct principal components. - Many statistical techniques, such as regression, classification, and clustering, can be easily adapted to use the $n \times M$ matrix whose columns are the first $M \ll p$ principal component score vectors, rather than using the full $n \times p$ data matrix. - Principal components can be used to impute impute the missing values through a process known as matrix completion.

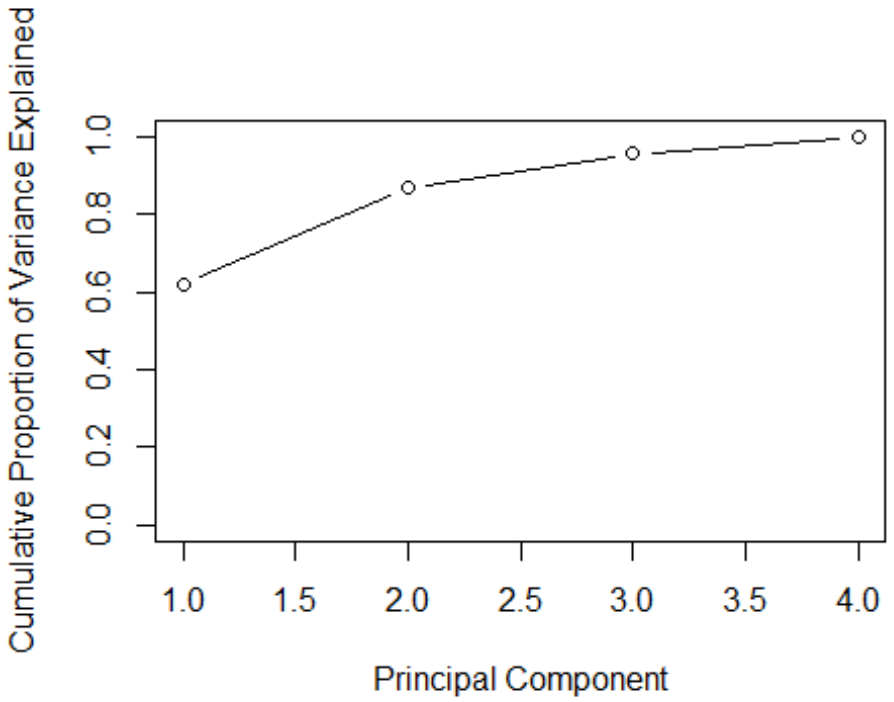
```
# To compute the proportion of variance
pr.var <- pr.out$sdev^2
pve <- pr.var / sum (pr.var)
round( pve, 2)

## [1] 0.62 0.25 0.09 0.04

plot (pve , xlab = "Principal Component", ylab = "Proportion of Variance Expl
ained", ylim = c(0, 1),
type = "b")
```




```
plot ( cumsum (pve), xlab = " Principal Component ", ylab = " Cumulative Prop
ortion of Variance Explained ", ylim = c(0, 1), type = "b")
```



```
# Matrix Completion
X <- data.matrix ( scale (USArrests))
pcob <- prcomp (X)
sX <- svd (X)
head(t(sX$d * t(sX$u)))
```

```
##            [,1]       [,2]        [,3]         [,4]
## [1,] -0.9756604  1.1220012 -0.43980366  0.154696581
## [2,] -1.9305379  1.0624269  2.01950027 -0.434175454
## [3,] -1.7454429 -0.7384595  0.05423025 -0.826264240
## [4,]  0.1399989  1.1085423  0.11342217 -0.180973554
## [5,] -2.4986128 -1.5274267  0.59254100 -0.338559240
## [6,] -1.4993407 -0.9776297  1.08400162  0.001450164
```

```
head(pcob$x) # these two are same
```

```
##                  PC1        PC2         PC3          PC4
## Alabama    -0.9756604  1.1220012 -0.43980366  0.154696581
## Alaska     -1.9305379  1.0624269  2.01950027 -0.434175454
## Arizona    -1.7454429 -0.7384595  0.05423025 -0.826264240
## Arkansas    0.1399989  1.1085423  0.11342217 -0.180973554
## California -2.4986128 -1.5274267  0.59254100 -0.338559240
## Colorado   -1.4993407 -0.9776297  1.08400162  0.001450164
```

k-means Clustering Methods:

