Introduction to Matrix Algebra

Matrix and its components; square, symmetric, diagonal, transpose, identity matrix; trace; matrix arithmetic by element, matrix-vector product, matrix-matrix product, matrix inverse, projection matrix, orthogonal matrix

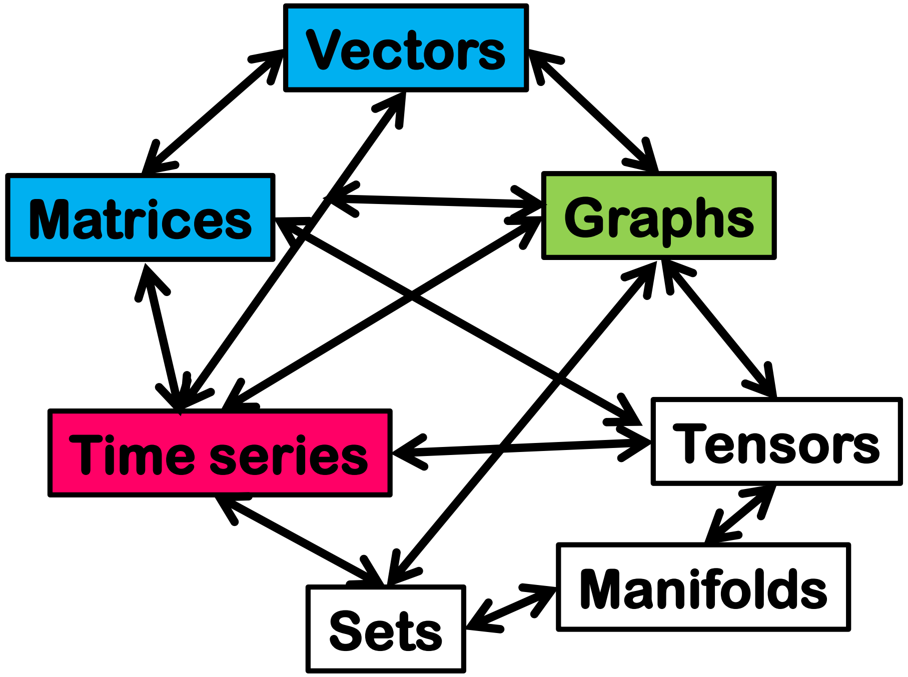
Nagiza F. Samatova, samatova@csc.ncsu.edu
Professor, Department of Computer Science
North Carolina State University

Matrix Algebra

MATRIX DATA MODEL IN DATA SCIENCE

Data Model in the Data Science (DS) Process

Application → Data → Questions → Data Objects + Features → Mathematical Data Representation (**Data Model**)



Different DS methods require different *abstractions for data representation*

Not one hat fits all

More than one models is needed

Models are related but often in a complementary way

Data Objects as Matrices

Example: A collection of text documents on the Web

Original Documents

D1:	Child Safety at Home
D2:	Infant & Toddler First Aid
D3:	Your Baby's Health and Safety: From Infant to Toddler

Parsed Documents

D1:	Child Safety Home
D2:	Infant Toddler
D3:	Bab Health Safety Infant Toddler

Terms=Features=Dimensions

T1:	Bab
T2:	Child
T3:	Health
T4:	Home
T5:	Infant
T6:	Safety
T7:	Toddler

t-d term-document matrix

	D1:	D2:	D3:
T1:	0	0	1
T2:	1	0	0
T3:	0	0	1
T4:	1	0	0
T5:	0	1	1
T6:	1	0	1
T7:	0	1	1

Mining such data ~ studying matrices

Vector & Matrix Algebra

VECTOR AND MATRIX OPERATORS IN PYTHON

Vector Operators in Python (x is a vector, A is a matrix)

Definition	Python Syntax	Example
To create x	x = xrange(x_n)	x = range(12)
To extract a diagonal from A as a vector	np.diag(A)	A = np.reshape(range(12),(3,4)) np.diag(A)
To find the length of x	len(x)	len(x) # 12
To access the i^{th} element in x	x$[i]$	x [3]

Vector Operators in R (x is a vector, A is a matrix)

Definition	R Syntax	Example
To create $\boldsymbol{x}$	$\boldsymbol{x} \leftarrow$ c $(x_1, x_2, \ldots, x_n)$	x <- c(1:12)
To extract a diagonal from $\boldsymbol{A}$ as a vector	diag (A)	A<-matrix(1:12, 3, 4) diag (A)
To find the length of $\boldsymbol{x}$	length (x)	length(x) # [1] 12
To access the i^{th} element in $\boldsymbol{x}$	x$[i]$	x [3]

Matrix Operators in Python (*A*, *B* are matrices)

Definition	Python Syntax	Example
To create *A* with np.reshape()	A = np.reshape(range(x_n), (rows, columns))	m=np.reshape(range(12),(3,4))
To create *A* from some other object	A = np.array([[values], [values]]) np.asmatrix(A)	m = np.array([[1, 2], [3, 4]]) np.asmatrix(m)
To create a diagonal matrix from the diagonals of *A*	np.diag(np.diag(A))	np.diag(np.diag(m))
Extract diagonal cells of A	np.diag(A, k) k>0 for diagonals above main, k<0 – below main diagonal	np.diag(m, 1)
To find the shape of *A*	A.shape	m.shape # (3, 4)
To find the number of rows in *A*	A.shape[0]	m.shape[0] # 3
To find the number of columns in *A*	A.shape[1]	m.shape[1] # 4

Matrix Operators in Python (A, B are matrices) (Cont.)

