Linear Algebra Review

Roi Yehoshua

Agenda

- Vectors
- Projections and orthogonality
- Matrices
- Properties of matrices: rank, norm, determinant, trace
- Eigenvalues and eigenvectors
- Quadratic forms and matrix definiteness
- Gradients, Hessian, and Jacobian
- Matrix calculus

Vectors

- A **vector** **x** in R^n is an ordered set of real numbers
  - *n* is called the **dimension** of the vector
- By convention, vectors are understood to be **column vectors**

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

- When we need to refer to a **row vector**, we use the transpose notation

$$\mathbf{x}^T = (x_1, x_2, \ldots, x_n)$$

Vectors

- In NumPy, vectors (one-dimensional arrays) are by default row vectors

```
import numpy as np
```

```
a = np.array([1, 2, 3, 4])
a
```

```
array([1, 2, 3, 4])
```

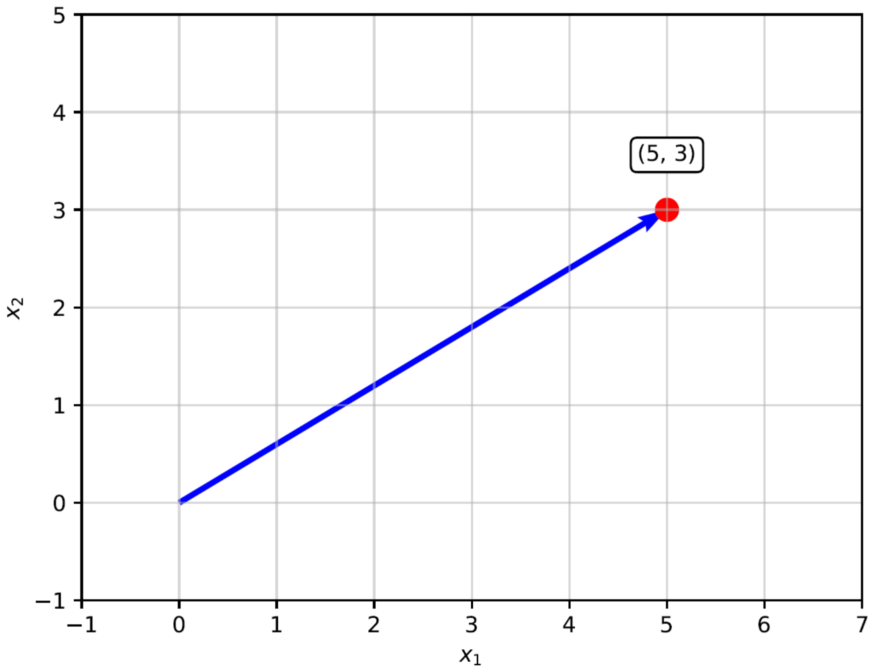
- To convert them to column vectors you need to use the **reshape()** method

```
a = a.reshape(-1, 1)
a
```

```
array([[1],
       [2],
       [3],
       [4]])
```

Vectors

- Vectors are often represented geometrically as arrows starting from the origin and extending to a point determined by its components



Basic Vector Operations

- Scalar multiplication

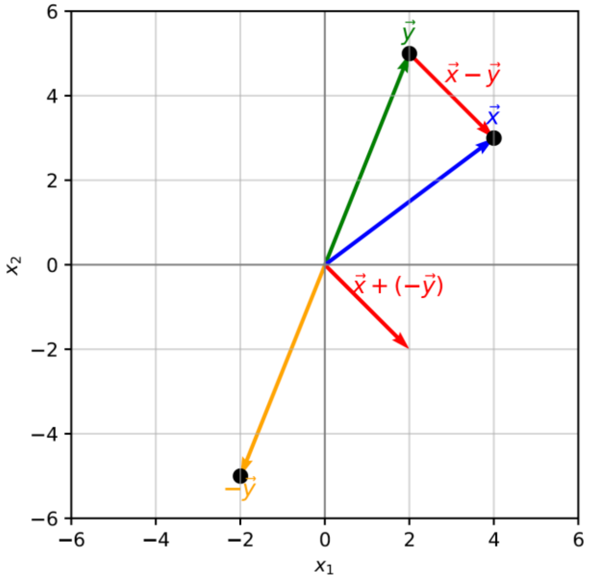
$$\alpha \mathbf{x} = (\alpha x_1, \alpha x_2, \ldots, \alpha x_n)$$

- Vector addition

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)$$

- Vector subtraction

$$\mathbf{x} - \mathbf{y} = (x_1 - y_1, x_2 - y_2, \ldots, x_n - y_n)$$



Vector-Vector Products

- **Dot product** or **inner product** of two vectors:

$$\mathbf{x}^T\mathbf{y} = (x_1, ..., x_n)\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \sum_{i=1}^{n} x_i y_i$$

```python
import numpy as np
```

```python
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

np.dot(a, b)
```

```
32
```

- **Outer product** of two vectors:

$$\mathbf{x}\mathbf{y}^T = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}(y_1, ..., y_n) = \begin{pmatrix} x_1y_1 & x_1y_2 & \cdots & x_1y_n \\ x_2y_1 & x_2y_2 & \cdots & x_2y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_my_1 & x_my_2 & \cdots & x_my_n \end{pmatrix}$$

```python
np.outer(a, b)
```

```
array([[ 4,  5,  6],
       [ 8, 10, 12],
       [12, 15, 18]])
```

- **Element-wise product** (Hadamard product)

$$\mathbf{x} \odot \mathbf{y} = (x_1, x_2, \ldots, x_n) \odot (y_1, y_2, \ldots, y_n) = (x_1y_1, x_2y_2, \ldots, x_ny_n)$$

```python
a * b
```

```
array([ 4, 10, 18])
```

Special Vectors

- The zero vector

$$\mathbf{0} = (0, 0, \ldots, 0)$$

- The ones vector

$$\mathbf{1} = (1, 1, \ldots, 1)$$

- Standard basis vectors

$$\mathbf{e}_1 = (1, 0, 0), \quad \mathbf{e}_2 = (0, 1, 0), \quad \mathbf{e}_3 = (0, 0, 1)$$

```
np.zeros(5)
```

```
array([0., 0., 0., 0., 0.])
```

```
np.ones(4)
```

```
array([1., 1., 1., 1.])
```

