College of Engineering & Applied Sciences

CSPB 2820

Linear Algebra With Computer Science Applications

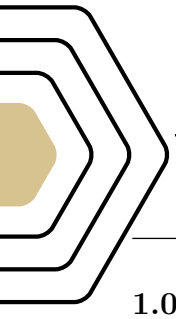
Class Notes

University Of Colorado

2024

Linear Algebra With Computer Science Applications - Class Notes

Vectors



Vectors

1.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- VMLS Chapter 1.1 - Vectors
- VMLS Chapter 1.2 - Vector Addition
- VMLS Chapter 1.3 - Scalar-Vector Multiplication
- VMLS Chapter 1.4 - Inner Product
- VMLS Chapter 1.5 - Complexity Of Vector Computations
- Python Companion Chapter 1.1 - Vectors
- Python Companion Chapter 1.2 - Vector Addition
- Python Companion Chapter 1.3 - Scalar-Vector Multiplication
- Python Companion Chapter 1.4 - Inner Product
- Python Companion Chapter 1.5 - Complexity Of Vector Computations

1.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

1.0.3 Lectures

The lectures for this week and their links can be found below:

- Why Linear Algebra? ≈ 10 min.
- Vectors ≈ 12 min.
- Adding & Scaling Vectors ≈ 22 min.
- Inner Product ≈ 13 min.
- Random Exercise: Chapter 1 - 1.9 ≈ 6 min.
- Implementing Vectors In Python ≈ 22 min.
- Vectors In Numpy ≈ 22 min.

1.0.4 Assignments

The assignment for this week is:

- Assignment 1 - Vectors

1.0.5 Quiz

The quiz for this week is:

- Quiz 1 - Vectors

1.0.6 Chapter Summary

The chapter that we will review this week is from **VMLS Chapter 1 - Vectors.** The first section that is going to be reviewed is **VMLS Section 1.1 - Vectors.**

VMLS Section 1.1 - Vectors

Overview

In the realm of computer science, vectors serve as fundamental mathematical constructs that hold immense significance in representing quantities possessing both magnitude and direction. Their versatile nature finds applications in various domains including graphics, physics simulations, machine learning, and data analysis.

A vector is essentially defined by its components across distinct dimensions. For instance, in a two-dimensional space, a vector could be denoted as (x, y), where 'x' and 'y' symbolize the components along the horizontal and vertical axes respectively. This concept extends to three-dimensional space where vectors have three components (x, y, z), and further to higher dimensions as required by the problem context.

The visual representation of vectors often takes the form of arrows originating from an origin point and extending to a specific location in space. The length of the arrow characterizes the magnitude of the vector, while its direction conveys the vector's alignment within the coordinate system.

At the heart of computer science, vector operations play a pivotal role. Key concepts include:

Vector Addition and Subtraction: Addition involves summing up corresponding components of vectors, while subtraction follows a similar principle. These operations are fundamental in simulations, motion calculations, and beyond.

Scalar Multiplication: Vectors can be scaled by a scalar value, modifying their magnitude while preserving direction. This operation is pivotal for scenarios such as magnifying a vector's impact by multiplying it with a factor.

Dot Product (Scalar Product): The dot product gauges the alignment of two vectors. Computed by summing the products of their corresponding components, it finds use in angle calculations and identifying vector orthogonality.

Cross Product (Vector Product): Resulting in a new vector orthogonal to its inputs, the cross product holds significance in graphics and physics, contributing to calculations involving surface normals and angular momentum.

Vector Spaces: The grouping of vectors into vector spaces, characterized by operations maintaining specific properties, provides the mathematical groundwork for numerous computational algorithms.

Machine Learning: Vectors find substantial application in machine learning, where data points are often represented in high-dimensional spaces. This facilitates efficient processing and analysis of complex data structures.

Graphics and Computer Animation: Vectors form the cornerstone of graphical representation, enabling efficient manipulation, transformation, and rendering of objects in 2D and 3D settings.

Data Analysis: Vectors are indispensable in data analysis, used to express features of data points. Techniques such as clustering, dimensionality reduction, and similarity calculations heavily depend on vector-based representations.

In summary, vectors serve as versatile mathematical tools that occupy a central role in the field of computer science. Their ability to capture both magnitude and direction renders them invaluable for resolving an extensive array of challenges across diverse domains.

Python Vectors

Below is an example of we can utilize vectors in the context of Python.

```
import numpy as np

# Creating vectors
vector_a = np.array([1, 2, 3])
vector_b = np.array([4, 5, 6])

print("Vector A:", vector_a)
print("Vector B:", vector_b)

```

The second section that we are going to review is **VMLS Section 1.2 - Vector Addition.**

VMLS Section 1.2 - Vector Addition

Addition

Vector addition is a fundamental mathematical operation that involves combining two or more vectors to create a new vector ($\mathbf{C}$). In this process, the individual components of the vectors are added together to form the components of the resulting vector. This operation finds widespread application in various fields, including physics, computer graphics, and engineering.

When adding vectors, their geometric properties are combined. If two vectors represent displacements or forces, for example, their addition results in a vector that encapsulates the net effect of both original vectors. The addition is performed by adding corresponding components along each dimension. In a two-dimensional space, if you have vectors $\mathbf{A} = (a_1, a_2)$ and $\mathbf{B} = (b_1, b_2)$, their sum $\mathbf{C} = \mathbf{A} + \mathbf{B}$ would be $(a_1 + b_1, a_2 + b_2)$.

Vector addition obeys the commutative property, meaning the order in which vectors are added does not affect the result: $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$. It also adheres to the associative property, allowing multiple vectors to be added in any order: $(\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C})$.

In computer graphics, vector addition is used to manipulate the positions and orientations of objects. In physics, it's employed to analyze forces and velocities acting on objects. In navigation and robotics, vector addition helps calculate resultant velocities or paths. Essentially, vector addition is a fundamental tool that enables us to combine different influences or movements in a coherent and mathematically rigorous manner.

Python Addition

Below is an example of vector addition in Python.

```python
import numpy as np

# Creating vectors
vector_a = np.array([1, 2, 3])
vector_b = np.array([4, 5, 6])

result_add = vector_a + vector_b
print("Vector A + Vector B:", result_add)

```

Subtraction

Vector subtraction is a fundamental mathematical operation that involves finding the difference between two vectors to create a new vector ($\mathbf{D}$). In this process, the individual components of one vector are subtracted from the corresponding components of the other vector, forming the components of the resulting vector. Like vector addition, vector subtraction is widely applicable in fields such as physics, computer graphics, and engineering.

When subtracting vectors, the geometric properties of their differences are considered. For instance, if one vector represents a displacement and the other vector represents a force, their subtraction yields a vector encapsulating the net effect of these vectors. The operation is carried out by subtracting corresponding components along each dimension. In a two-dimensional space, if you have vectors $\mathbf{A} = (a_1, a_2)$ and $\mathbf{B} = (b_1, b_2)$, their difference $\mathbf{D} = \mathbf{A} - \mathbf{B}$ would be $(a_1 - b_1, a_2 - b_2)$.

Similar to vector addition, vector subtraction follows the commutative property: $\mathbf{A} - \mathbf{B} = -(\mathbf{B} - \mathbf{A})$. This means that subtracting vector $\mathbf{B}$ from vector $\mathbf{A}$ is equivalent to adding the negation of vector $\mathbf{B}$ to vector $\mathbf{A}$.

Vector subtraction finds applications in scenarios where you need to find the difference between quantities, such as when determining the change in position or displacement between two points in space. Additionally, vector subtraction is fundamental in physics for analyzing relative velocities and net forces acting on objects.

Python Subtraction

Below is an example of vector subtraction in Python.

```python
import numpy as np

# Creating vectors
vector_a = np.array([1, 2, 3])
vector_b = np.array([4, 5, 6])

result_sub = vector_a - vector_b
print("Vector A - Vector B:", result_sub)

```

In summary, vector subtraction is a foundational operation in mathematics and has widespread application across various scientific and computational disciplines.

The next section that we are going to review is **VMLS Section 1.3 - Scalar-Vector Multiplication.**

VMLS Section 1.3 - Scalar-Vector Multiplication

Scalar-vector multiplication is a fundamental mathematical operation that involves scaling a vector by a scalar, resulting in a new vector with adjusted magnitude and direction. In this operation, each component of the vector is multiplied by the scalar value. Scalar-vector multiplication is a concept widely applied in various fields including physics, engineering, computer graphics, and linear algebra.

When a scalar is multiplied by a vector, the scalar effectively controls the stretching or shrinking of the vector without changing its direction. If the scalar is positive, the vector extends in the same direction while increasing in magnitude. If the scalar is negative, the vector reverses direction while its magnitude increases. If the scalar is zero, the resulting vector is the zero vector, regardless of the original vector's direction.

Scalar Vector Multiplication Defintion

Mathematically, if $\mathbf{v}$ is a vector and k is a scalar, then the scalar-vector multiplication is expressed as:

$$k \cdot \mathbf{v} = \begin{bmatrix} kv_1 \\ kv_2 \\ \vdots \\ kv_n \end{bmatrix}$$

Where $v_1, v_2, \ldots, v_n$ are the components of the vector $\mathbf{v}$ and n is the dimension of the vector.

Scalar-vector multiplication is essential in various applications. For instance, in physics, it can represent scaling physical quantities such as forces or velocities. In computer graphics, it's used to adjust the size and position of objects. In linear algebra, scalar-vector multiplication contributes to concepts like linear combinations and vector spaces.

Python Scalar Multiplication

Below is an example of vector inner produce in Python.

```
import numpy as np

# Creating a vector
vector_a = np.array([2, 4, 6])

# Scalar multiplication
scalar = 3
result_scalar_mul = scalar * vector_a

print("Vector A:", vector_a)
print("Scalar-Vector Multiplication:", result_scalar_mul)

```

In summary, scalar-vector multiplication is a foundational operation that enables the rescaling of vectors by multiplying them with scalar values. It's a versatile concept that plays a vital role in many mathematical and practical contexts across different fields.

The next section that we are going to review is **VMLS Section 1.4 - Inner Product.**

VMLS Section 1.4 - Inner Product

The inner product, also known as the dot product or scalar product, is a fundamental mathematical operation in vector spaces that measures the angle between two vectors and quantifies their alignment. It results in a scalar value and provides insight into how closely two vectors are oriented with respect to each other. The inner product is an essential concept in various fields, including linear algebra, physics, engineering, and computer graphics.

Inner Product Defintion

Mathematically, the inner product of two vectors $\mathbf{A}$ and $\mathbf{B}$ is denoted by $\langle \mathbf{A}, \mathbf{B} \rangle$ or $\mathbf{A} \cdot \mathbf{B}$, and it's calculated by summing the products of their corresponding components. For vectors in $\mathbb{R}^n$, the inner product is defined as:

$$\langle \mathbf{A}, \mathbf{B} \rangle = \mathbf{A} \cdot \mathbf{B} = a_1 \cdot b_1 + a_2 \cdot b_2 + \ldots + a_n \cdot b_n$$

Here, $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ are the components of vectors $\mathbf{A}$ and $\mathbf{B}$ in $\mathbb{R}^n$.

The inner product has several important properties, including symmetry ($\langle \mathbf{A}, \mathbf{B} \rangle = \langle \mathbf{B}, \mathbf{A} \rangle$), linearity ($\langle \mathbf{A} + \mathbf{B}, \mathbf{C} \rangle = \langle \mathbf{A}, \mathbf{C} \rangle + \langle \mathbf{B}, \mathbf{C} \rangle$), and positive definiteness ($\langle \mathbf{A}, \mathbf{A} \rangle \geq 0$, with equality only if $\mathbf{A}$ is the zero vector).

The inner product serves as the foundation for concepts like vector length (magnitude), orthogonality, projection, and angles between vectors. It plays a central role in defining norm and distance in vector spaces, leading to the concept of a inner product space.

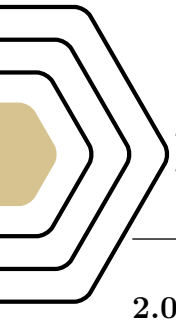
Python Inner Product

Below is an example of vector inner produce in Python.

```python
import numpy as np

# Creating vectors
vector_a = np.array([1, 2, 3])
vector_b = np.array([4, 5, 6])

inner_product = np.dot(vector_a, vector_b)
print("Inner Product of Vector A and Vector B:", inner_product)

```

In summary, the inner product is a mathematical operation that quantifies the alignment and relationship between vectors. It provides valuable geometric and algebraic insights into vector spaces and finds applications in various mathematical and practical contexts across multiple disciplines.

Linear Functions



Linear Functions

2.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- VMLS Chapter 2.1 - Linear Functions
- VMLS Chapter 2.2 - Taylor Approximations
- VMLS Chapter 2.3 - Regression Model
- Python Companion Chapter 2.1 - Linear Functions
- Python Companion Chapter 2.2 - Taylor Approximation
- Python Companion Chapter 2.3 - Regression Model

2.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

2.0.3 Lectures

The lectures for this week and their links can be found below:

- Linear Functions ≈ 28 min.
- Affine Functions ≈ 9 min.
- Taylor Approximations ≈ 16 min.
- Regression Models ≈ 9 min.
- Random Exercise ≈ 13 min.
- Taylor Series Overview ≈ 12 min.
- Calculus Refresher ≈ 21 min.

2.0.4 Assignments

The assignment for this week is:

- Assignment 2 - Linear Functions

2.0.5 Quiz

The quiz for this week is:

- Quiz 2 - Linear Functions

2.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 2 - Linear Functions.** The first section that we will examine this week is **VMLS Section 2.1 - Linear Functions.**

VMLS Section 2.1 - Linear Functions

Overview

In the context of linear algebra, a linear function, also known as a linear transformation or linear map, is a fundamental concept that describes a specific type of mathematical relationship between vector spaces. Linear functions play a central role in linear algebra, as they provide a way to describe how vectors and vector spaces are transformed or mapped to other vectors and vector spaces while preserving certain properties.

1. **Definition of a Linear Function:** A linear function T is a mapping or transformation from one vector space V to another vector space W that satisfies two key properties:
   (a) **Additivity:** For any vectors $\mathbf{u}$ and $\mathbf{v}$ in V, $T(\mathbf{u}+\mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$. This means that the function preserves vector addition; the sum of the images of vectors is equal to the image of their sum.
   (b) **Homogeneity:** For any vector $\mathbf{u}$ in V and any scalar c, $T(c\mathbf{u}) = cT(\mathbf{u})$. This property indicates that the function preserves scalar multiplication; scaling a vector and then applying the function is equivalent to applying the function first and then scaling the result.
2. **Examples of Linear Functions:** Linear functions are prevalent in various fields, including physics, engineering, and computer science. Here are a few common examples:
   - **Matrix Transformations:** In linear algebra, many transformations can be represented as matrix-vector products. These transformations include rotations, scaling, and shearing.
   - **Linear Operators:** In quantum mechanics and functional analysis, linear operators are used to model physical systems and mathematical functions. These operators are often represented by matrices or differential equations.
   - **Linear Regression:** In statistics and machine learning, linear regression models describe relationships between variables using linear functions. The coefficients in the model represent the linear transformation.
3. **Preservation of Linearity:** Linear functions are defined by their preservation of linearity, meaning that the properties of additivity and homogeneity hold true. This ensures that linear combinations of vectors in the domain map to linear combinations of vectors in the codomain. Consequently, linear functions are well-behaved and predictable.
4. **Representation and Matrix Form:** Many linear functions can be represented by matrices. The transformation of a vector $\mathbf{v}$ by a linear function T can be expressed as $T(\mathbf{v}) = A\mathbf{v}$, where A is the transformation matrix. This representation simplifies computations and allows for the efficient application of linear transformations to large datasets.
5. **Fundamental Concept in Linear Algebra:** Linear functions are a foundational concept in linear algebra. They are studied extensively because they provide insights into the structure and properties of vector spaces, and they are essential for solving systems of linear equations, understanding eigenvalues and eigenvectors, and more.

In summary, linear functions are mathematical mappings that maintain certain algebraic properties, specifically additivity and homogeneity. They are critical in linear algebra for understanding transformations, solving equations, and modeling various phenomena in mathematics, science, and engineering.

Superposition and Linearity

In the realm of linear algebra, the fundamental principles of superposition and linearity are crucial for comprehending the behavior of mathematical functions. Consider the inner product function denoted as f. This function is distinguished by a specific characteristic known as superposition.

Superposition, as articulated in the context of this inner product function, signifies that the function f adheres to the following property for all n-vectors x and y and all scalars a and b:

$$f(ax + by) = af(x) + bf(y)$$

Put differently, when the inner product function is applied to a linear combination of two vectors $(ax + by)$, it is tantamount to applying the function individually to each vector ($f(x)$ and $f(y)$) and subsequently merging the outcomes linearly with the scalars a and b.

This superposition property is intimately related to the concept of linearity. A function that fulfills the superposition property is denominated as "linear." Thus, in this specific context, the inner product with a fixed vector is acknowledged as a linear function because it complies with the superposition property.

In essence, linearity and superposition constitute foundational concepts in linear algebra, empowering us to elucidate and scrutinize mathematical functions that display predictable behavior when dealing with combinations of vectors and scalars. Profound comprehension of linearity is indispensable in diverse mathematical and scientific domains, as it simplifies the analysis of intricate systems and transformations.

Homogeneity & Additivity

The superposition equality is often deconstructed into two distinct properties, one involving scalar-vector product and the other involving vector addition in the argument. A function $f : \mathbb{R}^n \to \mathbb{R}$ is considered linear if it satisfies the following two fundamental properties:

1. **Homogeneity Property:** For any n-vector $\mathbf{x}$ and any scalar α, the function f obeys the homogeneity property, expressed as:
$$f(\alpha\mathbf{x}) = \alpha f(\mathbf{x})$$
This property signifies that scaling the vector argument is equivalent to scaling the function's value. In essence, it emphasizes the consistent behavior of the function under scalar-vector multiplication.

2. **Additivity Property:** For any n-vectors $\mathbf{x}$ and $\mathbf{y}$, the function f adheres to the additivity property, articulated as:
$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$
This property underscores that adding vector arguments yields the same result as adding the function values for each vector individually. It underscores the consistent behavior of the function under vector addition.