Definition	Python Syntax	Example
To access the $(i, j)^{th}$ elements in A	$A[i, j]$	m [2, 3]
To access the i^{th} row in A	$A[i,\ :]$ or $A[i,]$ or $A[i]$	m [2]
To access the j^{th} column in A	$A[:\ , j]$	m [:, 2]
To access a subset of rows in A	$A[i_1 : i_2, :]$ or $A[i_1 : i_2]$	m [1:3, :]
To access a subset of columns in A	$A[:\ , j_1 : j_2]$	m [:, 2:4]
To access a sub-matrix	$A[i_1 : i_2,\ j_1 : j_2]$	m [1:3, 1:4]

Matrix Operators in Python (*A*, *B* are matrices) (Cont.)

Definition	Python Syntax
Addition *A+B*, the same dimensions	A + B
Subtraction *A+B*, the same dimensions	A – B
Hadamard (element-by-element) multiplication, $A \odot B$	A * B
Multiplication $A \times B$, ncol (A) = nrow (B)	np.matmul(A,B)
Transpose A^T *or* A'	A.T
Inversion A^{-1}	np.linalg.inv(A)
Determinant	np.linalg.det(A)
Eigen-analysis of *A*	np.linalg.eig(A)
Trace of *A*	np.sum(np.diag(A))
To join vectors into *A* as columns	np.concatenate((A,B),axis=1)
To join vectors into *A* as rows	np.concatenate((A,B),axis=0)
To join *A* and *B* side-by-side (nrow(A) = nrow(B))	np.hstack((A, B))
To stack *A* and *B* on top of each other (ncol(A) = ncol(B))	np.hstack((A, B))

Matrix Operators in R (*A*, *B* are matrices)

Definition	R Syntax	Example
To create *A* with *m* rows and *n* cols	A ← matrix (data, nrow=m, ncol=n, byrow=F)	m<-matrix(1:12, 3, 4)
To create a diagonal matrix from	diag (c ($x_{11}, \dots, x_{nn}$)), diag (const, nrow, ncol)	diag(1, 3, 4)
To create a diagonal matrix from *A*	diag (diag (A))	diag (diag(m))
To create A from a dataframe *df*	data.matrix (df)	data.matrix (df)
To create A from some other object	as.matrix (object)	
To find the dimensions of *A*	dim (A)	dim(m) # [1] 3 4
To find the number of rows in *A*	nrow (A)	nrow(m) # 3
To find the number of columns in *A*	ncol (A)	ncol(m) # 4
To access the $(i, j)^{th}$ elements in *A*	$A[i, j]$	m [2, 3]
To access the i^{th} row in *A*	$A[i,]$	m [3,]
To access the j^{th} column in *A*	$A[, j]$	m [, 2]
To access a subset of rows in *A*	$A[i_1 : i_2,]$	m [2:3,]
To access a subset of columns in *A*	$A[, j_1 : j_2]$	m [, 3:4]
To access a sub-matrix	$A[i_1 : i_2, j_1 : j_2]$	m [2:3, 2:4]

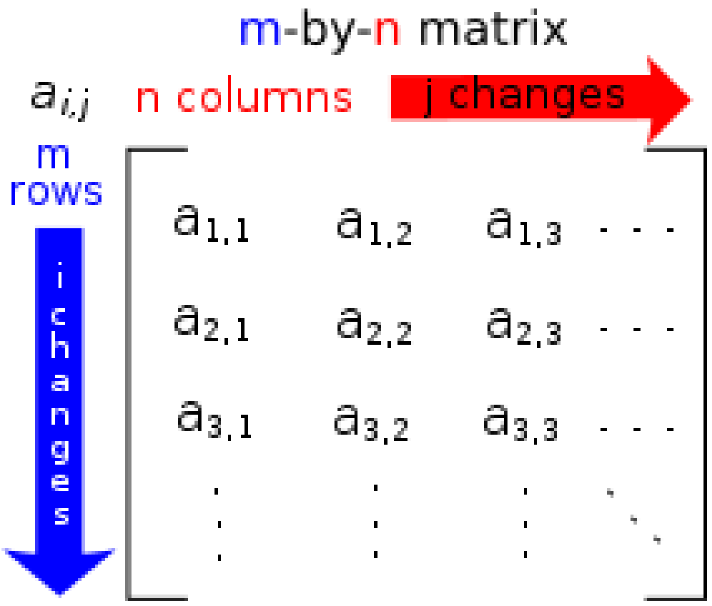
Matrix Operators in R (*A*, *B* are matrices) (Cont.)

