Multivariate Analysis Lecture 2: Matrix Operations

Zhaoxia Yu
Professor, Department of Statistics

2025-04-02

Subsection 1

Matrix Multiplication

Definition of Matrix Multiplication

- Consider matrix $A_{m\times n}$ and matrix $B_{n\times p}$.
- How does AB look like?
- AB has m rows and p columns
- What is the value in the ith row and jth column of AB? We often denote it as the (i,j)th element/entry of AB:

$$(AB)_{i,j} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Definition of Matrix Multiplication

If $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is an $n \times p$ matrix,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

the *matrix product* $\mathbf{C} = \mathbf{AB}$ (denoted without multiplication signs or dots) is defined to be the $m \times p$ matrix

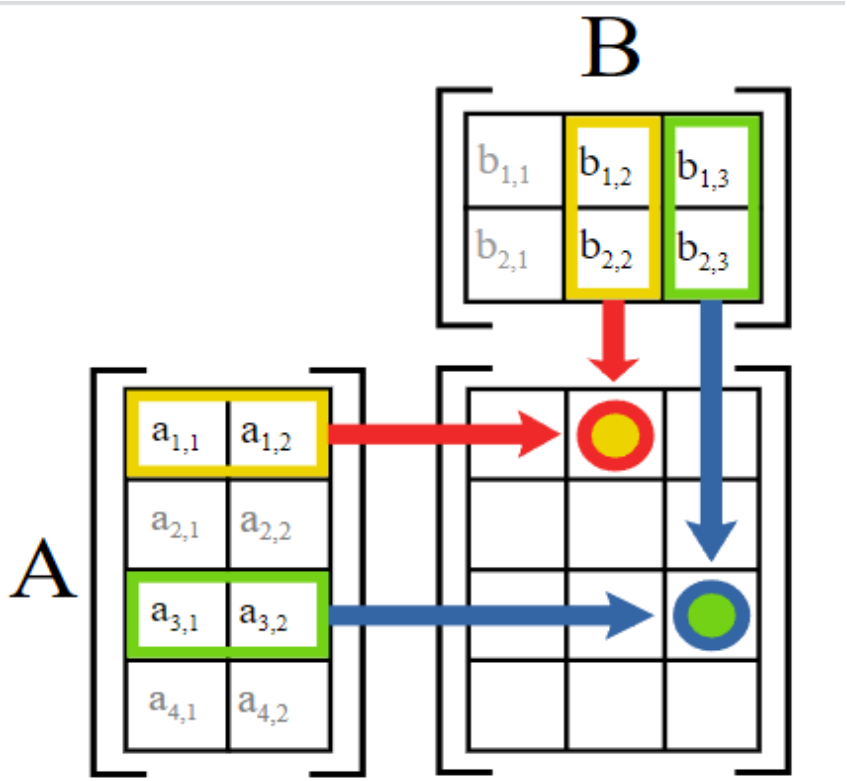
$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

such that

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj},$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, p$.

Matrix Multiplication



Conformable Matrices

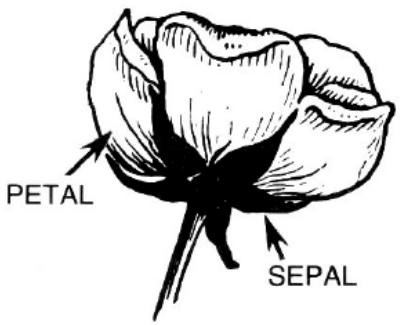
- Conformability refers to whether two matrices can be multiplied.
- Two matrices can be multiplied if they are conformable.
- E.g., if $dim(A) = n \times 10, dim(B) = 10 \times p$, then A and B are conformable because we can compute AB.
- Note: If A and B is conformable for calculating AB, it doesn't guarantee that they are also conformable for calculating BA.
- conformable for multiplication, then the operation is not defined.

Subsection 2

The Iris Data

Iris data

- Three species of iris flowers: setosa, versicolor, and virginica
- Four features: sepal length, sepal width, petal length, and petal width



Iris setosa

Iris versicolor

Iris virginica

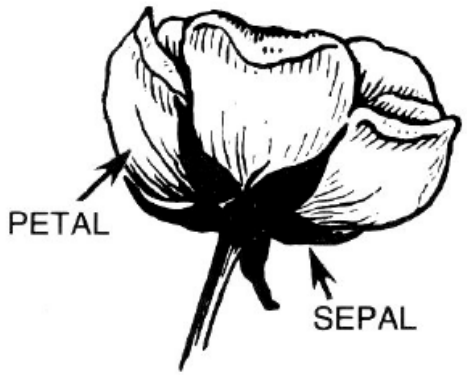
Introduction

- The iris dataset is a widely used dataset to illustrate various methods, algorithms, and problems.
- Iris dataset serves as an excellent example of how data can be used to gain insights into different fields.
- It is available in many places
  - It is available from UCI Machine Learning Repository.
  - It is also included as a built-in data set in R. Use "data()" to see the list of built-in data sets.
  - Wikipedia has a page for the dataset.
- Its simplicity and easy availability have made it a widely used dataset for educational and research purposes.