Vector Norms

- A **norm** of a vector $||\mathbf{x}||$ measures its length or magnitude
- The most commonly used norms are the L_p norms (**Minkowski norm**)

$$||\mathbf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

- L_1 norm (Manhattan norm): $$||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|$$
- L_2 norm (Euclidean norm): $$||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$
- L_∞ norm (Maximum norm): $$||\mathbf{x}||_\infty = \max_i |x_i|$$

```
a = np.array([1, 2, 3])
np.linalg.norm(a) # Euclidean norm
```

```
3.7416573867739413
```

```
np.linalg.norm(a, 1) # L1 norm
```

```
6.0
```

```
np.linalg.norm(a, np.inf)
```

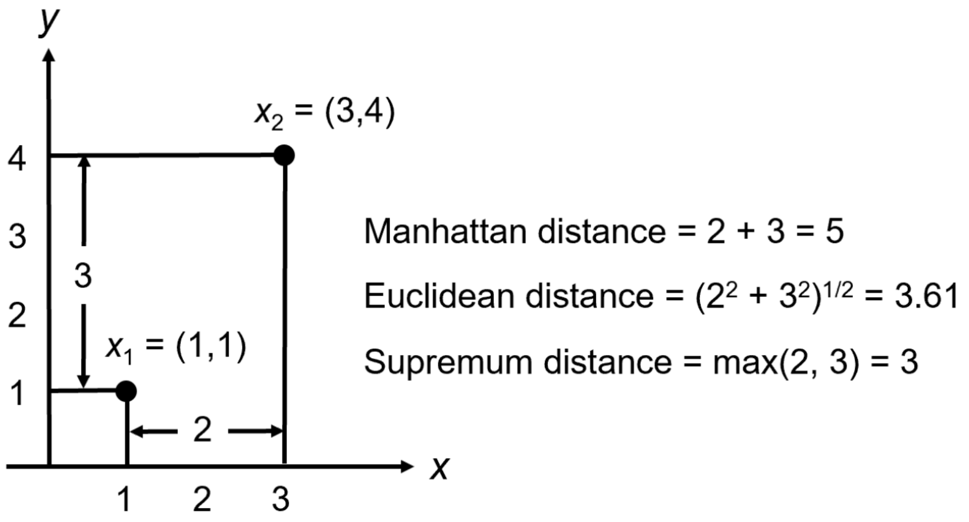
```
3.0
```

Distances between Vectors

- The distance between two vectors is the norm of their difference:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$$

- For example, Euclidean distance: $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$



Angles between Vectors

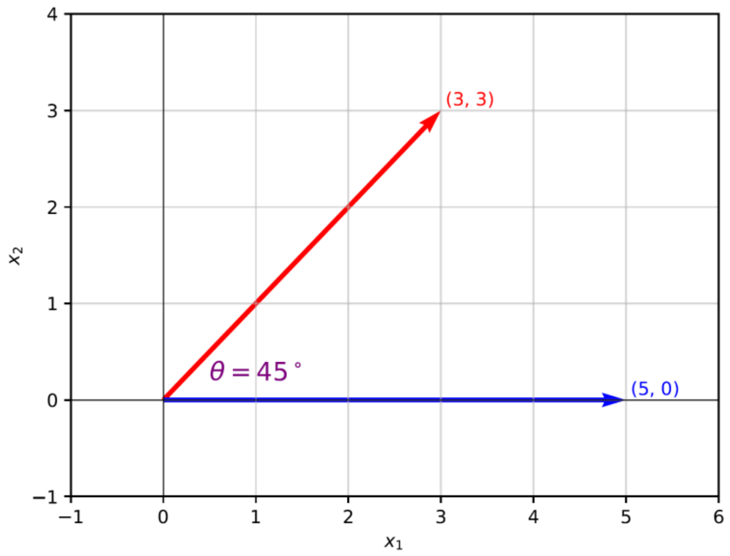
- The **angle** between two vectors **x** and **y** is

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||\,||\mathbf{y}||}$$

- For example, the angle between **x** = (3, 3) and **y** = (5, 0) is:

$$\cos\theta = \frac{3\cdot5+3\cdot0}{\sqrt{3^2+3^2}\cdot\sqrt{5^2+0^2}} = \frac{15}{\sqrt{18}\cdot\sqrt{25}} = \frac{1}{\sqrt{2}}$$

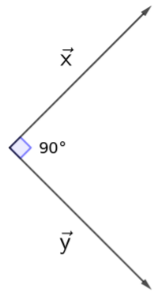
$$\theta = \arccos\left(\frac{1}{\sqrt{2}}\right) = \frac{\pi}{4} = 45°$$



Orthogonal Vectors

- Two vectors are **orthogonal** if the angle between them is 90°
  - Or equivalently, their dot product is 0
  - For example, **x** = (1, 2, 2) and **y** = (2, 3, -4) are orthogonal since:

$$\mathbf{x}^T\mathbf{y} = 1 \cdot 2 + 2 \cdot 3 + 2 \cdot (-4) = 0$$



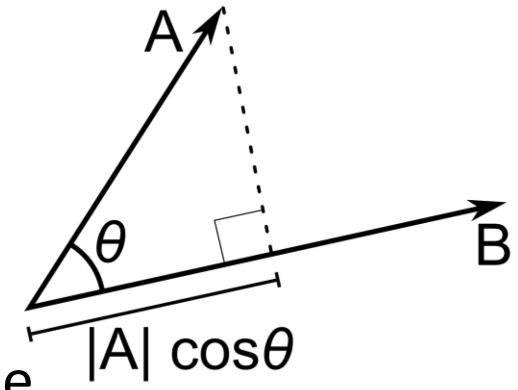
$$\vec{x} \cdot \vec{y} = |\vec{x}||\vec{y}|\cos(90^\circ) = 0$$