Definition	R Syntax
Addition ***A+B***, the same dimensions	A + B
Subtraction ***A+B***, the same dimensions	A – B
Hadamard (element-by-element) multiplication, $A \odot B$	A * B
Multiplication $A \times B$, ncol (A) = nrow (B)	A %*% B
Transpose A^T *or* A'	t (A)
Inversion A^{-1}	solve (A)
Determinant $\det(A)$ or $\|A\|$	det (A)
Eigen-analysis of ***A***	eigen (A)
Trace of ***A***	sum (diag (A))
To join vectors into ***A*** as columns	cbind $(vec_1, vec_2, \ldots, vec_n)$
To join vectors into ***A*** as rows	rbind $(vec_1, vec_2, \ldots, vec_m)$
To join ***A*** and ***B*** side-by-side (nrow(A) = nrow(B))	cbind (A, B)
To stack ***A*** and ***B*** on top of each other (ncol(A) = ncol(B))	rbind (A, B)

Matrix Algebra

MATRIX DEFINITIONS AND TYPES

Matrix and Its Components



An $m \times n$ **matrix** A is a rectangular array of **scalar numbers**:

- *m* rows and *n* columns, or
- *m* by *n* matrix, or
- a matrix of **order** $m \times n$, or
- *A* has dimensions *m* and *n*

$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 4 & 6 \end{pmatrix}$ is a 2×3 matrix

The numbers a_{ij} are the **components** (or **elements**) of *A*.

A [**column**] vector is a matrix with one column, or an $m \times 1$ matrix
A [**row**] vector is an $1 \times n$ matrix
A **square** matrix: $m = n$

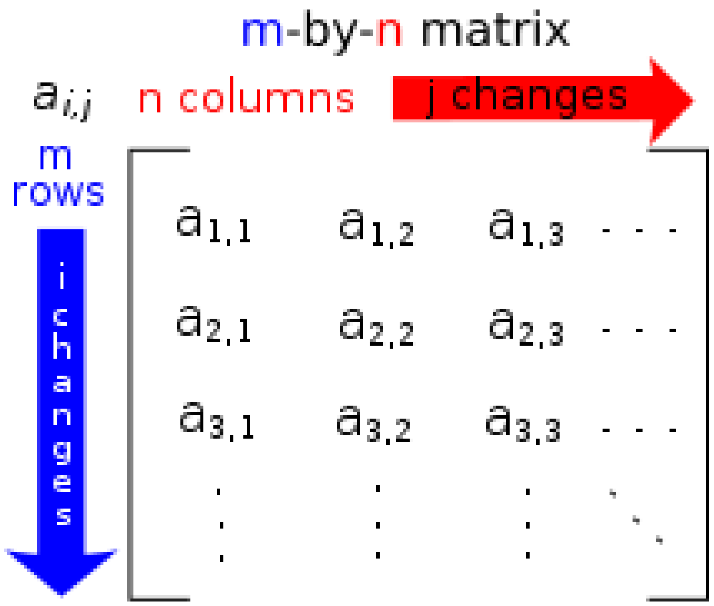
Example in R:

```r
A <- matrix (c (1,0,2,-6,3,0), nrow=2, ncol=3)
A
```

Example in Python:

```python
A = np.reshape([1,0,2,-6,3,0], (2,3))
A
```

Matrix Transpose, A^T or A'



$$\left(A^T\right)_{i,j} = A_{j,i}$$

$$\left(A^T\right)^T = A$$
$$(A')' = A$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -6 & 0 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 \\ 2 & -6 \\ 3 & 0 \end{bmatrix}$$

Example in R:

```
A <- matrix (c (1,0,2,-6,3,0), nrow=2, ncol=3)
A
B <- t (A)
B
t (B)
help (t)
```

Example in Python:

```
A = np.reshape([1,0,2,-6,3,0], (2,3))
B = A.T
B
B.T
help(np.transpose)
```

Ex #7: Transpose of the Matrix

Consider the following 3-by-2 matrix:

$$\mathbf{A}=\begin{bmatrix} 1 & -2 \\ 0 & 3 \\ 2 & -1 \end{bmatrix}$$

a. What is the value of $\boldsymbol{A}[2,1] = \quad ?$

b. Show its transpose matrix and validate with Python or R code: $\boldsymbol{A}^{\boldsymbol{T}} =$

c. What is the value of $\boldsymbol{A}^{\boldsymbol{T}}[1,2] = \quad ?$

d. If the matrix had m rows and n columns, then how many rows the transpose matrix will have?

Square Symmetric Matrix

A square matrix A is symmetric if:

- $A^T = A$ or
- $A' = A$ or
- $a_{ij} = a_{ji}$ for every element of A

Which matrix is symmetric?

