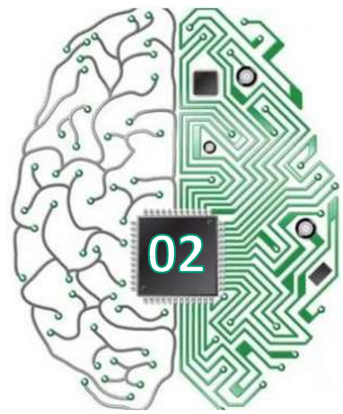
Open Elective Course [OE]
Course Code: CSO507
Winter 2023-24

Lecture#



Deep Learning

Unit-1: Linear Algebra for Machine/Deep Learning

Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India

Scalar

- Single number
  - In contrast to other objects in linear algebra, which are usually arrays of numbers
- Represented in **lower-case italics:** s, n, a, b, x
- Can be real-valued or be integers
  - E.g., let $s \in \mathbb{R}$ be the slope of the line
    - Defining a real-valued scalar
  - E.g., let $n \in \mathbb{N}$ be the number of data points
    - Defining a natural number scalar

	height	age
0	167	20
1	145	12
2	170	21
3	180	24
4	189	25
5	155	20
6	163	22
7	178	23
8	173	23
9	176	24

```
# Scalar
scalar = torch.tensor(9.42)
print("Scalar: ", scalar)
```

Output: `Scalar: tensor(9.4200)`

Scalar in NumPy

3

```python
import numpy as np
#Scalar
scalar2=9.24
print("The type of scalar1 is: ",type(scalar2))
print("The scalar1 is: ",scalar2)
print("Scalar2 is a scalar:", np.isscalar(scalar2))

scalar1=np.array(9.42)
print("\nThe type of scalar1 is: ",type(scalar1))
print("The scalar1 is: ",scalar1)
print("The dimension of the scalar1 is: ", scalar1.ndim)
print("The shape of the scalar1 is: ", scalar1.shape)
print("Scalar1 is a scalar:", np.isscalar(scalar1))
```

Output

```
The type of scalar1 is:  <class 'float'>
The scalar1 is:  9.24
Scalar2 is a scalar: True

The type of scalar1 is:  <class 'numpy.ndarray'>
The scalar1 is:  9.42
The dimension of the scalar1 is:  0
The shape of the scalar1 is:  ()
Scalar1 is a scalar: False
```


Vector

4

- 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

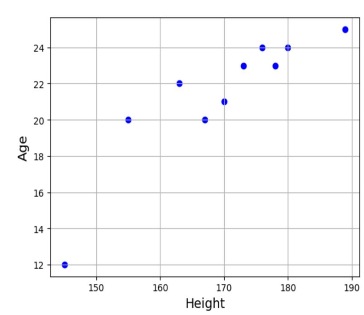
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

- If each element is in $\mathbb{R}$ then $\boldsymbol{x} \in \mathbb{R}^n$
- We can think of vectors as points in space
  – Each element gives coordinate along an axis
  – E.g. data points in the dataset

```python
# Vector
vector = torch.tensor([1, 2, 3])
print("Vector: ", vector)
```

Output: Vector: tensor([1, 2, 3])

	height	age
0	167	20
1	145	12
2	170	21
3	180	24
4	189	25
5	155	20
6	163	22
7	178	23
8	173	23
9	176	24



Vector in NumPy

```
#Vector
vector1 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print("\nThe vector1 is: ",vector1)
print("The dimension of the vector1 is: ",vector1.ndim)
print("The shape of the vector1 is: ",vector1.shape)
print("\nThe vector1 type is: ",type(vector1))
```