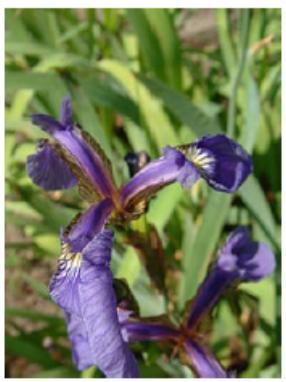
History

- The iris dataset has a rich history and has been used in many different areas of research
- It was first introduced by Ronald Fisher in 1936.
- The dataset contains measurements of four features of three species of iris flowers.
- Fisher used the iris dataset as an example of discriminant analysis

Iris data



Iris setosa



Iris versicolor

Iris virginica

Iris data in R

- The iris data is stored in two different formats:
  - as a 3D array: iris3
  - as a long matrix: iris

```
#?iris
dim(iris3)
```

```
## [1] 50  4  3
```

```
dim(iris)
```

```
## [1] 150   5
```

Iris data in R

```
dimnames(iris3)
```

```
## [[1]]
## NULL
##
## [[2]]
## [1] "Sepal L." "Sepal W." "Petal L." "Petal W."
##
## [[3]]
## [1] "Setosa"     "Versicolor" "Virginica"
```

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

Iris data in R

```
#the attributes is a useful function to understand data
attributes(iris3)
```

```
## $dim
## [1] 50  4  3
##
## $dimnames
## $dimnames[[1]]
## NULL
##
## $dimnames[[2]]
## [1] "Sepal L." "Sepal W." "Petal L." "Petal W."
##
## $dimnames[[3]]
## [1] "Setosa"     "Versicolor" "Virginica"
```

Iris data in R

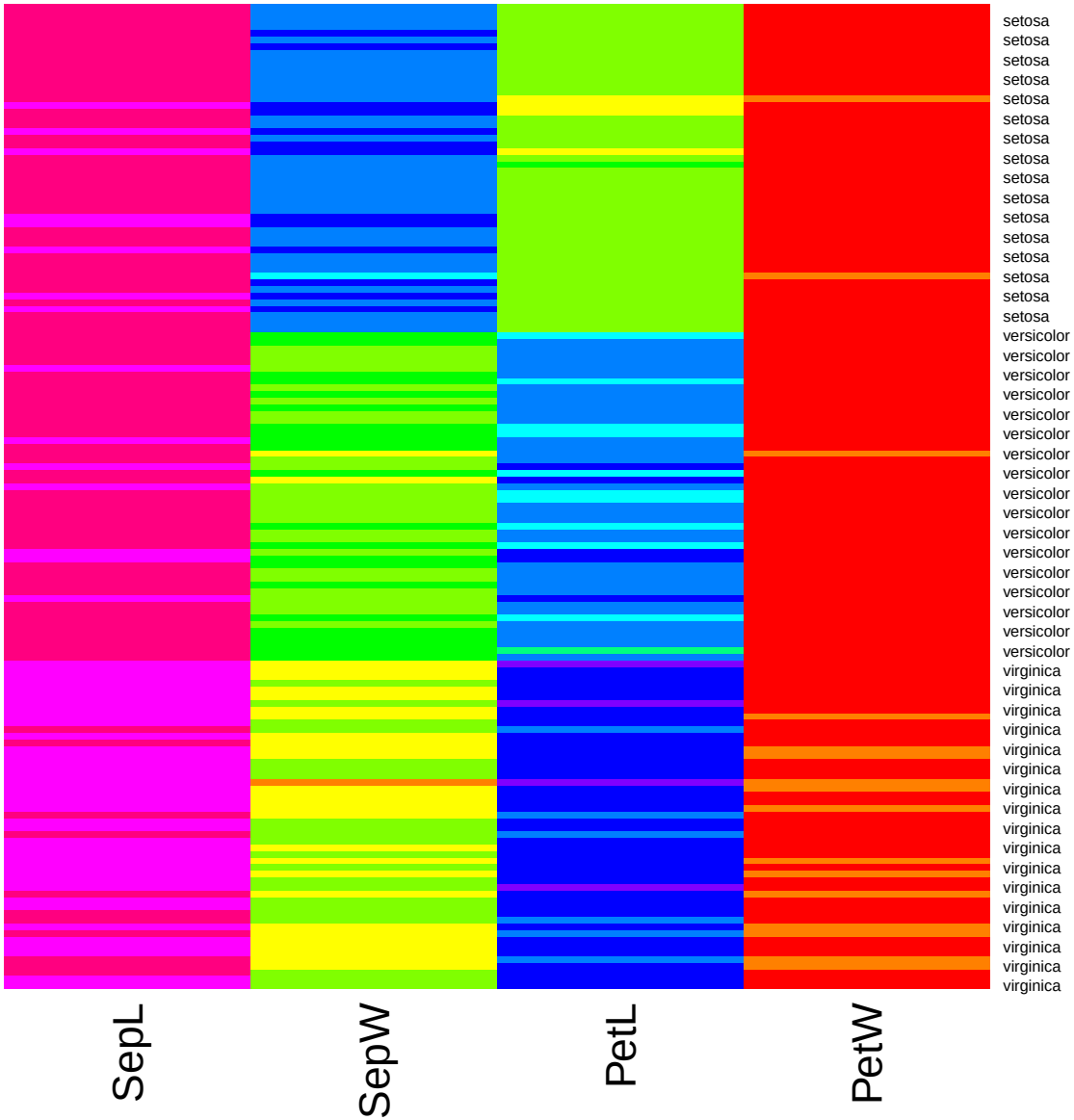
```r
#the attributes is a useful function to understand data
attributes(iris)
```

```
## $names
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
##
## $class
## [1] "data.frame"
##
## $row.names
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
## [145] 145 146 147 148 149 150
```