Together, the homogeneity and additivity properties collectively define the concept of linearity. A function is deemed linear when it satisfies both of these properties simultaneously. Linearity is a fundamental concept with broad applications in various mathematical and scientific disciplines, particularly in linear algebra. It simplifies the analysis of linear transformations and systems, making it indispensable in fields such as physics, engineering, economics, and computer science.

Linearity Example

Consider a linear transformation T that maps vectors from $\mathbb{R}^2$ to $\mathbb{R}$ defined as follows:

$$T\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = 2x - 3y$$

We will demonstrate how this transformation exhibits linearity by verifying the homogeneity and additivity properties.

---

1. **Homogeneity Property:**

   We'll show that T satisfies the homogeneity property. For any scalar α and vector $\mathbf{v} = \begin{bmatrix} x \\ y \end{bmatrix}$, we have:

   $$T(\alpha\mathbf{v}) = T\left(\alpha\begin{bmatrix} x \\ y \end{bmatrix}\right) = T\left(\begin{bmatrix} \alpha x \\ \alpha y \end{bmatrix}\right) = 2(\alpha x) - 3(\alpha y) = \alpha(2x - 3y) = \alpha T(\mathbf{v})$$

   This confirms the homogeneity property, demonstrating that scaling the input vector by α results in scaling the output by the same factor.

2. **Additivity Property:**

   Next, we'll show that T satisfies the additivity property. For any vectors $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$, we have:

   $$T(\mathbf{u} + \mathbf{v}) = T\left(\begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \end{bmatrix}\right) = 2(u_1 + v_1) - 3(u_2 + v_2) = (2u_1 - 3u_2) + (2v_1 - 3v_2) = T(\mathbf{u}) + T(\mathbf{v})$$

   This confirms the additivity property, demonstrating that the transformation of the sum of vectors is equal to the sum of the transformations of each vector individually.

Therefore, the transformation T is linear, as it satisfies both the homogeneity and additivity properties, making it a valid example of linearity.

Inner Product Representation of a Linear Function

The inner product representation of a linear function is a fundamental concept in linear algebra, establishing a relationship between linear functions and inner products. It reveals that linear functions can be expressed as the inner product of their argument with a fixed vector, and conversely, if a function is linear, it can be represented in this form.

Linearity and Inner Product

The initial observation establishes that a function $f(x)$ defined as the inner product of its argument with a fixed vector is linear. Mathematically, if $f(x)$ can be expressed as $f(x) = a^T x$, where a is a fixed vector, then $f(x)$ is a linear function.

The Converse

The converse statement is equally significant, stating that if a function is linear, it can be represented as the inner product of its argument with a fixed vector. This implies the existence of an n-vector a such that $f(x) = a^T x$ holds for all n-vectors x.

Deriving the Inner Product Representation

To understand the derivation of this representation, we begin by expressing an arbitrary n-vector x as a linear combination of basis vectors: $x = x_1 e_1 + x_2 e_2 + \ldots + x_n e_n$. Here, $e_1, e_2, \ldots, e_n$ represent the basis vectors, and $x_1, x_2, \ldots, x_n$ are the coefficients.

Given that f is linear, we apply multi-term superposition, demonstrating that $f(x)$ is the linear combination of $f(e_1), f(e_2), \ldots, f(e_n)$, weighted by the coefficients $x_1, x_2, \ldots, x_n$.

The resulting formula $f(x) = x_1 f(e_1) + x_2 f(e_2) + \ldots + x_n f(e_n)$ represents the inner product representation of f, with a being the vector $(f(e_1), f(e_2), \ldots, f(e_n))$.

In essence, this inner product representation illustrates how a linear function $f(x)$ can be expressed as a weighted sum of the inner products between the argument vector x and a set of basis vectors, where the weights are determined by the function's behavior on each basis vector.

This concept holds significant value in linear algebra and various mathematical applications, providing a deeper understanding of the interplay between linear functions and inner products.

Inner Product Representation Example

Let's illustrate the concept of the inner product representation of a linear function with an example.

Consider a scalar-valued function f defined on $\mathbb{R}^2$ as follows:

$$f(\mathbf{x}) = 3x_1 - 2x_2$$

We will demonstrate that f is a linear function and derive its inner product representation.

Verification of Linearity:

To verify the linearity of f, we need to check whether it satisfies the properties of linearity: homogeneity and additivity.

1. **Homogeneity Property:**

   Let's consider the homogeneity property. For any scalar α and vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, we have:

   $$f(\alpha \mathbf{x}) = 3(\alpha x_1) - 2(\alpha x_2) = \alpha(3x_1 - 2x_2) = \alpha f(\mathbf{x})$$

   This confirms that f satisfies the homogeneity property.

2. **Additivity Property:**

   Now, let's examine the additivity property. For any vectors $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$, we have:

   $$f(\mathbf{u} + \mathbf{v}) = 3(u_1 + v_1) - 2(u_2 + v_2) = (3u_1 - 2u_2) + (3v_1 - 2v_2) = f(\mathbf{u}) + f(\mathbf{v})$$

   This confirms that f satisfies the additivity property.

Inner Product Representation:

Now that we've established that f is a linear function, we can derive its inner product representation. We do this by expressing $\mathbf{x}$ in terms of basis vectors:

$$\mathbf{x} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2$$

Where $\mathbf{e}_1 = \begin{bmatrix}1\\0\end{bmatrix}$ and $\mathbf{e}_2 = \begin{bmatrix}0\\1\end{bmatrix}$ are the standard basis vectors for $\mathbb{R}^2$.

Now, we can calculate f in terms of these basis vectors:

$$f(\mathbf{x}) = 3x_1\mathbf{e}_1 - 2x_2\mathbf{e}_2$$

The coefficients of the basis vectors in this expression, $\mathbf{a} = \begin{bmatrix}3\\-2\end{bmatrix}$, represent the inner product representation of f.

Therefore, we have successfully derived the inner product representation of the linear function f as:

$$f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} = \begin{bmatrix}3 & -2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

This representation allows us to express f as the inner product of its argument $\mathbf{x}$ with the fixed vector $\mathbf{a}$, which simplifies computations and provides insights into the function's behavior.

In this example, we've demonstrated how a linear function can be represented using the inner product of its argument with a fixed vector.

Affine Functions

Affine functions play an essential role in numerous areas of mathematics and its applications, such as optimization, computer graphics, and machine learning. They provide an extension of linear functions while maintaining several beneficial properties. Here's an elaboration on the topic:

Definition

An affine function can be considered a "shifted" linear function. In a more formal sense, a function $f : \mathbb{R}^n \to \mathbb{R}$ is affine if it can be written in the form:

$$f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$$

where $\mathbf{a}$ is an n-dimensional vector, and b is a scalar. The term b is sometimes referred to as the "offset" because it represents a shift from the origin.

Superposition

One of the significant properties of linear functions is the superposition principle, which allows us to linearly combine multiple instances of the function. However, the superposition property for affine functions comes with a caveat. For the function to remain affine, the coefficients used in the linear combination must sum to one. This is referred to as the "restricted superposition property."

Intuitive Understanding

Imagine a linear function as a straight line going through the origin of a graph. When we talk about affine functions, they are also straight lines but don't necessarily pass through the origin. The offset b in the affine function determines how far and in which direction the line is shifted from the origin.

Practical Implications

The restricted superposition property of affine functions gives us a powerful tool to identify non-affine functions. If we can find vectors $\mathbf{x}, \mathbf{y}$ and scalars α, β such that $\alpha + \beta = 1$ and $f(\alpha\mathbf{x} + \beta\mathbf{y}) \neq \alpha f(\mathbf{x}) + \beta f(\mathbf{y})$, then the function f is not affine. This property acts as a litmus test.

For instance, the maximum function, which returns the largest value among its arguments, doesn't hold the superposition property when $n > 1$. This non-conformance can be explicitly illustrated using specific coefficients and vectors.

Affine Function Example

Let's delve deeper into the concept of affine functions with an example.

Consider the following function $f : \mathbb{R}^2 \to \mathbb{R}$:

$$f(\mathbf{x}) = 2x_1 - 3x_2 + 1$$

In this example, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ represents a two-dimensional vector. The function f is defined as an affine function because it follows the form:

$$f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$$

where $\mathbf{a} = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$ is an n-dimensional vector, and $b = 1$ is a scalar. The vector $\mathbf{a}$ represents the coefficients of the linear part of the function, and b is the offset.

This function is an example of an affine function because it combines a linear transformation of the input vector $\mathbf{x}$ with an offset b. It can be visualized as a plane in three-dimensional space, with the coefficients of $\mathbf{a}$ determining the plane's orientation and the offset b determining its position along the vertical axis.

Verification of Affine Property:

To verify that f is an affine function, we need to check if it satisfies the properties of an affine function: linearity and the offset.

1. **Linearity Property:**
   Let's consider the linearity property. For any scalar α and vectors $\mathbf{x}$ and $\mathbf{y}$, we have:

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) = 2(\alpha x_1 + (1-\alpha)y_1) - 3(\alpha x_2 + (1-\alpha)y_2) + 1$$

$$= \alpha(2x_1 - 3x_2 + 1) + (1-\alpha)(2y_1 - 3y_2 + 1) = \alpha f(\mathbf{x}) + (1-\alpha)f(\mathbf{y})$$

   This confirms that f satisfies the linearity property.

2. **Offset Property:**
   The offset property is satisfied since $b = 1$ is present in the function definition as an additive constant.

The next section that we will examine this week is **VMLS Section 2.2 - Taylor Approximations.**

VMLS Section 2.2 - Taylor Approximations

Overview

A Taylor approximation, often referred to as a Taylor series or Taylor expansion, is a mathematical technique used to approximate complex functions with simpler polynomial functions. It's named after the British mathematician Brook Taylor. The core idea behind a Taylor approximation is to represent a function as an infinite series of polynomial terms centered around a specific point. This approximation is particularly useful when dealing with functions that are difficult to work with directly.

The general form of a Taylor series for a function $f(x)$ centered at a point a is:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots$$

The general form of this equation is

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n.$$

Here's a breakdown of the key components:

1. $f(a)$: The value of the function at the center point a. This represents the constant term in the approximation.
2. $f'(a)$: The first derivative of the function at the point a. This term accounts for the linear behavior of the function around a.
3. $\frac{f''(a)}{2!}(x-a)^2$: The second derivative of the function at a divided by 2! (which is 2). This term captures the quadratic behavior of the function.
4. $\frac{f'''(a)}{3!}(x-a)^3$: The third derivative of the function at a divided by 3! (which is 6). This term accounts for cubic behavior, and the pattern continues for higher-order derivatives.

The more terms you include in the Taylor series, the closer the approximation gets to the actual function. In practice, the series is often truncated to a finite number of terms to simplify calculations. The accuracy of the approximation depends on how many terms are included and how close the approximation point a is to the point of interest.

Taylor approximations are widely used in various fields of mathematics, physics, engineering, and computer science to simplify complex functions, solve differential equations, and make numerical calculations more manageable. They serve as a powerful tool for understanding the behavior of functions near specific points.

Linear Algebra Taylor Approximation

The first-order Taylor approximation, denoted as $\hat{f}(x)$, proves to be a highly accurate estimate of the function $f(x)$ under the condition that the individual components x_i are in close proximity to their corresponding reference points z_i. This approximation is sometimes expressed with a secondary vector argument, $\hat{f}(x;z)$, indicating the central point of approximation, denoted as z. The primary component of the Taylor approximation remains constant, equaling the value of the function at the point of reference, i.e., $f(z)$. The remaining terms contribute to the approximate alteration in the function's value, stemming from perturbations in the components of x concerning z. Interestingly, $\hat{f}$ exhibits an affine relationship with x, despite being referred to as a linear approximation near z—a distinction driven by its general affine nature. A concise representation employs inner product notation:

$$\hat{f}(x) = f(z) + \nabla f(z)^T(x-z)$$

Here, $\nabla f(z)$ represents the gradient of f at point z, expressed as an n-vector:

$$\nabla f(z) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(z) \\ \vdots \\ \frac{\partial f}{\partial x_n}(z) \end{bmatrix}$$

This approximation permits an organized construction of an affine estimate for a function $f : \mathbb{R}^n \to \mathbb{R}$, particularly in proximity to a designated point z, given a descriptive formula or equation for f that possesses differentiability. This methodology is exemplified by a straightforward instance for $n = 1$. Over a comprehensive scale along the x-axis, the Taylor approximation $\hat{f}$ may exhibit discrepancies from the actual function f. Nevertheless, within the vicinity of z, the Taylor approximation consistently proves to be an excellent approximation.

First-Order Taylor Approximation Example

Let's illustrate the concept of a first-order Taylor approximation in the context of linear algebra.

Consider a real-valued function f that operates on vectors in $\mathbb{R}^3$ as follows:

$$f(\mathbf{x}) = 2x_1 - x_2 + 3x_3$$

We want to find the first-order Taylor approximation of f near a specific point $\mathbf{z} = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$.

Verification of Linearity:

To verify if f is linear, we need to check whether it satisfies the properties of linearity: homogeneity and additivity.

1. **Homogeneity Property:**

Let's consider the homogeneity property. For any scalar α and vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, we have:

$$f(\alpha \mathbf{x}) = 2(\alpha x_1) - (\alpha x_2) + 3(\alpha x_3) = \alpha(2x_1 - x_2 + 3x_3) = \alpha f(\mathbf{x})$$

This confirms that f satisfies the homogeneity property.

2. **Additivity Property:**

Now, let's examine the additivity property. For any vectors $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$, we have:

$$f(\mathbf{u} + \mathbf{v}) = 2(u_1 + v_1) - (u_2 + v_2) + 3(u_3 + v_3) = (2u_1 - u_2 + 3u_3) + (2v_1 - v_2 + 3v_3) = f(\mathbf{u}) + f(\mathbf{v})$$

This confirms that f satisfies the additivity property.

First-Order Taylor Approximation:

Now that we've established that f is a linear function, we can derive its first-order Taylor approximation near the point $\mathbf{z}$.

The first-order Taylor approximation can be expressed as:

$$\hat{f}(\mathbf{x}) = f(\mathbf{z}) + \nabla f(\mathbf{z})^T(\mathbf{x} - \mathbf{z})$$

Where $\nabla f(\mathbf{z})$ represents the gradient of f at point $\mathbf{z}$. In this case:

$$\nabla f(\mathbf{z}) = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}$$

Now, we can express the first-order Taylor approximation $\hat{f}(\mathbf{x})$ as:

$$\hat{f}(\mathbf{x}) = 2 - (\mathbf{x} - \mathbf{z})^T \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}$$

This representation allows us to approximate the function f near the point $\mathbf{z}$ using a linear function.

In this example, we've demonstrated how to find the first-order Taylor approximation of a linear function f near a specific point $\mathbf{z}$ in $\mathbb{R}^3$.

The last section that we will examine this week is **VMLS Section 2.3 - Regression Model.**

VMLS Section 2.3 - Regression Model

Overview

A regression model is a fundamental statistical tool used to analyze relationships between variables. It is primarily employed in predictive modeling, where the goal is to understand and predict the behavior of a dependent variable based on one or more independent variables. Regression analysis helps uncover patterns, associations, and dependencies within data, making it a cornerstone in various fields, including economics, finance, biology, and social sciences.

The core idea of a regression model is to establish a mathematical equation that describes the relationship between variables. Typically, it is used to predict a continuous outcome variable (also known as the dependent variable) based on one or more predictor variables (independent variables). This predictive equation can then be utilized to estimate the value of the dependent variable given specific values of the independent variables.

Key Components and Concepts

1. **Dependent Variable (Y)**: This is the variable we aim to predict or explain. It represents the outcome or response of interest and is often denoted as "Y" in the regression equation.
2. **Independent Variables (X)**: Also known as predictor variables or features, these are variables that are believed to have an influence on the dependent variable. The regression model attempts to quantify this relationship. Multiple independent variables can be used in multivariate regression.
3. **Regression Equation**: The core of the model, this equation mathematically expresses how the independent variables are related to the dependent variable. The equation has coefficients (slopes) associated with each independent variable, representing the strength and direction of their impact.
4. **Residuals**: Residuals are the differences between the observed values of the dependent variable and the values predicted by the regression equation. Analyzing residuals helps assess the goodness-of-fit of the model.
5. **Types of Regression Models**: There are various types of regression models, each suited to different data scenarios. Common types include linear regression, logistic regression, polynomial regression, ridge regression, and lasso regression, among others.
6. **Assumptions**: Regression models rely on several assumptions, such as linearity (the relationship between variables is linear), independence of errors (residuals are not correlated), and homoscedasticity (constant variance of residuals).
7. **Model Evaluation**: Techniques like R-squared (coefficient of determination), Mean Squared Error (MSE), and hypothesis testing are employed to evaluate the performance and significance of regression models.
8. **Overfitting and Underfitting**: Striking a balance between model complexity and predictive accuracy is crucial. Overfitting (when the model is too complex) and underfitting (when the model is too simplistic) are common challenges in regression modeling.
9. **Applications**: Regression models find applications in diverse fields, including predicting stock prices, analyzing the impact of marketing campaigns on sales, understanding the relationship between age and health, and much more.

In summary, regression models are a versatile and essential tool for analyzing and modeling relationships between variables. They provide a framework for making predictions, drawing insights, and making informed decisions based on data. However, the choice of the appropriate regression model and careful validation are critical to ensure meaningful and accurate results.

Simplified Regression Model Notation

The concept presented here relates to simplifying the notation used in regression models by leveraging vector stacking. In typical linear regression models, you have a set of weights (coefficients) denoted as β and an intercept term α. These parameters define the linear relationship between the features (x) and the predicted variable (y).

To streamline this notation and make it more convenient, vector stacking is employed. Instead of representing the weights and intercept separately, they are combined into a single parameter vector $\tilde{\beta}$. Additionally, a new feature vector $\tilde{x}$ is created. This vector includes all the original features (x) and one additional feature ($\tilde{x}_1$) that always has the value one. This extra feature essentially represents the constant term (intercept) in the regression model.

