Linear Algebra Essentials

Vectors, Matrices, and Operations for ML

Vectors

What is a Vector?

A **vector** is an ordered list of numbers. Think of it as:

- A list of values
- A point in space
- A direction with magnitude

Notation

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \text{ (column vector)}$$

$$\mathbf{v} = [v_1, v_2, \ldots, v_n] \text{ (row vector)}$$

Real World Examples

2D Vector (Position)

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Feature Vector (House)

$$\begin{bmatrix} \text{bedrooms} \\ \text{bathrooms} \\ \text{sqft} \\ \text{price} \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1500 \\ 350000 \end{bmatrix}$$

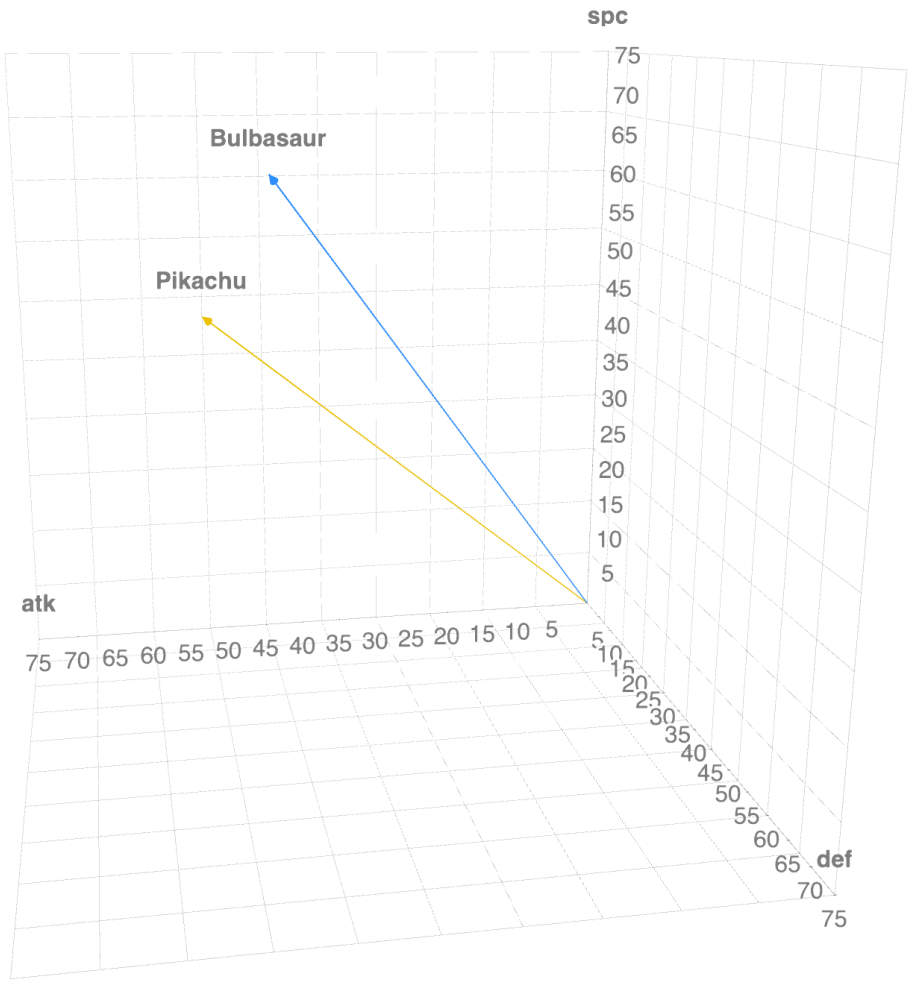
Pokemon as Vectors

We can think of a Pokémon as a vector of its attributes:

$$\mathbf{P} = \begin{bmatrix} \text{HP} \\ \text{Attack} \\ \text{Defense} \\ \text{Speed} \\ \text{Special Attack} \\ \text{Special Defense} \end{bmatrix} = \begin{bmatrix} 45 \\ 49 \\ 49 \\ 45 \\ 65 \\ 65 \end{bmatrix}$$

We can now imagine every pokemon as a point in a 6D space, where each dimension corresponds to one of its attributes.

