Data 345

Applied Linear Algebra for Statistical Learning

Class 3 (Sep. 4, 2025)

The Story So Far

- Created a basic <vector> object in Python.
- Use <__repr__> method to represent the object as an output.
- Define <__add__>, <__mul__>, and <__rmul__> methods to implement addition and scalar multiplication.

```python
class vector:
    def __init__(self, data):
        self.data = data

    def __repr__(self):
        return f"vector({self.data})"

    def __add__(self, other):
        return vector(data=[a + b for a,b in zip(self.data, other.data)])

    def __mul__(self, other):
        return vector(data=[a*other for a in self.data])

    def __rmul__(self, other):
        return self.__mul__(other)
```

✓ 0.0s Python

```python
vector([-1,3,9]) + -25*vector([44, 8, 3])
```

✓ 0.0s Python

```
vector([-1101, -197, -66])
```

While You Were Away…

```python
class vector:
    def __init__(self, data):
        self.data = data
```

```python
class vector:
    def __init__(self, data):
        try:
            # Convert input data to a float type for computation
            self.data = [float(x) for x in data]
        except (TypeError,ValueError) as e:
            raise type(e)(f"Input must be an iterable of numbers. Problem: {e}") from None

        # Insist that the list of elements is not empty
        if len(self.data) == 0:
            raise ValueError("Vector cannot be empty")
```

Converts input numbers to `<float>` type for computation

Basic input validation. Returns an error when:
- The input is not an iterable (list-like)
- The input is an iterable, but contains things that can't be converted to numbers
- The input list is empty

```
ValueError: Input must be an iterable of numbers. Problem: could not convert string to float: 'blah' Cell Execution Error
View Problem (Alt+F8)   No quick fixes available
vector([1, 2, 'blah'])
0.0s
```

```
TypeError: Input must be an iterable of numbers. Problem: 'int' object is not iterable Cell Execution Error
View Problem (Alt+F8)   No quick fixes available
vector(15345)
0.0s
```

```
ValueError: Vector cannot be empty Cell Execution Error
class vector(data: list[Any])
View Problem (Alt+F8)   No quick fixes available
vector([])
0.0s
```

While You Were Away…

```python
def __add__(self, other):
    return vector(data=[a + b for a,b in zip(self.data, other.data)])

def __mul__(self, other):
    return vector(data=[a*other for a in self.data])

def __rmul__(self, other):
    return self.__mul__(other)
```

More error handling – make sure we do not add non-vectors to vectors, or vectors of different sizes

```python
def __add__(self, other):
    # Check that we are adding vectors to vectors
    if not isinstance(other, vector):
        raise TypeError(f"Can only add vector to vector, got {type(other).__name__}")

    # Check that the vectors are the same length
    if len(self.data) != len(other.data):
        raise ValueError(f"Vector dimensions must match: {len(self.data)} vs {len(other.data)}")

    return vector(data=[a + b for a,b in zip(self.data, other.data)])

def __mul__(self, other):
    # Check that we are multiplying by a scalar.
    if not isinstance(other,(int, float)):
        raise TypeError(f"Cannot multiply vector by {type(other).__name__}")
    return vector(data=[a*other for a in self.data])

def __rmul__(self, other):
    return self*other
```

For scalar multiplication, we should only use scalars.