Mathematically, we define the parameter vector $\tilde{\beta}$ as $\tilde{\beta} = (\alpha; \beta)$, where α is the intercept, and β represents the weights for the original features. Consequently, the regression model, which initially looked like:

$$\hat{y} = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

can be expressed more succinctly in inner product form as:

$$\hat{y} = \tilde{x}^T \tilde{\beta}$$

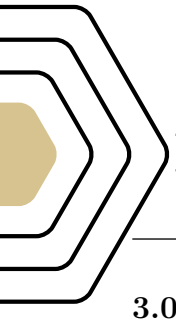
To simplify the notation further, we can often omit the tildes and write it simply as:

$$\hat{y} = x^T \beta$$

with the understanding that the first feature in x is always equal to one, representing the constant term. While this extra feature doesn't provide any informative value, it aids in simplifying the notation when working with regression models.

In essence, this technique allows for a more compact and manageable representation of regression models, making them easier to work with, especially in mathematical formulations and computations.

Norm And Distance



Norm And Distance

3.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- VMLS Chapter 3.1 - Norm
- VMLS Chapter 3.2 - Distance
- VMLS Chapter 3.3 - Standard Deviation
- VMLS Chapter 3.4 - Angle
- Python Companion Chapter 3.1 - Norm
- Python Companion Chapter 3.2 - Distance
- Python Companion Chapter 3.3 - Standard Deviation
- Python Companion Chapter 3.4 - Angle

3.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

3.0.3 Lectures

The lectures for this week and their links can be found below:

- Norms ≈ 25 min.
- Chebyshev Inequality ≈ 15 min.
- Standard Deviation ≈ 23 min.
- Angle ≈ 20 min.
- Random Exercise ≈ 14 min.

3.0.4 Assignments

The assignment for this week is:

- Assignment 3 - Norm And Distance

3.0.5 Quiz

The quiz for this week is:

- Quiz 3 - Norm And Distance

3.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 3 - Norm And Distance**. The first section that we will cover is **VMLS Section 3.1 - Norm**.

VMLS Section 3.1 - Norm

Overview

The norm of a vector is a mathematical concept that quantifies the length or magnitude of a vector in a multi-dimensional space. It provides a measure of how 'big' or 'small' a vector is. The most common norm is the Euclidean norm, also known as the 2-norm or L2-norm, which calculates the vector's length as the square root of the sum of its squared components. In general, the norm of a vector in an n-dimensional space is denoted as $||b||$, where b represents the vector. The norm is a fundamental concept in linear algebra and has various applications in fields such as physics, engineering, and machine learning, where it is used for tasks like vector normalization and distance calculations.

Norm Equation

The equation for calculating a norm is

$$||x|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Properties of Norm

Norms are mathematical functions used to measure the 'size' or 'length' of vectors in vector spaces. They possess fundamental properties that make them indispensable in various mathematical fields. These properties include non-negativity, meaning that the norm of a vector is always non-negative, being zero if and only if the vector is the zero vector. Homogeneity reflects that scaling a vector proportionally scales its norm. The triangle inequality asserts that the shortest path between two points in a vector space is a straight line. The submultiplicative property relates to the Hadamard product of vectors. Norms also have equivalents in the context of infinite-dimensional spaces and sequences. Equivalence of norms shows that different norms offer similar notions of vector size. Finally, norms satisfy the property that the norm of a linear combination of vectors is less than or equal to the sum of the norms of the individual vectors. These properties establish norms as a versatile and essential tool in mathematics, finding applications in optimization, functional analysis, and signal processing, among other areas. Precisely, we can summarize these properties as

- Nonnegative homogeneity: $||\beta x|| = ||\beta||||x||$. Multiplying a vector by a scalar multiplies the norm by the absolute value of the scalar.
- Triangle inequality: $||x + y|| \leq ||x|| + ||y||$. The Euclidean norm of a sum of two vectors is no more than the sum of their norms. (The name of this property will be explained later.) Another name for this inequality is subadditivity.
- Nonnegativity: $||x|| \geq 0$.
- Definiteness: $||x|| = 0$ only if $x = 0$.

Root Mean Square

The root-mean-square (RMS) value, denoted as RMS or rms, is a mathematical measure used to represent the effective magnitude or power of a varying quantity, such as a signal or dataset. It is computed by taking the square root of the mean (average) of the squares of the values. This calculation method allows it to capture both positive and negative contributions, making it particularly useful for oscillating or alternating quantities. The RMS value preserves the energy content of the dataset and is often used in fields like physics, engineering, and signal processing to quantify signal strength, power, or variability. It serves as a common metric for comparing varying quantities to constant values, providing a meaningful and widely applicable representation of their magnitude.

Root Mean Square Equation

The equation for the root mean square is

$$\textbf{rms}(x) = \sqrt{\frac{x_1^2 + \cdots + x_n^2}{n}} = \frac{||x||}{\sqrt{n}}.$$

Norm Of Sum

The norm of the sum of two vectors, often denoted as $\|x+y\|$, represents the length or magnitude of the resultant vector when two vectors, x and y, are added together. This concept is fundamental in vector algebra and geometry. The norm of the sum obeys the triangle inequality, which states that the magnitude of the sum of two vectors is always less than or equal to the sum of their individual magnitudes, i.e., $\|x + y\| \leq \|x\| + \|y\|$. This inequality signifies that the shortest path between two points in Euclidean space is a straight line. The norm of the sum plays a crucial role in various mathematical and scientific disciplines, such as physics and engineering, where it helps quantify the combined effect or magnitude of multiple vector quantities.

Norm Of Sum Equation

To calculate the norm of sum of vectors we use

$$||x + y|| = \sqrt{||x||^2 + 2x^T y + ||y||^2}.$$

Norm Of Block Vectors

The norm of block vectors, often used in linear algebra and functional analysis, extends the concept of vector norm to structured vectors composed of multiple sub-vectors or blocks. In this context, a block vector consists of several concatenated sub-vectors, each representing a distinct component or feature of the overall vector. The norm of a block vector is typically defined as the square root of the sum of the squares of the norms of its constituent sub-vectors. This allows for the measurement of the overall magnitude or length of the block vector, while considering the individual magnitudes of its components. Block vectors and their norms find applications in various areas, including multivariate statistics, data analysis, and numerical optimization, where structured data representations are common.

Norm Of Block Vectors Equation

To calculate the norm of block vectors we use

$$||(a, b, c)|| = \sqrt{||a||^2 + ||b||^2 + ||c||^2} = ||(||a||, ||b||, ||c||)||.$$

Chebyshev Inequality

The Chebyshev Inequality is a fundamental statistical theorem that provides bounds on the probability of a random variable deviating from its mean. It is particularly useful when the exact probability distribution of the variable is unknown or when dealing with data exhibiting variability. The inequality states that for any random variable with a finite mean and variance, the probability that the random variable deviates from its mean by more than a specified number of standard deviations is bounded. Specifically, it asserts that no more than 1 divided by the square of the specified number of standard deviations will be the probability of such an event occurring. This inequality is valuable for estimating the likelihood of extreme events or outliers in a dataset and is widely used in statistics and probability theory for setting confidence intervals and making probabilistic statements about data distributions.

Chebyshev Inequality Equation

We can represent the Chebyshev Inequality in terms of the root mean square of a vector with

$$\frac{k}{n} \leq \left(\frac{\mathbf{rms}(x)}{a} \right)^2.$$

The next section that we will review this week is **VMLS Section 3.4 - Angle**.

VMLS Section 3.4 - Angle

Overview

The concept of the angle between two vectors is fundamental in vector mathematics and linear algebra. It provides a measure of the angular separation between the directions of two vectors within a vector space. The angle between two vectors can be determined by utilizing the dot product and trigonometric functions. Specifically, the dot product formula relates the dot product of vectors A and B to the magnitudes of the vectors and the cosine of the angle between them. Solving for through inverse cosine ($\cos^{-1}$), we can express it as the arc cosine of the dot product divided by the product of the magnitudes. This mathematical approach offers a versatile method for calculating the angle between vectors in various applications, including physics, engineering, and computer graphics, where understanding spatial orientations is crucial.

Cauchy-Schwarz Inequality

The Cauchy-Schwarz inequality is a fundamental mathematical inequality that applies to inner product spaces, including Euclidean spaces. It states that for any two vectors, the absolute value of their inner product is always less than or equal to the product of their individual norms or magnitudes. Mathematically, for vectors $\mathbf{a}$ and $\mathbf{b}$, the inequality is expressed as $|\mathbf{a}^T\mathbf{b}| \leq \|\mathbf{a}\| \cdot \|\mathbf{b}\|$. Geometrically, this means that the angle between the vectors is related to their inner product and magnitudes, providing a measure of how much the vectors align. The Cauchy-Schwarz inequality has wide-ranging applications in mathematics, physics, engineering, and statistics, serving as a fundamental tool for establishing bounds and relationships between vectors in various contexts.

Cauchy-Schwarz Inequality Equation

To calculate the Cauchy-Schwarz inequality we use

$$|a^T b| \leq ||a||||b||.$$

Correlation Coefficient

The correlation coefficient is a statistical measure used to assess the strength and direction of the linear relationship between two variables. It quantifies how well the variations in one variable can be predicted by the variations in another. The coefficient typically ranges from -1 to 1, with -1 indicating a perfect negative linear relationship, 1 representing a perfect positive linear relationship, and 0 indicating no linear relationship. A positive coefficient implies that as one variable increases, the other tends to increase as well, while a negative coefficient suggests that as one variable increases, the other tends to decrease. The correlation coefficient is widely employed in fields like statistics, economics, and social sciences to analyze data, make predictions, and determine the degree of association between variables.

Correlation Coefficient Equation

Given two vectors a and b, their associated de-meaned vectors are

$$\tilde{a} = a - \mathbf{avg}(a)\mathbf{1} \ , \ \tilde{b} = b - \mathbf{avg}(b)\mathbf{1}.$$

The correlation coefficient ρ is then calculated with

$$\rho = \frac{\tilde{a}^T\tilde{b}}{||\tilde{a}||||\tilde{b}||}.$$

With

$$u = \frac{\tilde{a}}{\mathbf{std}(a)} \ , \ v = \frac{\tilde{b}}{\mathbf{std}(b)}$$

we can also calculate the correlation coefficient with

$$\rho = \frac{u^T v}{n}.$$

Standard Deviation Of Sum

The standard deviation of the sum of vectors is a statistical measure used to quantify the spread or variability of the resultant vector when multiple random vectors are added together. In the context of vectors, particularly in statistics and probability theory, this concept is valuable for understanding the uncertainty or dispersion that arises when combining multiple random variables. It extends the notion of standard deviation to vector spaces and is often employed in fields like finance and data analysis to assess risk or measure the overall variability of a portfolio

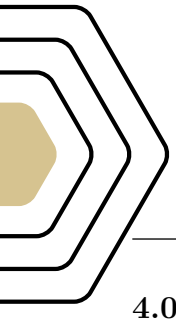
or dataset resulting from the aggregation of individual vector components. Calculating the standard deviation of the sum involves considering both the individual standard deviations of the vectors and their pairwise covariances, providing valuable insights into the behavior of the combined vector.

Standard Deviation Of Sum Equation

To calculate the standard deviation of sum we use

$$\mathbf{std}(a+b) = \sqrt{\mathbf{std}(a)^2 + 2\rho\mathbf{std}(a)\mathbf{std}(b) + \mathbf{std}(b)^2}.$$

Clustering



Clustering

4.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- **Sections 4.1, 4.2, 4.3, 4.4, 4.5** from VMLS.
- VMLS Chapter 4.1 - Clustering
- VMLS Chapter 4.2 - A Clustering Objective
- VMLS Chapter 4.3 - The K-Means Algorithm
- VMLS Chapter 4.4 - Examples
- VMLS Chapter 4.5 - Applications
- Python Companion Chapter 4.1 - Clustering
- Python Companion Chapter 4.2 - A Clustering Objective
- Python Companion Chapter 4.3 - The K-Means Algorithm
- Python Companion Chapter 4.4 - Examples
- Python Companion Chapter 4.5 - Applications

4.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

4.0.3 Lectures

The lectures for this week and their links can be found below:

- VMLS Chapter 4 - Part 1 $\approx$ 32 min.
- VMLS Chapter 4 - Part 2 $\approx$ 19 min.

4.0.4 Assignments

The assignment for this week is:

- Assignment 4 - Clustering

4.0.5 Quiz

The quiz for this week is:

- Quiz 4 - Clustering

4.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 4 - Clustering**. The first section that we will cover is **VMLS Section 4.1 - Clustering**.

VMLS Section 4.1 - Clustering

Overview

Clustering is a foundational technique in the field of machine learning and data analysis, particularly in unsupervised learning. At its core, clustering aims to uncover latent patterns or structures within a dataset by grouping similar data points together, making it an essential tool for understanding and organizing large datasets. This process does not rely on any prior knowledge of labels or categories, making it particularly valuable when exploring unfamiliar data.

One of the primary objectives of clustering is to reveal the inherent structure of data, which can provide valuable insights and facilitate decision-making processes. For example, in customer segmentation, businesses can employ clustering techniques to identify distinct groups of customers based on their purchasing behavior or demographic information. This information can then be used to tailor marketing strategies, recommend products, or improve customer experiences.

Various clustering algorithms exist to address different types of data and objectives. K-means clustering, one of the most widely used algorithms, partitions data points into clusters based on their proximity to a centroid. Hierarchical clustering, on the other hand, organizes data into a tree-like structure, allowing for both fine-grained and high-level cluster identification. Density-based spatial clustering of applications with noise (DBSCAN) identifies clusters based on data point density and is effective in detecting outliers.

The applications of clustering are vast and diverse, spanning various domains such as biology, finance, image processing, and social network analysis. In biology, clustering can help identify patterns in gene expression data, aiding in the understanding of genetic relationships. In finance, it can assist in portfolio optimization by grouping similar financial assets. In image processing, clustering can segment images based on pixel similarity, enabling object recognition. Overall, clustering plays a pivotal role in uncovering hidden structures within data, making it an indispensable tool in modern data analysis and machine learning.

The second section that we will cover is **VMLS Section 4.2 - A Clustering Objective**.

VMLS Section 4.2 - A Clustering Objective

Overview

A clustering objective is a fundamental component in the process of clustering data points into meaningful groups or clusters. It defines the criteria or goals that a clustering algorithm aims to achieve when partitioning data. The choice of clustering objective heavily influences the outcome and quality of the clustering results.

Common clustering objectives include minimizing the intra-cluster distance and maximizing the inter-cluster distance. In other words, clustering algorithms strive to create clusters where data points within the same cluster are similar to each other while being dissimilar to data points in other clusters. The specific distance metric used to measure similarity or dissimilarity varies depending on the algorithm and the nature of the data.

The choice of clustering objective depends on the problem and the characteristics of the data. For example, K-means clustering seeks to minimize the sum of squared distances between data points and their cluster centroids, making it suitable for spherical or globular clusters. In contrast, agglomerative hierarchical clustering aims to maximize the similarity between merged clusters, making it useful for identifying hierarchical structures in data.

Ultimately, the clustering objective guides the algorithm's decision-making process, helping it discover meaningful patterns or structures within the data. The effectiveness of a clustering algorithm is often evaluated based on how well it optimizes the chosen objective function. Therefore, selecting an appropriate clustering objective is a critical step in the clustering process and can greatly impact the interpretability and usefulness of the resulting clusters in various applications.

Optimal And Suboptimal Clustering

In the realm of clustering, the pursuit of optimal and suboptimal solutions plays a pivotal role. Achieving an optimal clustering solution involves finding the absolute best partitioning of data points into clusters, typically by minimizing or maximizing a specific objective function. However, in many real-world scenarios, finding the global optimal solution is computationally challenging, if not impossible. As a result, clustering algorithms often aim for

suboptimal solutions that provide reasonably good clusterings while sacrificing the guarantee of global optimality. Suboptimal clustering approaches, such as K-means and hierarchical clustering, offer efficient and practical means of partitioning data, making them widely used in various applications. The choice between optimal and suboptimal clustering hinges on the trade-off between computational complexity and the need for precise clustering results, with suboptimal methods serving as valuable tools when computational resources are limited or when approximate solutions are acceptable.

Cluster Centroid

In the context of clustering, a cluster centroid represents a central point within a cluster, often used as a reference or representative of that cluster. The centroid is typically computed as the mean or geometric center of all the data points within a cluster, and its coordinates serve as a summary or prototype of the cluster's characteristics. Cluster centroids play a crucial role in various clustering algorithms, such as K-means, where the goal is to iteratively adjust centroids to minimize the distance between data points and their assigned centroids, ultimately leading to well-defined clusters. Centroids serve as a useful reference for understanding the typical characteristics of each cluster and can be valuable in applications like data analysis, classification, and recommendation systems.

Cluster Centroid

The cluster centroid is calculated with

$$z_j = \frac{1}{|G_j|} \sum_{i \in G_j} x_i$$

where $|G_j|$ is standard mathematical notation for the number of elements in the set G_j (the size of group j).

The next section that we will cover is **VMLS Section 4.3 - The k-Means Algorithm**.

VMLS Section 4.3 - The k-Means Algorithm

Overview

The k-means algorithm is a versatile and widely used clustering technique in the field of machine learning and data analysis. Its primary purpose is to group similar data points into clusters while minimizing the overall variance within each cluster. This unsupervised learning method is particularly helpful in uncovering patterns and structures within datasets without the need for labeled data.

The algorithm's operation is relatively straightforward. It begins by randomly placing k centroids in the feature space, where k is a user-defined parameter representing the desired number of clusters. Each data point is then assigned to the nearest centroid based on a chosen distance metric, often the squared Euclidean distance. After the initial assignment, the algorithm calculates new centroids by computing the mean of all data points assigned to each cluster. This process iterates until convergence, either when the centroids no longer change significantly or after a predefined number of iterations.

K-means clustering is known for its simplicity and efficiency, making it suitable for various applications, including customer segmentation, image compression, and document organization. However, it has some limitations, such as its sensitivity to initial centroid placement, which can lead to suboptimal results. Additionally, k-means assumes that clusters are spherical and equally sized, making it less effective for datasets with irregularly shaped or differently sized clusters. To address these issues, variations of the algorithm, such as K-means++, hierarchical clustering, and DBSCAN, have been developed to provide more robust clustering solutions for complex datasets.