- Two vectors are **orthonormal** if they are orthogonal to each other and both have a unit length (norm 1)
  - Orthonormal vectors are especially useful for forming basis in vector spaces

Projections

- Projections measure the component of one vector in the direction of another
- The **scalar projection** of a vector **x** onto vector **y** gives the size of this component

$$\text{comp}_{\mathbf{y}}\, \mathbf{x} = \|\mathbf{x}\| \cos\theta = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{y}\|}$$



- The **vector projection** of **x** onto **y** gives both the direction and size

$$\text{proj}_{\mathbf{y}}\, \mathbf{x} = \text{comp}_{\mathbf{y}}\, \mathbf{x} \left(\frac{\mathbf{y}}{\|\mathbf{y}\|}\right) = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}$$

Linear Independence

- A set of vectors {$\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$} is **linearly dependent** if one of the vectors can be represented as a linear combination of the remaining vectors:

$$\mathbf{x}_n = \sum_{i=1}^{n-1} \alpha_i \mathbf{x}_i$$

  - for some scalar values $\alpha_1, \ldots, \alpha_{n-1} \in \mathbb{R}$
- Otherwise, the vectors are **linearly independent**
- For example, the vectors

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} 4 \\ 1 \\ 5 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} 2 \\ -3 \\ -1 \end{pmatrix}$$

are linearly dependent because $\mathbf{x}_3 = -2\mathbf{x}_1 + \mathbf{x}_2$

Matrices

- By $A \in \mathbb{R}^{m \times n}$ we denote a real-valued **matrix** with m rows and n columns:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} — & \mathbf{a}_1^T & — \\ — & \mathbf{a}_2^T & — \\ & \vdots & \\ — & \mathbf{a}_m^T & — \end{pmatrix}$$

- Examples:

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
A
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
A = np.arange(9).reshape(3, 3)
A
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Basic Matrix Operations

- Scalar multiplication

$$kA = \begin{pmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{pmatrix}$$

- Matrix addition

$$A + B = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{pmatrix}$$

- Matrix subtraction

$$A - B = \begin{pmatrix} a_{11} - b_{11} & a_{12} - b_{12} & \cdots & a_{1n} - b_{1n} \\ a_{21} - b_{21} & a_{22} - b_{22} & \cdots & a_{2n} - b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} - b_{m1} & a_{m2} - b_{m2} & \cdots & a_{mn} - b_{mn} \end{pmatrix}$$

Matrix Multiplication

- If $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$ then $C = AB \in \mathbb{R}^{m \times p}$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$A = \begin{pmatrix} 2 & 1 \\ 5 & 2 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 4 \\ 3 & 6 \end{pmatrix}$$

$$AB = \begin{pmatrix} 2 \cdot 1 + 1 \cdot 3 & 2 \cdot 4 + 1 \cdot 6 \\ 5 \cdot 1 + 2 \cdot 3 & 5 \cdot 4 + 2 \cdot 6 \end{pmatrix} = \begin{pmatrix} 5 & 14 \\ 11 & 32 \end{pmatrix}$$

```python
A = np.array([[2, 1], [5, 2]])
B = np.array([[1, 4], [3, 6]])

A @ B
```

```
array([[ 5, 14],
       [11, 32]])
```

- Properties
  - **Associative**: $(AB)C = A(BC)$
  - **Distributive**: $A(B + C) = AB + AC$
  - **Not commutative** (in general), i.e., it can be the case that $AB \neq BA$
    - e.g., the matrix BA doesn't even exist if $m \neq p$

Matrix-Vector Product

- Multiplying a matrix $A \in \mathbb{R}^{m \times n}$ by a vector $\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{y} = A\mathbf{x} = \begin{pmatrix} — & \mathbf{a}_1^T & — \\ — & \mathbf{a}_2^T & — \\ & \vdots & \\ — & \mathbf{a}_m^T & — \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathbf{a}_1^T \mathbf{x} \\ \mathbf{a}_2^T \mathbf{x} \\ \vdots \\ \mathbf{a}_m^T \mathbf{x} \end{pmatrix}$$

- Example:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 5 \\ 5 & 6 & 0 \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$$

$$A\mathbf{x} = \begin{pmatrix} 1 \cdot 2 + 2 \cdot 1 + 3 \cdot 3 \\ 0 \cdot 2 + 1 \cdot 1 + 5 \cdot 3 \\ 5 \cdot 2 + 6 \cdot 1 + 0 \cdot 3 \end{pmatrix} = \begin{pmatrix} 13 \\ 16 \\ 16 \end{pmatrix}$$

```
A = np.array([[1, 2, 3],
              [0, 1, 5],
              [5, 6, 0]])
x = np.array([2, 1, 3])

A @ x
```

```
array([13, 16, 16])
```

The Identity Matrix

- The **identity matrix** $I \in \mathbb{R}^{n \times n}$ is a square matrix with ones on the diagonal and zeros elsewhere

$$I_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

```
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

- For every matrix $A \in R^{m \times n}$,

$$AI_n = I_m A = A$$

Diagonal Matrices

- A **diagonal matrix** is a matrix where all off-diagonal elements are 0
- This is typically denoted $D = \text{diag}(d_1, d_2, ..., d_n)$ with

$$D_{ij} = \begin{cases} d_i & i = j \\ 0 & i \neq j \end{cases}$$

- Example:

```
np.diag([7, 3, 2, 8])
```

```
array([[7, 0, 0, 0],
       [0, 3, 0, 0],
       [0, 0, 2, 0],
       [0, 0, 0, 8]])
```

Triangular Matrices

- **Lower diagonal matrix**: all entries above the diagonal are zero

$$L = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix}$$

- **Upper diagonal matrix**: all entries below the diagonal are zero

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

Transpose

- The **transpose** of a matrix flips its rows and columns
- Given a matrix $A \in \mathbb{R}^{m \times n}$, its transpose is the $n \times m$ matrix whose entries are

$$(A^T)_{ij} = A_{ji}$$

- Properties of the transpose:

$$(A^T)^T = A$$

$$(A + B)^T = A^T + B^T$$

$$(AB)^T = B^T A^T$$