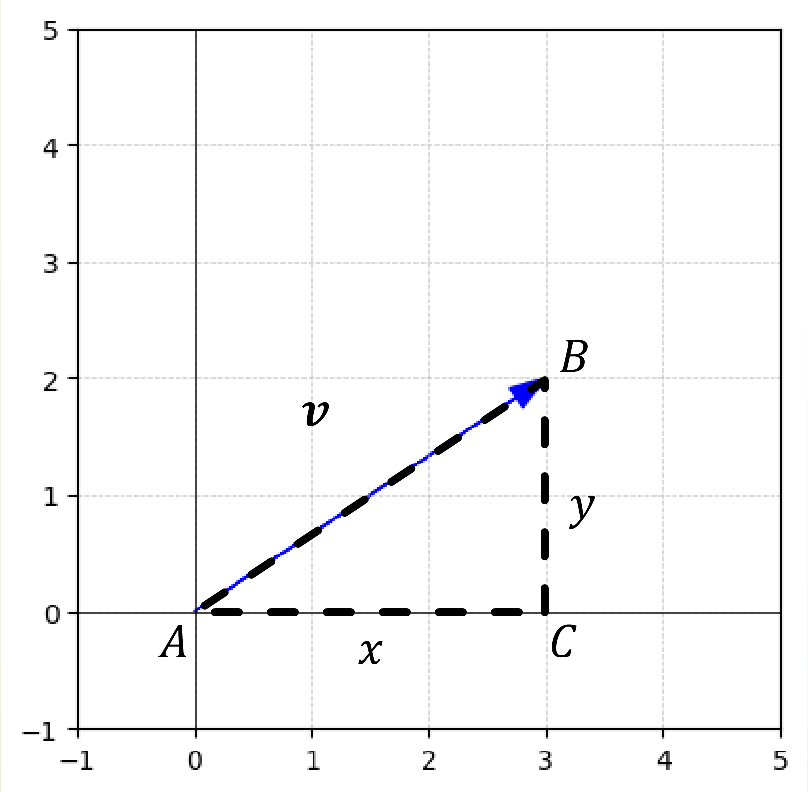
What's Next?

- Implement vector subtraction. If $v = [v_1, v_2, \dots, v_n]$ and $w = [w_1, w_2, \dots, w_n]$, then $v - w = [v_1 - w_1, v_2 - w_2, \dots, v_n - w_n]$. This is easy to do by combining both addition and scalar multiplication.
- Implement vector length as well as vector angles.
- Cauchy-Schwarz and Triangle inequalities.
- Matrices: matrix addition, scalar multiplication, and matrix multiplication.
- Row operations for matrices.

Vector Length

- Recall that we realized vectors geometrically as directed line segments emanating from the origin.
- The "length" of a vector, then, should correspond to the length of this line segment.

$\boldsymbol{v} = [x, y]$



$$\text{length}(\boldsymbol{v}) = \text{length}\left(\overline{AB}\right)$$

$$\left(\text{length}\left(\overline{AB}\right)\right)^2 = \text{length}\left(\overline{AC}\right)^2 + \text{length}\left(\overline{BC}\right)^2$$
$$= x^2 + y^2$$

Therefore,
$$\text{length}(\boldsymbol{v}) = \sqrt{x^2 + y^2}$$

Vector Length

- Let $\boldsymbol{v} = [v_1, v_2, \dots, v_n]$. Then the **Euclidean norm** of $\boldsymbol{v}$,written$||\boldsymbol{v}||$, is given by $||\boldsymbol{v}|| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$. This is the "usual" length assigned to a vector in $\mathbb{R}^n$.

$$\boldsymbol{v} = [v_1, v_2, v_3, \dots, v_n]$$

$$\boldsymbol{v} = [v_1, v_2, v_3, \dots, v_n]$$

$$||\boldsymbol{v}||^2 = (v_1 \cdot v_1) + (v_2 \cdot v_2) + (v_3 \cdot v_3) + \cdots + (v_n \cdot v_n)$$