Distances between these points can tell us how similar or different Pokémon are based on their attributes.



Vectors in Python (Lists)

```python
# Creating vectors with lists
vector_a = [1, 2, 3]
vector_b = [4, 5, 6]

# Accessing elements
first_element = vector_a[0]  # 1
print(f"First element: {first_element}")

# Vector length
length = len(vector_a)  # 3
print(f"Vector length: {length}")

# Adding vectors element-wise
vector_sum = [a + b for a, b in zip(vector_a, vector_b)]
print(f"Vector sum: {vector_sum}")  # [5, 7, 9]

# Scalar multiplication
scalar = 2
vector_scaled = [scalar * x for x in vector_a]
print(f"Scaled vector: {vector_scaled}")  # [2, 4, 6]
```

Vectors in Pandas

```
# Creating vectors as Series
vector_series = pd.Series([1, 2, 3, 4, 5], name='my_vector')
print(vector_series)

# From a DataFrame column (common in ML)
df = pd.DataFrame({
    'feature1': [1, 2, 3, 4],
    'feature2': [5, 6, 7, 8],
    'target': [10, 20, 30, 40]
})

# Extract column as vector
feature_vector = df['feature1']  # This is a Series
print(f"Feature vector:\n{feature_vector}")

# Dot product with another vector
other_vector = pd.Series([2, 3, 4, 5])
dot_product = feature_vector.dot(other_vector)
print(f"Dot product: {dot_product}")  # 70
```

```
0    1
1    2
2    3
3    4
4    5
```

Matrices

What is a Matrix?

A **matrix** is a 2D array of numbers arranged in rows and columns. Think of it as:

- A table of values
- A collection of vectors
- A transformation

Notation

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Size: $m \times n$ (m rows, n columns)

Real World Examples

Data Matrix (3 houses, 4 features)

$$\begin{bmatrix} 3 & 2 & 1500 & 350000 \\ 4 & 3 & 2000 & 450000 \\ 2 & 1 & 1000 & 250000 \end{bmatrix}$$

Identity Matrix (3×3)

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrices in Python (Lists of Lists)

```python
# Creating a matrix with lists of lists
matrix_A = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Accessing elements
element = matrix_A[0][1]  # 2 (row 0, column 1)
print(f"Element at (0,1): {element}")

# Getting dimensions
print(f"Shape: {len(matrix_A)}x{len(matrix_A[0])}")

# Extracting a row
first_row = matrix_A[0]  # [1, 2, 3]
print(f"First row: {first_row}")
```

```
Element at (0,1): 2
Shape: 3x3
First row: [1, 2, 3]
```

Matrices in Pandas (DataFrames)

```
# Creating a matrix as DataFrame
matrix_df = pd.DataFrame([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
], columns=['A', 'B', 'C'])

print("Matrix as DataFrame:")
print(matrix_df)

# Accessing elements
element = matrix_df.loc[0, 'B']  # 2
print(f"\nElement at row 0, column 'B': {element}")
```

```
Matrix as DataFrame:
   A  B  C
0  1  2  3
1  4  5  6
2  7  8  9

Element at row 0, column 'B': 2
```

```
# Extracting features matrix (common in ML)
df = pd.DataFrame({
    'bedrooms': [3, 4, 2],
    'bathrooms': [2, 3, 1],
    'sqft': [1500, 2000, 1000],
    'price': [350000, 450000, 250000]  # target variable
})

# Separate features and target
X = df[['bedrooms', 'bathrooms', 'sqft']]  # Feature matrix
y = df['price']  # Target vector

print(f"\nFeature matrix shape: {X.shape}")  # (3, 3)
print(f"Target vector shape: {y.shape}")  # (3,)
```

```
0    1
1    2
2    3
3    4
4    5
Name: my_vector, dtype: int64

Feature vector:
0    1
1    2
2    3
```