```
A = np.arange(9).reshape(3, 3)
A
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

```
A.T
```

```
array([[0, 3, 6],
       [1, 4, 7],
       [2, 5, 8]])
```

- A square matrix $A \in \mathbb{R}^{m \times n}$ is **symmetric** if it is equal to its transpose

$$A = A^T$$

Trace of a Matrix

- The **trace** of a square matrix $A \in \mathbb{R}^{n \times n}$ is the sum of the its diagonal elements:

$$\text{tr}(A) = \sum_{i=1}^{n} A_{ii}$$

- Example:

```
A = np.arange(9).reshape(3, 3)
A.trace()
```

```
12
```

- Properties:
  - $\text{tr}(A) = \text{tr}(A^T)$
  - $\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B)$
  - For $k \in \mathbb{R}$, $\text{tr}(kA) = k\text{tr}(A)$
  - For A, B such that AB is square $\text{tr}(AB) = \text{tr}(BA)$

System of Linear Equations

- A linear system of *m* equations in *n* variables:

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

- Can be expressed as the matrix equation $A\mathbf{x} = \mathbf{b}$:

$$
A\mathbf{x} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{n2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} = \mathbf{b}
$$

  - $A \in \mathbb{R}^{m \times n}$ is the coefficient matrix
  - $\mathbf{x} \in \mathbb{R}^n$ is the variable vector
  - $\mathbf{b} \in \mathbb{R}^m$ is the constant vector

Gaussian Elimination

- Use row operations to reduce A to a **row echelon form**

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

- Row operations include
  - Row swapping
  - Row scaling (multiplying a row by a nonzero scalar)
  - Adding a multiple of one row to another row

Gaussian Elimination Example

- Solve the following system

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 9 \\ 2x_1 + 4x_2 + 7x_3 = 20 \\ 3x_1 + 5x_2 + 6x_3 = 22 \end{cases}$$

- The augmented matrix is:

$$(A|\mathbf{b}) = \left(\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 2 & 4 & 7 & 20 \\ 3 & 5 & 6 & 22 \end{array}\right)$$

- Eliminate the entries below the pivot in the first column:

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 2 & 4 & 7 & 20 \\ 3 & 5 & 6 & 22 \end{array}\right) \xrightarrow[]{\substack{R_2 \leftarrow R_2 - 2R_1 \\ R_3 \leftarrow R_3 - 3R_1}} \left(\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 0 & 0 & 1 & 2 \\ 0 & -1 & -3 & -5 \end{array}\right)$$

Gaussian Elimination Example

- Swapping the second and third rows:

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 0 & 0 & 1 & 2 \\ 0 & -1 & -3 & -5 \end{array}\right) \xrightarrow{R_2 \leftrightarrow R_3} \left(\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 0 & -1 & -3 & -5 \\ 0 & 0 & 1 & 2 \end{array}\right)$$

- Back-substitution

$$\begin{aligned} x_3 &= 2 \\ -x_2 - 3x_3 &= -5 \;\Rightarrow\; -x_2 - 6 = -5 \;\Rightarrow\; x_2 = -1 \\ x_1 + 2x_2 + 3x_3 &= 9 \quad \Rightarrow\; x_1 + 2 \cdot (-1) + 3 \cdot 2 = 9 \;\Rightarrow\; x_1 = 5 \end{aligned}$$

- Thus, the solution is

$$\mathbf{x} = (5, -1, 2)^T$$

Solving Linear Equations in NumPy

- **np.linalg.solve**(*A*, *b*) computes the solution of the linear matrix equation *Ax* = *b*
  - If no unique solution exists (for nonsquare or singular matrix *A*), a LinAlgError is raised
- For example, let's find a solution to the same system of equations:

```
A = np.array([[1, 2, 3],
              [2, 4, 7],
              [3, 5, 6]])
b = np.array([9, 20, 22])
np.linalg.solve(A, b)
```

```
array([ 5., -1.,  2.])
```

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 9 \\ 2x_1 + 4x_2 + 7x_3 = 20 \\ 3x_1 + 5x_2 + 6x_3 = 22 \end{cases}$$

Rank of a Matrix

- The **column rank** of a matrix *A* is its largest number of linearly independent columns
- The **row rank** of *A* is its largest number of linearly independent rows
- **The rank theorem:** the column rank is always equal to the row rank
  - Both quantities are referred to collectively as the **rank** of A, denoted rank(*A*)
- Example:

$$A = \begin{pmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{pmatrix}$$

```
A = np.array([[1, 2, 1],
              [-2, -3, 1],
              [3, 5, 0]])
```

```
np.linalg.matrix_rank(A)
```

```
2
```

  - rank(*A*) = 2, as there are only 2 linearly independent rows

$$R_3 = R_1 - R_2$$

$$C_3 = -5C_1 + 3C_2$$

Rank of a Matrix

- Properties of the rank:
  - For $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) \leq \min(m, n)$
    - If $\text{rank}(A) = \min(m, n)$ the matrix is said to have **full rank**
  - $\text{rank}(A) = \text{rank}(A^T)$
  - $\text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$
  - $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$

Inverse of a Matrix

- The **inverse** of a square matrix $A \in \mathbb{R}^{n \times n}$, denoted A^{-1}, is the unique matrix such that

$$A^{-1}A = I = AA^{-1}$$

- A is **invertible** or **nonsingular** if A^{-1} exists
  - Otherwise, it is **non-invertible** or **singular**
- A is invertible if and only if it has full rank
- To compute the inverse, apply Gaussian elimination on the augmented matrix [A|I]

$$[A|I] \rightarrow [I|A^{-1}]$$

```
A = np.array([[1, 2, 3],
              [0, 1, 5],
              [5, 6, 0]])
```

```
np.linalg.inv(A)
```

```
array([[-6. ,  3.6,  1.4],
       [ 5. , -3. , -1. ],
       [-1. ,  0.8,  0.2]])
```