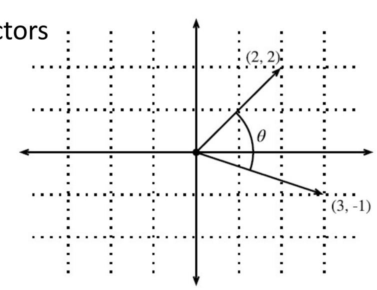
Output

```
The vector1 is:  [ 1  2  3  4  5  6  7  8  9 10]
The dimension of the vector1 is:  1
The shape of the vector1 is:  (10,)

The vector1 type is:  <class 'numpy.ndarray'>
```


Dot Product and Angles

- ***Dot product*** of vectors, $\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T\mathbf{v} = \sum_i u_i \cdot v_i$
  - It is also referred to as inner product, or scalar product of vectors
  - The dot product $\mathbf{u} \cdot \mathbf{v}$ is also often denoted by $\langle \mathbf{u}, \mathbf{v} \rangle$
- The dot product is a symmetric operation, $\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T\mathbf{v} = \mathbf{v}^T\mathbf{u} = \mathbf{v} \cdot \mathbf{u}$
- Geometric interpretation of a dot product: angle between two vectors
  - I.e., dot product $\mathbf{v} \cdot \mathbf{w}$ over the norms of the vectors is $\cos(\theta)$



$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\|\|\mathbf{v}\| cos(\theta) \qquad \cos\theta = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

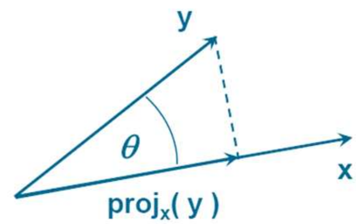
- If two vectors are orthogonal: $\theta = 90°$, i.e., $\cos(\theta) = 0$, then $\mathbf{u} \cdot \mathbf{v} = 0$
- Also, in ML the term $\cos\theta = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$ is sometimes employed as a measure of closeness of two vectors/data instances, and it is referred to as cosine similarity

Vector Projection

7

- ***Orthogonal projection*** of a vector $\mathbf{y}$ onto vector $\mathbf{x}$
  - The projection can take place in any space of dimensionality ≥ 2
  - The unit vector in the direction of $\mathbf{x}$ is $\frac{\mathbf{x}}{\|\mathbf{x}\|}$
    - A unit vector has norm equal to 1
  - The length of the projection of $\mathbf{y}$ onto $\mathbf{x}$ is $\|\mathbf{y}\| \cdot cos(\theta)$
  - The orthogonal project is the vector $\mathbf{proj_x}(\mathbf{y})$

$$\mathbf{proj_x}(\mathbf{y}) = \frac{\mathbf{x} \cdot \|\mathbf{y}\| \cdot cos(\theta)}{\|\mathbf{x}\|}$$



Norm of a Vector

8

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

$$\boxed{\begin{array}{ll} f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = 0 & \\ f(\boldsymbol{x}+\boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}) & \text{Triangle Inequality} \\ \forall \alpha \in R \quad f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x}) & \end{array}}$$

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

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Norm of a Vector

9

- For $p = 2$, we have ℓ_2 norm $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$
  - Also called **Euclidean norm**
  - It is the most often used norm
  - ℓ_2 norm is often denoted just as $\|\mathbf{x}\|$ without subscript 2

The squared ℓ_2 norm is more convenient to work with mathematically and computationally than the ℓ_2 norm itself.

- For $p = 1$, we have ℓ_1 norm $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$
  - Uses the absolute values of the elements
  - Discriminate between zero and non-zero elements

- In many contexts, the squared ℓ_2 norm may be undesirable
- It increases very slowly near the origin.
- Need a function that grows at the same rate in all locations: ℓ_1 norm

- For $p = \infty$, we have ℓ_∞ norm $\|\mathbf{x}\|_\infty = \max_i |x_i|$
  - Known as **infinity norm**, or **max norm**
  - Outputs the absolute value of the largest element
- ℓ_0 norm outputs the number of non-zero elements
  - It is not an ℓ_p norm, and it is not really a norm function either (it is incorrectly called a norm)

