Machine Learning:
Elements of **Linear Algebra**

Alexander G. Ororbia II

Introduction to Machine Learning

CSCI-635

8/30/2023

Special thanks to: Sargur N. Srihari

This is a Crash Course Review…

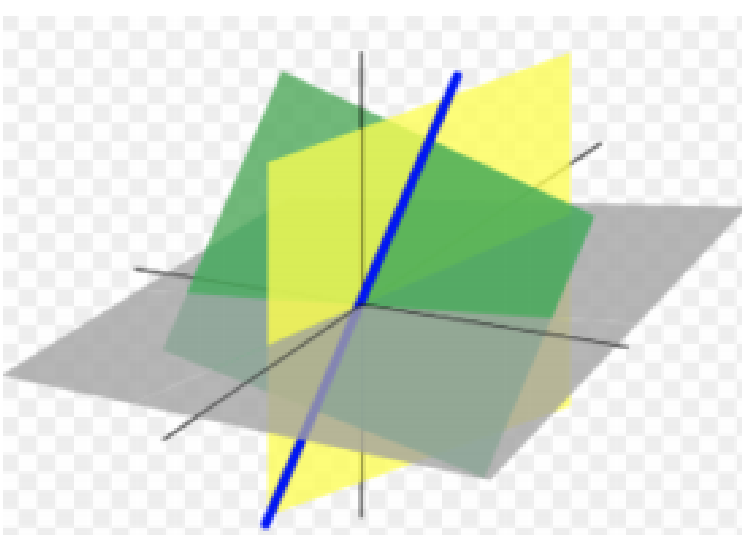
- Not a comprehensive survey of all of linear algebra
- Focused on subset most relevant to machine learning
- For a larger subset/treatment:
  ***Linear Algebra* by Georgi E. Shilov**
  - Also read: **“Linear Algebra for Dummies”**

What is Linear Algebra?

- Linear algebra is the branch of mathematics concerning linear equations such as

$$a_1 x_1 + \ldots.. + a_n x_n = b$$

  - In vector notation we say $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = b$
  - Called a linear transformation of $\boldsymbol{x}$
- Linear algebra is fundamental to geometry, for defining objects such as lines, planes, rotations



Linear equation $a_1 x_1 + \ldots.. + a_n x_n = b$ defines a plane in $(x_1, .., x_n)$ space
Straight lines define common solutions to equations

Linear algebra is used throughout engineering (based on continuous math)

Mathematical Notation in CSCI 635

$\subset$= proper subset (not the whole thing) $\subseteq$=subset

$\exists$ = there exists

$\forall$ = for every

$\in$= element of

$\bigcup$ = union (or)

$\bigcap$ = intersection (and)

s.t.= such that

w.r.t. = with respect to

$\Longrightarrow$ implies

$\Longleftrightarrow$ if and only if

$\sum$ = sum

$\prod$ = multiplication

$\setminus$ = set minus

$\therefore$= therefore

Scalars

- A scalar is a single number
- Integers, real numbers, rational numbers, etc.
- Denoted with italic font:

$$a, n, x$$

On number spaces/domains:

$\mathbb{R}$ $\mathcal{R}$ All real (continuous) numbers

$\mathbb{Z}$ $\mathcal{Z}$ All integers

$\mathbb{N}$ All natural (counting) numbers

Vectors

- An array of numbers arranged in order
- Each no. identified by an index
- Written in lower-case bold such as $\boldsymbol{x}$
  - its elements are in italics lower case, subscripted

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

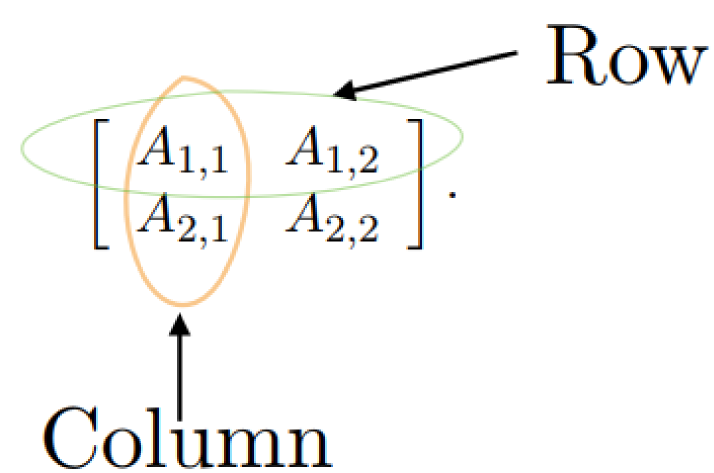
$$\mathcal{R}^n = \mathcal{R}^{n \times 1}$$

"All real numbers"

- If each element is in R then $\boldsymbol{x}$ is in R^n
- We can think of vectors as points in space
  - Each element gives coordinate along an axis

Matrices

- 2-D array of numbers
  - So each element identified by two indices
- Denoted by bold typeface $\boldsymbol{A}$
  - Elements indicated by name in italic but not bold
    - $A_{1,1}$ is the top left entry and $A_{m,n}$ is the bottom right entry
    - We can identify nos in vertical column j by writing : for the horizontal coordinate



The “slice” operator

    - $A_{i:}$ is i^{th} row of A, $A_{:j}$ is j^{th} column of $\boldsymbol{A}$
- If A has shape of height m and width n with real-values then $\boldsymbol{A} \in \mathbb{R}^{m \times n}$

Tensor

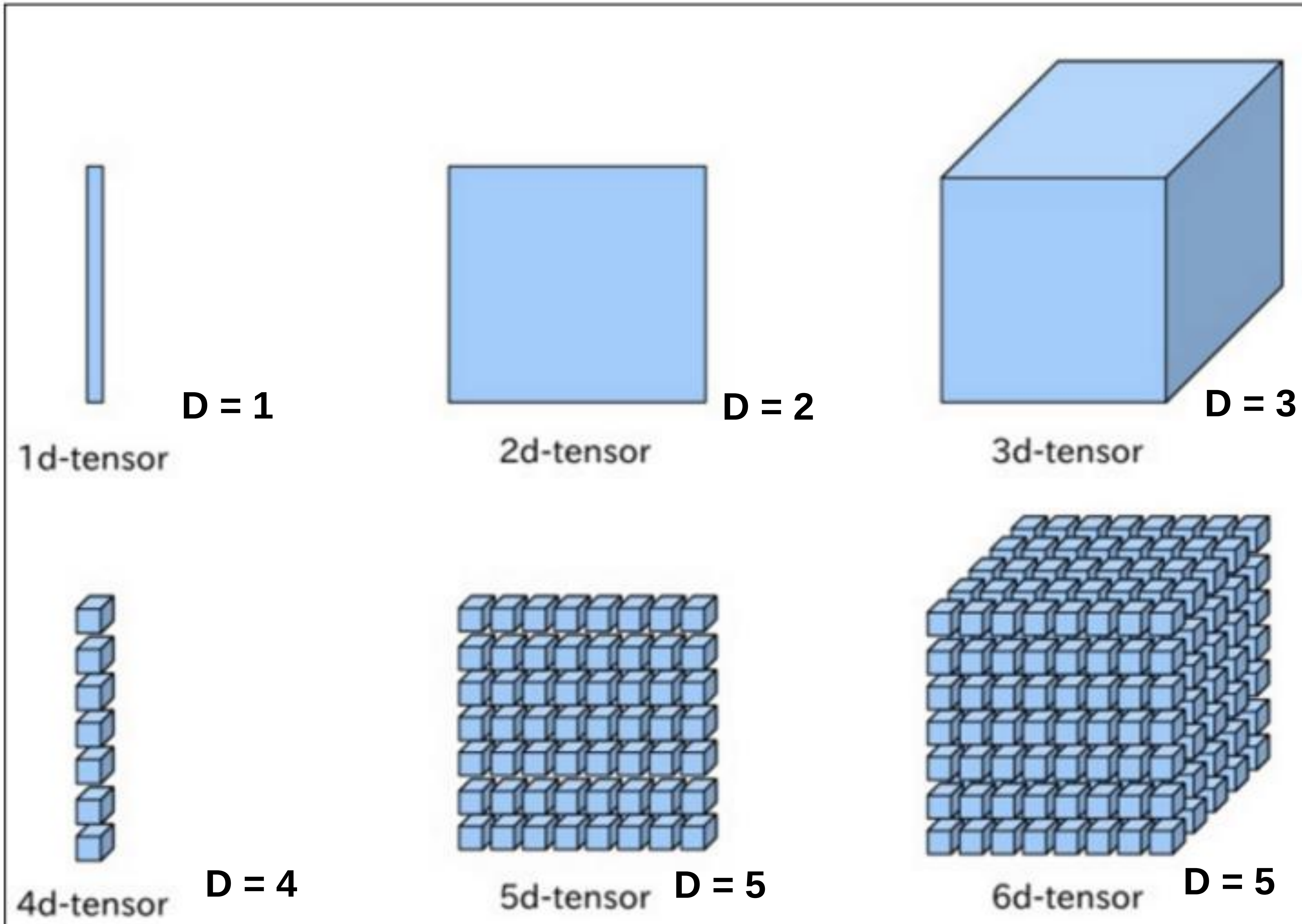
- Sometimes need an array with more than two axes
  - E.g., an RGB color image has three axes
- A tensor is an array of numbers arranged on a regular grid with variable number of axes