Inverse of a Matrix

- For any two invertible matrices

$$\text{(i)}\quad (A^{-1})^{-1} = A$$

$$\text{(ii)}\quad (AB)^{-1} = B^{-1}A^{-1}$$

$$\text{(iii)}\quad (A^{-1})^T = (A^T)^{-1}$$

- If A is an invertible matrix, then the solution to the linear system $A\mathbf{x} = \mathbf{b}$ is

$$\mathbf{x} = A^{-1}\mathbf{b}$$

Moore-Penrose Pseudoinverse

- Pseudoinverse of $A \in R^{m \times n}$, denoted A^+, generalizes the inverse to any matrix
  - Including non-square and nonsingular matrices
- If a linear system $A\mathbf{x} = \mathbf{b}$ has no solution, $A^+\mathbf{b}$ gives the **least-squares solution**
  - i.e., the solution that minimizes the squared norm of the error (residual)

$$\mathbf{x}^* = A^+\mathbf{b} = \underset{\mathbf{x}}{\text{argmin}} \|A\mathbf{x} - \mathbf{b}\|^2$$

```
A = np.array([[1, 2],
              [3, 4],
              [5, 6]])
np.linalg.pinv(A)
```

```
array([[-1.33333333, -0.33333333,  0.66666667],
       [ 1.08333333,  0.33333333, -0.41666667]])
```

The Determinant

- The **determinant** of a square matrix $A \in \mathrm{R}^{n \times n}$ is a scalar denoted by $|A|$ or $\det(A)$
- In the case of a 2 × 2 matrix the determinant is computed by:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

- The general recursive formula for the determinant is:

$$|A| = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} |A_{\backslash i, \backslash j}| \quad (\text{for any } j \in 1, \ldots, n)$$
$$= \sum_{j=1}^{n} (-1)^{i+j} a_{ij} |A_{\backslash i, \backslash j}| \quad (\text{for any } i \in 1, \ldots, n)$$

  - $A_{\backslash i, \backslash j}$ is the submatrix that results from deleting the *i*-th row and *j*-th column from *A*
    - The determinant of this matrix is called a **minor** of *A*

The Determinant

- Example:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 5 \\ 5 & 6 & 0 \end{pmatrix}$$

$$|A| = 1 \cdot (1 \cdot 0 - 5 \cdot 6) - 2 \cdot (0 \cdot 0 - 5 \cdot 5) + 3 \cdot (0 \cdot 6 - 1 \cdot 5)$$
$$= -30 + 50 - 15 = 5$$

```python
A = np.array([[1, 2, 3],
              [0, 1, 5],
              [5, 6, 0]])
```

```python
np.linalg.det(A)
```

```
4.999999999999995
```

- Determinant properties:
  - $|I| = 1$
  - $|A| = |A^T|$
  - $|AB| = |A||B|$
  - $|A| = 0$ if and only if A is singular
  - If A is nonsingular, $|A^{-1}| = 1/|A|$
  - The determinant of a triangular matrix equals the product of its diagonal elements

Orthogonal Matrix

- A square matrix $Q \in \mathbb{R}^{n \times n}$ is **orthogonal** if all its columns (and rows) are orthonormal
  - i.e., they are orthogonal to each other and have a unit length
- Example:

$$Q = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

- Properties:
  - The inverse of an orthogonal matrix is its transpose: $Q^T Q = Q Q^T = I$
  - The determinant of an orthogonal matrix is either +1 or -1

$$\det(Q) = \pm 1$$

  - Preserves the dot products between vectors and the Euclidean norm of vectors

$$(Q\mathbf{x})^T (Q\mathbf{y}) = \mathbf{x}^T \mathbf{y}$$

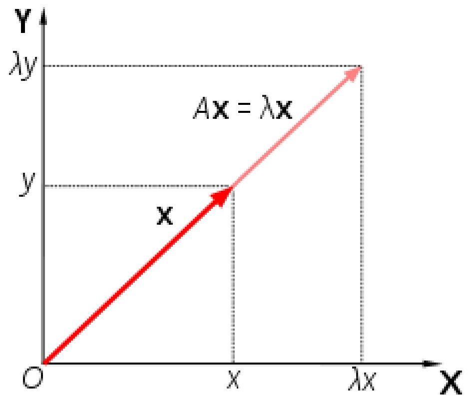
$$||Q\mathbf{x}||_2 = ||\mathbf{x}||_2$$

Eigenvalues and Eigenvectors

- Given a square matrix $A \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue** of A and $\mathbf{x} \neq \mathbf{0} \in \mathbb{C}^n$ is its corresponding **eigenvector** if

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

  - The pair $(\lambda, \mathbf{x})$ is called an **eigenpair**
- $A\mathbf{x}$ scales the vector $\mathbf{x}$ without changing its direction



- Each eigenvalue λ has infinitely many corresponding eigenvectors
  - Since any nonzero scalar multiple of $\mathbf{x}$ is also an eigenvector corresponding to λ
  - To avoid ambiguity, eigenvectors are often normalized to unit length

Eigenvalues and Eigenvectors

- To find the eigenpairs of *A*, we first rewrite the eigenvalue equation

$$(A - \lambda I)\mathbf{x} = \mathbf{0}$$

- For this equation to have a nontrivial solution, $A - \lambda I$ must be singular, i.e.,

$$|A - \lambda I| = 0$$

  - This is called the **characteristic equation** of *A*
- The solution is an *n*-degree polynomial in λ called the **characteristic polynomial**
  - Its roots give the *n* (possibly complex) eigenvalues of A
- The corresponding of eigenvector λ_i can be found by solving the system

$$(A - \lambda_i I)\mathbf{x} = \mathbf{0}$$

Eigenvalues and Eigenvectors

- For example, consider the matrix $$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$
- To find the eigenvalues of A, we compute $|A - \lambda I|$ and set it equal to 0:

$$|A - \lambda I| = \begin{vmatrix} 2-\lambda & 1 & 0 \\ 1 & 2-\lambda & 1 \\ 0 & 1 & 2-\lambda \end{vmatrix} = 0$$

$$(2-\lambda)\begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} - \begin{vmatrix} 1 & 1 \\ 0 & 2-\lambda \end{vmatrix} = 0$$

$$(2-\lambda)[(2-\lambda)^2 - 1] - (2-\lambda) = 0$$

$$(2-\lambda)(\lambda^2 - 4\lambda + 2) = 0$$

$$\lambda_1 = 2, \lambda_2 = 2 - \sqrt{2}, \lambda_3 = 2 + \sqrt{2}$$

Eigenvalues and Eigenvectors

- To find the corresponding eigenvectors we need to solve the following system of equations for each eigenvalue: $A\mathbf{x} = \lambda\mathbf{x}$

$$(A - \lambda I)\mathbf{x} = \mathbf{0}$$

- For example, for $\lambda = 2$ we get:

$$(A - \lambda I)\mathbf{x} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \begin{matrix} x_2 = 0 \\ x_1 + x_3 = 0 \end{matrix} \Rightarrow \mathbf{x} = \begin{pmatrix} a \\ 0 \\ -a \end{pmatrix}$$