Matrix-Vector Multiplication

How Matrix-Vector Multiplication Works

To multiply matrix $\mathbf{A}$ by vector $\mathbf{x}$:

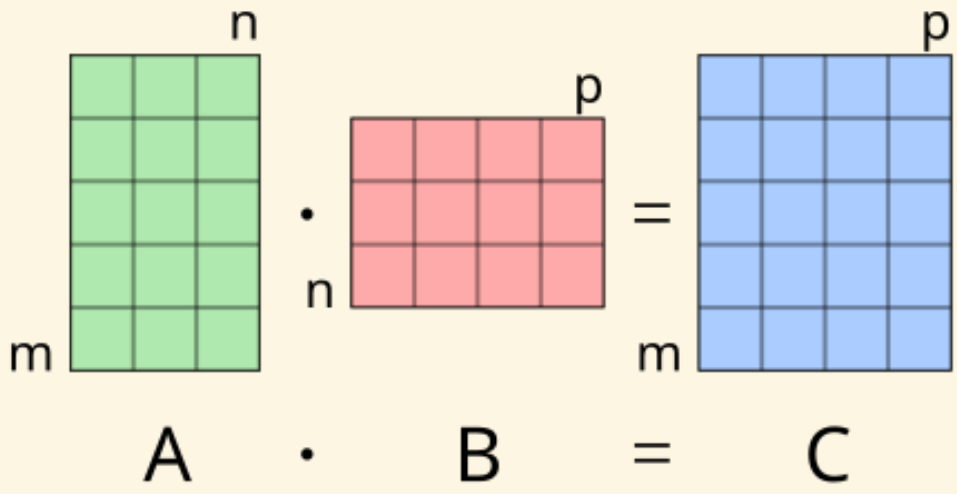
$$\mathbf{Ax} = \mathbf{y}$$

Rules:

- Matrix columns must equal vector length
- Result is a vector with length = matrix rows

Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 1(7) + 2(8) \\ 3(7) + 4(8) \\ 5(7) + 6(8) \end{bmatrix} = \begin{bmatrix} 23 \\ 53 \\ 83 \end{bmatrix}$$



Matrix-Vector Multiplication in Python (Lists)

```python
def matrix_vector_multiply(matrix, vector):
    result = []
    for row in matrix:
        sum_product = sum(a * b for a, b in zip(row, vector)) # dot product
        result.append(sum_product)
    return result

# Example
matrix_A = [
    [1, 2],
    [3, 4],
    [5, 6]
]
vector_x = [7, 8]

result = matrix_vector_multiply(matrix_A, vector_x)
print(f"Result: {result}")  # [23, 53, 83]
```

Matrix Operations with Pandas

```
# Create feature matrix
X = pd.DataFrame({
    'x1': [1, 2, 3],
    'x2': [4, 5, 6]
})

# Create coefficient vector
w = pd.Series([2, 3], index=['x1', 'x2'])

print("Feature matrix X:")
print(X)
print("\nCoefficients w:")
print(w)

# Alternative: Using DataFrame operations
result_df = X.dot(w)  # Pandas aligns by column names!
print(f"\nUsing DataFrame.dot():\n{result_df}")
```

Cheat Sheet: Vectors and Matrices

Vectors

- **Lists:** `[1, 2, 3]`
- **NumPy:** `np.array([1, 2, 3])`
- **Pandas:** `pd.Series([1, 2, 3])`

Matrices

- **Lists:** `[[1, 2], [3, 4]]`
- **NumPy:** `np.array([[1, 2], [3, 4]])`
- **Pandas:** `pd.DataFrame([[1, 2], [3, 4]])`

Matrix-Vector Multiplication

$$\mathbf{X}\mathbf{w} = \mathbf{y}$$

- **X**: Feature matrix (samples × features)
- **w**: Weight/coefficient vector
- **y**: Target/prediction vector

Python (w/Pandas)

```
y = X.dot(w)
```

- `X` : DataFrame of features
- `w` : Series of weights
- `y` : Series of predictions

Exercise

bigd103.link/matrix-multiplication