Matrices

10

- 2-D array of numbers
  - So each element identified by two indices

```
# Matrix
matrix = torch.tensor([[1, 2, 3], [4, 5, 6]])
print("Matrix: ", matrix)
```

Output: Matrix: tensor([[1, 2, 3],
        [4, 5, 6]])

- Denoted by **bold typeface (upper-case, italics)**: $\boldsymbol{A}$
  - Elements indicated by name in italics but not bold
    - $A_{1,1}$ is the top left entry and $A_{m,n}$ is the bottom right entry
    - E.g., $\boldsymbol{A} = [A_{i,j}] = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \cdots & \cdots & \cdots \\ A_{m,1} & \cdots & A_{m,n} \end{bmatrix}$
    - We can identify numbers in vertical column j by writing : for the horizontal coordinate
    - $A_{i:}$ is i th row of $\boldsymbol{A}$, $A_{:j}$ is j th column of $\boldsymbol{A}$
- If $\boldsymbol{A}$ has shape of height m and width n with real values then $\boldsymbol{A} \in \mathbb{R}^{m \times n}$

Matrix in NumPy

11

```
#Matrix
matrix1 = np.array([
        [1, 2],
        [3, 4]])
print("\nThe matrix1 is:\n",matrix1)
print("The dimension of the matrix1 is: ",matrix1.ndim)
print("The shape of the matrix1 is: ",matrix1.shape)
print("\nThe matrix1 type is: ",type(matrix1))
```

Output

```
The matrix1 is:
 [[1 2]
 [3 4]]
The dimension of the matrix1 is:  2
The shape of the matrix1 is:  (2, 2)

The matrix1 type is:  <class 'numpy.ndarray'>
```


Matrix Norms

12

- ***Frobenius norm***
  - measures the size of a matrix;
  - the square-root of the summed squares of the elements of matrix $\boldsymbol{X}$
  - This norm is similar to Euclidean norm of a vector

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij}^2}$$

- $\boldsymbol{L_{2,1}}$ ***norm*** – is the sum of the Euclidean norms of the columns of matrix $\boldsymbol{X}$

$$\|\mathbf{X}\|_{2,1} = \sum_{j=1}^{n}\sqrt{\sum_{i=1}^{m} X_{ij}^2}$$

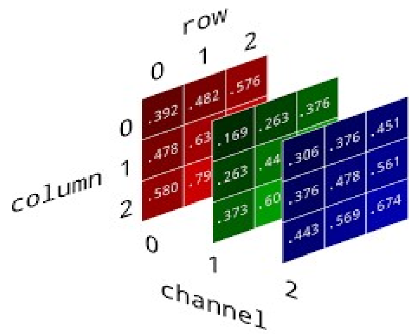
- ***Max norm*** – is the largest element of matrix $\boldsymbol{X}$

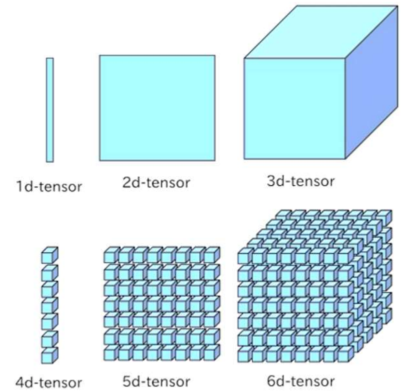
$$\|\mathbf{X}\|_{\max} = \max_{i,j}(X_{ij})$$

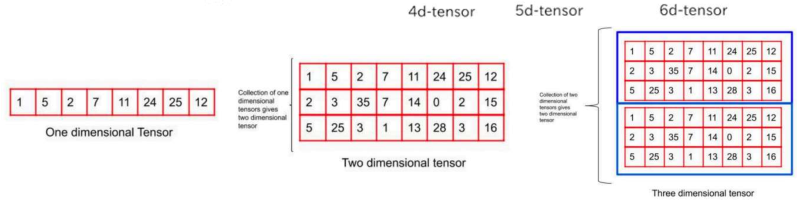
Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