1) $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

2) $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 4 & 6 \end{pmatrix}$

3) $A = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$

Python: Diagonal Matrix

A square matrix D_n with all elements NOT on the diagonal equal to zero (0) is diagonal <u>if</u>:

- $d_{ij} = d_{ji} = 0$ **for every (i, j) element of D_n, with $i \neq j$, AND**
- $d_{ii} \neq 0$ **for at least one value of i**

Which matrix is diagonal?

1) $A = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$

2) $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 6 \end{pmatrix}$

3) $A = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

Example in Python:

```
A = np.reshape(np.array([1,0,0,2]),(2,2))
print(A)
B = np.diag(np.array([1,0,0,2]))
print(B)
np.fill_diagonal(A,5)
print(A)
C = np.reshape(np.array([1,3,4,2]),(2,2))
print(C)
np.diag(np.diag(C))
```

R: Diagonal Matrix

A square matrix D_n with all elements NOT on the diagonal equal to zero (0) is diagonal if:

- $d_{ij} = d_{ji} = 0$ for every (i, j) element of D_n, with $i \neq j$, AND
- $d_{ii} \neq 0$ for at least one value of i

Which matrix is diagonal?

1) $A = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$

2) $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 6 \end{pmatrix}$

3) $A = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

Example in R:

```
A <- matrix (c (1,0,0,2), nrow=2, ncol=2)
A
B <- diag ( c (1,0,0,2) )
B
help (diag)
diag(A) <- 5
A
C <- matrix (c (1,3,4,2), nrow=2, ncol=2)
C
diag (diag (C) )
```

Identity Matrix

A diagonal matrix I_n with all the diagonal equal to one (1) is the identity matrix:

- $a_{ij} = a_{ji} = 0$ **for every (i, j) element of I_n, with $i \neq j$, AND**
- $a_{ii} = 1$ **for all values of i**

Which matrix is identical?

1) $A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

2) $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 6 \end{pmatrix}$

3) $A = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

Example in Python:

```
I = np.diag(np.array([1,1,1]))
I
```

Example in R:

```
I <- diag (c(1, 1, 1))
I
```

Trace of a Square Matrix

The trace of a square matrix A is the sum of all the diagonal elements of the matrix:

- $trace\,(A) = tr(A) = \sum_{i=1}^{n} a_{ii}$

What is the trace of the matrix?

1) $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

2) $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 4 & 6 \end{pmatrix}$

3) $A = \begin{pmatrix} 2 & 2 \\ 2 & 4 \end{pmatrix}$

Example in Python:

```
C = np.reshape(np.array([1,3,4,2]),(2,2))
C
trace = np.sum(np.diag(C))
trace
```

Example in R:

```
C <- matrix (c (1,3,4,2), nrow=2, ncol=2)
C
trace <-  sum ( diag (C) )
trace
```

Trace of the Transposed Matrix

$$\boxed{\boldsymbol{trace(A) = trace(A^T)}}$$

Matrix Algebra

MATRIX ARITHMETIC: ELEMENT-BY-ELEMENT

Matrix Operations: Element-by-Element

$$\boldsymbol{dim(A) = dim(B) = m \times n}$$

$$\boldsymbol{(\lambda \cdot A)_{ij} = \lambda \cdot A_{ij}}$$

Scalar Multiplications

$$\boldsymbol{(A \pm B)_{ij} = A_{ij} \pm B_{ij}}$$

Addition or Subtraction

$$\boldsymbol{(A \odot B)_{ij} = A_{ij} \cdot B_{ij}}$$

Hadamard Multiplication

$$\boldsymbol{(A/B)_{ij} = A_{ij}/B_{ij}}$$

Division

Matrix Operations: Element-by-Element

Example in Python:

```
A = np.reshape([54,49,49,41,26,43,49,50,58,71],(5,2))
B = np.reshape(range(1,11),(5,2))
print(A)
print(B)
```

Example in R:

```
A=matrix(c(54,49,49,41,26,43,49,50,58,71),nrow=5,ncol=2)
B=matrix(c(1:10), nrow=5, ncol=2)
A
B
```

```
2*A+3
A+B
A-B
A*B
A/B
```

To perform element-by-element operations

Examples: Element-by-Element Matrix Operations

$$\boldsymbol{A} = \begin{pmatrix} \mathbf{1} & \mathbf{2} \\ \mathbf{3} & \mathbf{4} \end{pmatrix} \qquad \boldsymbol{B} = \begin{pmatrix} \mathbf{5} & \mathbf{6} \\ \mathbf{7} & \mathbf{8} \end{pmatrix} \qquad \boldsymbol{\lambda} = \mathbf{2}$$