$$\boldsymbol{v} = [v_1, v_2, v_3, \dots, v_n]$$

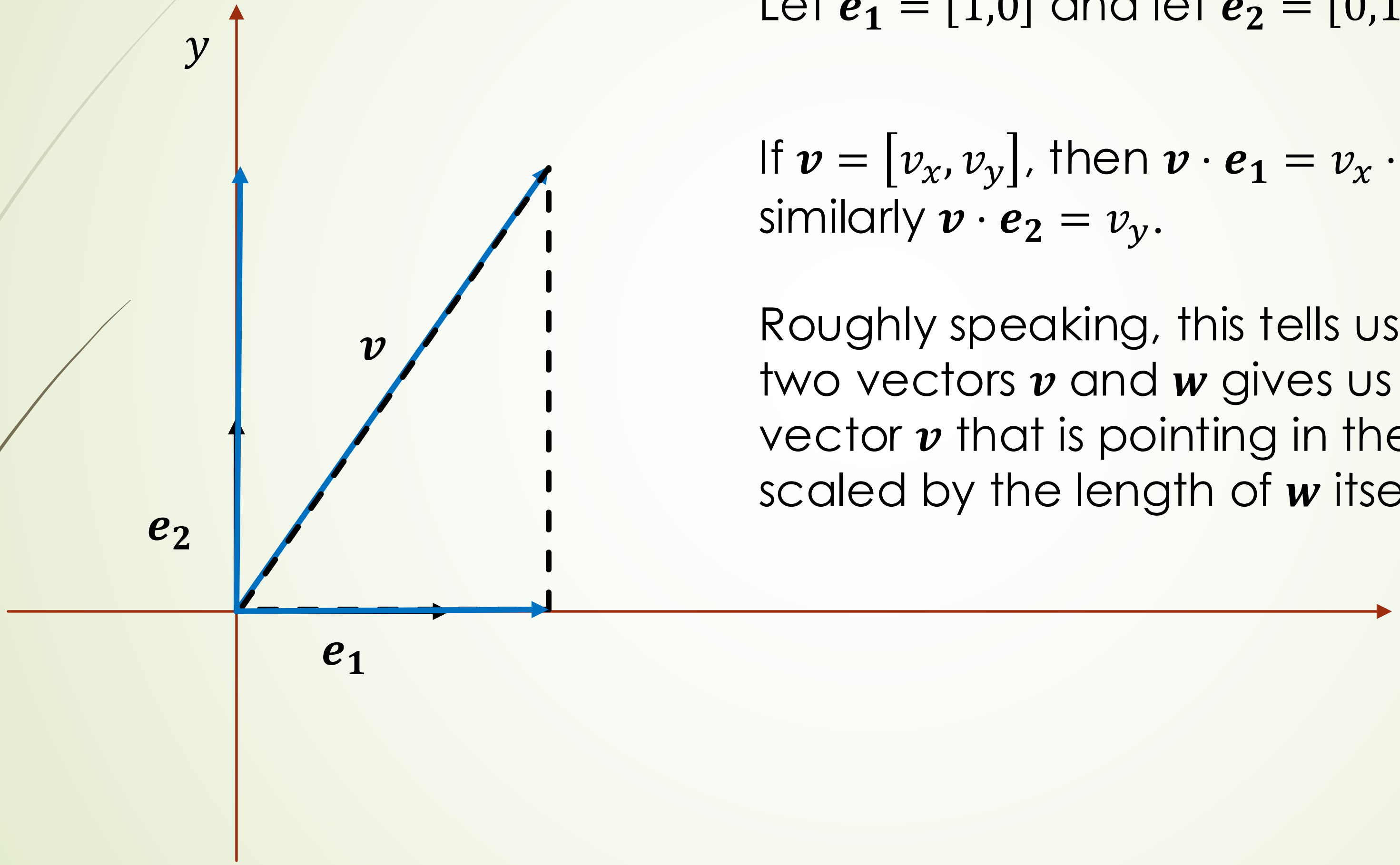
$$\boldsymbol{w} = [w_1, w_2, w_3, \dots, w_n]$$

$$(v_1 \cdot w_1) + (v_2 \cdot w_2) + (v_3 \cdot w_3) + \cdots + (v_n \cdot w_n)$$

Vector Dot Product

- Let $\boldsymbol{v} = [v_1, v_2, \dots, v_n]$ and $\boldsymbol{w} = [w_1, w_2, \dots, w_n]$ be two n-dimensional vectors. Then the **vector dot product** of $\boldsymbol{v}$ and $\boldsymbol{w}$ (or **scalar product** of $\boldsymbol{v}$ and $\boldsymbol{w}$), written as $\boldsymbol{v} \cdot \boldsymbol{w}$ (or $\langle \boldsymbol{v}, \boldsymbol{w} \rangle$), is given as $\boldsymbol{v} \cdot \boldsymbol{w} = (v_1 w_1) + (v_2 w_2) + \cdots + (v_n w_n)$.
- **Note:** this is <u>not</u> multiplication for vectors. The output of the vector dot product is a <u>scalar</u>, not a vector.
- From our earlier definition of $||\boldsymbol{v}||$, it is easy to see that $\boldsymbol{v} \cdot \boldsymbol{v} = ||\boldsymbol{v}||^2$. How should we interpret $\boldsymbol{v} \cdot \boldsymbol{w}$?