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: $\mathbf{A}$
- Element (i, j, k) of the tensor is denoted by $\mathrm{A}_{i,j,k}$







Tensor in NumPy and PyTorch

```
#Tensor
tensor1 = np.array([[
        [1, 2, 3],
        [4, 5, 6]],
        [[7, 8, 9],
        [10, 11, 12]]])
print("\nThe tensor1 is:\n",tensor1 )
print("The dimension of the tensor1  is: ",tensor1 .ndim)
print("The shape of the tensor1  is: ",tensor1 .shape)
print("\nThe tensor1  type is: ",type(tensor1))
```

Output

```
The tensor1 is:
 [[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
The dimension of the tensor1  is:  3
The shape of the tensor1  is:  (2, 2, 3)

The tensor1  type is:  <class 'numpy.ndarray'>
```

```
# Tensor
tensor = torch.tensor([[[1, 2, 3], [4, 5, 6]],
                       [[7, 8, 9], [10, 11, 12]]])
print("Tensor: ", tensor)
```

Output:

```
Tensor:  tensor([[[ 1,  2,  3],
         [ 4,  5,  6]],

        [[ 7,  8,  9],
         [10, 11, 12]]])
```

Matrix Transpose

15

- The transpose of a matrix $\boldsymbol{A}$ is denoted as $\boldsymbol{A}^{\mathrm{T}}$
- Defined as $(\boldsymbol{A}^{\mathrm{T}})_{i,j} = \boldsymbol{A}_{j,i}$
  - The mirror image across a diagonal line
    - Called the main diagonal , running down to the right starting from upper left corner

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

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

- Vectors are matrices with a single column: $\boldsymbol{x} = [x_1,\ x_2,\ \dots\ x_n\ ]^{\mathrm{T}}$
- A scalar is a matrix with one element: $a = a^{\mathrm{T}}$

Matrix Addition

16

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

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

- A scalar can be added to a matrix or multiplied

$$\boxed{D = aB + c \Rightarrow D_{i,j} = aB_{i,j} + c}$$

- Less conventional notation used in ML:
  - Vector added to matrix $\boxed{C = A + \boldsymbol{b} \Rightarrow C_{i,j} = A_{i,j} + b_j}$
    - Called broadcasting since vector $\boldsymbol{b}$ added to each row of $\boldsymbol{A}$

Matrix Multiplication

17

- 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 $m \times n$ and $\boldsymbol{B}$ is of shape $n \times p$ then matrix product $\boldsymbol{C}$ is of shape $m \times p$

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

  - Note that the standard product of two matrices is not just the product of two individual elements
    - Such a product does exist and is called the element-wise product or the Hadamard product $\boldsymbol{A} \odot \boldsymbol{B}$
- Dot product between two vectors x and y of same dimensionality is the matrix product $x^{\mathrm{T}} y$

Matrix Product Properties

18

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

Linear Transformation

19

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

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

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

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

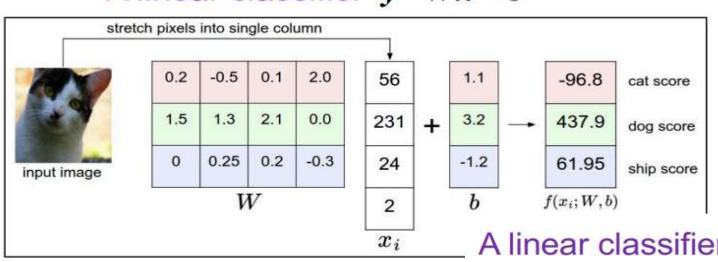
n x n $\qquad$ n x 1 $\qquad$ n x 1

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

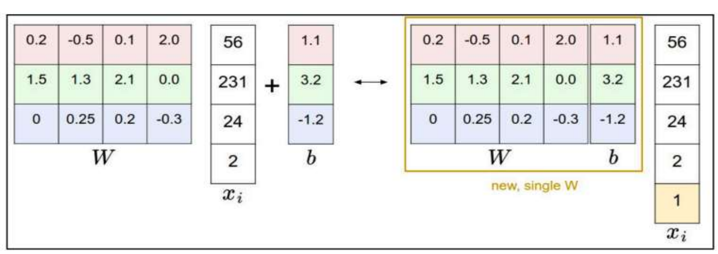
Example in the context of Machine Learning