Convergence

In the context of clustering, convergence refers to the point at which the clustering algorithm reaches a stable and optimal solution. The goal is to ensure that the clustering process stops when further iterations do not significantly change the cluster assignments. Convergence is crucial to identify the best grouping of data points and prevents unnecessary computational overhead. It is typically achieved when the centroids of clusters no longer shift or when a predefined criterion, such as a maximum number of iterations or a minimal change in cluster assignments, is

met. Convergence ensures that the clustering algorithm provides consistent and reliable results, facilitating the interpretation and utilization of the clustered data in various applications.

The next section that we will cover is **VMLS Section 4.4 - Examples**.

VMLS Section 4.4 - Examples

Examples

Examples in clustering play a pivotal role in various domains, offering valuable insights and aiding decision-making processes. One prominent application area is customer segmentation in marketing. Companies can use clustering algorithms to group customers based on their purchase histories, demographics, or online behavior. By identifying distinct customer segments, businesses can tailor marketing strategies, personalize product recommendations, and optimize advertising campaigns, thereby enhancing customer engagement and increasing revenue.

In the realm of computer vision, image segmentation is another compelling example. Clustering algorithms help partition images into regions or objects with similar visual characteristics, facilitating object recognition, background removal, and image editing tasks. This is particularly valuable in medical imaging for identifying and analyzing specific anatomical structures or anomalies within images, aiding in disease diagnosis and treatment planning.

In natural language processing, clustering assists with document categorization. By grouping documents with similar content or themes, it simplifies information retrieval and content organization. News articles, academic papers, or user-generated content can be efficiently categorized, making it easier for users to access relevant information. Moreover, in biology, clustering techniques are employed for species classification based on genetic data, enabling researchers to uncover evolutionary relationships and identify new species.

These diverse examples underscore the versatility and significance of clustering across different fields. They showcase how clustering algorithms empower organizations and researchers to extract meaningful patterns from data, improve processes, and make data-driven decisions, ultimately driving innovation and progress in numerous domains.

The last section that we will cover is **VMLS Section 4.5 - Applications**.

VMLS Section 4.5 - Applications

Applications

Clustering, a fundamental technique in data analysis, finds applications in various domains, offering insights, organization, and efficiency. In marketing, customer segmentation leverages clustering to group individuals with similar purchase histories, behaviors, or demographics. This enables businesses to personalize marketing strategies, target specific customer groups, and enhance overall customer satisfaction.

In computer vision, image segmentation is a prominent application. Clustering algorithms partition images into distinct regions based on visual characteristics, aiding in object recognition, background removal, and image analysis. This is valuable in medical imaging for identifying and analyzing anatomical structures, assisting in disease diagnosis and treatment planning.

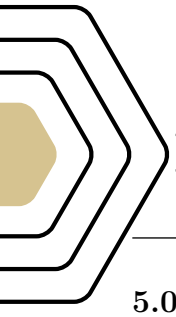
Text data benefits from clustering in natural language processing. Document clustering categorizes texts with similar content or themes, simplifying information retrieval and content organization. It's invaluable for organizing large document collections, such as news articles, academic papers, or user-generated content.

In biology, clustering techniques are applied to genetic data for species classification, revealing evolutionary relationships and identifying new species. This has far-reaching implications for biodiversity conservation and understanding the tree of life.

Moreover, clustering is used in recommendation systems to group users or items with similar preferences, enhancing personalized content recommendations. Anomaly detection in cybersecurity, network analysis in social sciences, and optimization in transportation are among other applications of clustering.

These applications underscore clustering's versatility and significance in extracting insights from data, improving decision-making processes, and advancing various fields.

Linear Independence



Linear Independence

5.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- **Sections 5.1, 5.2, 5.3, 5.4** from VMLS.
- VMLS Chapter 5.1 - Linear Dependence
- VMLS Chapter 5.2 - Basis
- VMLS Chapter 5.3 - Orthonormal Vectors
- VMLS Chapter 5.4 - Gram-Schmidt Algorithm
- Python Companion Chapter 5.1 - Linear Independence
- Python Companion Chapter 5.2 - Basis
- Python Companion Chapter 5.3 - Orthonormal Vectors
- Python Companion Chapter 5.4 - Gram-Schmidt Algorithm

5.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

5.0.3 Lectures

The lectures for this week and their links can be found below:

- 5.1 - Linear Independence ≈ 13 min.
- 5.2 - Basis ≈ 16 min.
- 5.3 - Orthogonal Vectors ≈ 18 min.
- 5.4 - Gram Schmidt Algorithm ≈ 14 min.
- 5.5 - Complexity ≈ 11 min.
- Random Exercise Chapter 5 ≈ 17 min.

5.0.4 Assignments

The assignment for this week is:

- Assignment 5 - Linear Independence

5.0.5 Quiz

The quiz for this week is:

- Quiz 5 - Linear Independence

5.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 5 - Linear Independence**. The first section that we will cover is **VMLS Section 5.1 - Linear Independence**.

VMLS Section 5.1 - Linear Independence

Overview

Linear independence is a fundamental concept in linear algebra that plays a central role in various mathematical and scientific fields. It provides a powerful framework for understanding the relationships between vectors within a vector space and forms the basis for many key concepts and operations in linear algebra.

One of the primary implications of linear independence is that it allows us to express vectors in a unique and efficient manner. When a set of vectors is linearly independent, each vector contributes a distinct direction or piece of information to the vector space. This uniqueness is valuable in representing data or systems of equations. For example, in the context of data analysis, linearly independent vectors can help us capture different features or dimensions of a dataset without redundancy. In contrast, linearly dependent vectors can introduce redundancy, making it more challenging to interpret or work with the data effectively.

Moreover, linear independence is closely related to the concept of a basis in linear algebra. A basis is a set of linearly independent vectors that spans a vector space, meaning that any vector in that space can be expressed as a unique linear combination of the basis vectors. Bases are essential because they provide a concise and systematic way to represent vectors within a space. For example, the standard unit vectors in Euclidean space (e.g., (1, 0, 0) and (0, 1, 0) in three-dimensional space) form a basis, allowing us to describe any point in that space using linear combinations of these unit vectors.

Additionally, linear independence is crucial for solving systems of linear equations. When vectors are linearly independent, the system of equations they represent has a unique solution. This property is essential in various scientific and engineering applications, such as solving electrical circuits, analyzing chemical reactions, or optimizing economic models. In contrast, linear dependence in a system of equations can lead to multiple solutions or no solution at all, making it harder to predict or control the behavior of the system.

Linear independence is a foundational concept in linear algebra that underpins many aspects of mathematics and science. It enables us to efficiently represent vectors, construct bases, and solve systems of equations. Whether in data analysis, engineering, physics, or computer science, an understanding of linear independence is essential for making sense of complex mathematical structures and real-world problems.

Linear Independence Definition

For a collection of vectors to be considered ***linearly independent*** of one another, the collection of vectors must not be linearly dependent. Mathematically this means

$$\sum_{i=1}^{k} \beta_i a_i = 0$$

where the above equation is only valid if $\beta_i = \cdots = \beta_k = 0$. For a collection of vectors to be ***linearly dependent***, the above equation is valid where not all of the coefficients β are equal to zero. Namely, $\beta_i \neq \ldots \neq \beta_k \neq 0$.

The next section that we will cover is **VMLS Section 5.2 - Basis**.

VMLS Section 5.2 - Basis

Overview

In the realm of linear algebra, the concept of a basis is fundamental and plays a pivotal role in understanding vector spaces and linear transformations. A basis is essentially a set of vectors that possesses unique and essential properties, making it a cornerstone of vector space theory.

One of the key characteristics of a basis is linear independence. Linear independence ensures that no vector in the basis can be expressed as a linear combination of the others. In simpler terms, the basis vectors are distinct and do not overlap in their representation of directions within the vector space. This property is crucial because it

guarantees that each vector in the space can be uniquely represented using the basis vectors. It's akin to having a set of directions where no two directions are redundant.

Another crucial property of a basis is its spanning ability. A basis should be capable of representing every vector within the vector space by forming linear combinations of its constituent vectors. This means that the basis vectors collectively cover the entire vector space, and any vector within that space can be expressed as a unique combination of these basis vectors. Think of it as having a versatile toolbox of vectors that can be used to build any vector you might encounter in the space.

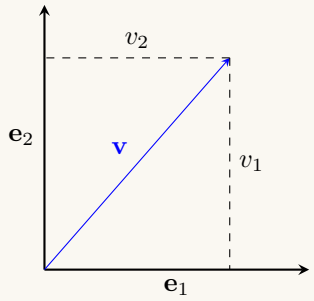
Bases are not arbitrary sets of vectors; they are meticulously constructed to satisfy these properties while being as minimal as possible. In other words, a basis should have the smallest number of vectors needed to span the entire space while preserving linear independence. This minimalist approach ensures that the basis is efficient and concise in representing the vector space.

Different vector spaces can have different types of bases. For example, in Euclidean space (R^n), the standard unit vectors (e.g., [1, 0, 0], [0, 1, 0], [0, 0, 1] for R^3) form a natural basis. In the space of polynomials of degree at most n, the monomials $\{1, x, x^2, \ldots, x^n\}$ can serve as a basis. These bases are tailored to the specific characteristics of their respective vector spaces.

In practice, bases are invaluable for solving linear equations, diagonalizing matrices, and understanding linear transformations. They provide structured and systematic ways to work with vectors and linear operators, simplifying complex problems by breaking them down into more manageable components. In essence, bases are the foundation upon which much of linear algebra is built, making them an essential concept for anyone delving into this mathematical field.

Basis Example

In linear algebra, a basis is a fundamental concept that forms the building blocks for vector spaces. A basis is a set of vectors that can be used to represent any vector within that vector space uniquely. To visualize this concept, let's consider a 2D vector space.



In the example provided, we have a 2D basis consisting of two vectors, $\mathbf{e}_1$ and $\mathbf{e}_2$. These basis vectors span the entire 2D space, meaning that any vector in this space can be expressed as a unique combination of these basis vectors. In this case, $\mathbf{e}_1$ points along the x-axis, and $\mathbf{e}_2$ points along the y-axis.

We also have a vector $\mathbf{v}$, represented in blue, which we want to express in terms of the basis vectors. This involves finding two scalars, v_1 and v_2, such that $\mathbf{v} = v_1\mathbf{e}_1 + v_2\mathbf{e}_2$. The dashed lines in the diagram represent these components: v_1 is the length along $\mathbf{e}_1$, and v_2 is the length along $\mathbf{e}_2$.

This illustration demonstrates how vectors within a vector space can be uniquely represented using a basis. The coefficients v_1 and v_2 are the coordinates of the vector $\mathbf{v}$ with respect to the basis vectors $\mathbf{e}_1$ and $\mathbf{e}_2$, respectively. The concept of a basis is crucial in linear algebra as it forms the foundation for various operations and transformations within vector spaces.

Consequently, there is a way that we can represent a vector as a linear combination of vectors for a given basis.

Basis Representation

For any n-vector b, we can write b as

$$b = b_1 e_1 + \cdots + b_n e_n$$

where e_i are the basis of b. In this context $\mathbf{e}$, represents unit vectors in this space.

The next section that we will cover is **VMLS Section 5.3 - Orthonormal Vectors**.

VMLS Section 5.3 - Orthonormal Vectors

Overview

Orthogonal vectors are a fundamental concept in linear algebra, representing vectors that are mutually perpendicular to each other. In other words, orthogonal vectors have a dot product of zero, which indicates that the angle between them is 90 degrees, forming a right angle. This geometric property makes them particularly useful in various mathematical and engineering applications.

Orthogonal vectors are essential in defining orthogonal bases for vector spaces. An orthogonal basis is a set of vectors where each vector is orthogonal to all others, and it can simplify vector calculations significantly. In such bases, finding the coefficients to represent a vector becomes straightforward, often involving simple dot products.

One special case of orthogonal vectors is orthonormal vectors, where orthogonal vectors have unit length, making them particularly convenient for calculations. Orthonormal bases are common in applications like linear transformations and signal processing, where they simplify computations and help maintain vector magnitudes.

Orthogonal vectors are vectors that are perpendicular to each other, with a dot product of zero. They play a crucial role in linear algebra, forming the basis for orthonormal bases that simplify vector computations and have applications in various fields, including physics, engineering, and computer science.

Orthonormal Basis

An orthonormal basis is a special type of basis in linear algebra that combines two essential properties: orthogonality and normalization. In an orthonormal basis, the basis vectors are not only mutually orthogonal (perpendicular to each other) but also normalized (each vector has a magnitude of 1). These two properties make orthonormal bases particularly useful in various mathematical and engineering applications.

The orthogonality property means that the dot product of any two distinct basis vectors is zero, indicating that they are geometrically perpendicular to each other. This property simplifies vector calculations significantly, making it easier to find coefficients when representing a vector in this basis. Additionally, the normalization property ensures that the basis vectors have unit length, which simplifies magnitude and projection calculations.

Orthonormal bases are especially valuable in linear transformations, where they simplify the transformation process and help maintain the magnitudes of vectors. They are commonly used in signal processing, quantum mechanics, and numerical methods. In quantum mechanics, for example, the states of a quantum system are often represented using an orthonormal basis, simplifying calculations related to measurements and state transformations.

An orthonormal basis consists of vectors that are both orthogonal and normalized. These bases are instrumental in linear algebra and various scientific and engineering disciplines, as they simplify calculations, facilitate vector representations, and ensure consistent magnitudes in mathematical operations.

Orthonormal Basis

To represent an n-vector x with an orthonormal basis $a_1, \ldots, a_n$ we use

$$x = \sum_{i=1}^{n} (a_i^T x) a_i.$$

x is expressed as a linear combination of $a_1, \ldots, a_n$.

The last section that we will cover is **VMLS Section 5.4 - Gram-Schmidt Algorithm**

VMLS Section 5.4 - Gram-Schmidt Algorithm

Overview

The Gram-Schmidt algorithm is a fundamental technique in linear algebra used to transform a set of linearly independent vectors into an orthonormal basis. This process is crucial in various applications, including solving systems of linear equations, eigenvalue problems, and signal processing. The algorithm works by taking a set of

linearly independent vectors and systematically orthogonalizing and normalizing them to create a new set of vectors that are both orthogonal (perpendicular) and normalized (unit length).

The Gram-Schmidt process begins with the first vector in the original set, which is automatically included in the new orthonormal basis as the first vector. Subsequent vectors are processed one by one. For each vector, the algorithm subtracts its projections onto the previously processed vectors from itself, effectively making it orthogonal to those vectors. Then, it normalizes the resulting orthogonal vector to ensure unit length.

Mathematically, the Gram-Schmidt algorithm takes a set of vectors $\{v_1, \ldots, v_k\}$ and transforms them into an orthonormal basis $\{u_1, \ldots, u_k\}$. The resulting basis vectors satisfy two crucial properties: they are mutually orthogonal, meaning their dot products are zero for different vectors, and they are normalized, having a magnitude of 1.

The Gram-Schmidt algorithm is a valuable tool for a wide range of applications in linear algebra and beyond. It helps simplify calculations, particularly in solving linear systems and finding eigenvalues and eigenvectors. Additionally, it plays a vital role in signal processing, quantum mechanics, and various scientific and engineering fields where orthogonal and normalized bases are essential.

The Gram-Schmidt algorithm is a method for converting a set of linearly independent vectors into an orthonormal basis. By ensuring orthogonality and normalization, it simplifies mathematical operations and is a fundamental tool in linear algebra and various scientific disciplines.

Gram-Schmidt Algorithm

Given n-vectors $\mathbf{a}_1, \ldots, \mathbf{a}_k$ for $i = 1, \ldots, k$, perform the following steps:

1. **Orthogonalization:** $\mathbf{q}_i = \mathbf{a}_i - (\mathbf{q}_1^T \mathbf{a}_i)\mathbf{q}_1 - \ldots - (\mathbf{q}_{i-1}^T \mathbf{a}_i)\mathbf{q}_{i-1}$
2. **Linear Independence:** If $\mathbf{q}_i = \mathbf{0}$, quit.
3. **Normalization:** $\mathbf{q}_i = \frac{\tilde{\mathbf{q}}_i}{\|\tilde{\mathbf{q}}_i\|}$

Exam 1

Exam 1

6.0.1 Piazza

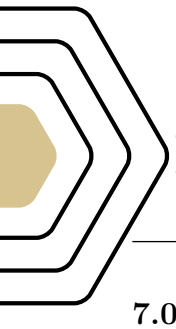
Must post / respond to at least **four** Piazza posts this week.

6.0.2 Exam

The exam for this week is:

- Exam 1 Notes

Matrices



Matrices

7.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- VMLS Chapter 6.1 - Matrices
- VMLS Chapter 6.2 - Zero And Identity Matrices
- VMLS Chapter 6.3 - Transpose, Addition, And Norm
- VMLS Chapter 6.4 - Matrix-Vector Multiplication
- Python Companion Chapter 6.1 - Matrices
- Python Companion Chapter 6.2 - Zero And Identity Matrices
- Python Companion Chapter 6.3 - Transpose, Addition, And Norm
- Python Companion Chapter 6.4 - Matrix-Vector Multiplication

7.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

7.0.3 Lectures

The lectures for this week and their links can be found below:

- 6.1 - Matrices $\approx$ 16 min.
- 6.2 - Matrix Operations $\approx$ 12 min.
- 6.3 - Matrix-Vector Multiply $\approx$ 22 min.
- Random Exercise: Chapter 6 $\approx$ 4 min.

7.0.4 Assignments

The assignment for this week is:

- Assignment 6 - Matrices

7.0.5 Quiz

The quiz for this week is:

- Quiz 6 - Matrices

7.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 6 - Matrices**. The first section that we will cover this week is **VMLS Section 6.1 - Matrices**.

VMLS Section 6.1 - Matrices

Overview

Matrices are fundamental mathematical structures used to represent and manipulate data in various fields, including linear algebra, statistics, and computer science. A matrix is essentially a two-dimensional array of numbers, symbols, or expressions arranged in rows and columns. It provides a compact and organized way to store and perform operations on data.