$$\boldsymbol{A} + \boldsymbol{B} = \begin{pmatrix} \mathbf{1} & \mathbf{2} \\ \mathbf{3} & \mathbf{4} \end{pmatrix} + \begin{pmatrix} \mathbf{5} & \mathbf{6} \\ \mathbf{7} & \mathbf{8} \end{pmatrix} = \begin{pmatrix} \mathbf{6} & \mathbf{8} \\ \mathbf{10} & \mathbf{12} \end{pmatrix} = \mathbf{2} \cdot \begin{pmatrix} \mathbf{3} & \mathbf{4} \\ \mathbf{5} & \mathbf{6} \end{pmatrix}$$

$$\boldsymbol{A} - \boldsymbol{B} = \begin{pmatrix} \mathbf{1} & \mathbf{2} \\ \mathbf{3} & \mathbf{4} \end{pmatrix} + \begin{pmatrix} \mathbf{5} & \mathbf{6} \\ \mathbf{7} & \mathbf{8} \end{pmatrix} = \begin{pmatrix} -\mathbf{4} & -\mathbf{4} \\ -\mathbf{4} & -\mathbf{4} \end{pmatrix}$$

$$\boldsymbol{A} \odot \boldsymbol{B} = \begin{pmatrix} \mathbf{1} & \mathbf{2} \\ \mathbf{3} & \mathbf{4} \end{pmatrix} + \begin{pmatrix} \mathbf{5} & \mathbf{6} \\ \mathbf{7} & \mathbf{8} \end{pmatrix} = \begin{pmatrix} \mathbf{5} & \mathbf{12} \\ \mathbf{21} & \mathbf{32} \end{pmatrix}$$

Matrix Algebra

MATRIX-VECTOR AND MATRIX-MATRIX PRODUCTS

The Inner Product of Two Vectors

$\boldsymbol{a}' = (a_1, a_2, \dots, a_n)$ **and** $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \dots \\ x_n \end{pmatrix}$

Inner Product

$$\boxed{\boldsymbol{a}'\boldsymbol{x} = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n}$$

<u>Problem</u>:

The company purchased the supplies for three products at a unit cost of \$3, \$1, and \$10, respectively. The total order size was 50, 100, and 30 units, resp.

What is the total cost for the order to the company?

<u>Solution</u> as a Vector Product:

$\boldsymbol{a}' = (\mathbf{3}, \mathbf{1}, \mathbf{10})$ **and** $\boldsymbol{x} = \begin{pmatrix} 50 \\ 100 \\ 30 \end{pmatrix}$

$$\begin{aligned} \boldsymbol{a}'\boldsymbol{x} &= (\mathbf{3}, \mathbf{1}, \mathbf{10}) \begin{pmatrix} 50 \\ 100 \\ 30 \end{pmatrix} \\ &= \mathbf{3}(\mathbf{50}) + \mathbf{1}(\mathbf{100}) + \mathbf{10}(\mathbf{30}) \\ &= \mathbf{550} \end{aligned}$$

Matrix-Vector Multiplication

$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$ is a matrix and $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ is a column vector

Matrix-vector multiplication:

$$Ax = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \end{pmatrix} \longleftarrow$$

by rows of A: inner product of a row of A with x

Problem:

The company purchased the supplies for three products at a unit cost of \$3, \$1, and \$10, respectively, in the first quarter and at a unit cost of \$2, \$2, and \$8, in the second quarter. The total order size was 50, 100, and 30 units, resp., in each quarter

What is the total cost for the order to the company for both quarters?

Solution as a Vector Product:

$A = \begin{pmatrix} 3 & 1 & 10 \\ 2 & 2 & 8 \end{pmatrix}$ and $x = \begin{pmatrix} 50 \\ 100 \\ 30 \end{pmatrix}$

$$Ax = \begin{pmatrix} 3 & 1 & 10 \\ 2 & 2 & 8 \end{pmatrix} \begin{pmatrix} 50 \\ 100 \\ 30 \end{pmatrix}$$
$$= \begin{pmatrix} 3(50) + 1(100) + 10(30) \\ 2(50) + 2(100) + 8(30) \end{pmatrix}$$
$$= \begin{pmatrix} 550 \\ 540 \end{pmatrix} = y$$