Heatmap of the Iris Data

```
Y=as.matrix(iris[, 1:4])
rownames(Y)=iris[,5]
colnames(Y)=c("SepL", "SepW", "PetL", "PetW")
heatmap(Y[150:1, ],col=rainbow(12),Rowv=NA, Colv=NA)
```

Heatmap of the Iris Data



Left multiply a matrix to Y: Example 1

- Consider

$$A = \frac{1}{150}\mathbf{1}_{1\times 150},$$

  i.e.,

$$A_{1\times 150} = (\frac{1}{150}, \cdots, \frac{1}{150})$$

- What is AY?
- In R, the correct syntax for matrix multiplication is

%*%

Left multiply a matrix to Y: Example 1

```
A=matrix(1/150, 1, 150)
A%*%Y
```

```
##           SepL     SepW  PetL     PetW
## [1,] 5.843333 3.057333 3.758 1.199333
```

```
dim(A%*%Y)
```

```
## [1] 1 4
```

Left multiply a matrix to Y: Example 1

- Let's check the results

```
A%*%Y
```

```
##          SepL     SepW  PetL     PetW
## [1,] 5.843333 3.057333 3.758 1.199333
```

```
colMeans(Y)
```

```
##     SepL     SepW     PetL     PetW
## 5.843333 3.057333 3.758000 1.199333
```

Left multiply a matrix to Y: Example 2

- Let

$$B_{3\times 150} = \frac{1}{50}\begin{pmatrix} \mathbf{1}_{1\times 50} & -\mathbf{1}_{1\times 50} & \mathbf{0}_{1\times 50} \\ \mathbf{1}_{1\times 50} & \mathbf{0}_{1\times 50} & -\mathbf{1}_{1\times 50} \\ \mathbf{0}_{1\times 50} & \mathbf{1}_{1\times 50} & -\mathbf{1}_{1\times 50} \end{pmatrix}$$

  i.e.,

$$B_{3\times 150} = \begin{pmatrix} \frac{1}{50} & \cdots & \frac{1}{50} & -\frac{1}{50} & \cdots & -\frac{1}{50} & 0 & \cdots & 0 \\ \frac{1}{50} & \cdots & \frac{1}{50} & 0 & \cdots & 0 & -\frac{1}{50} & \cdots & -\frac{1}{50} \\ 0 & \cdots & 0 & \frac{1}{50} & \cdots & \frac{1}{50} & -\frac{1}{50} & \cdots & -\frac{1}{50} \end{pmatrix}$$

- What does BY give us?

Left multiply a matrix to Y: example2

- $B_{3\times 150} Y_{150\times 4}$ gives the pairwise difference in group means for each feature

```
B=1/50*rbind(
  rep(c(1,-1, 0), each=50),
  rep(c(1, 0,-1), each=50),
  rep(c(0, 1,-1), each=50))
B%*%Y
```

```
##        SepL   SepW   PetL  PetW
## [1,] -0.930  0.658 -2.798 -1.08
## [2,] -1.582  0.454 -4.090 -1.78
## [3,] -0.652 -0.204 -1.292 -0.70
```

Left multiply a matrix to Y: example2

- Let's check results

```
#colMeans(iris[iris$Species=="setosa",1:4])
colMeans(Y[1:50,])
```

```
##  SepL  SepW  PetL  PetW
## 5.006 3.428 1.462 0.246
```

```
colMeans(Y[51:100,])
```

```
##  SepL  SepW  PetL  PetW
## 5.936 2.770 4.260 1.326
```

```
colMeans(Y[100:150,])
```

```
##     SepL     SepW     PetL     PetW
## 6.570588 2.970588 5.523529 2.011765
```

Right multiply a matrix to Y: convert cm to inch

- The lengths in the iris data were measured using cm
- 1cm = 0.393701inch. How to convert data to inch?
- Let

$$C = \begin{pmatrix} 0.393701 & 0 & 0 & 0 \\ 0 & 0.393701 & 0 & 0 \\ 0 & 0 & 0.393701 & 0 \\ 0 & 0 & 0 & 0.393701 \end{pmatrix}$$

- $Y_{150\times 4} C_{4\times 4}$ is the data in inch.