Matrices have several key characteristics and properties:

1. Dimensions: A matrix is defined by its dimensions, which specify the number of rows and columns. For example, an "m x n" matrix has m rows and n columns.
2. Elements: Each entry in a matrix is called an element, and it is identified by its position using row and column indices. Elements can be real numbers, complex numbers, or symbols.
3. Types: Matrices come in various types, such as square matrices (equal number of rows and columns), rectangular matrices (unequal number of rows and columns), row matrices (1 row and multiple columns), and column matrices (1 column and multiple rows).
4. Operations: Matrices support various operations, including addition, subtraction, scalar multiplication, and matrix multiplication. Matrix multiplication is a fundamental operation in linear algebra, often used for transformations and solving systems of linear equations.
5. Transposition: The transpose of a matrix involves switching its rows and columns. It's denoted by "A^T" and is useful for various mathematical operations and applications.
6. Inverse: Not all matrices have inverses, but those that do can be inverted to find a matrix that, when multiplied by the original matrix, results in the identity matrix.
7. Applications: Matrices find applications in diverse fields, including physics, engineering, computer graphics, data analysis, and machine learning. They are used for solving linear equations, representing transformations, and handling multi-dimensional data.

Matrices are versatile mathematical structures that play a crucial role in representing, organizing, and manipulating data in various mathematical and practical contexts. Their properties and operations make them invaluable tools in fields that deal with complex data and equations.

The next section that we will cover this week is **VMLS Section 6.2 - Zero And Identity Matrices**.

VMLS Section 6.2 - Zero And Identity Matrices

Overview

Zero and Identity matrices are special types of matrices with unique properties and applications in linear algebra and mathematics.

Zero Matrix

A Zero Matrix, denoted as $\mathbf{0}$ or $\mathbf{0}_{m\times n}$, is a matrix in which all of its elements are equal to zero. It has the following characteristics:

- All elements are zeros: $a_{ij} = 0$ for all i and j.
- Dimensions: It can have any dimensions, such as $m \times n$, where m represents the number of rows, and n represents the number of columns.
- Properties: When a matrix is added to a zero matrix, it remains unchanged ($A + \mathbf{0} = A$), and when multiplied by a zero matrix, the result is also a zero matrix ($A \cdot \mathbf{0} = \mathbf{0}$).

- Application: Zero matrices are often used in matrix operations to represent additive identity elements and for various algebraic manipulations.

Identity Matrix

An Identity Matrix, denoted as $\mathbf{I}$ or $\mathbf{I}_n$ (or simply $\mathbf{I}$ when the size is clear), is a square matrix in which all elements on the main diagonal (from the top-left to the bottom-right) are equal to 1, and all other elements are zero. It has the following characteristics:

- Diagonal elements are 1: $a_{ii} = 1$ for all i, and $a_{ij} = 0$ for $i \neq j$.
- Dimensions: It is always a square matrix, represented as $n \times n$.
- Properties: When a matrix A is multiplied by an identity matrix, it remains unchanged ($A \cdot \mathbf{I} = A$). The identity matrix serves as a multiplicative identity element in matrix multiplication.
- Application: Identity matrices are fundamental in linear transformations and solving systems of linear equations. They represent transformations that leave vectors unchanged in magnitude and direction.

Zero and Identity matrices are essential mathematical constructs in linear algebra. The Zero Matrix contains only zero elements and serves as an additive identity, while the Identity Matrix contains diagonal elements of 1 and serves as a multiplicative identity in matrix operations. These matrices play crucial roles in various mathematical operations and applications.

The next section that we will cover this week is **VMLS Section 6.3 - Transpose, Addition, And Norm**.

VMLS Section 6.3 - Transpose, Addition, And Norm

Overview

Matrix operations play a fundamental role in linear algebra, offering powerful tools for manipulating and analyzing data. The transpose of a matrix, denoted as A^T, involves swapping rows and columns and has properties such as self-inversion and distribution over addition. Matrix addition, $A + B$, combines two matrices element-wise, obeying commutative and associative properties. The norm of a matrix, like the Frobenius norm ($\|A\|_F$), quantifies the matrix's magnitude and is useful in measuring data size. These operations serve as building blocks for solving various mathematical problems, and their properties provide a solid foundation for understanding and working with matrices in diverse applications.

Transpose of a Matrix

The transpose of a matrix A, denoted as A^T, is a new matrix obtained by swapping its rows and columns. The transpose operation has the following properties:

- If A is an $m \times n$ matrix, then A^T is an $n \times m$ matrix.
- The transpose of a transpose matrix is the original matrix: $(A^T)^T = A$.
- Transpose distributes over addition: $(A + B)^T = A^T + B^T$.
- Transpose of a product is the product of transposes in reverse order: $(AB)^T = B^T A^T$.

Matrix Addition

Matrix addition is an operation that combines two matrices of the same dimensions element-wise. Let A and B be two $m \times n$ matrices. Their sum, denoted as $A + B$, is calculated by adding corresponding elements:

$$(A + B)_{ij} = A_{ij} + B_{ij} \text{ for } 1 \leq i \leq m,\ 1 \leq j \leq n.$$

Matrix addition has the following properties:

- Commutative: $A + B = B + A$.
- Associative: $(A + B) + C = A + (B + C)$.

- Identity element: There exists a zero matrix $\mathbf{0}$ such that $A + \mathbf{0} = A$ for any matrix A.
- Inverse element: For every matrix A, there exists an additive inverse matrix $(-A)$ such that $A + (-A) = \mathbf{0}$.

Norm of a Matrix

The norm of a matrix is a scalar value that quantifies the "size" or magnitude of the matrix. There are different ways to define the norm of a matrix, but one common norm is the Frobenius norm, denoted as $\|A\|_F$, for an $m \times n$ matrix A. The Frobenius norm is calculated as:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}.$$

The Frobenius norm satisfies the following properties:

- Non-negativity: $\|A\|_F \geq 0$, and $\|A\|_F = 0$ if and only if A is the zero matrix.
- Scalar multiplication: $\|\alpha A\|_F = |\alpha| \cdot \|A\|_F$ for any scalar α.
- Triangle inequality: $\|A + B\|_F \leq \|A\|_F + \|B\|_F$.

These properties make the Frobenius norm a useful tool for measuring the size of matrices in various applications, including linear algebra and optimization.

The last section that we will cover this week is **VMLS Section 6.4 - Matrix-Vector Multiplication**.

VMLS Section 6.4 - Matrix-Vector Multiplication

Overview

Matrix-vector multiplication is a fundamental operation in linear algebra where a matrix is multiplied by a vector to produce a new vector. This operation combines the linear transformations represented by the matrix with the vector's coordinates, resulting in a transformed vector. The process involves taking the dot product of each row of the matrix with the vector, and the resulting components of the new vector represent linear combinations of the matrix's rows. Matrix-vector multiplication is a cornerstone in solving systems of linear equations, representing linear transformations, and finding eigenvectors and eigenvalues, making it a crucial concept in various mathematical and scientific fields.

Matrix-Vector Multiplication

Consider an $m \times n$ matrix A and an n-dimensional column vector $\mathbf{v}$. The result of the matrix-vector multiplication $A\mathbf{v}$ is an m-dimensional column vector, denoted as $\mathbf{w}$. Each component of $\mathbf{w}$ is obtained by taking a linear combination of the columns of matrix A, where the coefficients are provided by the components of vector $\mathbf{v}$. In other words, $\mathbf{w}$ is computed as follows:

$$\mathbf{w} = A\mathbf{v} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} a_{1i}v_i \\ \sum_{i=1}^{n} a_{2i}v_i \\ \vdots \\ \sum_{i=1}^{n} a_{mi}v_i \end{bmatrix}$$

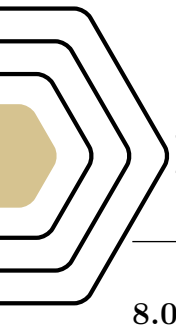
Key points about matrix-vector multiplication include:

1. **Linear Combination:** This operation computes a linear combination of the columns of matrix A, with the weights determined by the components of vector $\mathbf{v}$.
2. **Dimensions:** The dimensions of the resulting vector $\mathbf{w}$ depend on the dimensions of matrix A and vector $\mathbf{v}$. If A is of size $m \times n$, then $\mathbf{v}$ must be an n-dimensional vector, resulting in an m-dimensional vector $\mathbf{w}$.
3. **Applications:** Matrix-vector multiplication is used for solving linear equations, representing linear transformations, and finding solutions to optimization problems. It plays a vital role in computer graphics, data analysis, and scientific simulations.

4. **Computational Efficiency:** Efficient algorithms exist for matrix-vector multiplication, making it suitable for large matrices and enabling complex computations in various fields.

Matrix-vector multiplication serves as a fundamental building block for advanced linear algebraic operations and is a cornerstone of scientific and engineering applications.

Matrix Operations



Matrix Operations

8.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- **Sections 7.1, 7.2, 7.3, 7.4** from VMLS.
- VMLS Chapter 7.1 - Geometric Transformations
- VMLS Chapter 7.2 - Selectors
- VMLS Chapter 7.3 - Incidence Matrix
- VMLS Chapter 7.4 - Convolution
- Python Companion Chapter 7.1 - Geometric Transformations
- Python Companion Chapter 7.2 - Selectors
- Python Companion Chapter 7.3 - Incidence Matrix
- Python Companion Chapter 7.4 - Convolution

8.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

8.0.3 Lectures

The lectures for this week and their links can be found below:

- VMLS Chapter 7 Part 1 ≈ 6 min.
- VMLS Chapter 7 Part 2 ≈ 16 min.
- VMLS Chapter 7 Part 3 ≈ 16 min.

8.0.4 Assignments

The assignment for this week is:

- Assignment 7 - Matrix Operations

8.0.5 Quiz

The quiz for this week is:

- Quiz 7 - Matrix Operations

8.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 7 - Matrix Examples**. The first section that we will cover this week is **VMLS Section 7.1 - Geometric Transformations**.

VMLS Section 7.1 - Geometric Transformations

Overview

Geometric transformations refer to a fundamental concept in mathematics and computer graphics where the shape, position, or orientation of objects in a geometric space are altered. These transformations are commonly used to manipulate and manipulate shapes, figures, and images. There are several types of geometric transformations, including translation, rotation, scaling, shearing, and reflection, each serving a unique purpose in transforming objects in a 2D or 3D space.

1. **Translation:** Translation involves shifting an object's position without altering its shape or orientation. In 2D, this is done by adding or subtracting values to the x and y coordinates of each point, effectively moving the object in a specific direction.
2. **Rotation:** Rotation transforms an object by changing its orientation around a fixed point or axis. In 2D, this can be achieved through trigonometric functions to determine new coordinates based on a specified angle of rotation.
3. **Scaling:** Scaling changes the size of an object, making it either larger or smaller while maintaining its shape. Scaling factors are applied to the object's dimensions, modifying its proportions accordingly.
4. **Shearing:** Shearing involves skewing or distorting an object along one axis while keeping the other axis unchanged. This transformation is often used for creating perspective effects.
5. **Reflection:** Reflection flips an object over a line or plane, creating its mirror image. In 2D, this is achieved by reversing the sign of one coordinate while leaving the other unchanged.

Geometric transformations are widely used in computer graphics, computer-aided design (CAD), image processing, and various mathematical applications. They play a crucial role in manipulating and animating objects, simulating physical phenomena, and solving geometric problems. Understanding these transformations is essential for working with graphical data and mathematical modeling in various fields.

Rotations

Rotations are fundamental geometric transformations that involve altering the orientation of objects or vectors within a given space. In these transformations, objects are rotated about a specified point, axis, or origin by a certain angle, resulting in a new arrangement while preserving the objects' size and shape. Rotations are essential in various fields, including mathematics, physics, computer graphics, and engineering, where they are used to manipulate and analyze spatial data, model physical phenomena, and create realistic visual representations. These transformations are mathematically represented using rotation matrices, and they play a crucial role in understanding symmetry, transforming coordinate systems, and solving complex spatial problems.

Rotations Formula

The provided equation represents a 2D rotation transformation. In this transformation, a 2D vector x is rotated by an angle θ, resulting in a new vector y. The rotation is performed using a 2x2 matrix:

$$y = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} x.$$

Here's what each component of this equation means:

- $\cos(\theta)$ and $\sin(\theta)$ are trigonometric functions representing the cosine and sine of the rotation angle θ.
- The matrix structure represents the transformation. The top row corresponds to the new x-axis, and the bottom row corresponds to the new y-axis after the rotation.
- x is the input 2D vector that you want to rotate.
- y is the resulting vector after applying the rotation.

This equation essentially describes how to perform a 2D rotation of a vector x by an angle θ using a matrix. The matrix elements are determined by trigonometric functions of the rotation angle, ensuring that the rotated vector y retains its length and orientation in the new coordinate system. This concept is fundamental in geometry, graphics, and various engineering applications.

Principles of Rotation:

1. **Rotation Center:** A rotation is performed around a fixed point called the "center of rotation" or "pivot point." All points in the object move in a circular path around this center.
2. **Angle of Rotation:** The angle through which an object is rotated is a crucial parameter. It is measured in degrees or radians and determines the extent of the rotation.
3. **Direction:** Rotations can be clockwise or counterclockwise, depending on the choice of orientation. Conventionally, counterclockwise rotations are considered positive.

Properties of Rotation:

1. **Preservation of Length:** Rotations preserve the length of line segments. If two points are a certain distance apart before a rotation, they will remain at the same distance after the rotation.
2. **Preservation of Angles:** Rotations also maintain the angles between lines. If two lines intersect at a specific angle before a rotation, they will intersect at the same angle after the rotation.
3. **Center of Rotation:** The center of rotation is an invariant point. It remains fixed, and all other points move in circular paths around it.

Types of Rotations:

1. **2D Rotations:** In a two-dimensional space (2D), rotations involve changing the angle of an object around a fixed point. These rotations are often represented using trigonometric functions like sine and cosine.
2. **3D Rotations:** In a three-dimensional space (3D), rotations are more complex, involving changes in orientation about multiple axes. Representations like Euler angles, rotation matrices, and quaternions are used to describe 3D rotations.

Challenges in Rotations:

1. **Angle Representation:** Choosing the right angle representation (degrees or radians) and dealing with angle wrap-around can be challenging.
2. **Singularities:** Certain rotation representations may encounter singularities or gimbal lock, limiting their practicality.

Rotation is a versatile and powerful geometric transformation with wide-ranging applications in mathematics, science, engineering, art, and technology. Understanding its principles and properties is fundamental for anyone working with spatial transformations or manipulating objects in a 2D or 3D space.

Reflection

Reflection is a fundamental geometric transformation that involves flipping or mirroring an object or a point across a specified axis or plane. It plays a crucial role in various fields, including geometry, physics, computer graphics, and engineering. This transformation essentially changes the orientation of an object or point while preserving its size and shape.

Reflections Formula

The provided equation represents a 2D reflection transformation matrix. In this transformation, a 2D vector x is reflected across a line (or mirror line) at an angle 2θ with respect to the x-axis, resulting in a new vector y. The reflection matrix is defined as:

$$y = \begin{bmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{bmatrix} x.$$

Here's what each component of this equation means:

- $\cos(2\theta)$ and $\sin(2\theta)$ are trigonometric functions representing the cosine and sine of twice the reflection angle 2θ.
- The matrix structure represents the transformation. It describes how the vector x is reflected across the mirror line.
- x is the input 2D vector that you want to reflect.

- y is the resulting vector after applying the reflection.

This equation essentially describes how to perform a 2D reflection of a vector x across a line at a specified angle 2θ using a matrix. The matrix elements are determined by trigonometric functions of the reflection angle, ensuring that the reflected vector y retains its orientation with respect to the mirror line. Reflection matrices are fundamental tools in geometry, graphics, and various engineering applications.

Key points about reflection:

1. **Reflection Across a Line:** In 2D, reflection typically occurs across a line, often referred to as the "mirror line" or "axis of reflection." Anything on one side of this line is mirrored to the other side. The line of reflection is perpendicular to the mirror plane.
2. **Mirror Plane:** In 3D, reflection occurs across a plane called the "mirror plane." Objects or points are reflected from one side of the plane to the other. The normal vector of the mirror plane determines the direction of reflection.
3. **Mathematical Representation:** Reflection can be represented mathematically using matrices. In 2D, a reflection matrix is defined based on the angle of reflection. In 3D, reflection matrices depend on the orientation of the mirror plane.
4. **Preservation of Distance:** Reflection preserves the distance between points. If a point is a certain distance from the mirror line or mirror plane, its reflected counterpart will be the same distance on the other side.
5. **Symmetry:** Reflection is a symmetry operation. Objects that possess reflective symmetry are identical on both sides of the mirror line or plane.
6. **Multiple Reflections:** Multiple reflections can occur when an object is reflected multiple times across different mirror lines or planes. These transformations are cumulative.
7. **Reflection in Mathematics:** In mathematics, reflection is an essential concept in linear algebra, where reflection matrices are used for various transformations. It's also crucial in geometry when studying symmetry and transformations.

Reflection is a powerful tool for understanding the behavior of light, sound, and various physical phenomena. It also plays a significant role in creating realistic graphics and simulations, making it a fundamental concept in several scientific and technical disciplines.

The next section that we will cover this week is **VMLS Section 7.2 - Selectors**.

VMLS Section 7.2 - Selectors

Overview

Selectors in Linear Algebra

Selectors in the context of matrix operations in linear algebra refer to the process of choosing specific elements or subsets of a matrix to perform operations on or extract information from. These selectors play a crucial role in various matrix-related tasks, such as matrix addition, multiplication, transposition, and decomposition.

1. **Element Selection:** Selectors allow targeting individual elements within a matrix, denoted as A[i][j], where i and j represent the row and column indices.
2. **Row and Column Selection:** Entire rows or columns of a matrix can be selected, enabling operations involving entire vectors within the matrix.
3. **Submatrix Selection:** Selectors facilitate the extraction of submatrices from a larger matrix, useful for focusing on specific regions of interest.
4. **Block Matrix Operations:** Targeting blocks or submatrices of a larger matrix is possible using selectors, which is valuable for tasks like matrix multiplication.