Total Cost: sum(y) = 550+540

Matrix-Matrix Multiplication

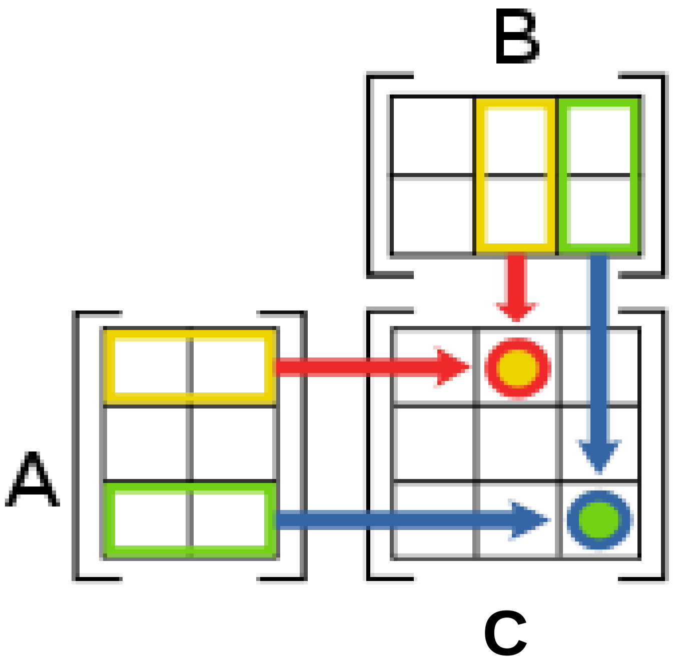
$\dim(A) = m \times n$

$C = A * B$

$\dim(B) = n \times k$

$\dim(C) = m \times k$

$$(A*B)_{i,j} = A_{i,1}B_{1,j} + A_{i,2}B_{2,j} + \ldots + A_{i,n}B_{n,j}$$
$$= \sum_{r=1}^{n} A_{i,r}B_{r,j}$$



$$1 * 1 + 0 * 1 + 2 * 0 = 1$$

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \end{bmatrix}.$$

2×3 $\quad$ 2×3 $\quad$ 3×2

In Python:

```
C = np.matmul(A,B)
```

In R:

```
C = A %*% B
```

Example: Matrix-Matrix Multiplication

Problem:
The company purchased the supplies for three products at a unit cost of \$3, \$1, and \$10, respectively, in the first quarter and at a unit cost of \$2, \$2, and \$8, in the second quarter.

The total order size for each quarter was 50, 100, and 30 units, resp., during the first year and 60, 80, and 40 units during the second year?

What is the total cost for the order to the company for two years?

Solution as a Matrix Product:

$$\boldsymbol{A} = \begin{pmatrix} \mathbf{3} & \mathbf{1} & \mathbf{10} \\ \mathbf{2} & \mathbf{2} & \mathbf{8} \end{pmatrix} \text{ and } \mathbf{B} = \begin{pmatrix} \mathbf{50} & \mathbf{60} \\ \mathbf{100} & \mathbf{80} \\ \mathbf{30} & \mathbf{40} \end{pmatrix}$$

$$\boldsymbol{AB} = \begin{pmatrix} \mathbf{3} & \mathbf{1} & \mathbf{10} \\ \mathbf{2} & \mathbf{2} & \mathbf{8} \end{pmatrix}\begin{pmatrix} \mathbf{50} & \mathbf{60} \\ \mathbf{100} & \mathbf{80} \\ \mathbf{30} & \mathbf{40} \end{pmatrix} = \begin{pmatrix} \mathbf{550} & \mathbf{660} \\ \mathbf{540} & \mathbf{600} \end{pmatrix} = \mathbf{C}$$

Total Cost: sum (*C*) = 550+540+660+600

Matrix Multiplication: Non-Commutative

$$A_{2\times 2} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad B_{2\times 2} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \qquad C_{2\times 2} = A \times B$$

$$\begin{aligned} A \times B &= \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \\ &= \begin{pmatrix} 1\times 5 + 2\times 7 & 1\times 6 + 2\times 8 \\ 3\times 5 + 4\times 7 & 3\times 6 + 4\times 8 \end{pmatrix} \\ &= \begin{pmatrix} 5 + 14 & 6 + 16 \\ 15 + 28 & 18 + 32 \end{pmatrix} \\ &= \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} B \times A &= \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \\ &= \begin{pmatrix} 5 + 18 & 10 + 24 \\ 7 + 24 & 14 + 32 \end{pmatrix} \\ &= \begin{pmatrix} 23 & 34 \\ 31 & 46 \end{pmatrix} \end{aligned}$$

$$\boxed{A \times B \neq B \times A}$$

Example: Multiplication of Rectangular Matrices

$$A_{2\times 3} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \qquad B_{3\times 2} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

$$A_{2\times \color{red}{3}} \times B_{\color{red}{3}\times 2} = C_{\color{blue}{2\times 2}} = \begin{pmatrix} 22 & 28 \\ 49 & 64 \end{pmatrix}$$

match

$$B_{3\times \color{red}{2}} \times A_{\color{red}{2}\times 3} = W_{\color{green}{3\times 3}} = \begin{pmatrix} 9 & 12 & 15 \\ 19 & 26 & 33 \\ 29 & 40 & 51 \end{pmatrix}$$

match

$$\boxed{A \times B \neq B \times A}$$

Multiplication with Identity Matrix: Commutative

Let I_m be the identity matrix of size $m \times m$ and I_n be the identity matrix of size $n \times n$

$$\boxed{A_{n\times m} \times I_m = I_n \times A_{n\times m} = A_{n\times m}}$$

$$A_{2\times 3} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \qquad I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **Verify that $A \times I = I \times A = A$**
- **Note: subscripts are not included if the context is clear**