Right multiply a matrix to Y: convert cm to inch

```
C=diag(0.393701, 4, 4)
Y%*%C
```

```
##                [,1]      [,2]      [,3]      [,4]
## setosa     2.007875 1.3779535 0.5511814 0.0787402
## setosa     1.929135 1.1811030 0.5511814 0.0787402
## setosa     1.850395 1.2598432 0.5118113 0.0787402
## setosa     1.811025 1.2204731 0.5905515 0.0787402
## setosa     1.968505 1.4173236 0.5511814 0.0787402
## setosa     2.125985 1.5354339 0.6692917 0.1574804
## setosa     1.811025 1.3385834 0.5511814 0.1181103
## setosa     1.968505 1.3385834 0.5905515 0.0787402
## setosa     1.732284 1.1417329 0.5511814 0.0787402
## setosa     1.929135 1.2204731 0.5905515 0.0393701
## setosa     2.125985 1.4566937 0.5905515 0.0787402
## setosa     1.889765 1.3385834 0.6299216 0.0787402
## setosa     1.889765 1.1811030 0.5511814 0.0393701
## setosa     1.692914 1.1811030 0.4330711 0.0393701
## setosa     2.283466 1.5748040 0.4724412 0.0787402
## setosa     2.244096 1.7322844 0.5905515 0.1574804
## setosa     2.125985 1.5354339 0.5118113 0.1574804
## setosa     2.007875 1.3779535 0.5511814 0.1181103
## setosa     2.244096 1.4960638 0.6692917 0.1181103
## setosa     2.007875 1.4960638 0.5905515 0.1181103
## setosa     2.125985 1.3385834 0.6692917 0.0787402
## setosa     2.007875 1.4566937 0.5905515 0.1574804
## setosa     1.811025 1.4173236 0.3937010 0.0787402
## setosa     2.007875 1.2992133 0.6692917 0.1968505
## setosa     1.889765 1.3385834 0.7480319 0.0787402
## setosa     1.968505 1.1811030 0.6299216 0.0787402
```

A homework problem:

- Find a matrix A such that AY gives the difference of means vectors between iris setosa and iris versicolor
- Find a matrix B such that YB is column-standardized, i.e., the standard deviation of each column/feature is 1.
- Check the following
  - Let $C = \mathbf{I}_{150} - \frac{1}{150} J$, where $J_{150\times 150}$ is an all-ones matrices . Use R to verify that CY centers each column/feature. The R code for C is "

```
C=diag(1,150) - \frac{150}matrix(1, 150, 150)
```

  - Let S be the sample covariance matrix. Use R to verify that each column of $CYS^{-1/2}$ has been centered and standardized. Hints:

```
S=cov(Y)
```

To compute $S^{-1/2}$, you may need a R package, such as "qtl2pleio".

Subsection 3

Spectral Decomposition

Spectral Decomposition

- Generate 1000 data points from a multivariation normal with mean $\mathbf{0}$ and covariance matrix

$$\Sigma = \begin{pmatrix} 0.8125000 & 0.3247595 \\ 0.3247595 & 0.4375000 \end{pmatrix}$$

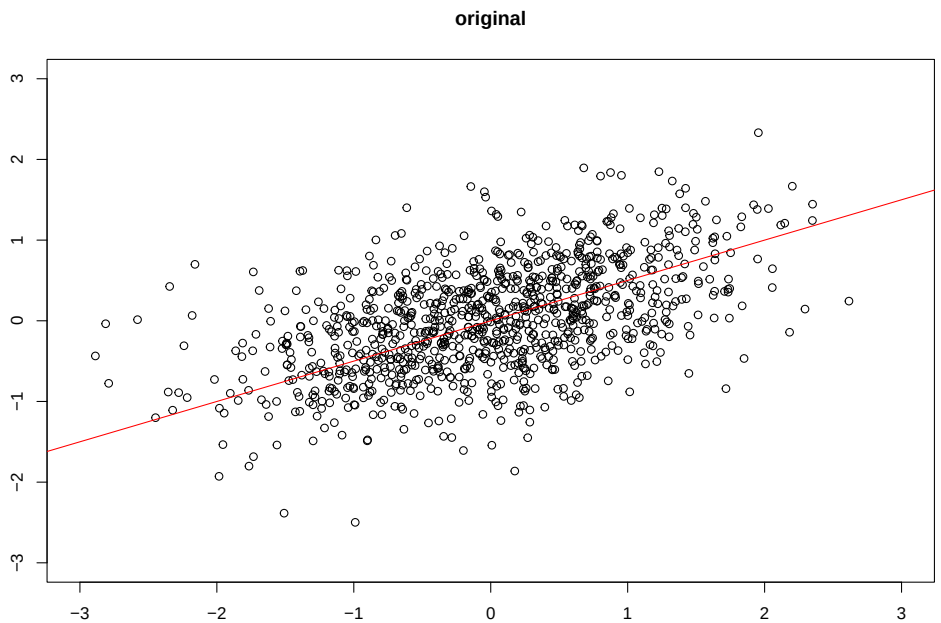
- How did I obtain this covariance matrix?

```
theta=-pi/6
R=matrix(c(cos(theta), -sin(theta), sin(theta), cos(theta)), 2, 2)
Lambda=diag(c(1,0.25))
Sigma=R%*%Lambda%*%t(R)
X=mvrnorm(1000, mu=rep(0,2), Sigma=Sigma)
```

Spectral Decomposition

- Scatter plot of the data X

```
plot(X, xlim=c(-3,3), ylim=c(-3,3), xlab="", ylab="", main="original")
abline(a=0, b=1/2, col="red")
```



Spectral Decomposition

- Conduct spectral decomposition of $cov(X)$.

```
R.obs=eigen(cov(X))$vectors
print(R.obs)
```

```
##            [,1]       [,2]
## [1,] -0.8714871  0.4904185
## [2,] -0.4904185 -0.8714871
```

```
Lambda.obs=diag(eigen(cov(X))$values)
print(Lambda.obs)
```

```
##          [,1]      [,2]
## [1,] 1.005821 0.0000000
## [2,] 0.000000 0.2706514
```