5. **Transpose Selection:** Selectors are used in transpose operations (A^T), rearranging rows and columns efficiently.
6. **Diagonal Selection:** Diagonal selectors are employed to extract or manipulate diagonal elements, essential in various linear algebra operations.
7. **Matrix Concatenation:** Selectors enable the combination or concatenation of matrices horizontally or vertically, particularly useful in block matrix operations.
8. **Element-wise Operations:** For element-wise operations, where corresponding elements of two matrices are combined using arithmetic operations, selectors are fundamental.

Selectors in linear algebra provide a flexible and precise way to access and manipulate elements within matrices. They are fundamental for performing a wide range of matrix operations and transformations, making them a crucial concept in linear algebra and its various applications.

Down Sampling

Down-sampling is a data reduction technique commonly used in signal processing, image processing, and data analysis. Its primary purpose is to decrease the amount of data by selecting a subset of data points while preserving essential information. In the context of digital signals or images, down-sampling involves reducing the sample rate or resolution, effectively reducing the size of the data.

Key points about down-sampling:

1. **Data Reduction:** Down-sampling is employed when working with large datasets to make data more manageable and computationally efficient.
2. **Anti-Aliasing:** In image processing, down-sampling is often accompanied by anti-aliasing filters to prevent aliasing artifacts, ensuring that the reduced data still represents the original information accurately.
3. **Applications:** Down-sampling is used in various fields, such as audio processing, where high-frequency components are removed to reduce file size without significant loss in audio quality.
4. **Sub-sampling:** Sub-sampling is a common down-sampling technique where every nth data point is selected, effectively reducing the data's resolution. For example, reducing the frame rate of a video.
5. **Down-sampling Factor:** The degree of down-sampling is determined by the down-sampling factor, which indicates how many data points are retained for every set of contiguous data points.
6. **Decimation:** Decimation is a more specific form of down-sampling where data points are removed, often used in digital signal processing.
7. **Loss of Information:** Down-sampling typically results in a loss of fine details or high-frequency components. The choice of down-sampling method and factor depends on the specific application and the acceptable level of information loss.
8. **Compression:** Down-sampling is a fundamental step in data compression techniques, helping reduce storage requirements while maintaining essential information.
9. **Preprocessing:** In machine learning and data analysis, down-sampling can be used to balance imbalanced datasets by reducing the size of the majority class.
10. **Trade-Off:** Down-sampling involves a trade-off between reducing data size and preserving critical information. The optimal down-sampling strategy depends on the specific task and data characteristics.

Down-sampling is a valuable technique for managing large datasets, reducing storage requirements, and simplifying data analysis. However, it requires careful consideration to balance the reduction in data size with the retention of essential information, making it a common preprocessing step in various fields.

The next section that we will cover this week is **VMLS Section 7.3 - Incidence Matrix**.

VMLS Section 7.3 - Incidence Matrix

Overview

An incidence matrix is a fundamental concept in graph theory and linear algebra used to represent relationships between elements of two sets, particularly nodes (vertices) and edges in a graph. Denoted as A, the incidence matrix has rows corresponding to nodes and columns corresponding to edges. Entries in the matrix indicate whether a node is incident to an edge. Key points about incidence matrices include:

1. **Binary Representation:** Incidence matrices are often binary, with entries of 0 or 1. A 1 in row i and column j indicates node i is incident to edge j, while 0 indicates no incidence.
2. **Directed and Undirected Graphs:** In directed graphs, matrices may contain 1 to indicate edge direction. For undirected graphs, entries are typically 0 and 1.
3. **Size:** The matrix size depends on the number of nodes and edges in the graph. For n nodes and m edges, the matrix size is $n \times m$.
4. **Sparse Matrices:** They are often sparse, particularly in large graphs where most nodes are not directly connected to all edges.
5. **Node Degrees:** Row sums of an incidence matrix reveal node degrees in the graph. In undirected graphs, the row sum represents the node's degree.
6. **Graph Connectivity:** Incidence matrices aid in studying graph connectivity and determining if a graph is connected.
7. **Linear Dependence:** In linear algebra, columns of an incidence matrix help analyze linear dependence relationships among edges.
8. **Spanning Trees:** They are essential for finding minimum spanning trees and solving network optimization problems.

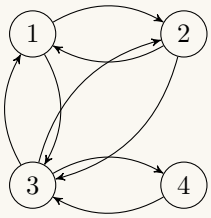
Incidence matrices are a powerful tool for efficiently representing and analyzing graph structures, making them a cornerstone of graph theory and related fields.

Incidence Matrix Example With Graph

The provided matrix and graph together represent an incidence matrix and its corresponding directed graph. In the context of graph theory, the incidence matrix illustrates the relationships between nodes and edges in a directed graph. Here's how to interpret this:

$$\begin{bmatrix} -1 & 1 & -1 & 0 \\ 1 & -1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

The matrix shown is an incidence matrix where each row represents a node (labeled 1, 2, 3, and 4), and each column represents an edge. The entries in the matrix indicate the incidence of each edge on each node. Specifically, a '-1' in the matrix at row 'i' and column 'j' signifies that node 'i' is the tail of edge 'j,' while a '1' indicates that node 'i' is the head of edge 'j.' Zeros represent no connection between the node and edge.



The directed graph depicted below the matrix shows the relationships between nodes (labeled 1, 2, 3, and 4) and directed edges (indicated by arrows). Each directed edge connects a source node (tail) to a target node (head), consistent with the entries in the incidence matrix. The graph visually represents the connections and flow of information or influence within the system.

This combination of an incidence matrix and directed graph provides a concise representation of the structure and relationships within a directed graph, making it a valuable tool in various fields such as network analysis, optimization, and transportation planning.

The last section that we will cover this week is **VMLS Section 7.4 - Convolution**.

VMLS Section 7.4 - Convolution

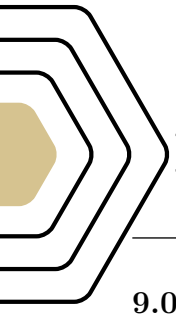
Overview

Convolution is a fundamental mathematical operation widely used in signal processing, image analysis, and deep learning. It involves the blending of two functions to produce a third function, which represents how one function modifies the other. In the context of discrete signals or images, convolution computes the output signal by sliding one function (often referred to as the kernel or filter) over the other while calculating the integral of their pointwise product. This operation has several key characteristics:

1. **Linearity:** Convolution is a linear operation, making it a fundamental building block in linear systems theory. This linearity property simplifies the analysis of complex systems composed of interconnected linear components.
2. **Translation-Invariance:** Convolution is translation-invariant, meaning that the same filter applied at different positions produces consistent results. This property is crucial in image processing and feature detection.
3. **Commutativity:** Convolution is commutative, meaning that the order in which two functions are convolved does not affect the result. This property is useful in various mathematical and computational contexts.
4. **Associativity:** Convolution is associative, allowing multiple convolutions to be combined into a single convolution operation. This property simplifies the implementation of deep neural networks.

Convolution finds applications in diverse fields, including image and audio processing for tasks like edge detection and noise reduction, as well as in deep learning for feature extraction and convolutional neural networks (CNNs) used in image recognition and natural language processing. Understanding convolution is essential for anyone working with digital signals, images, or neural networks.

Linear Functions



Linear Functions

9.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- VMLS Chapter 8.1 - Linear And Affine Functions
- VMLS Chapter 8.2 - Linear Function Models
- VMLS Chapter 8.3 - Systems Of Linear Equations
- Python Companion Chapter 8.1 - Linear And Affine Functions
- Python Companion Chapter 8.2 - Linear Function Models
- Python Companion Chapter 8.3 - Systems Of Linear Equations

9.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

9.0.3 Lectures

The lectures for this week and their links can be found below:

- 8.1 Linear Functions $\approx$ 22 min.
- 8.2 Regression Models $\approx$ 9 min.
- 8.3 Linear Equations $\approx$ 15 min.
- 8.3 Over And Underdetermined Systems $\approx$ 11 min.
- Random Exercise: Chapter 8 $\approx$ 8 min.

9.0.4 Assignments

The assignment for this week is:

- Assignment 8 - Linear Equations

9.0.5 Quiz

The quiz for this week is:

- Quiz 8 - Linear Equations

9.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 8 - Linear Equations**. The first section that we will cover this week is **VMLS Section 8.1 - Linear And Affine Functions**.

VMLS Section 8.1 - Linear And Affine Functions

Linear Functions:

Linear functions are perhaps the most straightforward mathematical relationships between variables. They adhere to two key principles:

1. **Additivity:** This means that when you add two inputs to the function, the corresponding outputs will add as well. In other words, $f(x + y) = f(x) + f(y)$. For example, if you're measuring the cost of producing a certain number of items, a linear cost function implies that the cost of producing 10 items is the sum of the costs of producing 5 items and then producing another 5 items.

2. **Homogeneity:** Homogeneity implies that when you scale the input by a factor (multiply it by a constant), the output scales accordingly. In mathematical terms, $f(ax) = af(x)$. In our cost example, if you double the number of items to produce, the cost should double as well if it follows a linear relationship.

Affine Functions:

Affine functions build upon the principles of linearity but introduce an extra element: a constant offset. This constant, often denoted as "b," adds versatility to the function:

1. **Additivity and Homogeneity:** Like linear functions, affine functions maintain additivity and homogeneity. However, the presence of the constant term "b" means that they may not pass through the origin (0,0) on a graph, unlike linear functions.

2. **Translation and Shifting:** The constant term "b" allows affine functions to model more complex relationships where there is an initial value or a shift. For instance, in finance, an affine function could represent a time series of stock prices, accounting for both the underlying trend and an initial value (offset) due to factors like an opening price.

Relationship:

The relationship between linear and affine functions is straightforward: every linear function is also an affine function, but not vice versa. This is because an affine function can have a constant term "b" that makes it non-linear when "b" is not zero.

In essence, linear and affine functions are essential tools for modeling and understanding relationships between variables in various scientific and practical contexts. They provide a mathematical foundation for describing how different quantities interact, evolve, and impact one another, making them indispensable in the toolkit of mathematicians, scientists, engineers, and economists.

The second section that we will cover this week is **VMLS Section 8.2 - Linear Function Models**.

VMLS Section 8.2 - Linear Function Models

Overview

Linear function models are a fundamental concept in the field of linear equations and mathematics. These models represent relationships between variables in a straightforward and easily interpretable way. A linear function is defined by two key principles:

1. **Linearity:** Linearity means that the relationship between variables is proportional and additive. In a linear function, if you double one variable, the output also doubles, and combining multiple variables is as simple as adding or subtracting their contributions. Mathematically, this property is expressed as $f(ax + by) = af(x) + bf(y)$, where a and b are constants.

2. **Constant Rate of Change:** Linear functions exhibit a constant rate of change. This means that as one variable increases or decreases by a fixed amount, the output changes by a consistent amount. The rate of change is represented by the slope of the line in a linear function.

Linear function models are widely used in various fields for their simplicity and applicability. Some key points to consider:

- **Graphical Representation:** Linear functions are represented by straight lines on a graph, where the slope of the line indicates the rate of change, and the y-intercept represents the initial value when the input is zero.
- **Real-Life Applications:** Linear function models are extensively applied in fields such as physics, economics, engineering, and data analysis. They are used to describe relationships like distance vs. time, cost vs. quantity, and temperature vs. time.
- **Regression Analysis:** Linear regression is a statistical technique that employs linear function models to analyze and predict relationships between variables. It is widely used in data science for predictive modeling.
- **Slope-Intercept Form:** Linear functions are often expressed in the slope-intercept form, $y = mx + b$, where m is the slope (rate of change) and b is the y-intercept (initial value).
- **Limitations:** While linear function models are powerful, they have limitations. They may not accurately represent complex, nonlinear relationships. In such cases, more sophisticated models, like quadratic or exponential functions, are needed.

Linear function models are a foundational concept in linear equations, providing a simple yet effective way to describe relationships between variables. Their versatility makes them a valuable tool for modeling, analysis, and prediction in various scientific, engineering, and real-world applications.

Taylor Approximations

Taylor Approximations are a powerful tool used in linear function models to approximate complex nonlinear functions with simpler linear representations. These approximations are based on Taylor series expansions and provide a means to locally linearize a function around a specific point.

Key points to understand about Taylor Approximations in linear function models:

- **Local Linearity:** Taylor Approximations are used when dealing with functions that exhibit nonlinear behavior. By selecting a reference point (usually denoted as 'a') within the function's domain, a Taylor series expansion is performed to approximate the function as a linear equation around that point. This results in a linear model that closely approximates the function's behavior within a small neighborhood of 'a.'
- **Taylor Series Expansion:** The Taylor series expansion of a function f(x) around a point 'a' involves finding the function's derivatives at 'a.' The general form of a first-order Taylor Approximation (linearization) is given by: $f(x) \approx f(a) + f'(a)(x - a)$, where f'(a) represents the first derivative of the function at 'a.' Higher-order approximations consider additional derivatives for increased accuracy.
- **Applications:** Taylor Approximations are extensively applied in various fields, including physics, engineering, and numerical analysis. They are particularly useful for approximating nonlinear physical phenomena and simplifying complex mathematical models. For instance, in physics, they are used to linearize equations of motion, making them easier to solve.
- **Error Analysis:** The accuracy of Taylor Approximations depends on the proximity of the selected point 'a' to the point of interest. The smaller the interval around 'a,' the more accurate the linear approximation. Error analysis is essential to assess the quality of the approximation and determine its suitability for a given application.
- **Higher-Order Approximations:** While first-order Taylor Approximations provide local linearizations, higher-order approximations incorporate additional derivatives and result in polynomial models of higher degrees. These can offer even more accurate representations of nonlinear functions but are more computationally intensive.
- **Limitations:** Taylor Approximations are valid only in a local neighborhood of the chosen point 'a.' They may not accurately represent the entire function if it exhibits significant nonlinear behavior over a wide range.

Taylor Approximations are valuable tools in linear function modeling when dealing with nonlinear functions. They allow for the creation of local linear models that simplify complex relationships and facilitate analysis and prediction. The choice of 'a' and the order of the approximation depend on the specific problem and the desired level of accuracy.

Regression Model

Regression models are fundamental tools within the realm of linear function models, aimed at understanding and predicting relationships between variables. These models assume that the relationships between the response variable and predictor variables can be represented linearly. Here are key insights into regression models in this context:

- **Linear Assumption:** In linear function models, regression models assume that the relationship between the response variable (usually denoted as 'Y') and predictor variables ($'X_1$,$'X_2$,$'$ etc.) can be expressed as a linear combination of these variables. The relationship is represented as $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \epsilon$, where $\beta_0, \beta_1, \beta_2$, etc., are coefficients, and ϵ represents the error term.
- **Simple Linear Regression:** Simple linear regression models the relationship between a single predictor variable (X) and the response variable (Y). It aims to find the best-fitting linear equation $Y = \beta_0 + \beta_1 X$ that minimizes the sum of squared differences between observed and predicted values.
- **Multiple Linear Regression:** Multiple linear regression extends the simple linear model to incorporate multiple predictor variables (X_1, X_2, etc.). It seeks to find coefficients $\beta_0, \beta_1, \beta_2$, etc., that optimize the linear equation $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \epsilon$.
- **Model Interpretation:** In linear function models, the coefficients (β values) provide insights into the strength and direction of the relationships between predictor variables and the response variable. Positive coefficients indicate a positive relationship, while negative coefficients signify a negative relationship.
- **Assumptions:** Regression models rely on assumptions such as linearity, independence of errors, constant variance of errors (homoscedasticity), and normality of error terms. Violations of the1se assumptions may impact model accuracy.
- **Model Evaluation:** Regression models are evaluated based on metrics like the coefficient of determination (R-squared), mean squared error (MSE), and others. These metrics gauge the model's fit and predictive performance.
- **Applications:** Regression models are widely used in various fields, including economics, social sciences, finance, and machine learning. They help researchers and analysts uncover relationships, make predictions, and inform decision-making.
- **Extensions:** Linear function models can be extended to include interactions, polynomial terms, and other transformations to capture complex relationships. These extensions allow for modeling nonlinear patterns when necessary.

Regression models are essential components of linear function models, providing a framework to understand and predict relationships between variables. They are versatile tools with applications across diverse domains, and their interpretability makes them valuable for making informed decisions based on data analysis.

The last section that we will cover this week is **VMLS Section 8.3 - Systems Of Linear Equations**.

VMLS Section 8.3 - Systems Of Linear Equations

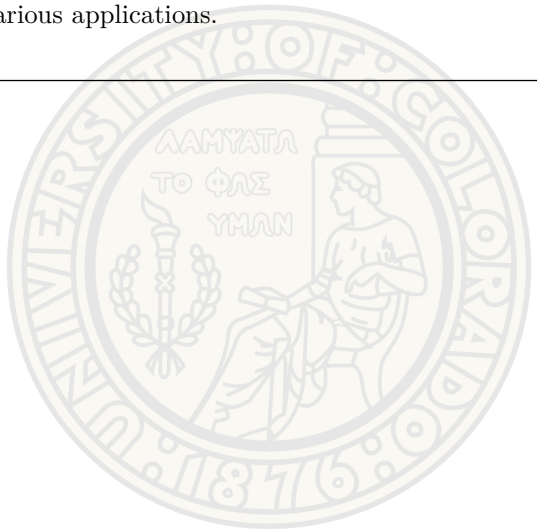
Overview

A system of linear equations is a collection of two or more linear equations involving the same set of variables. These systems are fundamental in various areas of mathematics, science, and engineering. Here are key insights into systems of linear equations:

- **Linear Equations:** Each equation in the system is a linear equation, meaning it can be expressed in the form of $ax + by + cz + \cdots = d$, where 'a,' 'b,' 'c,' etc., are constants, and $'x$,$'y$,$'z$,$'$ etc., are variables. The goal is to find values for the variables that satisfy all equations simultaneously.
- **Variables and Coefficients:** Variables represent unknown quantities, while coefficients represent known constants. The system may involve more variables than equations, resulting in an overdetermined or underdetermined system.
- **Matrix Representation:** Systems of linear equations can be represented using matrices and vectors. The coefficient matrix $'A'$ contains the coefficients of the variables, the vector $'X'$ represents the variables, and the constant vector $'B'$ holds the constants from the right-hand side of the equations. The system is written as $'AX = B.'$

- **Solution Space:** The set of all possible solutions to the system forms a solution space. The system can have one unique solution, infinitely many solutions, or no solution at all, depending on the relationships between equations.
- **Methods of Solution:** Various methods are employed to solve systems of linear equations, including Gaussian elimination, matrix inversion, Cramer's rule (for square systems), and numerical techniques like the Gauss-Jordan method and iterative methods.
- **Applications:** Systems of linear equations have widespread applications in diverse fields. In physics, they describe equilibrium conditions and electrical circuits. In economics, they model supply and demand relationships. In engineering, they solve problems related to structural analysis and control systems.
- **Linear Independence:** The concept of linear independence is crucial in determining whether a system has a unique solution. Linearly independent equations provide the best chance of a unique solution.
- **Homogeneous Systems:** When all equations in a system have a right-hand side of zero, it's referred to as a homogeneous system. Such systems always have a trivial solution (all variables equal to zero) and may have non-trivial solutions.
- **Nonlinear Systems:** While linear equations deal with linear relationships, nonlinear systems involve equations with nonlinear relationships between variables. Solving nonlinear systems is generally more complex and may require numerical methods.

