Lecture *1* – Multivariate linear models

AI in Genetics

ZOO6927 / BOT6935 / ZOO4926

Vectors, matrices, and tensors

Matrix: Rectangular array of numbers:

$$A = \begin{bmatrix} 1402 & 191 \\ 1371 & 821 \\ 949 & 1437 \\ 147 & 1448 \end{bmatrix}$$

Dimension of matrix: number of rows x number of columns

Dim(A) = 4 x 2

Matrix Elements (entries of matrix)

$$A = \begin{bmatrix} 1402 & 191 \\ 1371 & 821 \\ 949 & 1437 \\ 147 & 1448 \end{bmatrix}$$

$A_{ij} =$ "$i\,,j$ entry" in the i^{th} row, j^{th} column.

Vector: An n x 1 matrix.

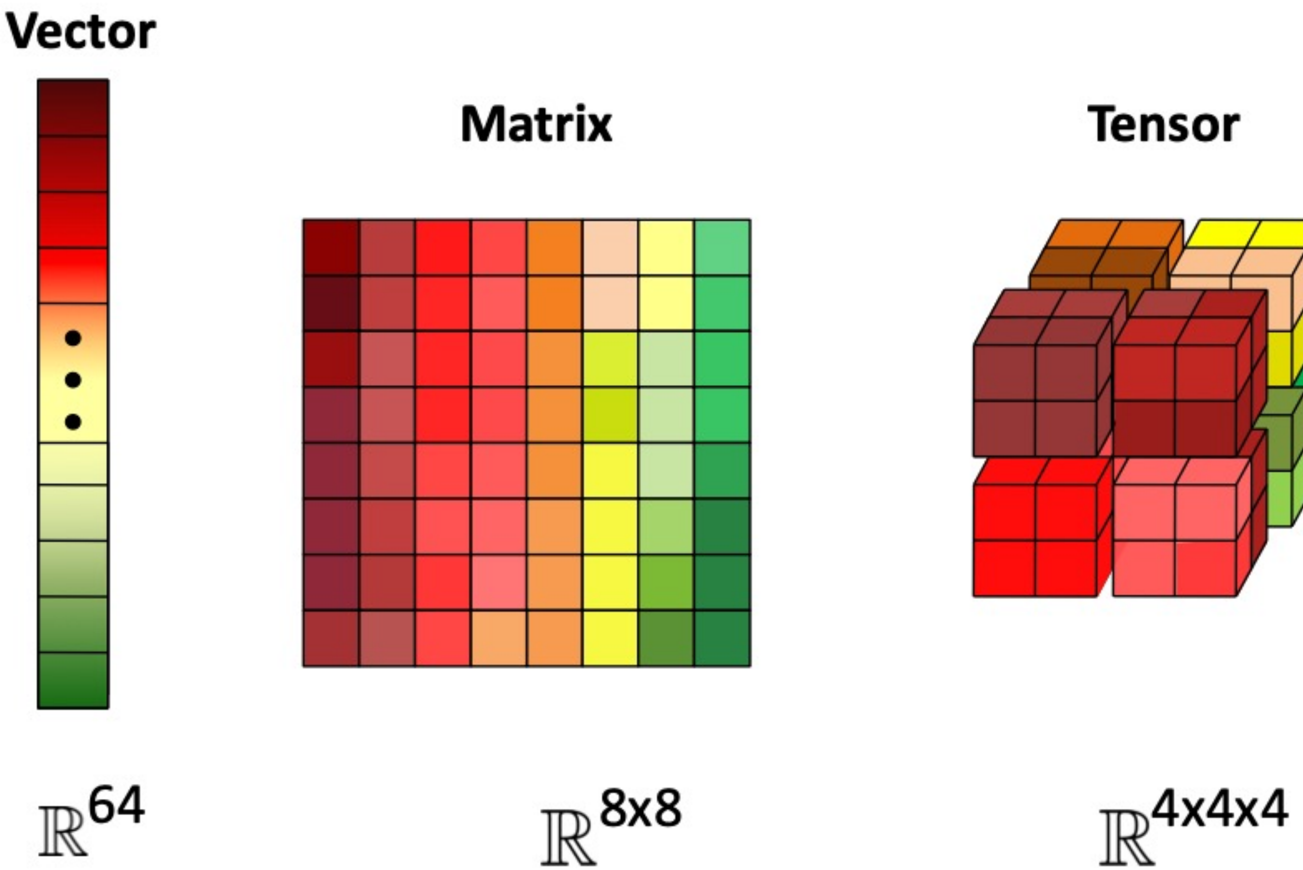
$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$y_i = i^{th}$ element