Spectral Decomposition

- Verify the spectral decomposition of $cov(X)$.

```
print(cov(X))
```

```
##           [,1]      [,2]
## [1,] 0.8290054 0.3142068
## [2,] 0.3142068 0.4474673
```

```
print(R.obs%*% Lambda.obs %*% t(R.obs))
```

```
##           [,1]      [,2]
## [1,] 0.8290054 0.3142068
## [2,] 0.3142068 0.4474673
```

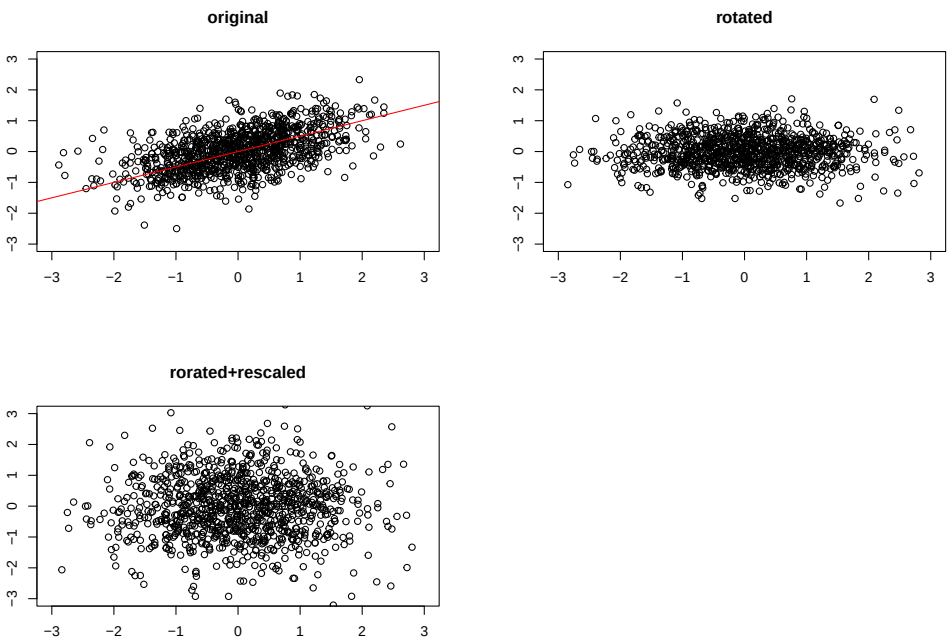
Spectral Decomposition

- standardizing X is equivalent to two steps:
- 1. rotate the n-by-2 data matrix X by multiplying R.
- 2. scale Y by multipling Lambdaˆ{-1/2} to obtain Z

```
Y=X%*%R.obs
Z=Y%*%diag(1/sqrt(diag(Lambda.obs)))
#note: for each 2-by-1 vector xi, the rotated vector yi is given by
#yi=R.obsˆT xi
```

Spectral Decomposition

```
par(mfrow=c(2,2))
plot(X, xlim=c(-3,3), ylim=c(-3,3), xlab="", ylab="", main="original")
abline(a=0, b=1/2, col="red")
plot(Y, xlim=c(-3,3), ylim=c(-3,3), xlab="", ylab="", main="rotated")
plot(Z, xlim=c(-3,3), ylim=c(-3,3), xlab="", ylab="", main="rorated+rescaled")
```



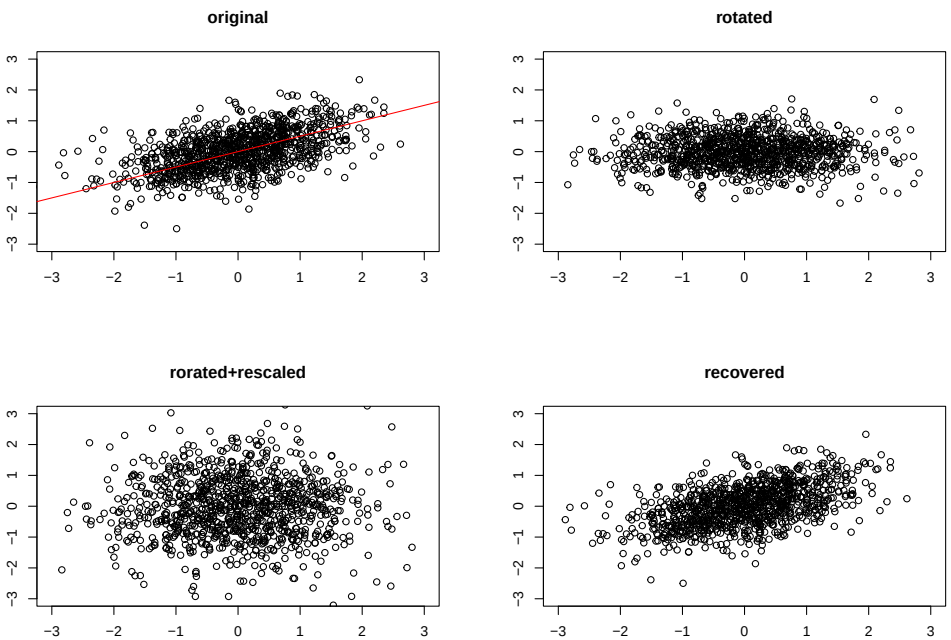
Spectral Decomposition

- Let's reverse the process
- 1. rescale Z by multiplying Lambda.obs^{1/2} to obtain Y
- 2. rotate Y by multiplying R^T to obtain X

```
Y=Z%*%sqrt(Lambda.obs)
X.recovered=Y%*%t(R.obs)
```