- Denote a tensor with this bold typeface: **A**
- Element (i,j,k) of tensor denoted by $\mathsf{A}_{i,j,k}$

Types of notation accepted (just be consistent & mean what you write):

$$\mathcal{R}^{1\times1\times1} = \mathcal{R}^{1\times1} = \mathcal{R}^{1} = \mathcal{R}$$

Shapes of Tensors



Transpose of a Matrix

- An important operation on matrices
- The transpose of a matrix $\boldsymbol{A}$ is denoted as $\boldsymbol{A}^{\mathrm{T}}$
- Defined as

$$(\mathbf{A}^{\mathrm{T}})_{i,j} = A_{j,i}$$

  – The mirror image across a diagonal line
    - Called the main diagonal , running down to the right starting from upper left corner

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \\ A_{1,3} & A_{2,3} & A_{3,3} \end{bmatrix}$$

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

$$(\boldsymbol{AB})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top$$ ← Useful property of transpose

Vector as Special Case of Matrix

- Vectors are matrices with a single column
- Often written in-line using transpose

$$\boldsymbol{x} = [x_1,..,x_n]^{\mathrm{T}}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \Rightarrow \boldsymbol{x}^T = [x_1, x_2, .. x_n]$$

- A scalar is a matrix with one element

$$a = a^{\mathrm{T}} \longleftarrow \mathcal{R}^{1\times 1}$$

Matrix Addition/Subtraction

- Assume column-major matrices (for efficiency)
- Add/subtract operators follow basic properties of normal add/subtract
  - Matrix A + Matrix B is computed element-wise

0.5	-0.7
-0.69	1.8

+

0.5	-0.7
-0.69	1.8

=

.5 + .5 = 1.0	-.7 - .7 = -1.4
-.69 - .69 = -1.38	1.8 + 1.8 = 3.6

Matrix Addition

- We can add matrices to each other if they have the same shape, by adding corresponding elements
  - If A and B have same shape (height m, width n)

$$C = A + B \Rightarrow C_{i,j} = A_{i,j} + B_{i,j}$$

- A scalar can be added to a matrix or multiplied by a scalar $D = aB + c \Rightarrow D_{i,j} = aB_{i,j} + c$
- Less conventional notation used in ML:
  - Vector added to matrix $C = A + \boldsymbol{b} \Rightarrow C_{i,j} = A_{i,j} + b_j$
    - Called broadcasting since vector $\boldsymbol{b}$ added to each row of A

Multiplying Matrices

- For product $\boldsymbol{C}=\boldsymbol{AB}$ to be defined, $\boldsymbol{A}$ has to have the same no. of columns as the no. of rows of $\boldsymbol{B}$
  - If $\boldsymbol{A}$ is of shape mxn and B is of shape nxp then *matrix product* $\boldsymbol{C}$ is of shape mxp

$$C = AB \Rightarrow C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

- Dot product between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of same dimensionality is the matrix product $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}$
- We can think of matrix product $C=AB$ as computing C_{ij} the dot product of row i of A and column j of B

Matrix-Matrix Multiply

- Matrix-Matrix multiply (outer product)
  - Vector-Vector multiply (dot product)
- The usual workhorse of machine learning
- Vectorizes sums of products (builds on dot product)

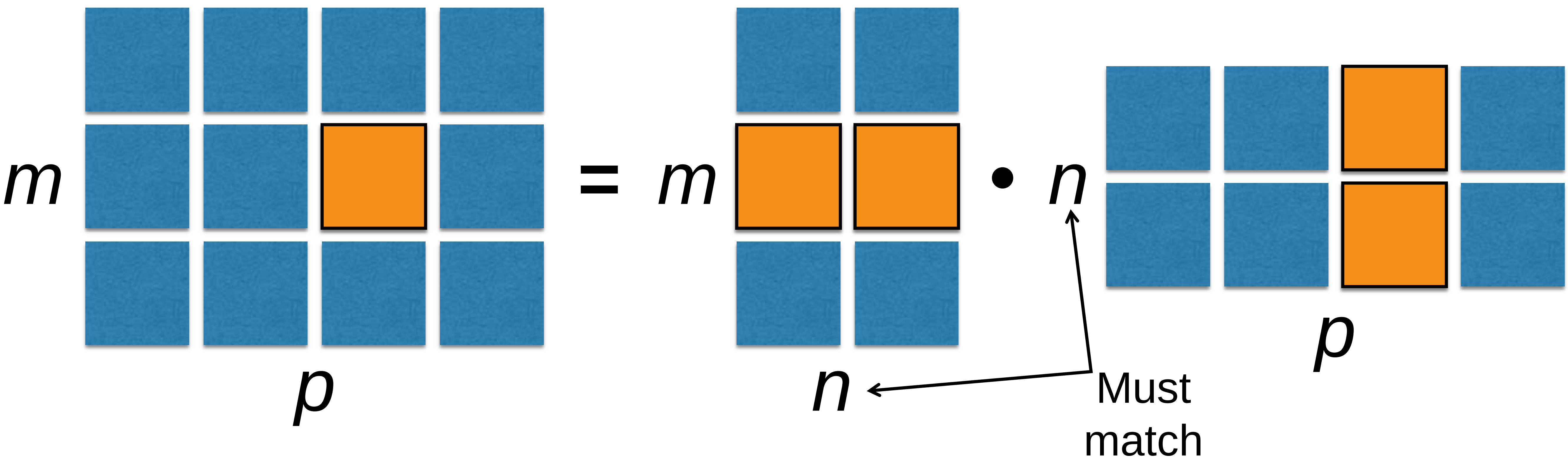
0.5	-0.7
-0.69	1.8

*

0.5	-0.7
-0.69	1.8

=

(.5 * .5) + (-.7 * -.69)	(.5 * -.7) + (-.7 * 1.8)
(-.69 * .5) + (1.8 * -.69)	(-.69 * -.7) + (1.8 * 1.8)



Referred to sometimes as matrix-matrix product or matrix-vector product (or multiply)

