Labsheet - 3

# Principal Component Analysis (PCA) & Singular Value Decomposition (SVD)

Machine Learning (BITS F464)

I Semester 2021-22

## Principal Component Analysis

PCA is used to perform an orthogonal transformation that converts a set of observations having correlated attributes into a set of attributes of linearly uncorrelated variables called principal components.

### Example:

Use mtcars dataset, which is built into R. The dataset consists of data on 32 models of car. For each car, you have 11 features, as mpg (Fuel consumption), cyl (Number of cylinders), disp (Displacement), hp (Gross horsepower), drat (Rear axle ratio), wt (Weight), qsec (speed and acceleration), vs (Engine block), am (Transmission automatic or manual) gear (Number of forward gears), carb (Number of carburetors). Let us see what is there in mtcars

```
head ( mtcars )
```

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |

Two of the features vs and am are categorical so drop them.

```
d01 <- mtcars [ ,c(1:7,10,11)]
head ( d01 )
```

| | mpg | cyl | disp | hp | drat | wt | qsec | gear | carb |
|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160 | 110 | 3.90 | 2.620 | 16.46 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 | 3.90 | 2.875 | 17.02 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.85 | 2.320 | 18.61 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.15 | 3.440 | 17.02 | 3 | 2 |
| Valiant | 18.1 | 6 | 225 | 105 | 2.76 | 3.460 | 20.22 | 3 | 1 |

Relationship between every pair of attributes could be seen by plot(d01). Let us apply PCA on the data and see the summary.

```
d01 <- mtcars [,c(1:7,10 , 11)]
d01 . pca <-  princomp (d01 ,cor=TRUE, score=TRUE)
summary( d01   .  pca )
```

```
                           Comp.1    Comp.2     Comp.3    Comp.4
Standard deviation     2.3782219 1.4429485 0.71008086 0.5148082
Proportion of Variance 0.6284377 0.2313445 0.05602387 0.0294475
Cumulative Proportion  0.6284377 0.8597822 0.91580607 0.9452536
                           Comp.5    Comp.6     Comp.7      Comp.8
Standard deviation     0.42797037 0.3518426 0.32413257 0.241896155
Proportion of Variance 0.02035096 0.0137548 0.01167355 0.006501528
Cumulative Proportion  0.96560453 0.9793593 0.99103287 0.997534402
                            Comp.9
Standard deviation     0.148964367
Proportion of Variance 0.002465598
Cumulative Proportion  1.000000000
```

It is interesting to note that the first component itself has 92.70% variance. Added with 2nd component it becomes. 99.93%. Therefore, it looks like only two values could suffice to describe the data for most of the applications. Try to see plot of the components

```
plot(d01.pca)
```

How many PCs you should pick?

```
# scree plot
plot(d01.pca, type='l')
summary(pc) # 4 components is both 'elbow' and explains >85% variance
```

There is a very important variable called loading that specifies how individual attributes contribute to the components.

```
d01.pca$loadings
```

```
Loadings:
     Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
mpg   0.393         0.221         0.321  0.720  0.381  0.125  0.115
cyl  -0.403         0.252        -0.117  0.224  0.159 -0.810  0.163
disp -0.397                -0.339  0.487         0.182        -0.662
hp   -0.367  0.269                0.295  0.354 -0.696  0.166  0.252
drat  0.312  0.342 -0.150 -0.846 -0.162                -0.135
wt   -0.373 -0.172 -0.454 -0.191  0.187         0.428  0.198  0.569
qsec  0.224 -0.484 -0.628         0.148  0.258 -0.276 -0.356 -0.169
gear  0.209  0.551 -0.207  0.282  0.562 -0.323        -0.316
carb -0.245  0.484 -0.464  0.214 -0.400  0.357  0.206  0.108 -0.320

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.111  0.111  0.111  0.111  0.111  0.111  0.111  0.111
Cumulative Var  0.111  0.222  0.333  0.444  0.556  0.667  0.778  0.889
               Comp.9
SS loadings     1.000
Proportion Var  0.111
Cumulative Var  1.000
```

Transformed components could be obtained using scores

```
d02 <- d01.pca $scores
head (d02)
```

What is the relationship between principal components and the original features?

## Singular Value Decomposition

- Based on a theorem in Linear Algebra, according to which a rectangular matrix can be decomposed into 3 matrices - an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V

$$A = \mathrm{U}\Sigma\mathrm{V}^T$$

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

```
A = as.matrix(data.frame(c(3,1,1), c(-1,3,1)))
A
```

The singular value decomposition of the matrix is computed using the svd() function.

```
A.svd <- svd(A)
A.svd
```

## Singular Value Decomposition Step-by-Step

SVD can be performed step-by-step with R by calculating $A^T\mathrm{A}$ and $\mathrm{A}A^T$ then finding the eigenvalues and eigenvectors of the matrices. However, it should be noted this is only for demonstration and not recommended in practice as the results can be slightly different than the output of the svd(). This is due to somewhat random changes in signs of the eigenvectors from the eigen() function as the eigenvectors can be scaled by −1

First, find $A^T\mathrm{A}$ and $\mathrm{A}A^T$

```
ATA <- t(A) %*% A
ATA
```

The V component of the singular value decomposition is then found by calculating the eigenvectors of the resultant $A^T\mathrm{A}$ matrix.

```
ATA.e <- eigen(ATA)
v.mat <- ATA.e$vectors
v.mat
```

Here we see the V matrix is the same as the output of the svd() but with some sign changes. These sign changes can happen, as mentioned earlier, as the eigenvector scaled by −1 is still the same eigenvector, just scaled. We will alter the signs of our calculated VV to match the output of the svd() function.

```
v.mat[,1:2] <- v.mat[,1:2] * -1
v.mat
```

The same routine is done for the $A^T A$ and $AA^T$ matrix.

```
AAT <- A %*% t(A)
AAT
```

The eigenvectors are again found for the computed AATAAT matrix.

```
AAT.e <- eigen(AAT)
u.mat <- AAT.e$vectors
u.mat
u.mat <- u.mat[,1:2]
```

As mentioned earlier, the singular values r are the square roots of the non-zero eigenvalues of the $A^T A$ and $AA^T$ matrices.

```
r <- sqrt(ATA.e$values)
r <- r * diag(length(r))[,1:2]
r
```

Our answers align with the output of the svd() function. We can also show that the matrix AA is indeed equal to the components resulting from singular value decomposition.

```
svd.matrix <- u.mat %*% r %*% t(v.mat)
svd.matrix
```

# Exercise

- How you can use SVD for dimensionality reduction?
- Apply SVD on 'mtcars' data and compare the result with PCA.