Spectral Decomposition

```
par(mfrow=c(2,2))
plot(X, xlim=c(-3,3), ylim=c(-3,3), xlab="", ylab="", main="original")
abline(a=0, b=1/2, col="red")
plot(Y, xlim=c(-3,3), ylim=c(-3,3), xlab="", ylab="", main="rotated")
plot(Z, xlim=c(-3,3), ylim=c(-3,3), xlab="", ylab="", main="rorated+rescaled")
plot(X.recovered, xlim=c(-3,3), ylim=c(-3,3), xlab="", ylab="", main="recovered")
```



```
#SVD of a matrix X is given by X=UDV^T
#where U cand V
```

Subsection 4

Singular Value Decomposition (SVD)

Introduction to SVD

- SVD was developed by a number of researchers independently over time. Key contributors
  - Eugenio Beltrami (1873): an Italian mathematician, first introduced the concept of singular value decomposition in his work on mathematical physics, although the modern formulation of SVD was not fully realized at that time.
  - Camille Jordan (1874), a French mathematician, independently discovered it.
  - . . . Many other contributors . . .
  - Gene H. Golub and Charles F. Van Loan (1965): American mathematicians, are credited with the development of the modern and widely used algorithm for computing the singular value decomposition, known as the Golub-Reinsch algorithm.

What is SVD?

- SVD is a useful factorization of a rectangular matrix A into three matrices
- If A is an $n \times p$ matrix of rank r,then A can be written as

$$A_{n\times p} = U_{n\times r} D_{r\times r} V^T_{r\times p}$$

where

- $U_{n\times r}$ and $V_{r\times p}$ are column orthogonal matrices, i.e.,

$$U^T U = V^T V = \mathbf{I}_r$$

- U is the left singular vectors matrix, contains the eigenvectors of AA^T (n x n matrix)
- V^T is right singular vectors matrix, contains the eigenvectors of $A^T A$ (p x p matrix)
- D is diagonal matrix of singular values, contains the square root of the eigenvalues of AA^T and $A^T A$

Intuition behind SVD

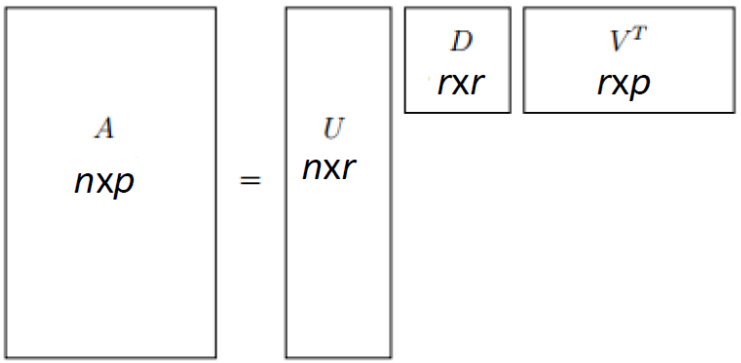
- SVD helps us understand the linear transformation of a matrix in terms of its components: rotation, scaling, and reflection.
- U and V^T represent rotations in the input and output spaces respectively, while D represents scaling along the axes.
- SVD allows us to find a lower-dimensional representation of the data by capturing the most important singular values and their corresponding singular vectors.

Applications of SVD

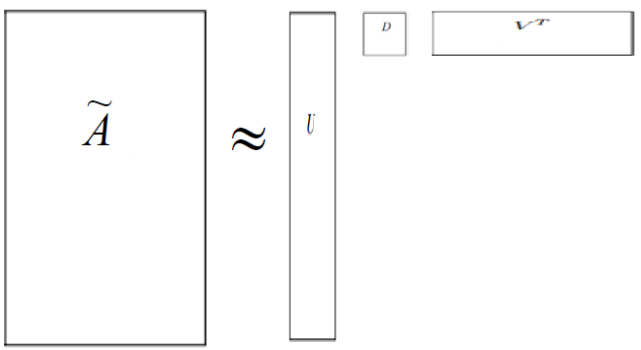
- SVD can be used for various applications such as image compression, recommendation systems, and data analysis.
  - Image Compression: SVD can be used to reduce the size of an image by compressing it while retaining its important features.
  - Recommendation Systems: SVD can be used to make personalized recommendations in recommendation systems by factorizing user-item interaction matrices.
  - Data Analysis: SVD can be used for dimension reduction, noise reduction, and feature extraction in data analysis tasks such as clustering, classification, and regression.