Matrix Product Properties

- Distributivity over addition: $A(B+C)=AB+AC$
- Associativity: $A(BC)=(AB)C$
- Not commutative: $AB=BA$ is not always true
- Dot product between vectors is commutative: $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}=\boldsymbol{y}^{\mathrm{T}}\boldsymbol{x}$
- Transpose of a matrix product has a simple form: $(AB)^{\mathrm{T}}=B^{\mathrm{T}}A^{\mathrm{T}}$

Hadamard Product

- Multiply each A(i, j) to each corresponding B(i, j)
  - Element-wise multiplication

0.5	-0.7
-0.69	1.8

$\odot$

0.5	-0.7
-0.69	1.8

=

.5 * .5 = .25	-.7 * .7 = .49
-.69 * -.69 = .4761	1.8 * 1.8 = 3.24

Elementwise Functions

- Applied to each element (i, j) of matrix/vector argument
  - Could be *cos(.)*, *sin(.)*, *tanh(.)*, etc. (the "." means argument)
- *Identity*: $\phi(\mathbf{v}) = \mathbf{v}$
- *Logistic Sigmoid*: $\phi(\mathbf{v}) = \sigma(\mathbf{v}) = \frac{1}{1+e^{-\mathbf{v}}}$
- *Softmax*: $\phi(\mathbf{v}) = \frac{\exp(\mathbf{v})}{\sum_{c=1}^{C} \exp(\mathbf{v}_c)}$
- *Linear Rectifier*: $\phi(\mathbf{v}) = \max(0, \mathbf{v})$