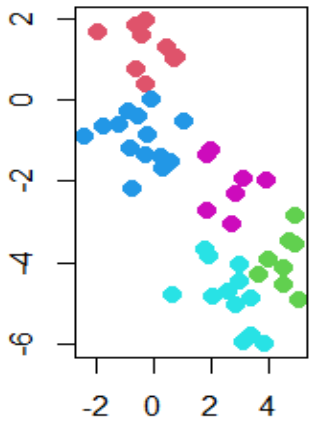
When we cluster the observations of a data set, we seek to partition them into distinct groups so that the observations within each group are quite similar to each other, while observations in different groups are quite different from each other. K-means algorithm finds a local rather than a global optimum. One disadvantage of K-means clustering is that it requires us to pre-specify the number of clusters K.

```
set.seed (2)
x <- matrix ( rnorm (50 * 2), ncol = 2)
x[1:25, 1] <- x[1:25, 1] + 3
x[1:25, 2] <- x[1:25, 2] - 4
km.out <- kmeans (x, 5, nstart = 20)
km.out$cluster
```

```
##  [1] 4 5 2 5 4 4 2 5 2 5 4 5 4 4 2 4 4 4 2 4 2 4 2 2 4 3 1 1 1 3 1 3 3 3 3
1 3 3
## [39] 3 1 1 1 3 5 3 5 1 3 3 3
```

```
par (mfrow = c(1, 2))
plot (x, col = (km.out$cluster + 1),
xlab = "", ylab = "", pch = 20, cex = 2) # K- Means Clustering Results with K
= 5
km.out <- kmeans (x, 5, nstart = 1)
km.out$tot.withinss
```