Systems of linear equations are essential tools for modeling real-world problems and finding solutions to unknown variables. Their mathematical representation, methods of solution, and applications make them a fundamental concept in mathematics and its various applications.



SIR/Modeling/Dynamical Systems

SIR/Modeling/Dynamical Systems

10.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- VMLS Chapter 9.1 - Linear Dynamical Systems
- VMLS Chapter 9.2 - Population Dynamics
- VMLS Chapter 9.3 - Epidemic Dynamics
- VMLS Chapter 9.4 - Motion Of A Mass
- VMLS Chapter 9.5 - Supply Chain Dynamics
- Python Companion Chapter 9.1 - Linear Dynamical Systems
- Python Companion Chapter 9.2 - Population Dynamics
- Python Companion Chapter 9.3 - Epidemic Dynamics
- Python Companion Chapter 9.4 - Motion Of A Mass
- Python Companion Chapter 9.5 - Supply Chain Dynamics

10.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

10.0.3 Lectures

The lectures for this week and their links can be found below:

- Introduction To Modelling With Systems Of Non-linear Differential Equations ≈ 13 min.
- Calculus in Context
- VMSL - Dynamical Systems Overview - Chapter 9 ≈ 39 min.

10.0.4 Assignments

The assignment for this week is:

- SIR Model

10.0.5 Quiz

The quiz for this week is:

- Chapter 9

10.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 9 - Linear Dynamical Systems**. The first section that we will cover this week is **VMLS Section 9.1 - Linear Dynamical Systems**.

VMLS Section 9.1 - Linear Dynamical Systems

Overview

Linear dynamical systems are mathematical models used to describe the time-evolution of variables in various fields such as physics, engineering, economics, and biology. They are particularly useful for studying systems where linear relationships between variables exist. Key points about linear dynamical systems include:

- **Linearity:** LDS assume that the relationships governing the system's behavior are linear. This means that the evolution of each variable is a linear combination of its current value and other variables' values, weighted by constant coefficients.
- **State-Space Representation:** LDS are often represented in a state-space form. The state vector contains all relevant variables, and the dynamics are described by a set of linear differential or discrete equations. This representation simplifies the analysis and control of complex systems.
- **Time-Invariance:** Linear dynamical systems are time-invariant, meaning that system behavior does not change over time. The coefficients in the equations governing the system remain constant.
- **Stability and Control:** Stability analysis plays a crucial role in LDS. Engineers and scientists use stability criteria to determine if a system's behavior remains bounded or diverges over time. Control theory leverages LDS to design controllers that regulate system behavior.
- **Eigenvalues and Eigenvectors:** Eigenvalues and eigenvectors of the system matrix are essential for characterizing the behavior of LDS. Eigenvalues indicate stability, while eigenvectors determine the system's response to different initial conditions.
- **Applications:** Linear dynamical systems find applications in a wide range of domains, including mechanical systems, electrical circuits, chemical reactions, population dynamics, and financial modeling. They are instrumental in predicting system responses and designing control strategies.
- **Matrix Exponentiation:** Solving linear dynamical systems often involves matrix exponentiation. The system matrix is exponentiated to obtain the time evolution operator, which allows us to predict future states of the system.

Linear dynamical systems provide a foundational framework for understanding and predicting the behavior of dynamic systems. Their simplicity, analytical tractability, and wide applicability make them indispensable tools for engineers, scientists, and researchers in various fields.

Markov Model

A Markov model, named after Russian mathematician Andrey Markov, is a mathematical framework used to describe stochastic processes where future events or states depend solely on the current state and are independent of previous states. Markov models are widely applied in various fields, including physics, engineering, economics, and biology. Key points about Markov models include:

- **Markov Property:** The central assumption in a Markov model is the Markov property, which states that the probability of transitioning to a future state depends only on the current state and not on the sequence of past states.
- **State Space:** Markov models are defined by a finite or countable set of states, representing possible conditions or configurations of the system under study. Transitions between states are governed by transition probabilities.
- **Transition Probabilities:** Transition probabilities specify the likelihood of moving from one state to another in a single time step. These probabilities are often represented in a transition matrix, where each entry corresponds to the probability of transitioning between specific states.
- **Homogeneous and Non-Homogeneous Models:** Markov models can be homogeneous, where transition probabilities remain constant over time, or non-homogeneous, allowing transition probabilities to change over time.
- **Applications:** Markov models find applications in various fields, including weather forecasting, stock market analysis, natural language processing, epidemiology (e.g., disease spread modeling), and machine learning (e.g., Hidden Markov Models in speech recognition).

- **Steady-State Analysis:** In many Markov models, the long-term behavior is of interest. Steady-state analysis involves finding the equilibrium distribution of states, which represents the probabilities of being in each state over time.
- **Limitations:** Markov models assume that future events are conditionally independent of the past given the current state. While this simplifying assumption works well in many scenarios, it may not capture complex dependencies in some real-world systems.

Markov models provide a powerful framework for modeling and analyzing systems that exhibit random and sequential behavior. Their simplicity, along with their ability to capture dynamic processes, makes them valuable tools for understanding and predicting various phenomena.

The next section that we will cover this week is **VMLS Section 9.2 - Population Dynamics**.

VMLS Section 9.2 - Population Dynamics

Overview

Population dynamics refers to the study of how the size and composition of populations change over time. In the context of linear dynamical systems, population dynamics are often modeled using differential equations and matrices to describe the interactions and evolution of populations. Key points about population dynamics in linear systems include:

- **State Variables:** In population dynamics, state variables represent characteristics of a population, such as the number of individuals in different age groups or species. These state variables are typically organized into vectors or matrices.
- **Linear Differential Equations:** Linear dynamical systems describe how state variables change over time using linear differential equations. These equations capture birth, death, migration, and other population processes as linear transformations.
- **Matrix Representation:** Matrices play a crucial role in linear dynamical systems, with each matrix representing the rates of change or interactions between different subpopulations. Matrix multiplication allows for efficient modeling of complex population structures.
- **Eigenvalues and Eigenvectors:** Eigenvalues and eigenvectors of population matrices provide insights into the long-term behavior of populations. Stable eigenvalues indicate population equilibrium, while unstable eigenvalues suggest population instability.
- **Steady States:** Steady states represent population configurations where population sizes remain constant over time. Finding steady states involves solving linear systems of equations and often requires considering external factors like resource availability.
- **Limitations:** Linear dynamical models assume linear relationships between population variables and may not capture all nuances of real-world population dynamics, such as nonlinear interactions or demographic stochasticity.
- **Applications:** Linear dynamical models are applied in ecology, epidemiology, economics, and other fields to understand and predict population trends, disease spread, economic growth, and more.

Population dynamics within linear dynamical systems provide a valuable framework for studying and managing populations in various contexts. By mathematically modeling population interactions and changes, researchers gain insights into the stability and sustainability of ecosystems, economies, and communities.

The next section that we will cover this week is **VMLS Section 9.3 - Epidemic Dynamics**.

VMLS Section 9.3 - Epidemic Dynamics

Overview

Epidemic dynamics is a crucial field of study that focuses on understanding the spread and control of infectious diseases within populations. In the context of linear dynamical systems, epidemic dynamics involves modeling the transmission of diseases using mathematical equations and matrices. Key points about epidemic dynamics in linear systems include:

- **State Variables:** In epidemic models, state variables represent different populations or groups involved in the disease transmission process. These may include susceptible individuals, infected individuals, and recovered individuals, often organized into vectors or matrices.
- **Compartmental Models:** Epidemic models are often compartmental, where individuals move between compartments based on their disease status. Common compartments include Susceptible (S), Infected (I), and Recovered (R), giving rise to the SIR model.
- **Linear Differential Equations:** Linear dynamical systems are used to describe how the number of individuals in each compartment changes over time. These systems capture disease transmission rates, recovery rates, and other factors as linear transformations.
- **Matrix Representation:** Matrices represent the interactions and transitions between compartments. The basic reproduction number (R_0) is determined by the eigenvalues of these matrices, providing insights into epidemic outcomes.
- **Epidemic Threshold:** Linear models often identify a critical threshold value of R_0 below which an epidemic cannot sustain itself, leading to disease extinction. Above this threshold, epidemics can occur and may reach an endemic equilibrium.
- **Control Strategies:** Linear models help evaluate the impact of various control strategies, such as vaccination, quarantine, and social distancing. These strategies can be represented as changes in model parameters.
- **Limitations:** Linear dynamical models assume constant transmission rates and homogeneous mixing, which may not capture all real-world complexities. More advanced models, including nonlinear and stochastic approaches, are often used for greater accuracy.
- **Applications:** Epidemic dynamics models have been instrumental in understanding and managing infectious diseases, including the study of COVID-19, HIV, influenza, and many others. They inform public health policies and interventions.

Epidemic dynamics within linear dynamical systems provide a valuable framework for predicting disease outbreaks, assessing the effectiveness of interventions, and guiding public health decisions. These models offer insights into the complex interplay between disease spread, population behavior, and control measures.

The next section that we will cover this week is **VMLS Section 9.4 - Motion Of A Mass**.

VMLS Section 9.4 - Motion Of A Mass

Overview

The motion of a mass within linear dynamical systems is a fundamental concept in physics and engineering. It involves describing the behavior of a physical mass (or object) under the influence of external forces using linear equations and matrices. Key points about the motion of a mass in linear dynamical systems include:

- **State Variables:** In this context, the state variables represent the position, velocity, and acceleration of the mass. These variables are often organized into vectors or matrices to describe the system comprehensively.
- **Newton's Second Law:** The motion of a mass is governed by Newton's second law of motion, which states that the force acting on an object is equal to the mass of the object times its acceleration ($F = ma$). Linear systems translate this law into differential equations and matrix equations.
- **Linear Differential Equations:** Linear dynamical systems model the motion of a mass using linear differential equations that relate position, velocity, and acceleration. These equations may include external forces, such as springs or dampers, which are represented as linear transformations.

- **Matrix Representation:** Matrices are used to represent the linear relationships between state variables. For example, in a mechanical system, matrices describe how forces, displacements, and velocities are related.
- **Eigenvalues and Stability:** The eigenvalues of the system matrices provide insights into the stability and behavior of the mass's motion. Stable systems lead to bounded and predictable motion, while unstable systems may exhibit unbounded behavior.
- **Control and Feedback:** Linear dynamical systems allow for the application of control strategies to modify the mass's motion. Feedback control, where system states are measured and adjusted in real-time, plays a significant role in engineering applications.
- **Applications:** Understanding the motion of a mass is essential in various fields, including mechanical engineering, robotics, aerospace engineering, and physics. It is used in designing structures, optimizing control systems, and analyzing the behavior of physical systems.
- **Limitations:** Linear dynamical models assume linearity and may not capture nonlinear effects or extreme conditions. In such cases, more advanced nonlinear dynamical systems are employed.

The study of the motion of a mass in linear dynamical systems provides a foundation for analyzing and controlling the behavior of mechanical and physical systems. It serves as a fundamental concept in engineering and physics, enabling the design and optimization of a wide range of systems and structures.

The last section that we will cover this week is **VMLS Section 9.5 - Supply Chain Dynamics**.

VMLS Section 9.5 - Supply Chain Dynamics

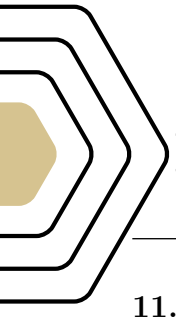
Overview

In the context of linear dynamical systems, supply chain dynamics refer to the analysis of how goods, materials, and information flow through a supply chain network over time. These systems are typically modeled using linear equations and matrices to understand and optimize supply chain behavior. Key aspects of supply chain dynamics in linear dynamical systems include:

- **Network Representation:** Supply chains are represented as networks with nodes representing entities like suppliers, manufacturers, and retailers. Links denote the flow of products and information.
- **Demand and Supply:** Linear models incorporate supply and demand equations to describe production, distribution, and consumption within the supply chain.
- **Lead Times:** Lead times, representing product transit times, affect inventory and order quantities.
- **Inventory Management:** Linear models optimize inventory levels and reorder points for efficient operations.
- **Production Planning:** Planning production quantities considers demand forecasts and capacity constraints.
- **Transportation and Logistics:** Optimization includes route planning, carrier selection, and logistics to minimize costs.
- **Uncertainty and Variability:** Linear models incorporate stochastic elements for probabilistic analysis.
- **Performance Metrics:** Metrics like order fill rates and inventory turnover assess supply chain effectiveness.
- **Sustainability:** Models evaluate environmental impact and sustainability in supply chain decisions.
- **Real-Time Optimization:** Linear systems aid real-time decisions for adapting to changing market conditions.
- **Integration:** Data integration from various sources enhances model accuracy and responsiveness.

Supply chain dynamics in linear dynamical systems offer a structured approach to managing modern supply chains, optimizing efficiency, reducing costs, and adapting to market changes.

Matrix Multiplication



Matrix Multiplication

11.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- **Sections 10.1, 10.2, 10.3, 10.4** from VMLS.
- VMLS Chapter 10.1 - Matrix-Matrix Multiplication
- VMLS Chapter 10.2 - Composition Of Linear Functions
- VMLS Chapter 10.3 - Matrix Power
- VMLS Chapter 10.4 - QR Factorization
- Python Companion Chapter 10.1 - Matrix-Matrix Multiplication
- Python Companion Chapter 10.2 - Composition Of Linear Functions
- Python Companion Chapter 10.3 - Matrix Power
- Python Companion Chapter 10.4 - QR Factorization

11.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

11.0.3 Lectures

The lectures for this week and their links can be found below:

- 10.1 Matrix Multiply ≈ 24 min.
- Transpose Proof ≈ 14 min.
- 10.1 Matrix Multiplication Identities ≈ 19 min.
- 10.2 Linear Function Composition ≈ 14 min.
- 10.3 Matrix Power ≈ 12 min.
- 10.4 QR Factorization ≈ 22 min.
- Random Exercise: Chapter 10 ≈ 16 min.

11.0.4 Assignments

The assignment for this week is:

- Assignment 10 - Matrix Multiplication

11.0.5 Quiz

The quiz for this week is:

- Quiz 10 - Matrix Multiplication

11.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 10 - Matrix Multiplication**. The first section that we will cover this week is **VMLS Section 10.1 - Matrix-Matrix Multiplication**.

VMLS Section 10.1 - Matrix-Matrix Multiplication

Overview

Matrix multiplication is a central operation in linear algebra that finds application in various disciplines such as computer science, physics, engineering, and economics. It involves the multiplication of two matrices to yield a third matrix, known as the product matrix.

Concept

Matrix-matrix multiplication, also referred to as the matrix product, is an operation that is distinct from element-wise multiplication. It involves a row-by-column multiplication of two matrices.

Conditions

The number of columns in the first matrix must equal the number of rows in the second matrix for multiplication to be defined. If we have a first matrix A of size $m \times n$ and a second matrix B of size $n \times p$, then the resulting product matrix C will have a size of $m \times p$.

Mathematical Formula

The element at the i-th row and j-th column of the product matrix C is calculated as follows:

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

where c_{ij} is computed by summing the products of the corresponding elements of the i-th row of matrix A and the j-th column of matrix B.

Steps

1. Select the i-th row from matrix A.
2. Select the j-th column from matrix B.
3. Multiply the corresponding elements and sum up the products.
4. Place the result in the i-th row and j-th column of the product matrix C.
5. Repeat the process for all rows and columns of the resulting matrix.

Properties

- Matrix multiplication is associative, i.e., $(AB)C = A(BC)$, but not commutative since $AB \neq BA$ generally.
- An identity matrix I serves as a multiplicative identity for matrix multiplication: $AI = IA = A$.
- The distributive property is applicable: $A(B + C) = AB + AC$ and $(B + C)A = BA + CA$.

Context

In computer science, matrix multiplication is utilized in algorithms, graphics, optimization problems, among other areas. Proficiency in executing matrix multiplication efficiently is critical for enhancing the performance of many computational tasks.

The next section that we will be covering this week is **VMLS Section 10.2 - Composition Of Linear Functions**.

VMLS Section 10.2 - Composition Of Linear Functions

Overview

Matrix multiplication is intimately linked with the composition of linear functions. Each matrix represents a linear transformation, and multiplying two matrices corresponds to composing two such transformations.

Composition of Linear Functions

A linear transformation can be seen as a function that maps vectors from one vector space to another while preserving vector addition and scalar multiplication. If we represent linear transformations $T : U \to V$ and $S : V \to W$ by matrices A and B, the composition of S and T, denoted as $S \circ T$, is the function that applies T to a vector, followed by S.

Mathematical Representation

The matrix representation of the composition $S \circ T$ is obtained by the matrix product BA, where A and B represent T and S, respectively. For a vector $\boldsymbol{x}$ in U, the composition is defined as:

$$(S \circ T)(\boldsymbol{x}) = S(T(\boldsymbol{x})) = B(A\boldsymbol{x})$$

Context

The concept of composing linear functions is essential in many mathematical and computational applications. In computer graphics, transformations applied to objects are represented by matrices. The cumulative effect of multiple transformations is achieved by the product of the corresponding matrices, illustrating the practical significance of matrix multiplication in the composition of linear functions.