$\mathbf{v} \in \mathbb{R}^C$

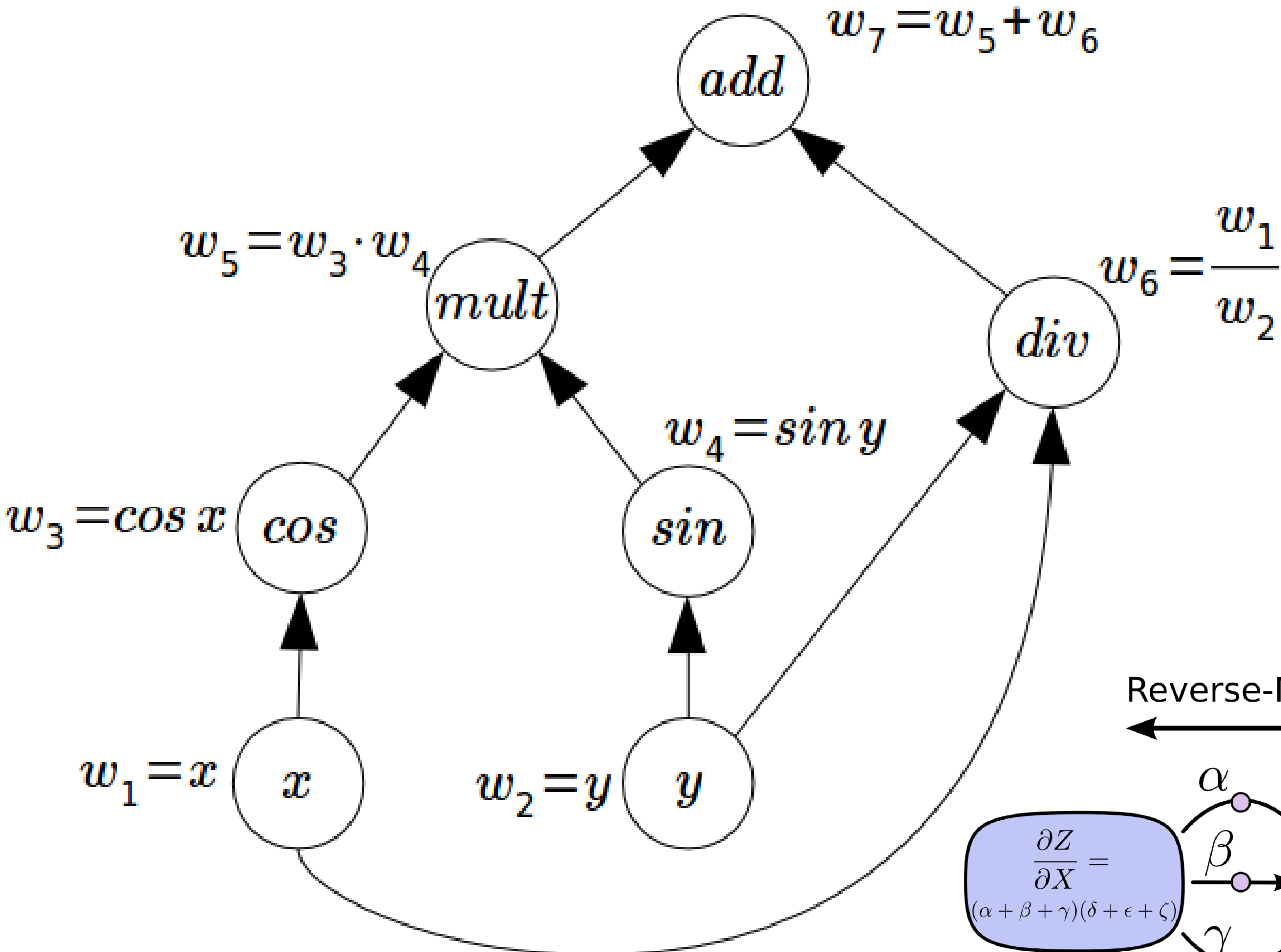
φ(

1.0	-1.4
-0.69	1.8

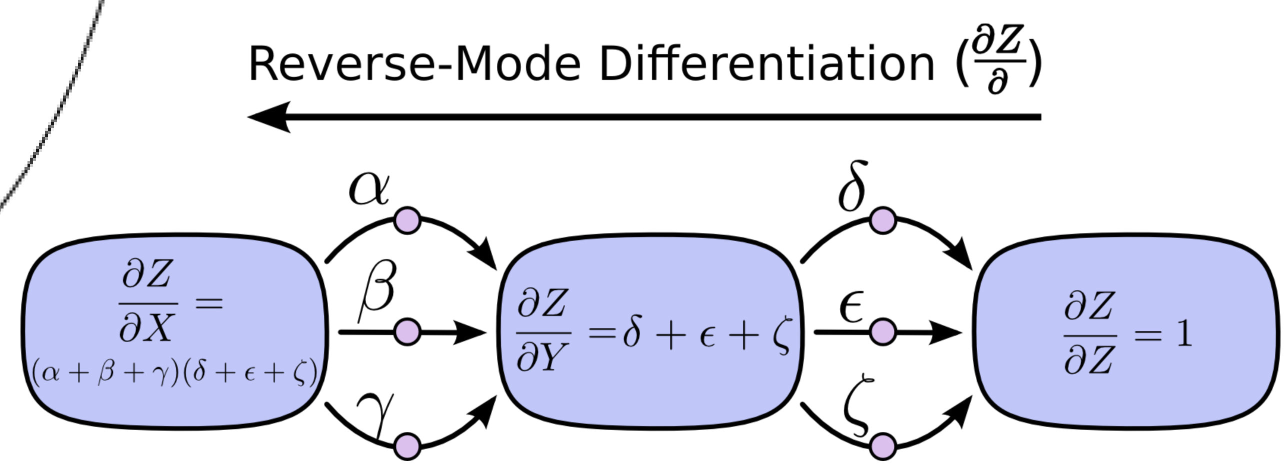
) =

$\varphi(1.0) = 1$	$\varphi(-1.4) = 0$
$\varphi(-0.68) = 0$	$\varphi(1.8) = 1.8$

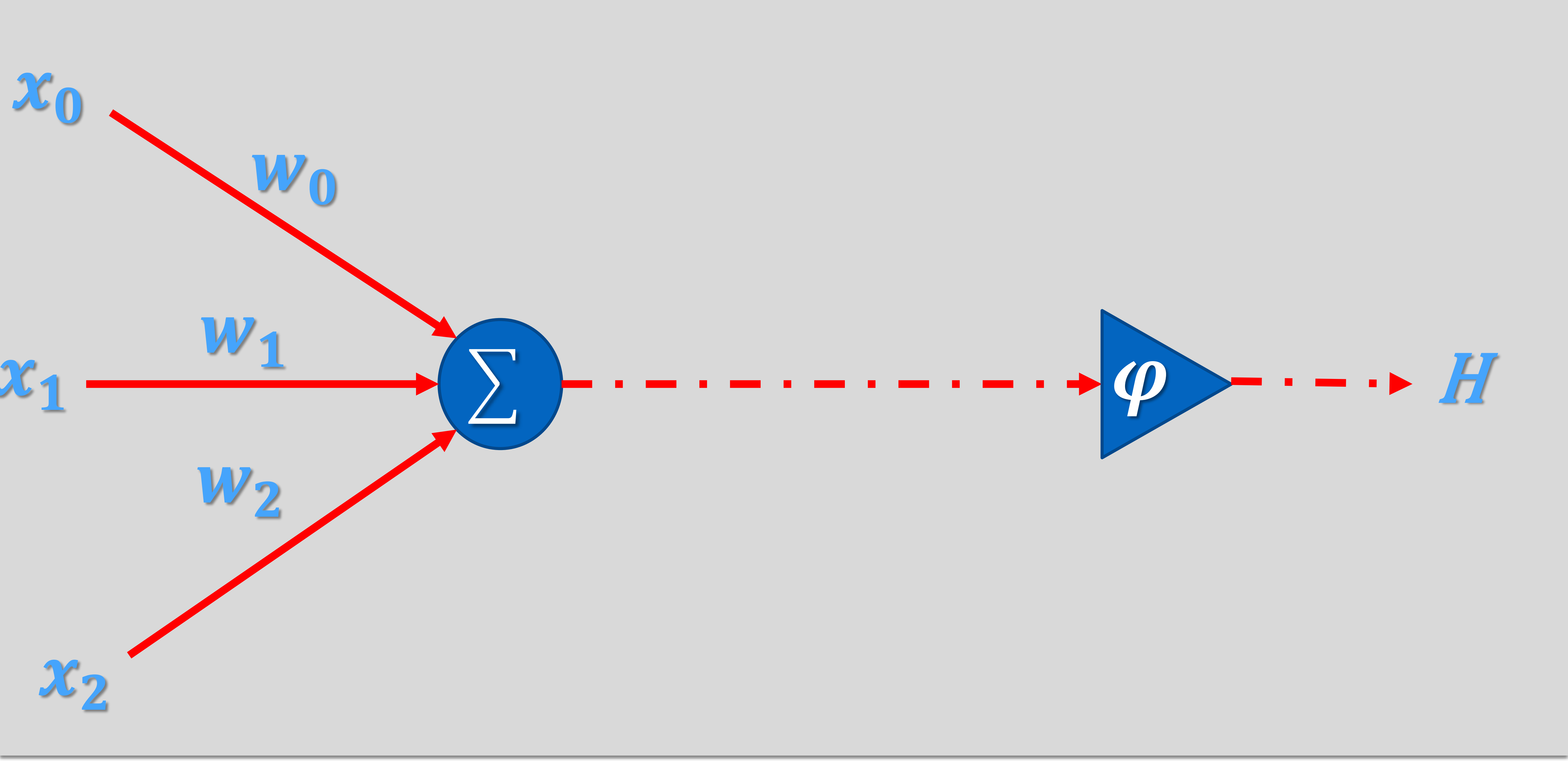
Why Do We Care? Computation Graphs



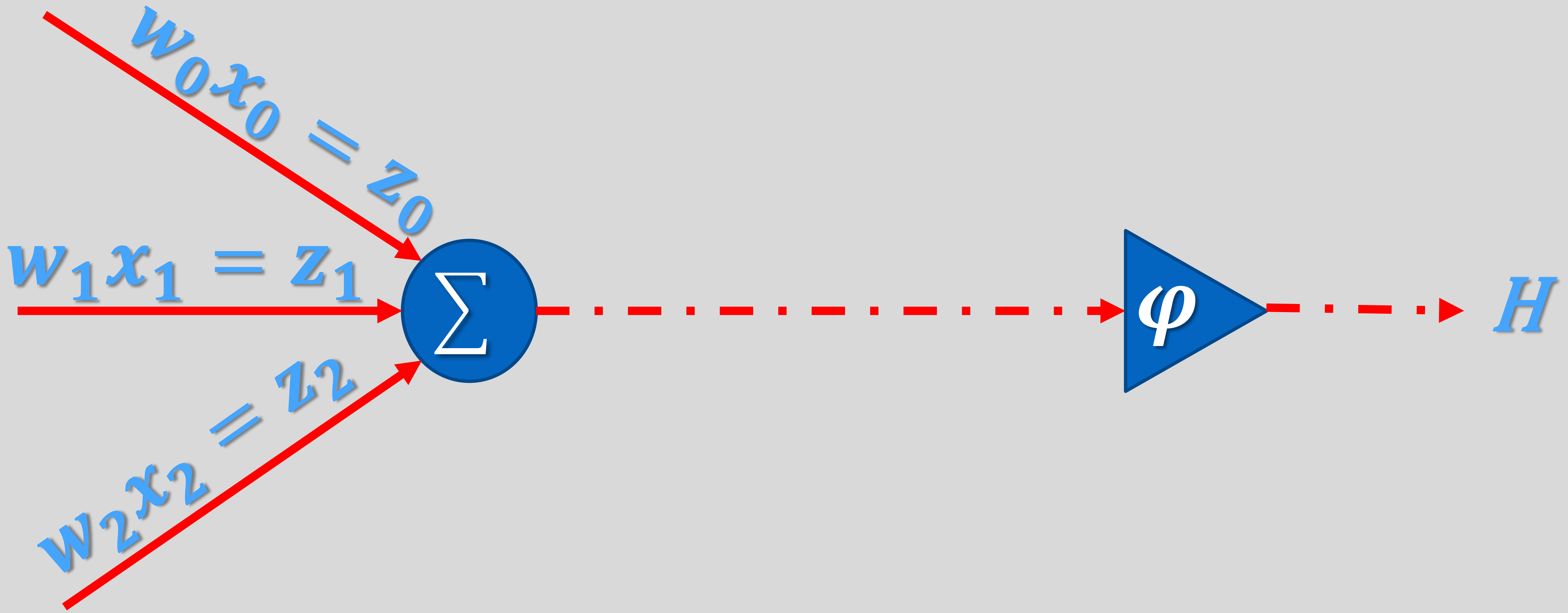
Linear algebra operators arranged in a directed graph!