20

A linear classifier $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}^{\mathrm{T}} + \boldsymbol{b}$



A linear classifier with bias eliminated $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}^{\mathrm{T}}$



Systems of Equations

21

- **Consider following equations:**

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m$$

- **Can be expressed in the form:**

$\mathbf{A}\mathbf{x} = \mathbf{b}$

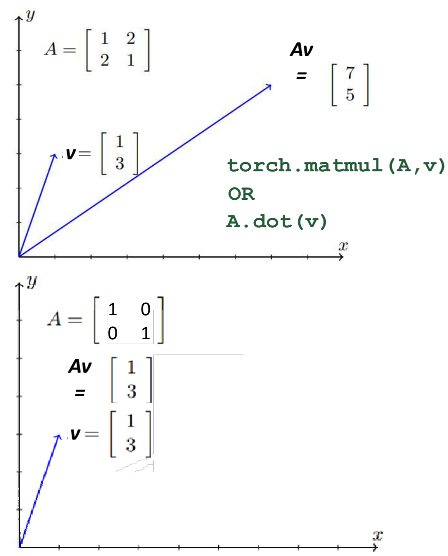
$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

- **A linear system of equations may have**
  - No solution
  - Infinite number of solutions
  - Exactly one solution
- If $\boldsymbol{x}$ and $\boldsymbol{y}$ are solutions then $\boldsymbol{z=\alpha\, x + (1-\alpha)\, y}$ is a solution for any real α

Identity Matrix

22

- 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 $\boldsymbol{I}_n$
  - Formally $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ and $\forall_{\boldsymbol{x}} \in \mathbb{R}^n, \boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{x}$
  - Example of $\boldsymbol{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$



Inverse Matrix

23

- Matrix inversion is **a powerful tool to analytically solve $\boldsymbol{Ax} = \boldsymbol{b}$**
- Inverse of square matrix $\boldsymbol{A}$ defined as $\boldsymbol{A^{-1}A} = \boldsymbol{I_n}$
- We can now solve $\boldsymbol{Ax} = \boldsymbol{b}$ as follows:

$$\boldsymbol{Ax} = \boldsymbol{b}$$
$$\boldsymbol{A}^{-1}\boldsymbol{Ax}=\boldsymbol{A}^{-1}\boldsymbol{b}$$
$$\boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$
$$\boldsymbol{x}=\boldsymbol{A}^{-1}\boldsymbol{b}$$

Matrix cannot be inverted if
- More rows than columns
- More columns than rows
- Redundant rows/columns

("linearly dependent", "low rank")

- This depends on being able to find $\boldsymbol{A^{-1}}$
- If $\boldsymbol{A^{-1}}$exists there are several methods for finding it
- **In order for A^{-1} to exist, $Ax = b$ must have exactly one solution for every value of b**
- **Is it possible to solve $Ax = b$ if A is not square or square but singular?**
  - **Yes**; Methods **other than matrix inversion** are used

Systems of Equations [contd.]