- For it to be a unit vector we need $2a^2 = 1 \Rightarrow a = 1/\sqrt{2}$

$$\hat{\mathbf{x}} = \begin{pmatrix} 1/\sqrt{2} \\ 0 \\ -1/\sqrt{2} \end{pmatrix}$$

Eigenvalues and Eigenvectors

- **np.linalg.eig**(*A*) computes the eigenvalues and eigenvectors of the matrix *A*
  - The eigenvectors are returned as normalized column vectors

```
A = np.array([[2, 1, 0],
              [1, 2, 1],
              [0, 1, 2]])
```

```
eigen_vals, eigen_vecs = np.linalg.eig(A)
eigen_vals
```

```
array([3.41421356, 2.        , 0.58578644])
```

```
eigen_vecs
```

```
array([[-5.00000000e-01,  7.07106781e-01,  5.00000000e-01],
       [-7.07106781e-01,  4.05925293e-16, -7.07106781e-01],
       [-5.00000000e-01, -7.07106781e-01,  5.00000000e-01]])
```

Properties of Eigenvalues and Eigenvectors

- The trace of A is equal to the sum of its eigenvalues

$$\mathrm{tr}(A) = \sum_{i=1}^{n} \lambda_i$$

- The determinant of A is equal to the product of its eigenvalues

$$|A| = \prod_{i=1}^{n} \lambda_i$$

- If A is nonsingular with eigenvalue λ associated with eigenvector **x**, then $1/\lambda$ is an eigenvalue of A^{-1} with an associated eigenvector **x**
- The eigenvalues of a diagonal/triangular matrix are its diagonal entries
- If $A \in \mathrm{R}^{n \times n}$ is a symmetric matrix
  - All eigenvalues $\lambda_1, \dots, \lambda_n$ of A are real numbers (not necessarily distinct)
  - It has a full set of orthogonal eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_n$

Eigendecomposition of a Matrix

- Let V be a matrix whose columns are the eigenvectors of A and Λ a diagonal matrix with the corresponding eigenvalues then

$$AV = V\Lambda$$

- If V is invertible (i.e., A has n linearly independent eigenvectors) then

$$A = V\Lambda V^{-1}$$

  - This is called an **eigendecomposition** of A
  - If such decomposition exists, A is said to be **diagonalizable**
- This decomposition can greatly simplify computations, e.g., of powers of A

$$A^k = V\Lambda^k V^{-1}$$

The Spectral Theorem

- If A is a symmetric matrix, then it is diagonalizable with an orthogonal matrix

$$A = Q\Lambda Q^T$$

  - This is called a **spectral decomposition** of A
  - The columns of Q form an orthonormal basis of eigenvectors

```
A = np.array([[4, 1, 2],
              [1, 3, 0],
              [2, 0, 2]])

λ, Q = np.linalg.eigh(A)
λ
```

```
array([0.63853123, 2.83255081, 5.52891796])
```

```
Q
```

```
array([[-0.54739786,  0.15351016, -0.8226726 ],
       [ 0.23180398, -0.91675668, -0.32530617],
       [ 0.80412841,  0.36877068, -0.46624638]])
```

Matrix Norms

- Norms can also be defined for matrices
- **Frobenius norm** is analogous to Euclidean norm of vectors:
  - The most commonly used norm

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2} = \sqrt{\mathrm{tr}(A^T A)}$$

```
A = np.arange(9).reshape(3, 3)
np.linalg.norm(A)
```

```
14.2828568570857
```

Quadratic Forms

- Given a square matrix and a vector, the scalar value $\mathbf{x}^T A\mathbf{x}$ is called a **quadratic form**
- We can write it explicitly as follows:

$$\mathbf{x}^T A\mathbf{x} = \sum_{i=1}^{n} x_i (A\mathbf{x})_i = \sum_{i=1}^{n} x_i \left(\sum_{j=1}^{n} A_{ij} x_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} x_i x_j$$

- For example, if $A = \begin{pmatrix} 3 & -2 \\ -2 & 7 \end{pmatrix}$
- Then

$$\mathbf{x}^T A\mathbf{x} = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 3 & -2 \\ -2 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 3x_1^2 - 4x_1 x_2 + 7x_2^2$$

Matrix Definiteness

- A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is:
  - **positive definite** (PD) if for all nonzero vectors $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^T A \mathbf{x} > 0$
    - all eigenvalues must be positive
  - **positive semidefinite** (PSD) if for all vectors $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^T A \mathbf{x} \geq 0$
    - all eigenvalues must be nonnegative
  - **negative definite** (ND) if for all nonzero vectors $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^T A \mathbf{x} < 0$
    - all eigenvalues must be negative
  - **negative semidefinite** (NSD) if for all vectors $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^T A \mathbf{x} \leq 0$
    - all eigenvalues must be nonpositive
  - **indefinite** if it is neither positive semidefinite nor negative semidefinite
    - A has both positive and negative eigenvalues
- Matrix definiteness plays a central role in optimization

Matrix Definiteness

- <u>Example</u>: show that the following matrix is positive definite

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

- For any nonzero vector **x** ∈ R²

$$\mathbf{x}^T A \mathbf{x} = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 2x_1^2 - 2x_1 x_2 + 2x_2^2$$