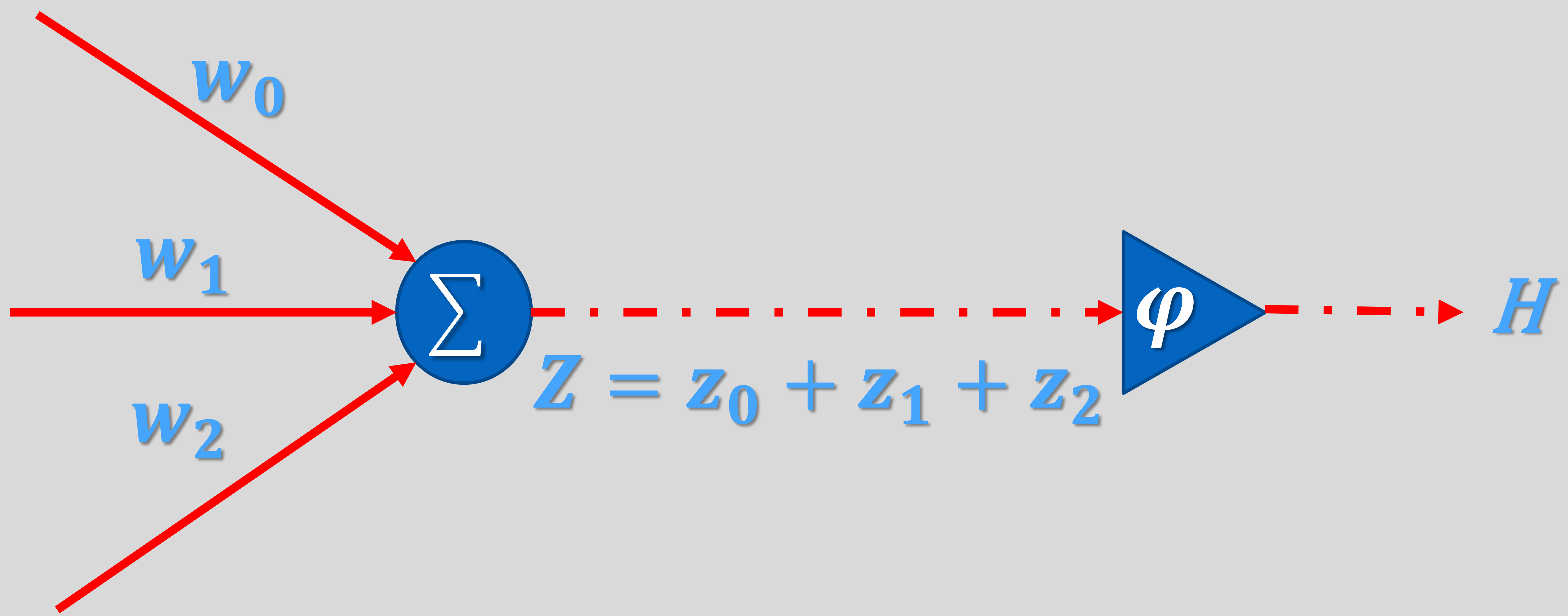
A Simple Linear Predictor



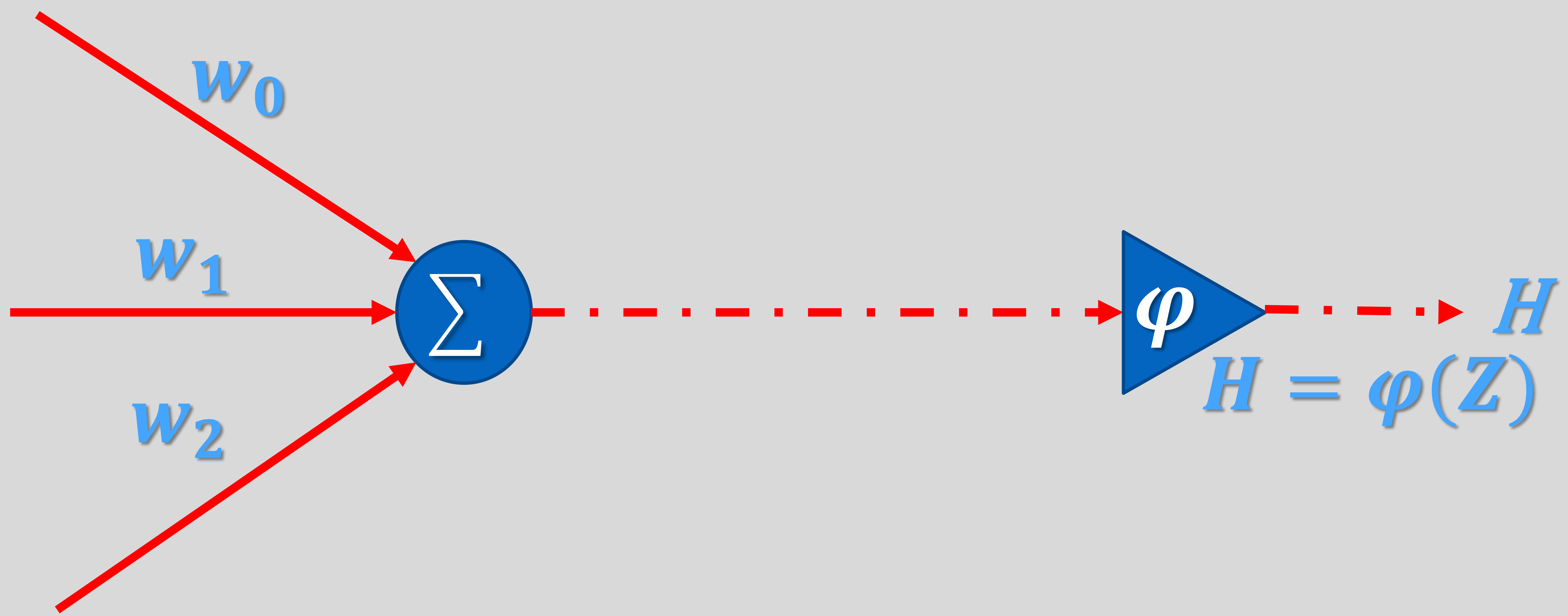
A Simple Linear Predictor



A Simple Linear Predictor



A Simple Linear Predictor



Vector Form (One Unit)

This calculates activation value of single (output) unit that is connected to 3 (input) sensors.

$$h_0: \begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \left[\varphi(w_0 * x_0 + w_1 * x_1 + w_2 * x_2) \right]$$

Vector Form (Two Units)

This vectorization easily generalizes to multiple (3) sensors feeding into multiple (2) units.

$$\begin{matrix} h_0: \\ h_1: \end{matrix} \begin{bmatrix} w_0 & w_1 & w_2 \\ w_3 & w_4 & w_5 \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \varphi(w_0 * x_0 + w_1 * x_1 + w_2 * x_2) \\ \varphi(w_3 * x_0 + w_4 * x_1 + w_5 * x_2) \end{bmatrix}$$

Known as vectorization!

Now Let Us Fully Vectorize This!

This vectorization is also important for formulating **mini-batches**.
(Good for GPU-based processing.)

h_0:
h_1:

$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$
$\mathbf{w}_3$	$\mathbf{w}_4$	$\mathbf{w}_5$