1-indexed vs 0-indexed:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \qquad y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Tensor: High-dimensional array



Basic matrix algebra

Scalar Multiplication

$$\text{If } A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}_{m \times n}$$

then for any scaler ‘k’

$$kA = \begin{bmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \vdots & \vdots & & \vdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{bmatrix}_{m \times n}$$

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

Matrix Addition is element-wise

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \text{ and } B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & a_{13} + b_{13} \\ a_{21} + b_{21} & a_{22} + b_{22} & a_{23} + b_{23} \\ a_{31} + b_{31} & a_{32} + b_{32} & a_{33} + b_{33} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \\ 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \end{bmatrix} =$$

Matrix transpose

$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}_{2 \times 3} \qquad A^{T} = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}_{3 \times 2}$$

$$(\mathbf{A}^{\top})_{ij} = A_{ji}$$

Transposing a column vector results in a row vector, and vice versa

$$V = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \rightarrow V^T = \begin{bmatrix} a & b & c \end{bmatrix}$$

Vector-Vector Product

"Inner product" or "dot product" $$\langle \mathbf{x}, \mathbf{y} \rangle \triangleq \mathbf{x}^\top \mathbf{y} = \sum_{i=1}^{n} x_i y_i .$$

$$\langle \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \begin{bmatrix} e \\ f \\ g \end{bmatrix} \rangle = [a \quad b \quad c] \cdot \begin{bmatrix} e \\ f \\ g \end{bmatrix} = a \cdot e + b \cdot f + c \cdot g$$

Matrix-vector multiplication

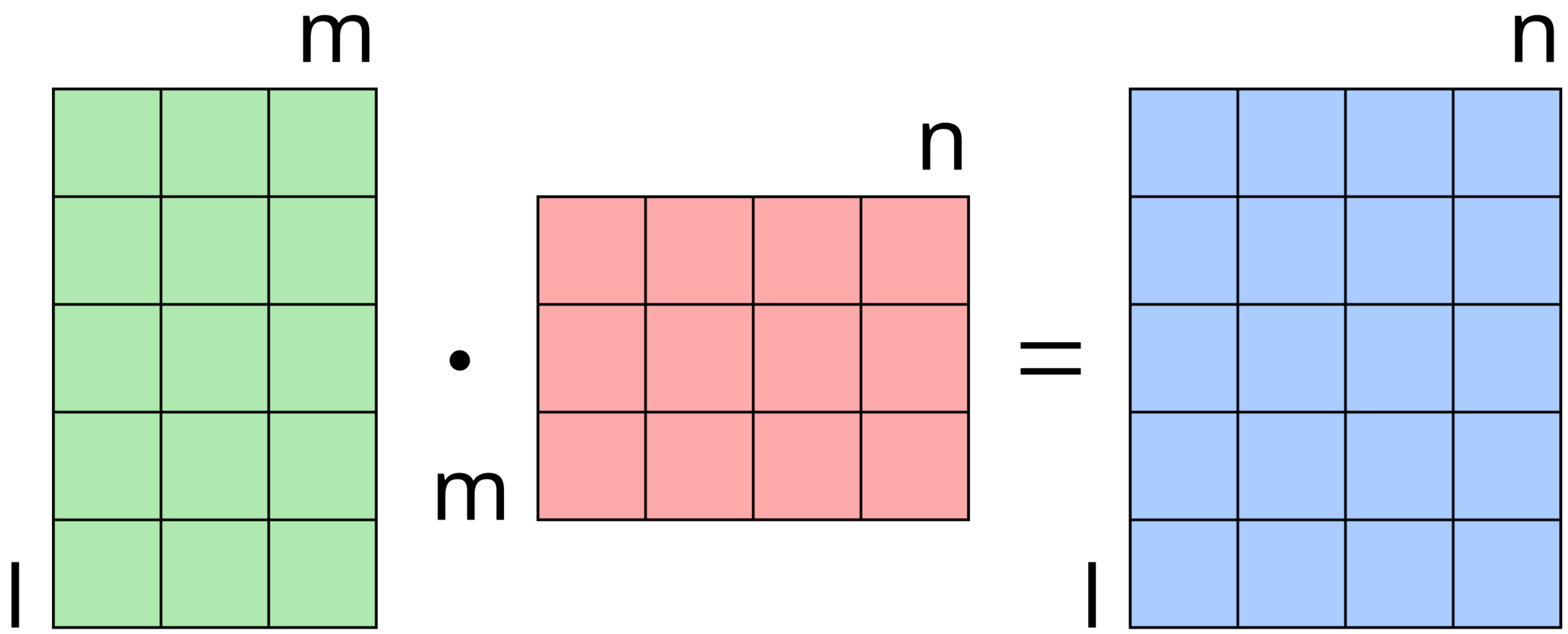
$$\begin{bmatrix} A & B \\ C & D \\ E & F \end{bmatrix} \times \begin{bmatrix} G \\ H \end{bmatrix} = \begin{bmatrix} A \times G + B \times H \\ C \times G + D \times H \\ E \times G + F \times H \end{bmatrix}$$