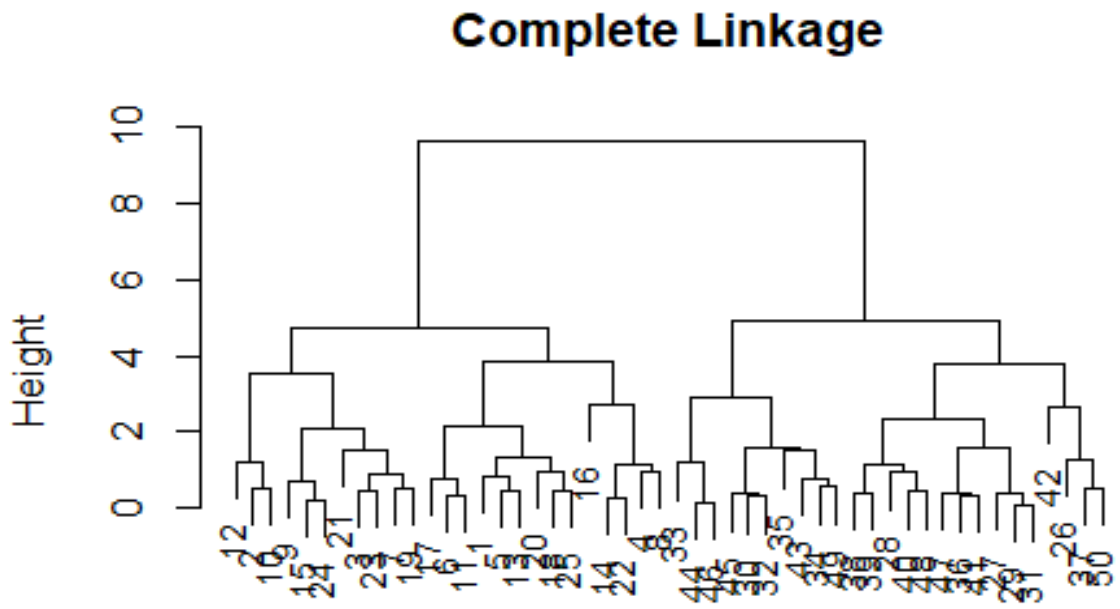
```
## [1] 50.89555
```



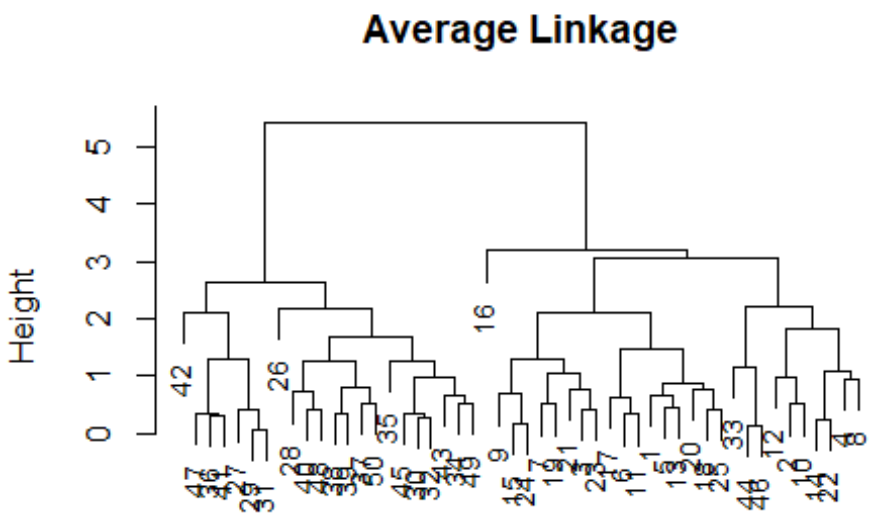
Hierarchical Clustering:

It is a tree-based representation of the observations, called a dendrogram. To group the datasets into clusters, it follows the bottom-up approach. It means, this algorithm considers each dataset as a single cluster at the beginning and then start combining the closet pair of clusters together. It does this until all the clusters together. It dose this until all the clusters are merged into a single cluster that contains all the datasets.

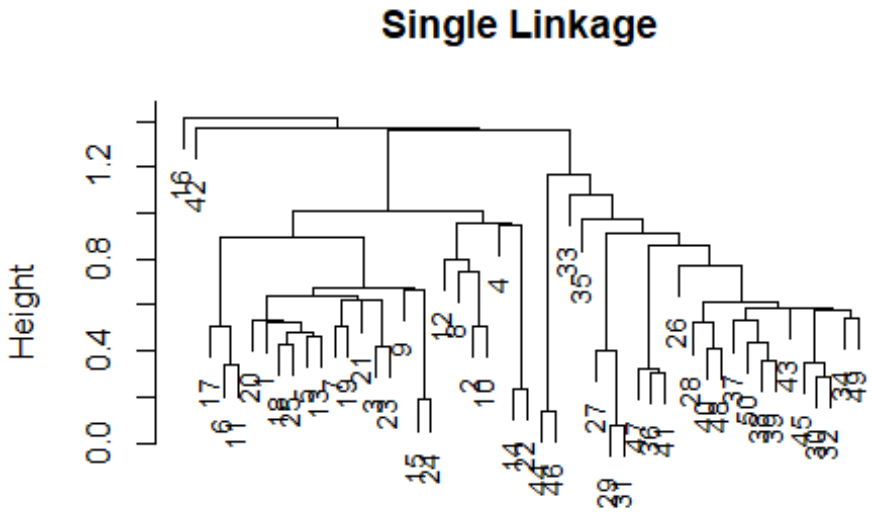
```
hc.complete <- hclust ( dist (x), method = "complete")
hc.average <- hclust ( dist (x), method = "average")
hc.single <- hclust ( dist (x), method = "single")
plot (hc.complete, main = " Complete Linkage ", xlab = "", sub = "", cex = .9
)
```




```
plot (hc.average , main = " Average Linkage ", xlab = "", sub = "", cex = .9)
```




```
plot (hc.single, main = " Single Linkage ", xlab = "", sub = "", cex = .9)
```




```
cutree (hc.complete, 2) # To determine the cluster labels
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2

xsc <- scale (x)
plot ( hclust ( dist (xsc), method = "complete") ,
main = " Hierarchical Clustering with Scaled Features ")
```