*

$\mathbf{x}_0$	$\mathbf{x}_3$
x_1	x_4
x_2	x_5

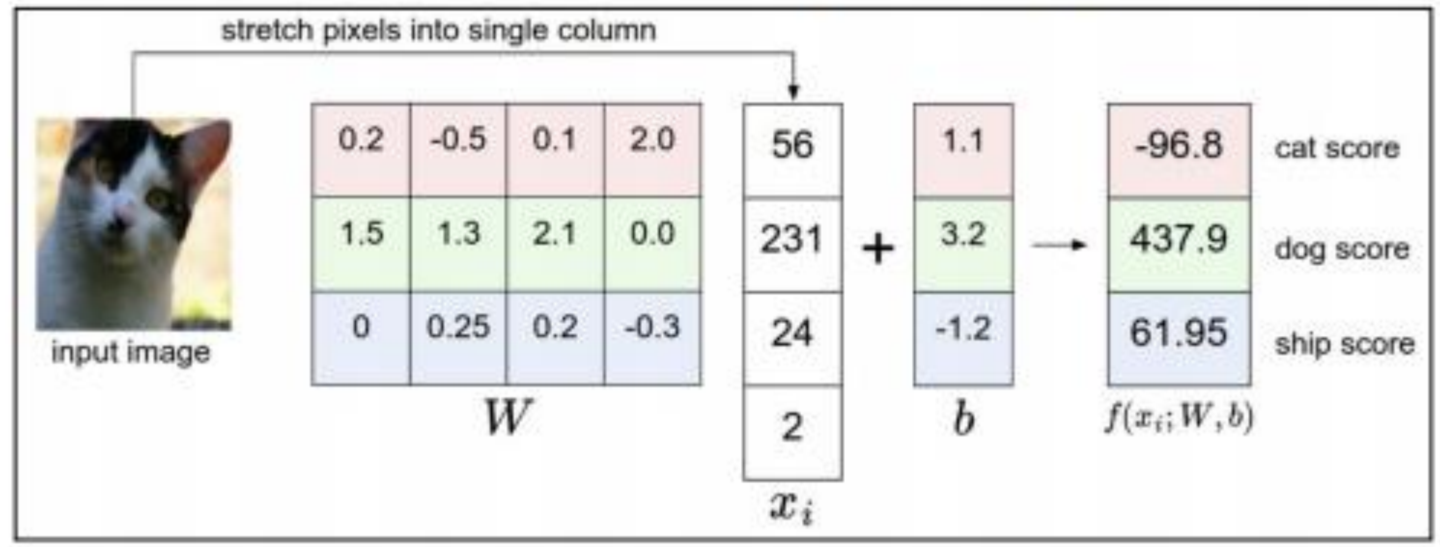
=

$\varphi(\mathbf{w}_0 * \mathbf{x}_0 + \ldots)$	$\varphi(\mathbf{w}_0 * \mathbf{x}_3 + \ldots)$
$\varphi(\mathbf{w}_3 * \mathbf{x}_0 + \ldots)$	$\varphi(\mathbf{w}_3 * \mathbf{x}_3 + \ldots)$

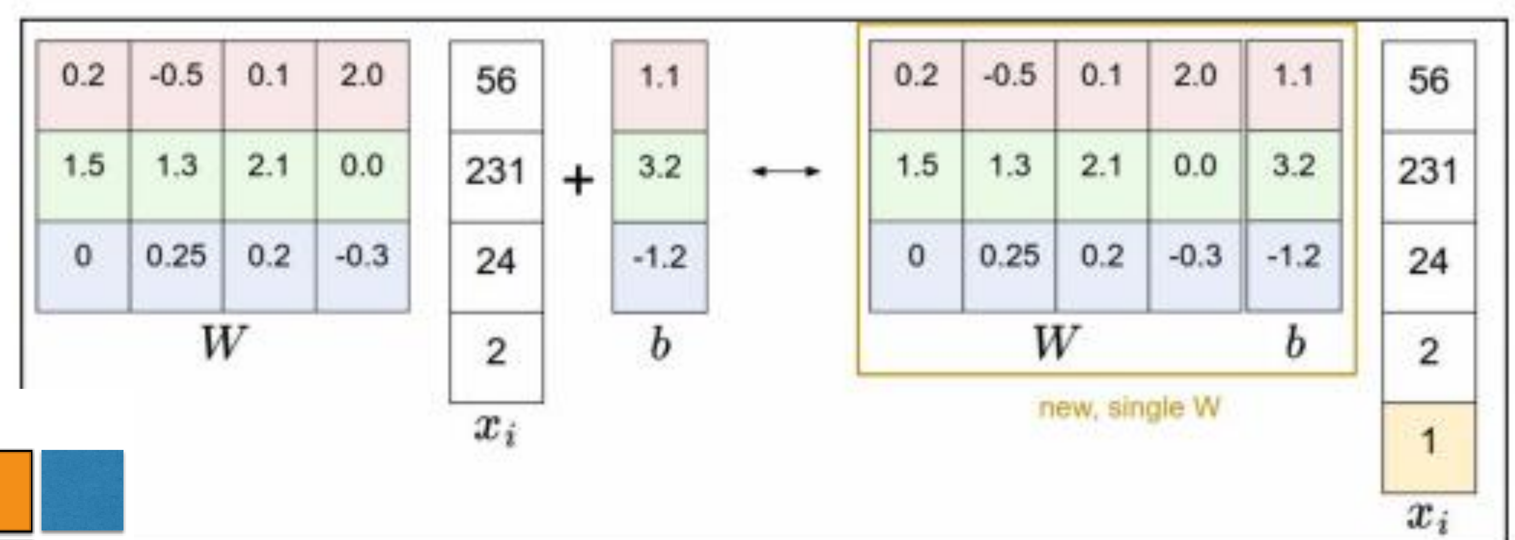
Tensors in Machine Learning

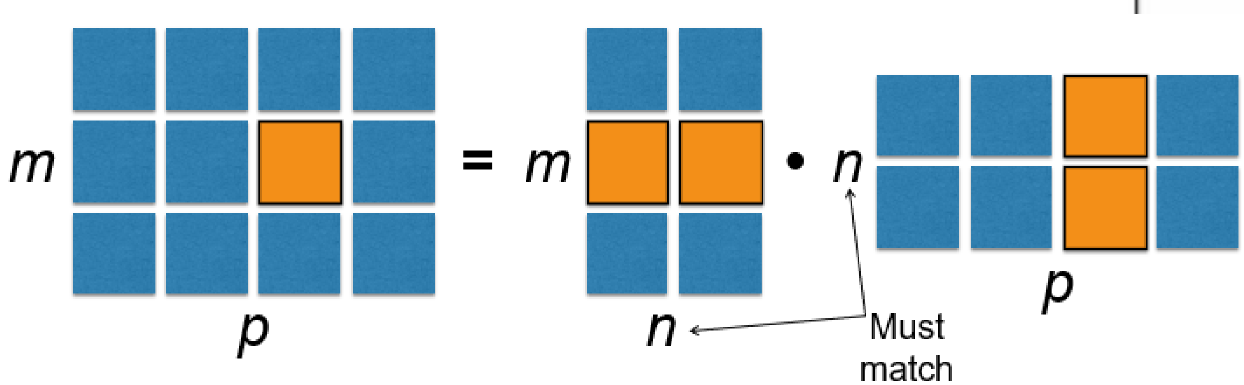
Vector x is converted into vector y by multiplying x by a matrix W

A linear classifier $y = Wx + b$



A linear classifier with bias eliminated $y = Wx$





Identity and Inverse Matrices

- Matrix inversion is a powerful tool to analytically solve $A\boldsymbol{x}=\boldsymbol{b}$
- Needs concept of Identity matrix
- Identity matrix does not change value of vector when we multiply the vector by identity matrix
  - Denote identity matrix that preserves n-dimensional vectors as I_n
  - Formally $I_n \in \mathbb{R}^{n\times n}$ and $\forall \mathbf{x} \in \mathbb{R}^n, I_n \mathbf{x} = \mathbf{x}$
  - Example of I_3
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Norms

- Used for measuring the size of a vector
- Norms map vectors to non-negative values
- Norm of vector $\boldsymbol{x} = [x_1,..,x_n]^{\mathrm{T}}$ is distance from origin to $\boldsymbol{x}$
  - It is any function f that satisfies:

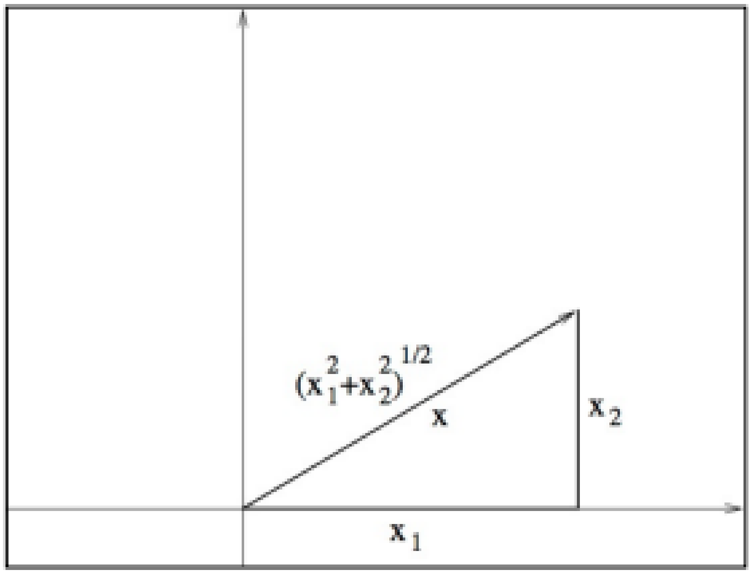
$$f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = 0$$