- Completing the square:

$$\mathbf{x}^T A \mathbf{x} = 2(x_1^2 - x_1 x_2 + x_2^2) = 2\left[\left(x_1 - \frac{x_2}{2}\right)^2 + \frac{3}{4}x_2^2\right]$$

- The expression is nonnegative and equals zero only when both $x_1 = x_2 = 0$, thus

$$\mathbf{x}^T A \mathbf{x} > 0$$

Matrix Calculus

- Generalizes differential calculus to functions involving vectors and matrices
- Allows to use linear algebra to compute derivatives in compact matrix form
  - Instead of computing partial derivatives component-by-component
- Greatly simplifies operations such as finding the maximum of multivariate functions
- Used in different areas of machine learning
  - Closed-form solution to linear regression
  - Automatic differentiation (neural network training)

Reminder: Gradient of a Multivariable Function

- A **scalar-valued** function f: R$^n \to$ R maps a vector **x** $\in$ R^n to a scalar in R
- The gradient of f is the vector that contains all partial derivatives of f

$$\nabla_{\mathbf{x}} f = \frac{\partial f}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

- Example: $f(\mathbf{x}) = f(x_1, x_2) = x_1^2 + x_1 x_2 + 2x_2$

$$\nabla_{\mathbf{x}} f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 \\ x_1 + 2 \end{pmatrix}$$

Reminder: Gradient of a Multivariable Function

- What is the gradient of the squared norm function?

$$f(\mathbf{x}) = ||\mathbf{x}||^2 = x_1^2 + \cdots + x_n^2$$

- Solution:
  - For every $i = 1, ..., n$

$$\frac{\partial f}{\partial x_i} = \frac{\partial \left(\sum_{k=1}^n x_k^2\right)}{\partial x_i} = 2x_i$$

- Thus the gradient is:

$$\nabla_{\mathbf{x}} f = 2\mathbf{x}$$

Rules for Computing Gradients

- The following rules follow directly from the properties of partial derivatives
- Constant rule $\nabla_{\mathbf{x}} c = \mathbf{0}, \quad \text{for any constant } c$
- Sum rule $\nabla_{\mathbf{x}}(f + g) = \nabla_{\mathbf{x}} f + \nabla_{\mathbf{x}} g$
- Scalar multiplication rule $\nabla_{\mathbf{x}}(af) = a\nabla_{\mathbf{x}} f$
- Product rule $\nabla_{\mathbf{x}}(fg) = f(\mathbf{x})\nabla_{\mathbf{x}} g + g(\mathbf{x})\nabla_{\mathbf{x}} f$
  - where

$$f(\mathbf{x})\nabla_{\mathbf{x}} g = \begin{pmatrix} f(\mathbf{x})\frac{\partial g}{\partial x_1} \\ f(\mathbf{x})\frac{\partial g}{\partial x_2} \\ \vdots \\ f(\mathbf{x})\frac{\partial g}{\partial x_n} \end{pmatrix}$$

Rules for Computing Gradients

- Example: given the functions

$$f(\mathbf{x}) = x_1^2 + x_2^2, \quad g(\mathbf{x}) = x_1 x_2$$

- We want to compute the gradient of their product

$$\nabla_{\mathbf{x}} \left(f(\mathbf{x}) \cdot g(\mathbf{x}) \right) = \nabla_{\mathbf{x}} \left((x_1^2 + x_2^2)(x_1 x_2) \right)$$

- Using the product rule:

$$\nabla_{\mathbf{x}}(fg) = f(\mathbf{x}) \cdot \nabla_{\mathbf{x}} g + g(\mathbf{x}) \cdot \nabla_{\mathbf{x}} f$$

$$\nabla_{\mathbf{x}}(fg) = (x_1^2 + x_2^2) \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} + (x_1 x_2) \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix} = \begin{pmatrix} (x_1^2 + x_2^2)x_2 + 2x_1^2 x_2 \\ (x_1^2 + x_2^2)x_1 + 2x_1 x_2^2 \end{pmatrix} = \begin{pmatrix} 3x_1^2 x_2 + x_2^3 \\ x_1^3 + 3x_1 x_2^2 \end{pmatrix}$$

Gradients of Linear Functions

- For $\mathbf{x} \in R^n$, let $f(\mathbf{x}) = \mathbf{u}^T\mathbf{x}$ for some constant vector $\mathbf{u} \in R^n$
- Then

$$\nabla_{\mathbf{x}} f = \nabla_{\mathbf{x}}(\mathbf{u}^T\mathbf{x}) = \mathbf{u}$$

- <u>Proof</u>: for each $1 \leq i \leq n$ we have:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{\partial}{\partial x_i}\mathbf{u}^T\mathbf{x} = \frac{\partial}{\partial x_i}\sum_{k=1}^{n} u_k x_k = u_i$$

Gradients of Quadratic Functions

- For $A \in R^{n \times n}$ a square matrix, $\mathbf{x} \in R^n$, and $f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x}$
- The gradient of f is:

$$\nabla_{\mathbf{x}}(\mathbf{x}^T A \mathbf{x}) = (A + A^T)\mathbf{x}$$

  - You will prove this in the homework
- If A is symmetric, then

$$\nabla_{\mathbf{x}}(\mathbf{x}^T A \mathbf{x}) = 2A\mathbf{x}$$

The Hessian Matrix

- A square matrix that contains all second-order partial derivatives of a function
- Provides important information about the curvature of the function

$$H_f(\mathbf{x}) = \nabla_{\mathbf{x}}^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

- If the second-order partial derivatives are continuous then *H* is symmetric

The Hessian Matrix

- Example: given the function

$$f(x, y) = x^2 + 3xy$$

- Its second-order partial derivatives are:

$$f_{xx} = \frac{\partial^2 f}{\partial x^2} = 2, \quad f_{yy} = \frac{\partial^2 f}{\partial y^2} = 0, \quad f_{xy} = \frac{\partial^2 f}{\partial x \partial y} = 3, \quad f_{yx} = \frac{\partial^2 f}{\partial y \partial x} = 3$$