SVD for image compression

SVD



A low-rank approximation



SVD: Example 1

- The original image

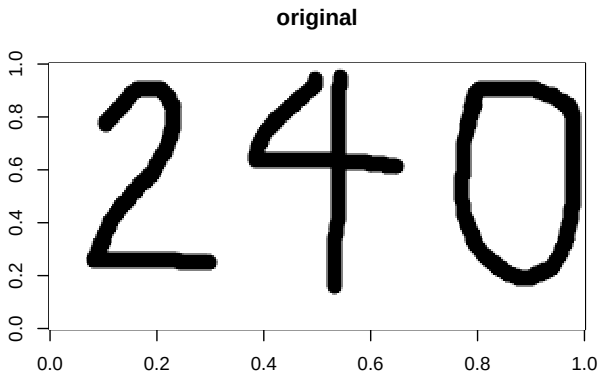
SVD: Example 1

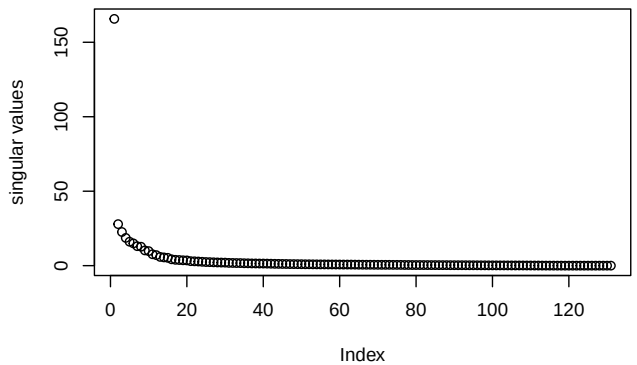
```
#install and load "jpeg"
library(jpeg)
mydata=readJPEG("img/svdexample.jpg", native = FALSE)
mydata=mydata[,,1] #279-by-131
obj=svd(mydata)
#obj$u: a 279-by-131 matrix
#obj$d: a vector of 131 nonnegative values
#obj$v: a 131-by-131 matrix
```

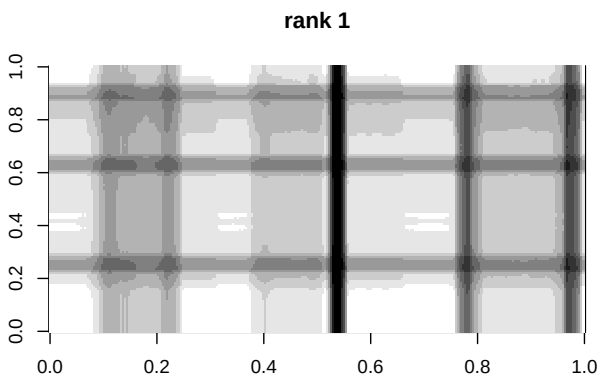
SVD Example 1: R

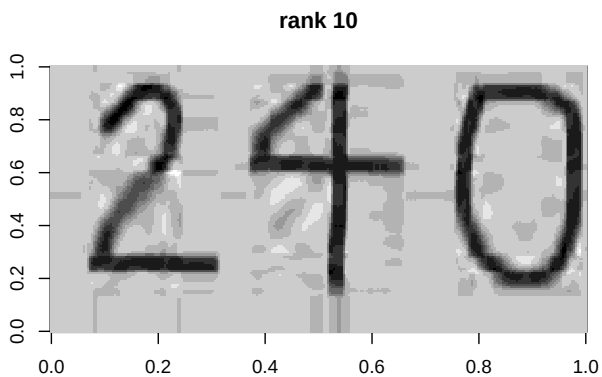
```r
par(mfrow=c(2,2))
#image(mydata,col=gray(c(0:10)/10))

image(t(mydata)[,nrow(mydata):1],col=gray(c(0:10)/10),
      xlab="", ylab="", main="original")

plot(obj$d, ylab="singular values")

n=1
mydata.approx=obj$u[,1] %*% t(obj$v[,1])*obj$d[1]
image(t(mydata.approx)[, nrow(mydata.approx):1], main="rank 1", col=gray(c(0:10)/10))

n=10
mydata.approx=obj$u[,1:n]%*% diag(obj$d[1:n]) %*% t(obj$v[,1:n])
image(t(mydata.approx)[, nrow(mydata.approx):1], main="rank 10", col=gray(c(0:10)/10))
```