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

Example

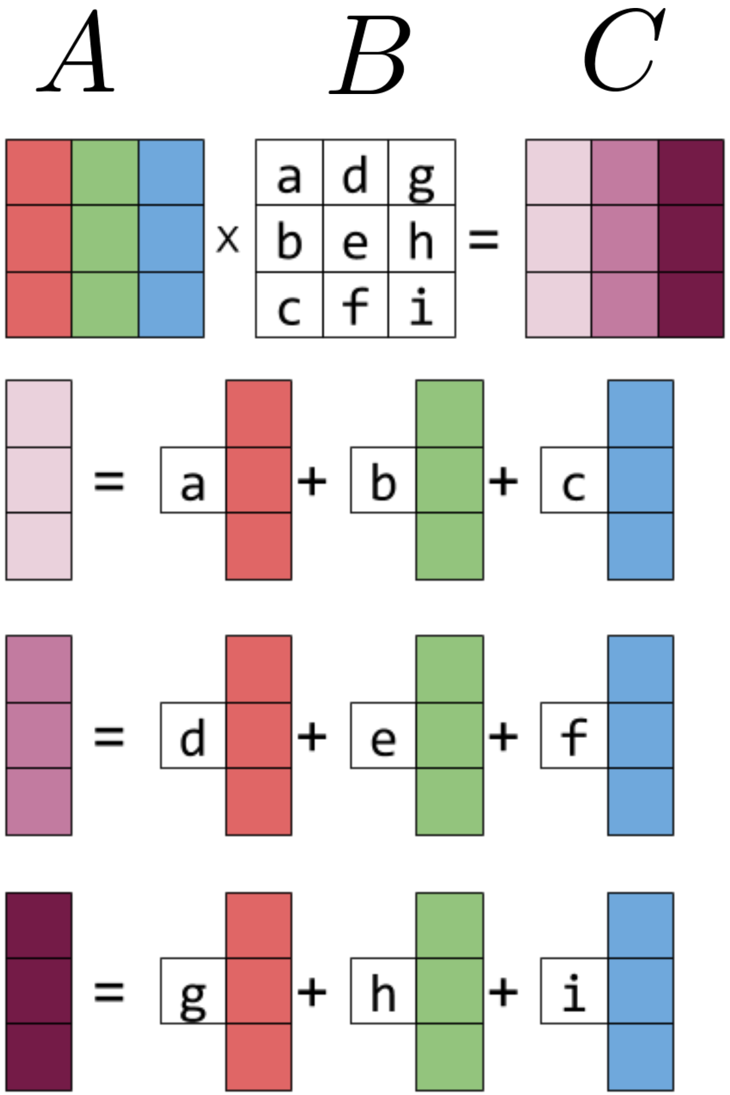
$$\begin{bmatrix} 1 & 2 & 1 & 5 \\ 0 & 3 & 0 & 4 \\ -1 & -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} =$$

Matrix-Matrix Product



$$A \cdot B = C$$

Matrix-Matrix Product



The *i*-th column of the matrix ***C*** is obtained by multiplying ***A*** with the ***i*** column of ***B***