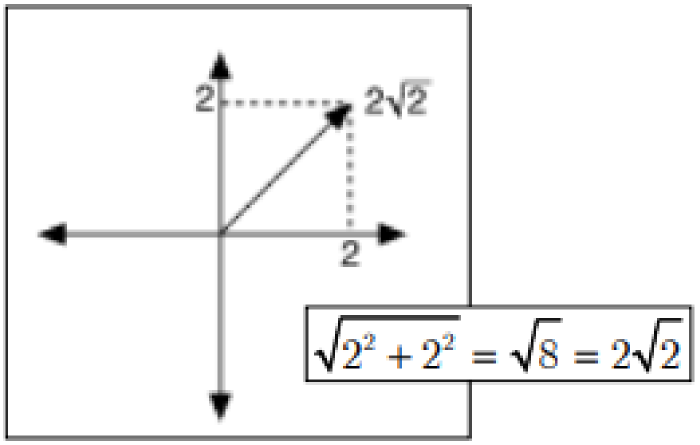
$$f(\boldsymbol{x}+\boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}) \quad \text{Triangle Inequality}$$

$$\forall \alpha \in R \quad f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$$

L^P Norm

- Definition: $$\|\boldsymbol{x}\|_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}$$





  - L^2 Norm
    - Called Euclidean norm
      - Simply the Euclidean distance between the origin and the point $\boldsymbol{x}$
      - written simply as $\|\boldsymbol{x}\|$
      - Squared Euclidean norm is same as $\boldsymbol{x}^T\boldsymbol{x}$
  - L^1 Norm
    - Useful when 0 and non-zero have to be distinguished
      - Note that L^2 increases slowly near origin, e.g., $0.1^2=0.01$)
  - L^∞ Norm $$\|\boldsymbol{x}\|_\infty = \max_i |x_i|$$
    - Called max norm

The Frobenius Norm

Similar to L^2 norm

$$\|A\|_F = \left(\sum_{i,j} A_{i,j}^2 \right)^{\frac{1}{2}}$$

$$A = \begin{bmatrix} 2 & -1 & 5 \\ 0 & 2 & 1 \\ 3 & 1 & 1 \end{bmatrix} \quad \|A\| = \sqrt{4+1+25+..+1} = \sqrt{46}$$

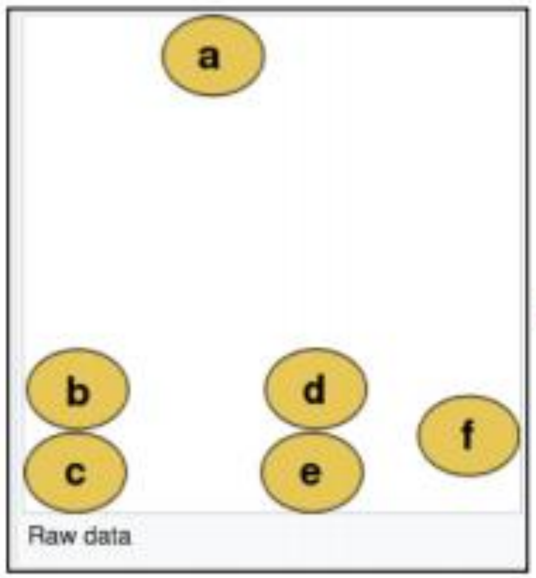
Norms Can Serve as Building Blocks for Distance Measurements!

- Distance between two vectors $(\boldsymbol{v}, \boldsymbol{w})$
  - $\mathrm{dist}(\boldsymbol{v}, \boldsymbol{w}) = ||\boldsymbol{v} - \boldsymbol{w}||$

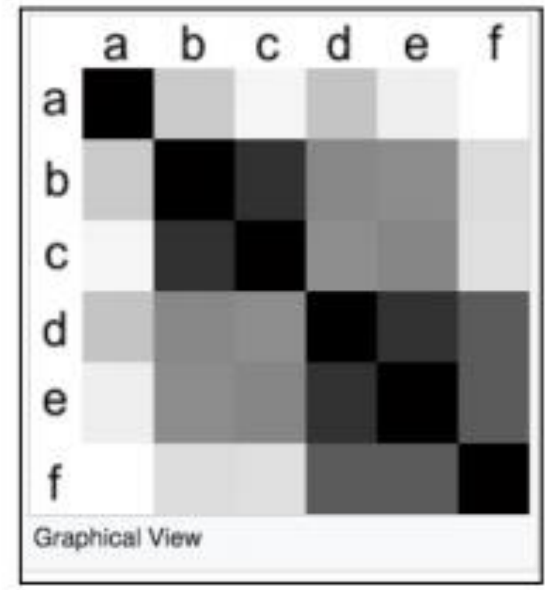
$$= \sqrt{(v_1 - w_1)^2 + .. + (v_n - w_n)^2}$$

Special kind of Matrix: Symmetric

- A symmetric matrix equals its transpose: $A = A^T$
  - E.g., a distance matrix is symmetric with $A_{ij} = A_{ji}$

