Linear Algebra: Foundations, Systems, and Decompositions

1 Introduction to Linear Algebra

Linear algebra is the mathematical foundation for handling data in machine learning and other computational sciences. It studies vectors, matrices, and linear transformations, which are essential for representing datasets, features, and model parameters [2, 1].

1.1 Key Concepts

- **Vector**: An ordered array of numbers (e.g., features of a data point).
- **Matrix**: A 2D array of numbers, often used to represent datasets or transformations.
- **Scalar**: A single number, used to scale vectors or matrices.
- **Linear Transformation**: A function that maps vectors to other vectors, preserving vector addition and scalar multiplication.

1.2 Properties of Linear Transformations

A transformation T is linear if:

$$T(u+v) = T(u) + T(v) \quad \text{(Additivity)}$$
$$T(cv) = cT(v) \quad \text{(Homogeneity, for any scalar } c\text{)}$$

2 NumPy Basics for Linear Algebra

```
import numpy as np

# Vector
v = np.array([1, 2, 3])

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

```
# Scalar multiplication
v_scaled = 5 * v

# Matrix multiplication
B = np.array([[5, 6], [7, 8]])
C = A @ B  # or np.dot(A, B)
```

3 Systems of Linear Equations

A system of linear equations consists of two or more equations with the same variables. The general form is $AX = B$, where:

- A: Coefficient matrix
- X: Column vector of variables
- B: Column vector of constants [3, 4]

3.1 Types of Systems

- **Consistent system**: At least one solution exists.
- **Inconsistent system**: No solution exists.
- **Independent system**: Exactly one solution.
- **Dependent system**: Infinitely many solutions (equations are multiples of each other).

4 Matrix Properties and Operations

- **Shape**: $m \times n$ (rows $\times$ columns)
- **Row vector**: 1 row, n columns
- **Column vector**: m rows, 1 column
- **Transpose**: Flips rows and columns (A^T)
- **Inverse**: Exists only for square, non-singular matrices (A^{-1})
- **Determinant**: Indicates if a matrix is invertible

4.1 Matrix Operations in NumPy

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

# Addition
C = A + B

# Multiplication
D = A @ B

# Transpose
A_T = A.T

# Inverse (if possible)
A_inv = np.linalg.inv(A)
```

5 Solving Systems of Linear Equations

5.1 Methods

- **Direct solution**: For square, invertible matrices ($x = A^{-1}b$)
- **NumPy solver**: `np.linalg.solve(A, b)` (preferred for numerical stability)
- **Least squares**: For over/underdetermined or singular systems (`np.linalg.lstsq(A, b)`)
- **LU/QR Decomposition**: Efficient for large systems or repeated solutions [6, 7]

5.2 NumPy Implementation Examples

```
import numpy as np

# Example: 2x + 3y = 8, 5x + 4y = 13
A = np.array([[2, 3], [5, 4]])
b = np.array([8, 13])

# 1. Direct solution
try:
    x = np.linalg.solve(A, b)
    print("Solution:", x)
except np.linalg.LinAlgError:
    print("Matrix A is singular or not square.")

# 2. Least squares (over/underdetermined)
```

```
x_ls, residuals, rank, s = np.linalg.lstsq(A, b, rcond=None)
print("Least squares solution:", x_ls)
```

5.3 Step-by-Step Example

Given the system:

$$A = \begin{bmatrix} 8 & 3 & -2 \\ -4 & 7 & 5 \\ 3 & 4 & -12 \end{bmatrix}, \quad b = \begin{bmatrix} 9 \\ 15 \\ 35 \end{bmatrix}$$

```
x = np.linalg.solve(A, b)
print(x)  # Outputs: [-0.582..., 3.228..., -1.985...]
```

6 LU and QR Decomposition

6.1 LU Decomposition

LU decomposition factors a square matrix A into three matrices: P (permutation), L (lower triangular), and U (upper triangular), such that $PA = LU$.

```
import numpy as np
import scipy.linalg as la

A = np.array([[2, 4, 5],
              [1, 3, 2],
              [4, 2, 1]])

P, L, U = la.lu(A)
print("P:\n", P)
print("L:\n", L)
print("U:\n", U)
```

This decomposition is useful for solving systems of equations and matrix inversion efficiently [8, 9].

6.2 QR Decomposition

QR decomposition factors a matrix A into Q (orthogonal) and R (upper triangular) matrices, such that $A = QR$.

```
import numpy as np

A = np.array([[1, 2, 3],
              [3, 4, 5]])
Q, R = np.linalg.qr(A)
print("Q:\n", Q)
print("R:\n", R)
```

QR decomposition is widely used in least squares problems and eigenvalue algorithms [10, 11].

7 Types of Systems: Examples

7.1 Consistent System (At Least One Solution)

A system is consistent if there is at least one solution. This can be:

- Unique solution (independent)
- Infinitely many solutions (dependent)

7.2 Inconsistent System (No Solution)

A system is inconsistent if there is no solution (the equations contradict each other).

7.3 Independent System (Exactly One Solution)

A system is independent if it has exactly one solution; the equations represent lines (or planes) that intersect at a single point.

7.4 Dependent System (Infinitely Many Solutions)

A system is dependent if the equations are multiples of each other, representing the same line (or plane) and thus have infinitely many solutions.

8 Applications in Machine Learning

- **Data representation**: Features as vectors, datasets as matrices.
- **Dimensionality reduction**: PCA, SVD use matrix decompositions.
- **Model training**: Linear regression, neural networks use matrix operations.
- **Optimization**: Gradient descent and backpropagation rely on linear algebraic computations [1, 2].

9 References

References

[1] https://www.geeksforgeeks.org/machine-learning/ml-linear-algebra-operations/

[2] https://www.britannica.com/science/linear-algebra

[3] https://math.libretexts.org/Bookshelves/Algebra/Algebra_and_Trigonometry_1e_(OpenStax)/11:_Systems_of_Equations_and_Inequalities/11.01:Systems_of_Linear_Equations-_Two_Variables

[4] https://www.geeksforgeeks.org/system-linear-equations/

[5] https://en.wikipedia.org/wiki/Matrix_(mathematics)

[6] https://stackabuse.com/solving-systems-of-linear-equations-with-pythons-numpy/

[7] https://www.linkedin.com/pulse/solving-systems-linear-equations-numpy-mohamed-riyaz-

[8] https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.lu.html

[9] https://scicoding.com/how-to-calculate-lu-decomposition-in-python/

[10] https://www.geeksforgeeks.org/calculate-the-qr-decomposition-of-a-given-matrix-using

[11] https://scicoding.com/how-to-calculate-qr-decompsition-in-python/