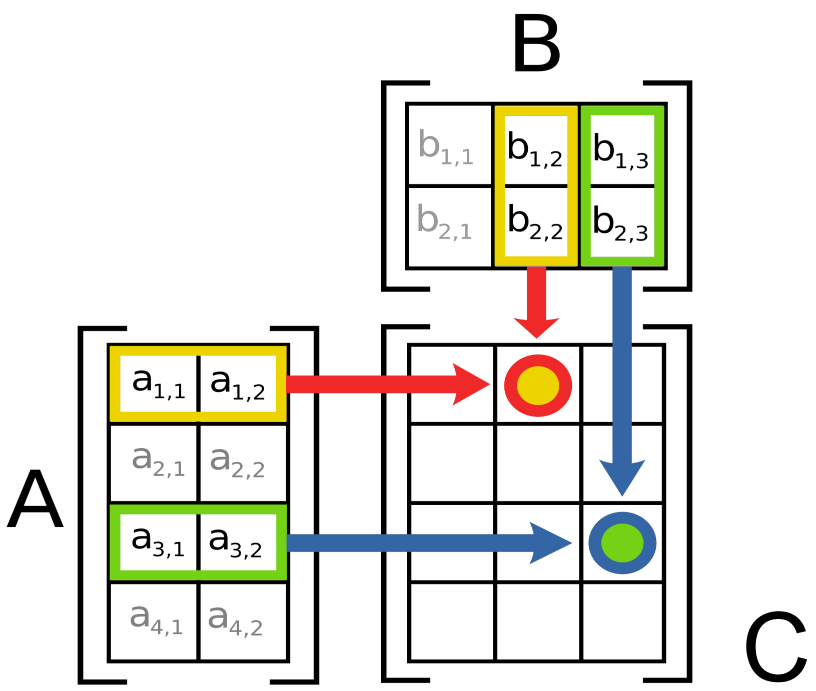
Example

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \end{bmatrix} =$$

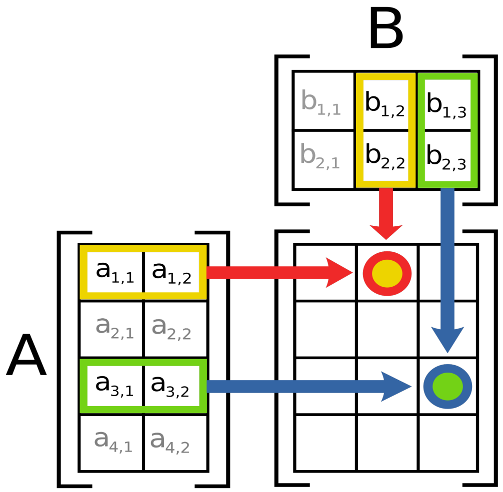
$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} =$$

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} — & \mathbf{a}_1^\top & — \\ — & \mathbf{a}_2^\top & — \\ & \vdots & \\ — & \mathbf{a}_m^\top & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^\top \mathbf{b}_1 & \mathbf{a}_1^\top \mathbf{b}_2 & \cdots & \mathbf{a}_1^\top \mathbf{b}_p \\ \mathbf{a}_2^\top \mathbf{b}_1 & \mathbf{a}_2^\top \mathbf{b}_2 & \cdots & \mathbf{a}_2^\top \mathbf{b}_p \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_m^\top \mathbf{b}_1 & \mathbf{a}_m^\top \mathbf{b}_2 & \cdots & \mathbf{a}_m^\top \mathbf{b}_p \end{bmatrix}$$



The ***I,j***-th element of the matrix ***C*** is the dot product of the ***i***-th row of ***A*** with the ***j***-th column of ***B***

A and ***B*** must have conforming dimensions!



Dim(A) = m x n
Dim(B) = n x p
Dim(AB) = m x p

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} - & \mathbf{b}_1^\top & - \\ - & \mathbf{b}_2^\top & - \\ & \vdots & \\ - & \mathbf{b}_n^\top & - \end{bmatrix} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^\top \ .$$

*The matrix ***C*** is the sum of outer product of ***i***-th column of A and ***i***-th row of B

Back to the simple linear regression example

House sizes:

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function (MSE):

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

House sizes:

Size in feet^2 (x)	**Price ($) in 1000's (y)**
2104	460
1416	232
1534	315
852	178
…	…

Hypothesis:

$$h_\theta(x) = -40 + 0.25x$$

Use matrix-vector multiplication:

$$\underbrace{\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix}}_{\boldsymbol{X}} \cdot \underbrace{\begin{bmatrix} -40 \\ 0.25 \end{bmatrix}}_{\boldsymbol{\theta}} = \underbrace{\begin{bmatrix} 486.0 \\ 314.0 \\ 343.5 \\ 173.0 \end{bmatrix}}_{\widehat{\mathbf{y}}}$$