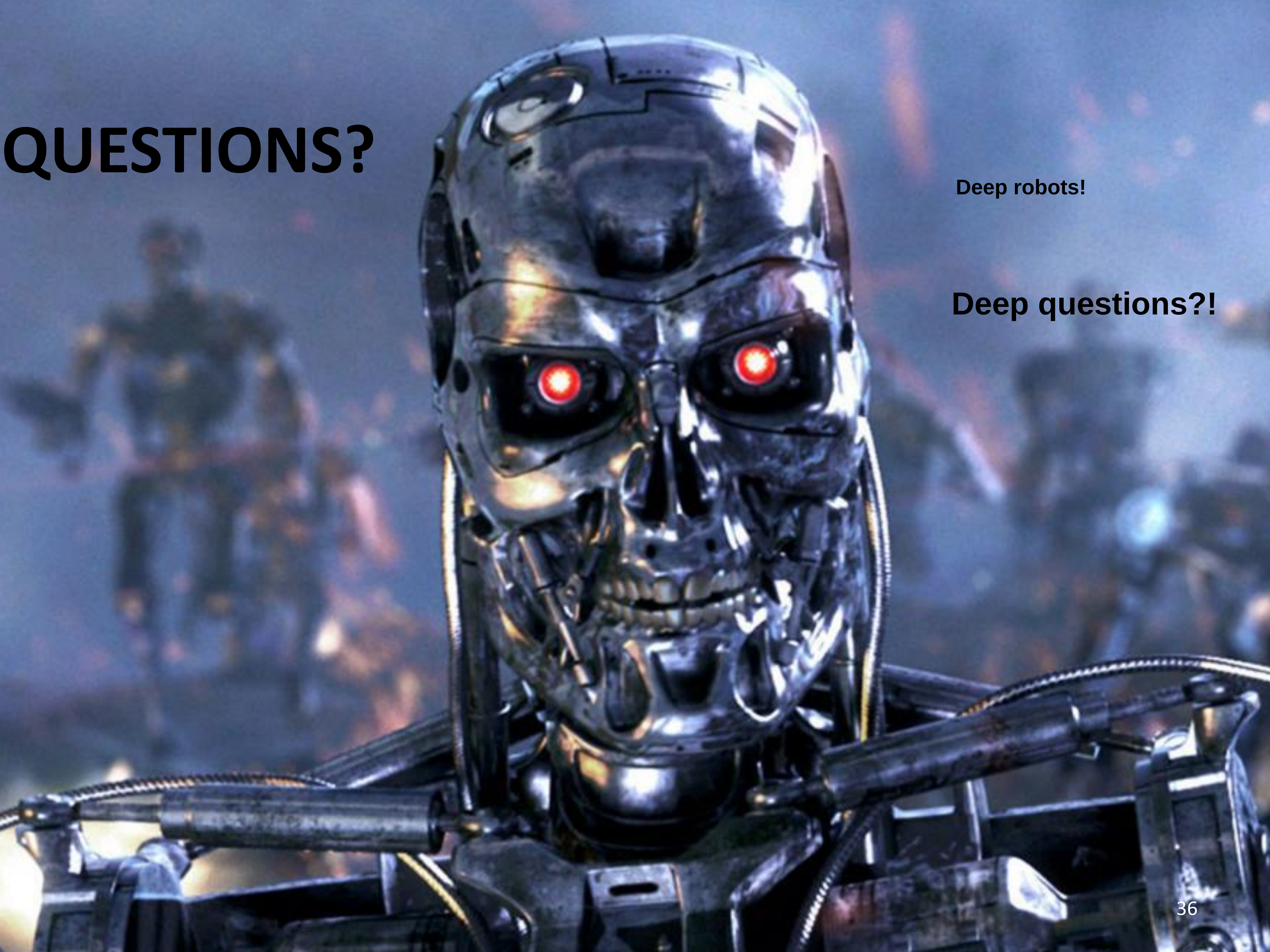


	a	b	c	d	e	f
a	0	184	222	177	216	231
b	184	0	45	123	128	200
c	222	45	0	129	121	203
d	177	123	129	0	46	83
e	216	128	121	46	0	83
f	231	200	203	83	83	0



  - E.g., covariance matrices are symmetric

$$\Sigma = \begin{pmatrix} 1 & .5 & .15 & .15 & 0 & 0 \\ .5 & 1 & .15 & .15 & 0 & 0 \\ .15 & .15 & 1 & .25 & 0 & 0 \\ .15 & .15 & .25 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & .10 \\ 0 & 0 & 0 & 0 & .10 & 1 \end{pmatrix}$$



QUESTIONS?

Deep robots!

Deep questions?!

Linear Transformation

- $\boldsymbol{Ax}=\boldsymbol{b}$
  - where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^{n}$
  - More explicitly

$$
\begin{aligned}
A_{11}x_1 + A_{12}x_2 + \ldots. + A_{1n}x_n &= b_1 \\
A_{21}x_1 + A_{22}x_2 + \ldots. + A_{2n}x_n &= b_2 \\
\\
A_{n1}x_1 + A_{m2}x_2 + \ldots. + A_{n,n}x_n &= b_n
\end{aligned}
$$

n equations in
n unknowns

$$
A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \vdots & \vdots \\ A_{n,1} & \cdots & A_{nn} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}
$$

n x n $\quad$ n x 1 $\quad$ n x 1

Can view A as a *linear transformation* of vector $\boldsymbol{x}$ to vector $\boldsymbol{b}$

- Sometimes we wish to solve for the unknowns $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ when A and $\boldsymbol{b}$ provide constraints

Use of a Vector in Regression

- A design matrix
  - N samples, D features

	# hours studied	# hours playing games	# classes missed
Student #1	10	3	0
Student #2	8	20	2
Student #3	5	1	5

→

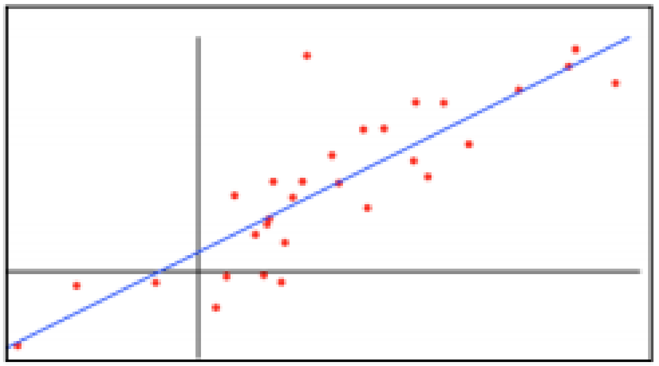
Grade
87
75
63

  - Feature vector has three dimensions
- This is a regression problem

Use of norm in Regression

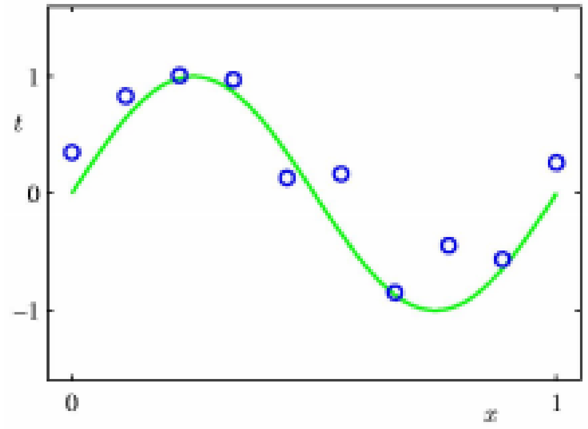
- Linear Regression

  $\boldsymbol{x}$: a vector, $\boldsymbol{w}$: weight vector



$$y(\boldsymbol{x},\boldsymbol{w}) = w_0 + w_1 x_1 + .. + w_d x_d = \boldsymbol{w}^T \boldsymbol{x}$$

  With nonlinear basis functions ϕ_j



$$y(\boldsymbol{x},\boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\boldsymbol{x})$$

- Loss Function

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\boldsymbol{x}_n,\boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}||\,\boldsymbol{w}^2\,||$$

  Second term is a weighted norm

  called a regularizer (to prevent overfitting)

Matrix Inverse

- Inverse of square matrix A defined as $A^{-1}A = I_n$
- We can now solve $A\boldsymbol{x}=\boldsymbol{b}$ as follows:

$$A\boldsymbol{x} = \boldsymbol{b}$$
$$A^{-1}A\boldsymbol{x} = A^{-1}\boldsymbol{b}$$
$$I_n\boldsymbol{x} = A^{-1}\boldsymbol{b}$$
$$\boldsymbol{x} = A^{-1}\boldsymbol{b}$$

- This depends on being able to find A^{-1}
- If A^{-1} exists there are several methods for finding it

Numerically unstable, but useful for abstract analysis

Closed-form solutions

- Two closed-form solutions
  1. Matrix inversion $\boldsymbol{x} = \mathrm{A}^{-1}\boldsymbol{b}$
  2. Gaussian elimination

- If A^{-1} exists, the same A^{-1} can be used for any given $\boldsymbol{b}$
  - But A^{-1} cannot be represented with sufficient precision
  - It is not used in practice
- Gaussian elimination also has disadvantages
  - numerical instability (division by small no.)
  - $O(n^3)$ for n x n matrix

Invertibility

- Matrix can’t be inverted if…
  - More rows than columns
  - More columns than rows
  - Redundant rows/columns (“linearly dependent”, “low rank”)

Matrix Rank

- Matrix **rank** is defined as (a) maximum number of linearly independent *column* vectors in matrix or (b) maximum number of linearly independent *row* vectors in matrix (equivalent definitions)

- Linear independence/dependence

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

$$\mathbf{b} = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$

$$\mathbf{c} = \begin{bmatrix} 5 & 7 & 9 \end{bmatrix}$$

$$\mathbf{d} = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix}$$

$$\mathbf{e} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$