SVD Example 1: R





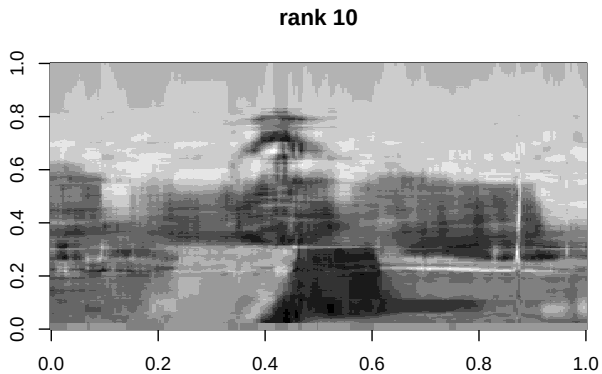


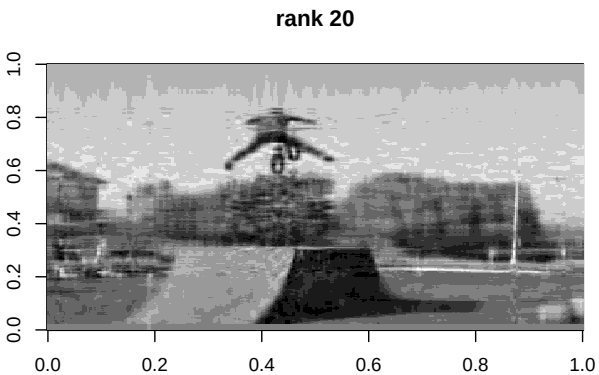


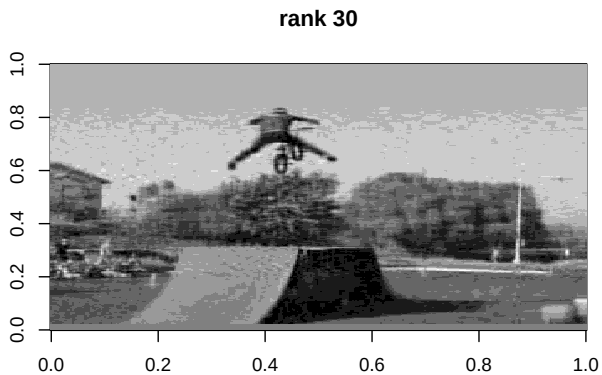
SVD: Example 2

```
BikeMan=readJPEG("BikeMan.jpg", native = FALSE)
BikeMan=BikeMan[,,1]

par(mfrow=c(2,2))
image(BikeMan, col=gray(c(0:10)/10))
obj.bm=svd(BikeMan)

n=10
BikeMan.approx=obj.bm$u[,1:n]%*% diag(obj.bm$d[1:n]) %*% t(obj.bm$v[,1:n])
image(BikeMan.approx, main="rank 10", col=gray(c(0:10)/10))

n=20
BikeMan.approx=obj.bm$u[,1:n]%*% diag(obj.bm$d[1:n]) %*% t(obj.bm$v[,1:n])
image(BikeMan.approx, main="rank 20", col=gray(c(0:10)/10))

n=30
BikeMan.approx=obj.bm$u[,1:n]%*% diag(obj.bm$d[1:n]) %*% t(obj.bm$v[,1:n])
image(BikeMan.approx, main="rank 30", col=gray(c(0:10)/10))
```

SVD: Example 2







Subsection 5

Another Matrix Operation: Kronecker Product

Definition

- The Kronecker product, denoted by $\otimes$ is a mathematical operation that combines two matrices to create a larger matrix.

- According to wikipedia, "The Kronecker product is named after the German mathematician Leopold Kronecker (1823–1891), even though there is little evidence that he was the first to define and use it."
- Why is it useful? It provide compact Representation. Kronecker product allows for compact representation of large matrices by expressing them as a combination of smaller matrices.

Definition

Let

$$A_{m\times n} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}, B_{p\times q} = \begin{pmatrix} b_{11} & \cdots & b_{1q} \\ \cdots & \cdots & \cdots \\ b_{p1} & \cdots & b_{pq} \end{pmatrix}$$

Then,

$$A \bigotimes B = \begin{pmatrix} a_{11} * B & a_{12} * B & \ldots & a_{1n} * B \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} * B & a_{m2} * B & \ldots & a_{mn} * B \end{pmatrix}$$

Definition

more explicitly:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix}.$$

Subsection 6

Futher reading

Futher reading

- Chronecker product: Gerald S. Rogers. (1984) Kronecker Products in ANOVA-A First Step. The American Statistician, 38: 197-202.
- SVD: Biclustering via Sparse Singular Value Decomposition" https://www.unc.edu/~haipeng/publication/ssvd.pdf

Subsection 7

Homework 1

Problem 1

- Suppose X_1, X_2, Y_1, Y_2 are mutually independent.
  - X_1 and X_2 are iid from $N(\mu = 0, \sigma_x^2 = 2^2)$
  - Y_1 and Y_2 are iid from $N(\mu = 0, \sigma_y^2 = 1^2)$ Consider the two pairs (X_1, X_2) and (Y_1, Y_2). Which pair tends to have a larger difference? To answer the question, please calculate and estimate the following two probabilities:

$$P(|X_1 - X_2| > 4), P(|Y_1 - Y_2| > 4)$$

- The hints for calculating/estimating $P(|X_1 - X_2| > 4)$ can be found in the two slides. Using similar strategies, you can calculate/estimate $P(|Y_1 - Y_2| > 4)$

Calculate $P(|X_1 - X_2| > 4)$

- Hints for calculating $P(|X_1 - X_2| > 4)$.
  - First find the distribution of $X_1 - X_2$. Then standard it to have mean 0 and SD 1.
  - Second, express the probability to $P(|Z| > z)$, where $Z \sim N(0, 1)$.
  - Next, expression the probability in terms of $\Phi(\cdot)$, the CDF of the standard normal distribution.
  - Last, use the "pnorm" function in R to find the numerical value.

Estimate $P(|X_1 - X_2| > 4)$

- The probability can be estimated by doing simulations/sampling.
- If you sample many (say 10,000) pairs of X_1 and X_2, count how many pairs satisfying $|X_1 - X_2| > 4$. The probability can be used to estimate $P(|X_1 - X_2| > 4)$

Problem 2

- Find a matrix A such that AY gives the difference of mean vectors between iris setosa and iris versicolor
- Find a matrix B such that YB is column-standardized, i.e., the standard deviation of each column/feature is 1.
- Check the following
  - Let $C = \mathbf{I}_{150} - \frac{1}{150} J$, where $J_{150\times 150}$ is an all-ones matrices . Use R to verify that CY centers each column/feature. The R code for C is "

```
C=diag(1,150) - \frac{1}{150}matrix(1, 150, 150)
```

  - Let S be the sample covariance matrix. Use R to verify that each column of $CYS^{-1/2}$ has been centered and standardized (in fact, the columns have also been de-correlated). Hints:

```
S=cov(Y)
```

To compute $S^{-1/2}$, you may need an R package, such as "qtl2pleio".

Problem 3

- Choose a picture you like and conduct approximations using singular value decomposition (SVD).