Ex #8: Matrix-Matrix Multiplication

1. Generate in Python a 3-by-3 matrix A filled with one's everywhere but the diagonal; and with 3's on the diagonal.
2. Generate in Python a 3-by-2 matrix B filled with two's.

a. Multiply matrix A by matrix B and print the resulting matrix C:

$$C = AB =$$

b. What is the size of this new matrix C, i.e. number of rows and cols?

c. Can you multiply matrix B by A to get matrix D (why or why not):

$$\mathbf{D} = BA =$$

d. Show with manual calculations how you will get the value of:

$$C[2,2] =$$

Matrix Algebra

PROJECTION AND ORTHOGONAL MATRIX

Ex #9: Projection Matrix

1. Generate in Python a 4-by-2 matrix A and plot its 4 rows as points in 2-dim.
2. Generate a 2-by-2 matrix with diagonal elements as one's and off-diagonal matrix as zero's (aka ***identity*** matrix, I).

a. Multiply in Python matrix A by matrix I and print the resulting matrix C:

$$\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{I} =$$

b. Set the $\boldsymbol{I}[2,2]$ element to zero and assign the new matrix to P.

c. Multiply matrix A by the modified matrix I to get matrix D:

$$\mathbf{D} = \boldsymbol{A} \times \boldsymbol{P} =$$

d. Add the rows of matrix D as four points in 2-dim. to your original plot of A.
Do you observe that these new points are projections of the original points?
What axis are the points of A projected to in D?

e. How will you modify the identity matrix I to project the points of A on the other axis?

Orthogonal Matrix

A square $m \times m$ matrix A is orthogonal if:

$\boxed{A \times A^T = A^T \times A = I_m \text{ – the identity matrix}}$

Example: Verify (manually or using Python) that the following matrices are orthogonal:

$$A_{2\times 2} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \qquad B_{2\times 2} = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$$

You can choose any θ, e.g., $\theta = \frac{\pi}{2}, \frac{\pi}{4}, \frac{\pi}{3}$

Side Note: An orthogonal matrix whose elements are all $+1$ or -1 is known as Hadamard matrix; it has a role in statistical experimental design.

Matrix Algebra

INVERSES

Inverse of a Square Matrix, nrow(A)=ncol(A)=*n*

Identity Matrix, I_n

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix Inverse, A^{-1}

$$\boxed{A \times A^{-1} = A^{-1} \times A = I_n}$$

Examples: Verify (manually) the inverse of the matrix

$$A = \begin{pmatrix} 2 & 3 \\ 3 & 4 \end{pmatrix} \qquad A^{-1} = \begin{pmatrix} -4 & 3 \\ 3 & -2 \end{pmatrix}$$

$$A = \begin{pmatrix} 3 & 4 \\ 2 & 3 \end{pmatrix} \qquad A^{-1} = \begin{pmatrix} 3 & -2 \\ -4 & 3 \end{pmatrix}$$

Inverse of a Square Matrix, nrow(A)=ncol(A)=*n*

Identity Matrix, I_n

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix Inverse, A^{-1}

$$\boxed{A \times A^{-1} = A^{-1} \times A = I_n}$$

Example in R:

```
A <- matrix (rnorm(16), nrow=4,ncol=4)
options (digits=3)
A
IA <- solve (A)
IA
A %*% IA
IA %*% A
help (solve)
```

Example in Python:

```
A = np.random.rand(4,4)
np.set_printoptions(precision=3)
print(A)
IA = np.linalg.inv(A)
print(IA)
print(np.matmul(A,IA))
print(np.matmul(IA,A))
```

Inverse of Inverse

$$\boxed{\left(A^{-1}\right)^{-1} = A}$$

Example in Python:

```
A = np.reshape(np.array([2,3,3,4]),(2,2))
print(A)
IA = np.linalg.inv(A)
print(IA)
```

Example in R:

```
A <- matrix ( c(2, 3, 3, 4), nrow=2, ncol=2, byrow=F)
A
IA <- solve (A)
IA
solve (IA)
```

Python: Inverse of Matrix Product

$$\boxed{(\boldsymbol{A} \times \boldsymbol{B})^{-1} = \boldsymbol{B}^{-1} \times \boldsymbol{A}^{-1}}$$

Example in Python:

```
A = np.reshape([2,3,3,4],(2,2),order='F')
print(A)
B = np.reshape([3,4,2,3],(2,2),order='F')
print(B)
C = np.matmul(A,B)
print(C)
IC = np.linalg.inv(C)
print(IC)
np.matmul(np.linalg.inv(B),np.linalg.inv(A))
```

```
[[2 3]
 [3 4]]
[[3 2]
 [4 3]]
[[18 13]
 [25 18]]
[[-18.  13.]
 [ 25. -18.]]

array([[-18.,  13.],
       [ 25., -18.]])
```