House sizes:

Size in feet^2 (x)	**Price ($) in 1000's** (y)
2104	460
1416	232
1534	315
852	178
…	…

Model prediction

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \cdot \begin{bmatrix} -40 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 486.0 \\ 314.0 \\ 343.5 \\ 173.0 \end{bmatrix}$$

$$X \qquad \theta \qquad \hat{\mathbf{y}}$$

Cost

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$J(\boldsymbol{\theta}) = \frac{1}{2m}(\hat{\mathbf{y}} - \mathbf{y})^T(\hat{\mathbf{y}} - \mathbf{y}) = \frac{1}{2m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Model prediction

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \cdot \begin{bmatrix} -40 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 486.0 \\ 314.0 \\ 343.5 \\ 173.0 \end{bmatrix}$$

$$X \qquad \theta \qquad \hat{\mathbf{y}}$$

Goal: $\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Solution

$$\hat{\boldsymbol{\theta}} = \mathbf{argmin}_{\boldsymbol{\theta}}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Cost

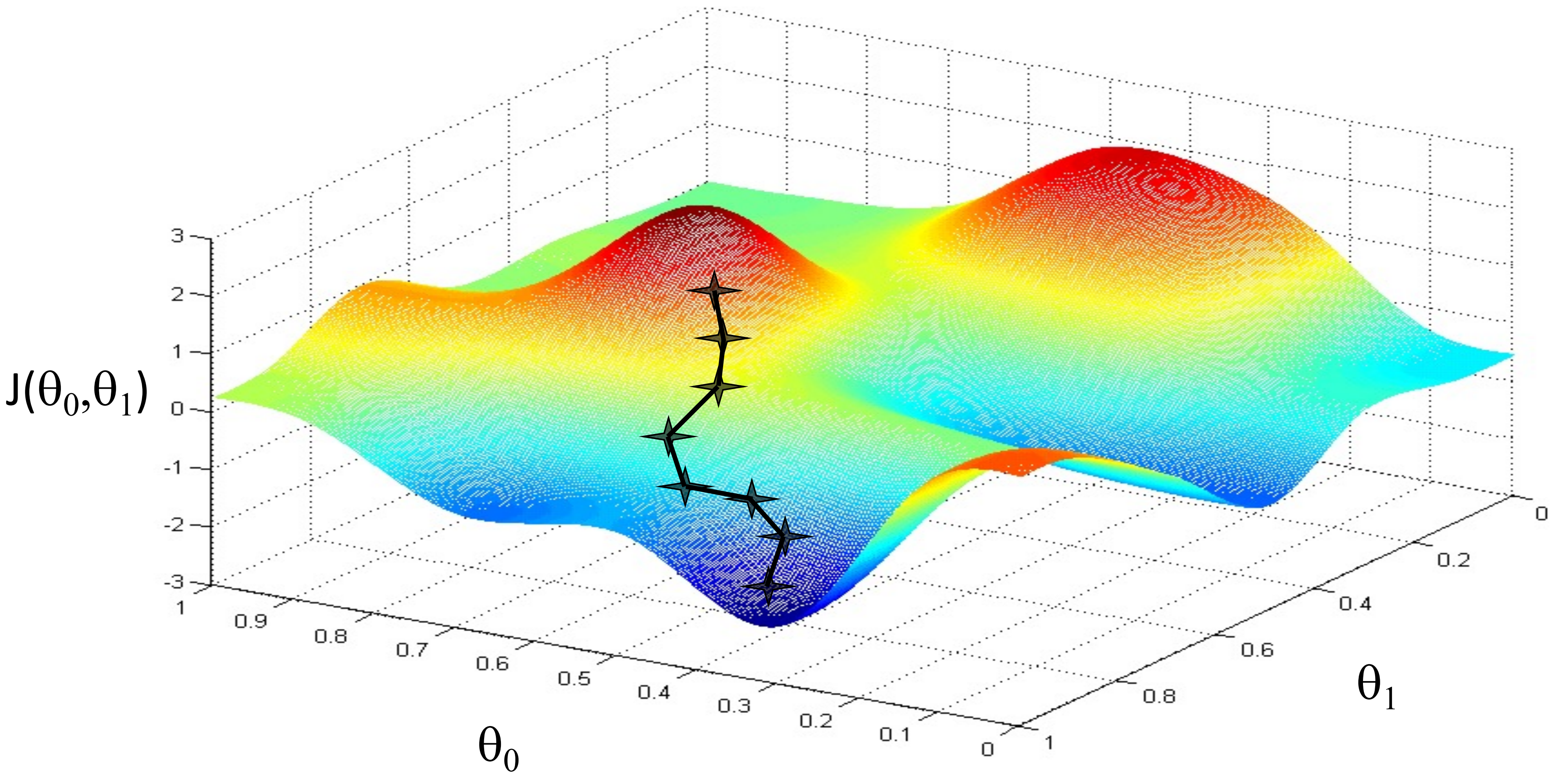
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J(\boldsymbol{\theta}) = \frac{1}{2m}(\hat{\mathbf{y}} - \mathbf{y})^T(\hat{\mathbf{y}} - \mathbf{y}) = \frac{1}{2m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