24

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m$$

$$\Rightarrow \quad x_1\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \cdot \\ \cdot \\ \cdot \\ a_{m1} \end{bmatrix} + x_2\begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \\ \cdot \\ \cdot \\ \cdot \\ a_{m2} \end{bmatrix} + x_3\begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \\ \cdot \\ \cdot \\ \cdot \\ a_{m3} \end{bmatrix} + \ldots\ldots + x_n\begin{bmatrix} a_{1n} \\ a_{2n} \\ a_{3n} \\ \cdot \\ \cdot \\ \cdot \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{bmatrix}$$

- $\boldsymbol{Ax} = \sum_i \boldsymbol{x_i A_{:,i}}$
- **Column** can be thought of as specifying ***direction from origin***
- **Each element** of $\boldsymbol{x}$ specify ***how far we should move*** in each of these direction
- Formally, this is a linear combination of the set of vectors
- **Span of set of vectors** is the set of all points obtainable by linear combination of the original vectors
- Hence, $\boldsymbol{Ax} = \boldsymbol{b}$ represents:
  - Testing whether $\boldsymbol{b}$ is in span of column of $\boldsymbol{A}$
  - Span is known as column space or range of $\boldsymbol{A}$

Linear Independence

25

- A set of n vectors $\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_n}$ is linearly independent if no vector in the set is a linear combination of the remaining $n-1$ vectors
- In other words, the only solution to $c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2} + c_3\boldsymbol{v_3} + \dots + c_n\boldsymbol{v_n} = 0$ is $c_1 = c_2 = \cdots = c_n = 0$ (c_i's are scalars)
- In general, **given a set of linearly independent vectors $\boldsymbol{u_1}, \boldsymbol{u_2}, \dots, \boldsymbol{u_n} \in \mathbb{R}^n$, we can express any vector $\boldsymbol{z} \in \mathbb{R}^{\boldsymbol{n}}$ as a linear combination of these vectors**.
  - This set of vectors is called a basis

$$z = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_n u_n$$

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \alpha_1 \begin{bmatrix} u_{11} \\ u_{12} \\ \vdots \\ u_{1n} \end{bmatrix} + \alpha_2 \begin{bmatrix} u_{21} \\ u_{22} \\ \vdots \\ u_{2n} \end{bmatrix} + \ldots + \alpha_n \begin{bmatrix} u_{n1} \\ u_{n2} \\ \vdots \\ u_{nn} \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} u_{11} & u_{21} & \ldots & u_{n1} \\ u_{12} & u_{22} & \ldots & u_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ u_{1n} & u_{2n} & \ldots & u_{nn} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}$$

- We can now find the α_i's using Gaussian Elimination (Time Complexity: O(n^3))
- If we have **orthonormal basis** the complexity will be O(n^2)

Special kinds of Matrices

26

- A **symmetric matrix** equals to its transpose: $\boldsymbol{A} = \boldsymbol{A}^{\mathrm{T}}$
  - E.g., a distance matrix is symmetric with $A_{i,j} = A_{j,i}$
- **Diagonal Matrix** has mostly zeros, with nonzero entries only in diagonal
  - $A_{i,j} = 0, i \neq j$
  - E.g., identity matrix, where all diagonal entries are 1
- diag($\boldsymbol{v}$) denotes a **square diagonal matrix** with diagonal elements given by entries of vector $\boldsymbol{v}$
- Multiplying vector $\boldsymbol{x}$ by a diagonal matrix is efficient
  - To compute diag($\boldsymbol{v}$)$\boldsymbol{x}$ we only need to scale each x_i by v_i
- **Inverting a square diagonal matrix is efficient**
  - Inverse exists iff every diagonal entry is nonzero, in which case diag $(\boldsymbol{v})^{-1}$=diag($[1/v_i, \ldots, 1/v_n]^{\mathrm{T}}$)

Special Kinds of Vectors

27

- **Unit Vector**
  - A vector with unit norm $\|x\|_2 = 1$
- **Orthogonal Vectors**
  - A vector $\boldsymbol{x}$ and a vector $\boldsymbol{y}$ are orthogonal to each other if $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} = 0$
  - If vectors have nonzero norm, vectors at 90 degrees to each other
- **Orthonormal Vectors**
  - Vectors are orthogonal and have unit norm
- **Orthogonal Matrix**
  - A square matrix whose rows are mutually orthonormal: $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{I}$
  - $\boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}}$

28

Questions?