R: Inverse of Matrix Product

$$\boxed{(A \times B)^{-1} = B^{-1} \times A^{-1}}$$

Example in R:

```
A <- matrix ( c(2, 3, 3, 4), nrow=2, ncol=2, byrow=F)
A
B <- matrix ( c(3, 4, 2, 3), nrow=2, ncol=2, byrow=F)
B
C <- A %*% B
C
IC <- solve (C)
IC
solve (B) %*% solve(A)
```

```
> C <- A %*% B
> C
     [,1] [,2]
[1,]   18   13
[2,]   25   18
>
> IC <- solve (C)
> IC
     [,1] [,2]
[1,]  -18   13
[2,]   25  -18
>
> solve (B) %*% solve(A)
     [,1] [,2]
[1,]  -18   13
[2,]   25  -18
```

Python: Inverse of Transpose

$$\boxed{\left(A^{T}\right)^{-1} = \left(A^{-1}\right)^{T}}$$

Example in Python:

```
A = np.reshape(np.array([2,3,3,4]),(2,2),order='F')
T = A.T
print(A)
print(T)
IT = np.linalg.inv(T)
print(IT)
IA = np.linalg.inv(A)
print(IA.T)
```

```
[[2 3]
 [3 4]]
[[2 3]
 [3 4]]
[[-4.  3.]
 [ 3. -2.]]
[[-4.  3.]
 [ 3. -2.]]
```

R: Inverse of Transpose

$$\boxed{\left(A^T\right)^{-1} = \left(A^{-1}\right)^T}$$

Example in R:

```
A <- matrix ( c(2, 3, 3, 4), nrow=2, ncol=2, byrow=F)
T <- t(A)
A
T
IT <- solve (T)
IT
IA <- solve (A)
t (IA)
help(t)
```

```
> A
     [,1] [,2]
[1,]    2    3
[2,]    3    4
> T
     [,1] [,2]
[1,]    2    3
[2,]    3    4
> IT <- solve (T)
> IT
     [,1] [,2]
[1,]   -4    3
[2,]    3   -2
> IA <- solve (A)
> t (IA)
     [,1] [,2]
[1,]   -4    3
[2,]    3   -2
```

Existence of the Inverse Matrix, det(A)≠0

Matrix Inverse, A^{-1}

$$\boxed{A \times A^{-1} = A^{-1} \times A = I_n}$$

$$\boxed{A^{-1} \textbf{ exists if and only if } det(A) \neq 0}$$

Determinant, *det(A)*, |A|

scalar: $\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$

2-by-2 Matrix Inverse

$$\boxed{A^{-1} = \frac{1}{\det(A)}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}}$$

***n-by-n* Matrix Inverse Methods:**

- LU-factorization
- Gaussian elimination
- Gauss-Jordan elimination

Existence of the Inverse Matrix, det(A)≠0

Matrix Inverse, A^{-1}

$$\boxed{A \times A^{-1} = A^{-1} \times A = I_n}$$

A^{-1} **exists if and only if** $det(A) \neq 0$

Example in Python:

```
A = np.random.rand(4,4)
print(A)
np.linalg.det(A)
```

Example in R:

```
A <- matrix (rnorm(16), nrow=4,ncol=4)
options (digits=3)
A
det (A)
help (det)
```

Matrix Algebra

TRANSPOSE AND TRACE OF MATRIX PRODUCT

Python: Transpose of the Matrix Product

$$\boxed{(A \times B)^T = B^T \times A^T}$$

Example in Python:

```
A = np.reshape(np.array([1,2,3,4,5,6]),(2,3))
B = np.reshape(np.array([1,2,3,4,5,6]),(3,2))
U = np.matmul(A,B)
print(U.T)
V = np.matmul(B.T, A.T)
print(V)
W = np.matmul(A.T, B.T)
print(W)
```

```
[[22 49]
 [28 64]]
[[22 49]
 [28 64]]
[[ 9 19 29]
 [12 26 40]
 [15 33 51]]
```

R: Transpose of the Matrix Product

$$\boxed{(A \times B)^T = B^T \times A^T}$$

Example in R:

```
A <- matrix (c(1, 2, 3, 4, 5, 6), nrow=2, ncol=3, byrow=T)
B <- matrix (c(1, 2, 3, 4, 5, 6), nrow=3, ncol=2, byrow=T)

U <-  A %*% B
t(U)
V <- t(B) %*% t(A)
V
W <- t(A) %*% t(B)
W
```

```
> U <-   A %*% B
> t(U)
     [,1] [,2]
[1,]   22   49
[2,]   28   64
> V <- t(B) %*% t(A)
> V
     [,1] [,2]
[1,]   22   49
[2,]   28   64
>
> W <- t(A) %*% t(B)
> W
     [,1] [,2] [,3]
[1,]    9   19   29
[2,]   12   26   40
[3,]   15   33   51
```

Trace of the Matrix Product

$$\boxed{\boldsymbol{trace}(\boldsymbol{A} \times \boldsymbol{B}) = \boldsymbol{trace}\ (\boldsymbol{B} \times \boldsymbol{A})}$$