How to find the solution - calculus

$$\widehat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T \mathbf{y}$$

How to find the solution – gradient descent

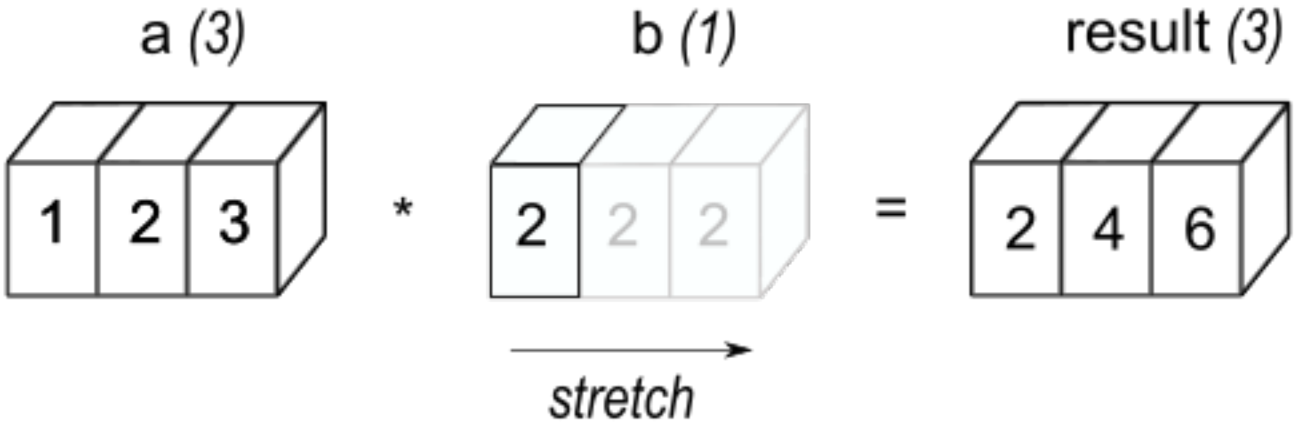


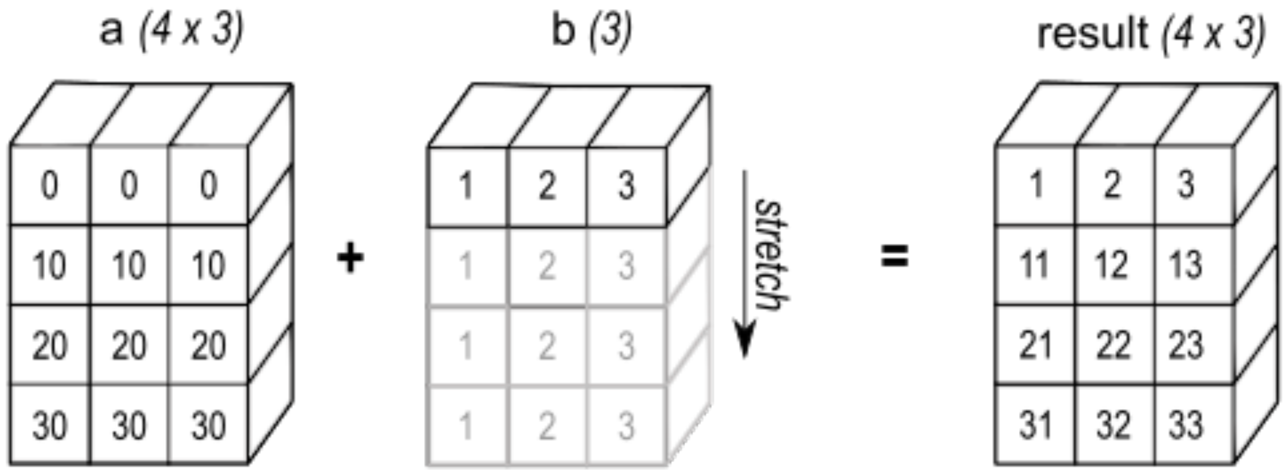
Broadcasting

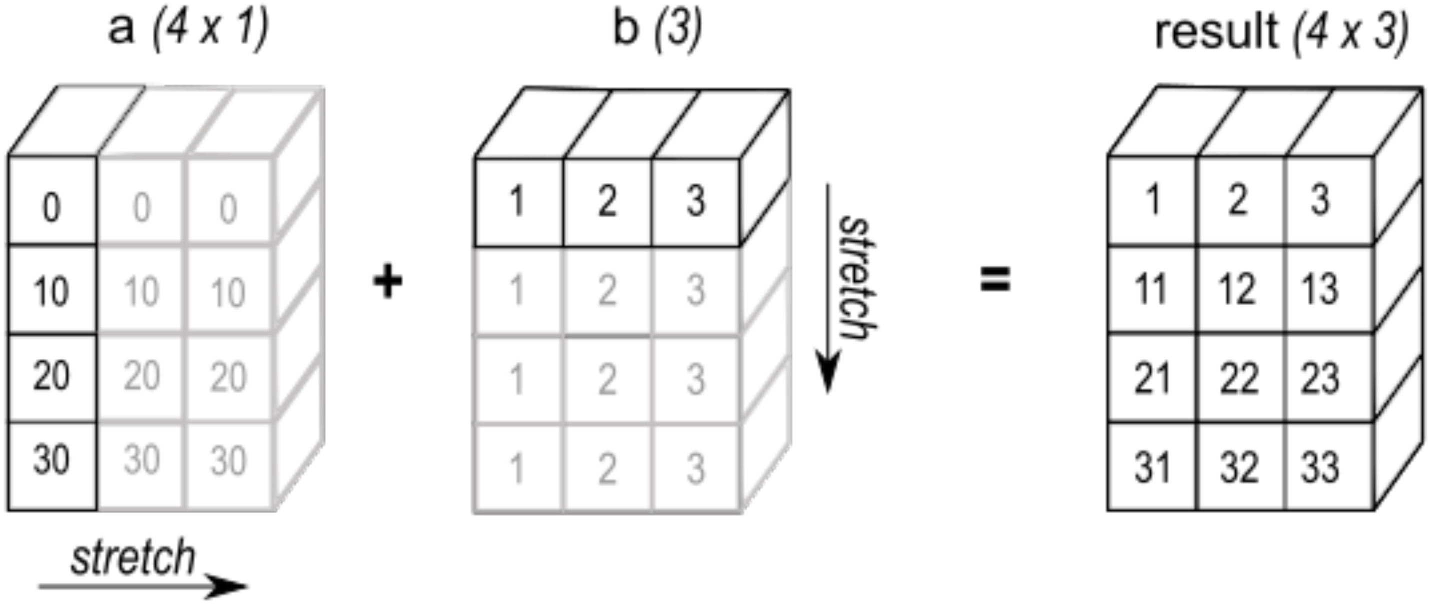
- Broadcasting: how NumPy (torch, tensorflow) treats arrays with different shapes during arithmetic operations.

- When performing operations between two arrays, broadcasting automatically expands one or both arrays so they have compatible shapes. This allows element-wise operations without the need for explicitly reshaping arrays.

Generalized tensor operation: Broadcasting




a (4 x 3)
b (3)
result (4 x 3)
0 0 0
10 10 10
20 20 20
30 30 30
+
1 2 3
1 2 3
1 2 3
1 2 3
stretch
=
1 2 3
11 12 13
21 22 23
31 32 33


a (4 x 1)
b (3)
result (4 x 3)
0
0
0
10
10
10
20
20
20
30
30
30
+
1
2
3
1
2
3
1
2
3
1
2
3
stretch
=
1
2
3
11
12
13
21
22
23
31
32
33
stretch

Concise Code: It reduces the need for explicit loops and reshaping, making the code more concise and easier to read.

Performance: Broadcasting is implemented in a memory-efficient manner, avoiding unnecessary copies of data.

Flexibility: It allows operations on arrays of different shapes without needing them to be exactly the same size.