The next section that we will be covering this week is **VMLS Section 10.3 - Matrix Power**.

VMLS Section 10.3 - Matrix Power

Overview

Matrix-matrix products provide a foundation for understanding the composition of linear functions. Given matrices A of size $m \times p$ and B of size $p \times n$, we can associate them with linear functions $f : \mathbb{R}^p \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^p$, respectively. The composition of these functions is $h(x) = f(g(x)) = A(Bx) = (AB)x$, where $h : \mathbb{R}^n \to \mathbb{R}^m$. This composition is a linear function, which can be written as $h(x) = Cx$ where $C = AB$.

Matrix multiplication, when viewed in this way, clarifies why $AB \neq BA$ in general, as the composition order of linear functions matters. The section uses 2×2 matrices as an example to illustrate this point. It further discusses the concept of a second difference matrix, which is a product of two consecutive difference matrices, and how it represents the second difference function in terms of linear functions.

The composition of affine functions is also an affine function. If $f(x) = Ax + b$ and $g(x) = Cx + d$ are affine functions, then their composition h is $h(x) = A(Cx + d) + b = (AC)x + (Ad + b)$.

Moreover, the chain rule of differentiation is explored, showing that the derivative matrix of a composed function h can be expressed as the product of the derivative matrices of the individual functions, which can be interpreted in terms of matrix multiplication.

This also touches upon linear dynamical systems with state feedback, explaining that the state x_t is affected by the input u_t, and with state feedback, the input is a linear function of the state. This relationship can be captured using a state-feedback gain matrix K, and the closed-loop dynamics matrix is $A + BK$.

Matrix powers are explained with their properties, including the effect of multiplying a matrix by itself multiple times and how this relates to paths in a directed graph represented by the matrix. The section uses the adjacency matrix of a directed graph to exemplify how matrix powers can enumerate paths of a certain length within the graph.

Lastly, the section discusses the application of matrix powers in linear dynamical systems, describing how the power of a matrix A can propagate the system state forward in time.

Key Points

- Matrix multiplication corresponds to the composition of linear functions.
- The order of multiplication reflects the order in which functions are applied.
- Affine functions compose to form another affine function, with the matrix of the composition being the product of the individual matrices.
- The chain rule in differentiation can be expressed as a matrix-matrix product.
- Matrix powers are useful in understanding the dynamics of systems and enumerating paths in graphs.

The last section that we will cover this week is **VMLS Section 10.4 - QR Factorization**.

VMLS Section 10.4 - QR Factorization

Overview

System State

The section discusses the concept of matrix powers in the context of time-invariant linear dynamical systems with inputs. The evolution of the system's state over time is described by the equation:

$$x_{t+\tau} = A^\tau x_t + \sum_{j=0}^{\tau-1} A^j B u_{t+\tau-j-1}$$

where x_t is the state at time t, A is the system matrix, B is the input matrix, and u_t is the input at time t. This demonstrates how the state at a future time point is influenced by the powers of A and the inputs up to that time.

Properties of Matrices with Orthonormal Columns

- The condition for n-vectors $a_1, \ldots, a_k$ to be orthonormal is given by the matrix equation $A^T A = I$, where A is the matrix with columns $a_1, \ldots, a_k$.
- For an $m \times n$ matrix A with orthonormal columns and n-vectors x and y, we have the following properties:
  - Norm preservation: $\|Ax\| = \|x\|$.
  - Inner product preservation: $(Ax)^T(Ay) = x^T y$.
  - Angle preservation: The angle between Ax and Ay is the same as the angle between x and y.

QR Factorization

QR factorization is a method to decompose a matrix A into a product of two matrices Q and R, where Q has orthonormal columns and R is upper triangular. If A is an $n \times k$ matrix with linearly independent columns, the QR factorization is expressed as:

$$A = QR$$

This factorization is derived from the Gram-Schmidt process, which is used to obtain the orthonormal vectors that form the columns of Q. The matrix R is then constructed to reflect the coefficients used to express the

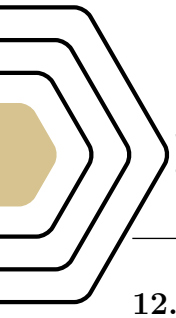
> original columns of A as linear combinations of the columns of Q.

The section also mentions alternative algorithms for QR factorization that are more stable in the presence of round-off errors and are efficient for sparse matrices.

Key Takeaways

- The QR factorization is useful for expressing a matrix as the product of an orthogonal (or orthonormal) matrix and an upper triangular matrix.
- The properties of norm, inner product, and angle preservation by orthonormal matrices have significant implications in various applications.
- Understanding the QR factorization and matrix powers is essential for analyzing linear dynamical systems and their behavior over time.

Matrix Inverses



Matrix Inverses

12.0.1 Assigned Reading

The reading assignments for this week are from, Introduction To Applied Linear Algebra, Vectors, Matrices, And Least Squares and Introduction to Applied Linear Algebra, Vectors, Matrices, And Least Squares - Python Companion:

- VMLS Chapter 11.1 - Left And Right Inverses
- VMLS Chapter 11.2 - Inverse
- VMLS Chapter 11.3 - Solving Linear Equations
- VMLS Chapter 11.4 - Examples
- VMLS Chapter 11.5 - Pseudo-Inverso
- Python Companion Chapter 11.1 - Left And Right Inverses
- Python Companion Chapter 11.2 - Inverse
- Python Companion Chapter 11.3 - Solving Linear Equations
- Python Companion Chapter 11.4 - Pseudo Inverse

12.0.2 Piazza

Must post / respond to at least **four** Piazza posts this week.

12.0.3 Lectures

The lectures for this week and their links can be found below:

- 11.1 Left & Right Inverses ≈ 18 min.
- 11.2 Inverse ≈ 20 min.
- 11.2 Inverse: Inverse Of Products ≈ 2 min.
- 11.3 Solving Linear Equations ≈ 28 min.
- 11.4 Pseudo-Inverse ≈ 15 min.
- 11.5 Finding Pseudo-Inverse With QR Factorization ≈ 14 min.
- Eigenvalues And Eigenvectors ≈ 21 min.
- Eigenvalues In Dynamical Systems ≈ 27 min.

12.0.4 Assignments

The assignment for this week is:

- Assignment 11 - Matrix Inverses

12.0.5 Quiz

The quiz for this week is:

- Quiz 11 - Matrix Inverses

12.0.6 Chapter Summary

The chapter that we will review this week is **VMLS Chapter 11 - Matrix Inverses**. The first section that we will be covering this week is **VMLS Section 11.1 - Left And Right Inverses**.

Section 11.1 - Left And Right Inverses

Overview

The concept of left and right inverses in matrices extends the idea of a multiplicative inverse from numbers to matrices. However, due to the non-commutative nature of matrix multiplication, the distinction between left and right inverses becomes crucial.

Left Inverse

- **Definition:** A matrix X is a left inverse of matrix A if it satisfies $XA = I$, where I is the identity matrix.
- **Existence and Uniqueness:** A left inverse exists if and only if the columns of A are linearly independent. It may not be unique unless A is square.
- **Mathematical Representation:**
$$XA = I$$
where X is an $n \times m$ matrix if A is an $m \times n$ matrix.

Right Inverse

- **Definition:** A matrix Y is a right inverse of matrix A if it satisfies $AY = I$.
- **Existence and Uniqueness:** A right inverse exists if and only if the rows of A are linearly independent. It is not unique unless A is square.
- **Mathematical Representation:**
$$AY = I$$
where Y is an $n \times m$ matrix if A is an $m \times n$ matrix.

Invertibility

- A matrix with both a left and a right inverse is invertible or nonsingular, and the left and right inverses are equal.
- **Mathematical Representation:**
$$AX = YA = I$$
Here, X and Y are equal and represent the inverse of A, denoted as A^{-1}.
- **Conditions for Invertibility:** A matrix is invertible if it is square and its columns (or rows) are linearly independent.

Applications

1. Solving linear systems: If A is left-invertible, $Ax = b$ has a unique solution $x = Xb$. If A is right-invertible, any solution to $Ax = b$ can be found using a right inverse.
2. Simplifying matrix equations like $AX = B$ or $YA = B$.

Properties of Inverses

- **Transpose Inverses:** If A has a right (or left) inverse, then A^T has a left (or right) inverse, given by $(A^{-1})^T$.
- **Product Inverses:** For invertible matrices A and B, $(AB)^{-1} = B^{-1}A^{-1}$.

Implications

The existence of left and right inverses is crucial in linear algebra and relates to linear independence, dimensionality, solvability of linear systems, and stability of solutions.

The next section that we will cover this week is **VMLS Section 11.2 - Inverse**.

VMLS Section 11.2 - Inverse

Overview

This section focuses on the concept of matrix inverses, their properties, and their applications in solving linear equations and other mathematical problems.

Left and Right Inverse of Matrix Product:

- If matrices A and D are compatible for multiplication and both have right inverses B and E respectively, then EB is a right inverse of AD.
- Similarly, if A and D have left inverses C and F, then FC is a left inverse of AD.

Uniqueness of Inverses:

- If a matrix is both left- and right-invertible, the left and right inverses are unique and equal.
- A matrix with both a left inverse Y and a right inverse X is called invertible or nonsingular, and denoted as A^{-1}.
- Invertible matrices must be square, as tall matrices are not right-invertible and wide matrices are not left-invertible.

Solving Linear Equations with Inverse:

- For square systems of linear equations $Ax = b$, if A is invertible, then the unique solution is $x = A^{-1}b$.
- The solution x is a linear function of the right-hand side vector b.

Examples and Special Cases:

- The identity matrix I is invertible with its inverse equal to itself.
- A diagonal matrix A is invertible if and only if its diagonal entries are nonzero.
- Orthogonal matrices (square matrices with orthonormal columns) are invertible with the inverse equal to their transpose.

Inverse of Transpose and Matrix Product:

- The inverse of a matrix transpose (A^T) is the transpose of the inverse $(A^{-1})^T$.
- For invertible matrices A and B of the same size, $(AB)^{-1} = B^{-1}A^{-1}$.

Negative Matrix Powers and Triangular Matrices:

- Negative matrix powers are defined for square invertible matrices, e.g., $(A^{-1})^k$ for positive integer k.
- Triangular matrices with nonzero diagonal elements are invertible.

Inverse via QR Factorization:

- The QR factorization provides a method to compute the inverse of an invertible square matrix.
- For a matrix A with QR factorization $A = QR$, where Q is orthogonal and R is upper triangular, the inverse is $A^{-1} = R^{-1}Q^T$.

This section provides a comprehensive understanding of matrix inverses, their properties, and applications, especially in the context of solving linear equations and determining the invertibility of matrices.

The next section that we will be covering this week is **VMLS Section 11.3 - Solving Linear Equations**.

VMLS Section 11.3 - Solving Linear Equations

Overview

This section starts with an algorithm for solving linear equations $Rx = b$ where R is an upper triangular $n \times n$ matrix with nonzero diagonal entries, making it invertible. The process, known as back substitution, involves solving for the variables one at a time, starting from x_n and working backwards.

Algorithm For Back Substitution:

Given an upper triangular matrix R and an n-vector b.

For $i = n$ down to 1, calculate x_i using the formula:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} R_{ij}x_j}{R_{ii}}$$

This method computes $x = R^{-1}b$ and is guaranteed to work due to nonzero diagonal entries of R.

Complexity Analysis:

- The number of floating-point operations (flops) for back substitution is n^2.

Solving Equations Using QR Factorization

The section introduces a method to solve square systems of linear equations $Ax = b$ using QR factorization, particularly useful when A is invertible.

Algorithm for Solving via QR Factorization:

- Given an invertible $n \times n$ matrix A and an n-vector b.
- Compute the QR factorization $A = QR$.
- Calculate $y = Q^T b$.
- Solve the triangular equation $Rx = y$ using back substitution.

Complexity Analysis:

- The total number of flops for this method is approximately $2n^3$, indicating a cubic order of complexity.
- The QR factorization step dominates the computational cost.

Factor-Solve Methods

Factor-solve schemes are common in solving linear equations and involve two main steps: factorization and solving. The factor step involves decomposing the coefficient matrix, while the solve step involves solving one or more linear equations using these factors.

Multiple Right-Hand Sides:

- For solving multiple sets of equations $AX = B$ with the same coefficient matrix A but different right-hand sides, the factorization step can be reused.
- The cost of solving multiple sets is approximately the same as solving one set, due to the reuse of factorization.

Computing the Matrix Inverse

The section describes a method to compute the inverse $B = A^{-1}$ of an invertible $n \times n$ matrix A using QR factorization.

Algorithm for Computing the Inverse:

- Perform QR factorization of A.
- For each $i = 1$ to n, solve $Rb_i = \tilde{q}_i$ using back substitution, where b_i and $\tilde{q}_i$ are columns of B and Q^T, respectively.

Complexity Analysis:

- The total computational cost is around $3n^3$ flops.

This section provides a thorough understanding of various methods to solve linear equations, emphasizing the importance of factorization techniques and their practical applications in computational efficiency.

The next section that we will be covering this week is **VMLS Section 11.4 - Examples**.

VMLS Section 11.4 - Examples

Sparse Linear Equations

- Systems of linear equations with sparse coefficient matrices are common in various applications.
- These equations can be solved more efficiently by exploiting the sparsity of the coefficient matrix, typically using a variant of the QR factorization algorithm.
- The memory usage and computational complexity depend on the sparsity pattern of the coefficient matrix, often requiring significantly fewer resources than for non-sparse matrices.

Polynomial Interpolation

- The chapter discusses using matrix inverses for polynomial interpolation.
- Given a cubic polynomial $p(x) = c_1 + c_2x + c_3x^2 + c_4x^3$, the coefficients can be determined by solving a system of linear equations $Ac = b$, where A is a specific Vandermonde matrix.
- The unique solution for the coefficients is given by $c = A^{-1}b$.

Balancing Chemical Reactions

- The process involves setting up and solving a set of linear equations to balance atoms and charges in a chemical reaction.
- The reactant and product matrices are formed and used to balance the equation.
- The coefficients for reactants and products are determined by solving the corresponding linear equations.

Heat Diffusion

- The chapter examines a diffusion system with nodes having fixed potential and others with zero external source.
- This models a thermal system where some nodes are in constant contact with an external heat source, while others are internal without heat sources.
- The system is represented as a set of linear equations $Bs + Ce = d$, where B and C are diagonal matrices, and d is a vector representing fixed potential and zero source conditions.
- The entire system, including flow conservation and boundary conditions, is assembled into a larger set of linear equations and solved assuming the coefficient matrix is invertible.

This section provides a diverse range of practical applications of matrix inverses, illustrating their significance in solving real-world problems in various fields.

The last section that we will cover this week is **VMLS Section 11.5 - Pseuod-Inverse**.

VMLS Section 11.5 - Pseuod-Inverse

Overview

The pseudo-inverse, denoted as $A^\dagger$ or A^+, is a generalization of the matrix inverse that applies to matrices that are not necessarily square or invertible. It plays a crucial role in solving over-determined or under-determined systems of linear equations and has significant applications in various fields.

Linearly Independent Columns and Gram Invertibility

- An $m \times n$ matrix A has linearly independent columns if and only if its $n \times n$ Gram matrix A^TA is invertible.
- The pseudo-inverse for a square or tall matrix with linearly independent columns is given by:

$$A^\dagger = (A^TA)^{-1}A^T$$

Pseudo-Inverse of Square or Wide Matrix

- For a square or wide matrix A, a right inverse is given by:

$$A^\dagger = A^T(AA^T)^{-1}$$

- This formula also defines the pseudo-inverse and reduces to the ordinary inverse when A is square.

General Cases

The pseudo-inverse is defined for any matrix, including cases where A is tall with linearly dependent columns or wide with linearly dependent rows, and even when A is square but not invertible.

Pseudo-Inverse via QR Factorization

- For matrices that are left-invertible, the pseudo-inverse can be computed using QR factorization.
- The formula is given by:

$$A^\dagger = R^{-1}Q^T$$

- This computation involves QR factorization followed by back substitution on the columns of Q^T.
- The complexity of this method is $3mn^2$ flops for an $n \times m$ matrix.

Solving Over and Under-Determined Systems

- The pseudo-inverse provides a method to solve over-determined systems (with linearly independent columns) and under-determined systems (with linearly independent rows).
- For an over-determined system $Ax = b$, the solution is $x = A^{\dagger}b$ if it exists.
- For an under-determined system $Ax = b$, any vector b has a solution given by $x = A^{\dagger}b$.

Numerical Example

- A numerical example using a 3×2 matrix A with linearly independent columns demonstrates the application of the pseudo-inverse in solving an over-determined system of linear equations.
- The unique solution of $Ax = b$ is found using $A^{\dagger}$.

This chapter elaborates on the utility of the pseudo-inverse in various contexts, especially in solving linear systems that do not have a unique solution or where the coefficient matrix is not square. The concept of the pseudo-inverse extends the applicability of matrix inverses to a broader range of matrices, enhancing the problem-solving toolkit in linear algebra.

Exam 2

Exam 2

13.0.1 Piazza

Must post / respond to at least **four** Piazza posts this week.

13.0.2 Lectures

The lectures for this week and their links can be found below:

- Linear Regression Introduction ≈ 12 min.
- Linear Regression With Higher-Order Terms: Polynomial Regression ≈ 13 min.
- Matrix Factorization Introduction ≈ 11 min.
- PCA Introduction ≈ 8 min.

13.0.3 Exam

The exam for this week is:

- Exam 2 Notes

Eigenvalues & Eigenvectors

Eigenvalues & Eigenvectors

14.0.1 Piazza

Must post / respond to at least **four** Piazza posts this week.

14.0.2 Lectures

The lectures for this week and their links can be found below:

- Eigenvalues And Eigenvectors ≈ 21 min.
- Eigenvalues In Dynamical Systems ≈ 27 min.

14.0.3 Assignments

There is no assignment for this week.

14.0.4 Quiz

The quiz for this week is:

- Eigenvalues And Eigenvectors Quiz