Vector Dot Product



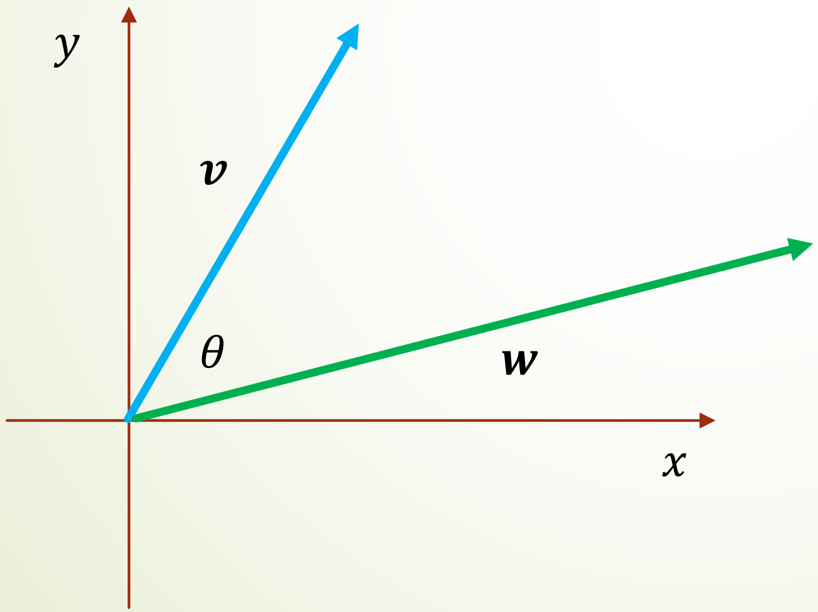
Let $\boldsymbol{e_1} = [1,0]$ and let $\boldsymbol{e_2} = [0,1]$.

If $\boldsymbol{v} = [v_x, v_y]$, then $\boldsymbol{v} \cdot \boldsymbol{e_1} = v_x \cdot 1 + v_y \cdot 0 = v_x$, and similarly $\boldsymbol{v} \cdot \boldsymbol{e_2} = v_y$.

Roughly speaking, this tells us that the dot product of two vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ gives us the length "part" of the vector $\boldsymbol{v}$ that is pointing in the same direction as $\boldsymbol{w}$, but scaled by the length of $\boldsymbol{w}$ itself.

Vector Dot Product

- Geometrically, the dot product has an additional useful realization:



Could be negative

$$\boldsymbol{v} \cdot \boldsymbol{w} = ||\boldsymbol{v}|| \cdot ||\boldsymbol{w}|| \cdot \cos(\theta)$$

Always nonnegative

Vector Dot Product

- Recall that we discussed the notion of **simultaneous linear equations** in multiple variables.

$$3x_1 + 2x_2 + x_3 = 39$$
$$2x_1 + 3x_2 + x_3 = 34$$
$$x_1 + 2x_2 + 3x_3 = 26$$

- Crucially, we can also set up this system of equations in terms of vector dot products.
- Setting $\boldsymbol{v_1} = [3, 2, 1]$, $\boldsymbol{v_2} = [2, 3, 1]$, $\boldsymbol{v_3} = [1, 2, 3]$, and $\boldsymbol{x} = [x_1, x_2, x_3]$, the system of equations becomes:

$$\boldsymbol{v_1} \cdot \boldsymbol{x} = 39$$
$$\boldsymbol{v_2} \cdot \boldsymbol{x} = 34$$
$$\boldsymbol{v_3} \cdot \boldsymbol{x} = 26$$

Goals

- Implement a vector subtraction method
- Implement a vector length or `<norm>` attribute for the `<vector>` class (take a vector $\boldsymbol{v} = [v_1, v_2, \ldots, v_n]$ and output $\sqrt{v_1^2 + v_2^2 + \cdots v_n^2}$)
- Implement a vector `<dot>` method, which will take in two like-sized vectors $\boldsymbol{v} = [v_1, v_2, \ldots, v_n]$ and $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]$, and then output $v_1 w_1 + v_2 w_2 + \cdots + v_n w_n$
- Utilize the properties of the vector dot product and a math computation library to calculate the angle between two vectors.