- Thus, the Hessian matrix is:

$$H_f(x, y) = \begin{pmatrix} 2 & 3 \\ 3 & 0 \end{pmatrix}$$

The Jacobian Matrix

- A **vector-valued** function **f**: $R^n \rightarrow R^m$ maps a vector in R^n to a vector in R^m

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ f_2(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{pmatrix}$$

- The Jacobian matrix of **f** contains all the first-order partial derivatives of **f**

$$J_{\mathbf{f}}(\mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

  - Each row in the matrix corresponds to the gradient of one component function f_i

The Jacobian Matrix

- Example: Given the function

$$\mathbf{f}(x,y) = \begin{pmatrix} x^2 + y \\ \sin x \\ e^y \end{pmatrix}$$

- Its Jacobian matrix is:

$$J_{\mathbf{f}}(x,y) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x & 1 \\ \cos x & 0 \\ 0 & e^y \end{pmatrix}$$

The Multivariable Chain Rule

- Let f be a function of m variables $z_1, z_2, \ldots, z_m$, each of which is a function of n variables $x_1, x_2, \ldots, x_n$
- Then the partial derivative of f with respect to x_i is:

$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial f}{\partial z_j} \frac{\partial z_j}{\partial x_i}$$

- Example:

$$f(g, h) = g^2 h + \sin h, \quad \text{where} \quad g(x) = x + 1, \quad h(x) = e^x$$

$$\begin{aligned}
\frac{d}{dx} f &= \frac{\partial f}{\partial g} \cdot \frac{dg}{dx} + \frac{\partial f}{\partial h} \cdot \frac{dh}{dx} \\
&= 2gh \cdot \frac{d}{dx}(x+1) + (g^2 + \cos h) \frac{d}{dx} e^x \\
&= 2gh \cdot 1 + (g^2 + \cos h) e^x \\
&= 2(x+1)e^x + ((x+1)^2 + \cos(e^x)) e^x \\
&= e^x (x^2 + 4x + 3 + \cos(e^x))
\end{aligned}$$

The Gradient Chain Rule

- Let f: $R^m \to R$ be a function that depends on a vector-valued function **g**: $R^n \to R^m$

$$f(\mathbf{x}) = f(g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x}))$$

- The gradient of f with respect to **x** is:

$$\nabla_{\mathbf{x}} f = J_{\mathbf{g}}^T(\mathbf{x}) \nabla_{\mathbf{g}} f$$

  - where $J_{\mathbf{g}}$ is the Jacobian matrix of **g**

$$J_{\mathbf{g}}(\mathbf{x}) = \frac{\partial \mathbf{g}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \cdots & \frac{\partial g_m}{\partial x_n} \end{pmatrix}$$

The Gradient Chain Rule

- Example: consider the function

$$f(\mathbf{x}) = \|\mathbf{g}(\mathbf{x})\|^2 \qquad \mathbf{g}(\mathbf{x}) = \begin{pmatrix} x_1^2 + x_2 \\ \sin x_1 + x_2^2 \end{pmatrix}$$

- Then using the chain rule: $\nabla_{\mathbf{x}} f = J_{\mathbf{g}}(\mathbf{x})^T \nabla_{\mathbf{g}} f$

$$J_{\mathbf{g}}(\mathbf{x}) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 & 1 \\ \cos x_1 & 2x_2 \end{pmatrix} \qquad \nabla_{\mathbf{g}} f = \nabla_{\mathbf{g}} \|\mathbf{g}\|^2 = 2\mathbf{g} = 2 \begin{pmatrix} x_1^2 + x_2 \\ \sin x_1 + x_2^2 \end{pmatrix}$$

$$\nabla_{\mathbf{x}} f = 2J_{\mathbf{g}}(\mathbf{x})^T \mathbf{g} = 2 \begin{pmatrix} 2x_1 & \cos x_1 \\ 1 & 2x_2 \end{pmatrix} \begin{pmatrix} x_1^2 + x_2 \\ \sin x_1 + x_2^2 \end{pmatrix} = \begin{pmatrix} 4x_1^3 + 4x_1 x_2 + 2\cos x_1 \sin x_1 + 2x_2^2 \cos x_1 \\ 2x_1^2 + 2x_2 + 4x_2 \sin x_1 + 4x_2^3 \end{pmatrix}$$

Exercise

- Compute the gradient $\nabla_{\mathbf{x}} \left(\|\mathbf{x}\|^2 \cdot \sin(\mathbf{a}^T \mathbf{x}) \right)$

- Using the product rule: $f(\mathbf{x}) = \|\mathbf{x}\|^2, \quad g(\mathbf{x}) = \sin(\mathbf{a}^T \mathbf{x})$

$$\nabla_{\mathbf{x}}(fg) = f(\mathbf{x}) \cdot \nabla_{\mathbf{x}} g + g(\mathbf{x}) \cdot \nabla_{\mathbf{x}} f$$

- The gradient of *f* is: $\nabla_{\mathbf{x}} f = 2\mathbf{x}$
- The gradient of *g* can be computed using the chain rule:

$$h(\mathbf{x}) = \mathbf{a}^T \mathbf{x}, \quad g(\mathbf{x}) = \sin(h(\mathbf{x}))$$

$$\nabla_{\mathbf{x}} g = \frac{\partial g}{\partial \mathbf{x}} = \frac{\partial g}{\partial h} \frac{\partial h}{\partial \mathbf{x}} = \cos(h(\mathbf{x})) \cdot \mathbf{a} = \cos(\mathbf{a}^T \mathbf{x}) \cdot \mathbf{a}$$

- Thus $\nabla_{\mathbf{x}} \left(\|\mathbf{x}\|^2 \cdot \sin(\mathbf{a}^T \mathbf{x}) \right) = \|\mathbf{x}\|^2 \cdot \cos(\mathbf{a}^T \mathbf{x}) \cdot \mathbf{a} + \sin(\mathbf{a}^T \mathbf{x}) \cdot 2\mathbf{x}$

Further Readings

- Zico Kolter, [Linear Algebra Review]
- Book: Steven J. Leon, Linear Algebra with Applications