Intro_to_Matrices_in_NumPy

September 9, 2024

1 Introduction to Matrices in NumPy

Matrices are a fundamental data structure in scientific computing, and `numpy` provides powerful tools for creating and manipulating matrices. This guide will introduce the basics of working with matrices in `numpy` by covering the following topics:

1. **Creating Matrices**
2. **Accessing Elements**
3. **Basic Matrix Operations**
4. **Extracting Rows and Columns**
5. **Row and Column Operations**

You should read the guide carefully and run the code cells. Feel free to edit individual cells for experimentation.

(This guide was created by ChatGPT and edited by Dr. White)

1.1 1. Creating Matrices

In `numpy`, matrices are created as 2D arrays. Here are several ways to create them:

[ ]:
```
import numpy as np
```

1.1.1 a. Creating a Matrix from a List of Lists

[ ]:
```
# Create a 3x3 matrix of ints
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])
matrix
```

[ ]:
```
# Create a 3x3 matrix with 64-bit float data types
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]], np.float64)
matrix
```

1.1.2 b. Creating Matrices with Built-in Functions

[ ]:
```
# Create a 3x3 matrix of zeros
zeros_matrix = np.zeros((3, 3))

# Create a 3x3 matrix of ones
ones_matrix = np.ones((3, 3))

# Create a 3x3 identity matrix
identity_matrix = np.eye(3)

# Create a 3x3 matrix with random values
random_matrix = np.random.rand(3, 3)

zeros_matrix, ones_matrix, identity_matrix, random_matrix
```

1.2 2. Accessing Elements

You can access elements, rows, or columns of a matrix using indexing and slicing:

1.2.1 a. Accessing Individual Elements

[ ]:
```
# Access element at row 1, column 2 (remember, indexing starts from 0)
element = matrix[1, 2]
element
```

1.2.2 b. Accessing Rows and Columns

[ ]:
```
# Access the second row (index 1)
second_row = matrix[1, :]

# Access the third column (index 2)
third_column = matrix[:, 2]

second_row, third_column
```

1.2.3 c. Modifying parts of a matrix

[ ]:
```
matrix_2 = matrix
matrix_2[1,:] = matrix_2[1,:] + 1
matrix, matrix_2
```

1.2.4 d. Matrix references

Notice the above matrix_2 copied a reference to matrix so any change to one changed both. To duplicate a matrix you need to create a new matrix. Notice the two are different in the following cell.

[ ]:
```
matrix_2 = np.array(matrix)
matrix_2[1,:] = matrix_2[1,:] / 100
matrix, matrix_2
```

1.3 3. Basic Matrix Operations

NumPy allows you to perform various matrix operations easily.

1.3.1 a. Matrix Addition and Subtraction

[ ]:
```
# Add two matrices
result_add = matrix + identity_matrix

# Subtract two matrices
result_subtract = matrix - identity_matrix

result_add, result_subtract
```

1.3.2 b. Matrix Multiplication

[ ]:
```
# Element-wise multiplication
element_wise_product = matrix * matrix

# Matrix multiplication, two ways
matrix_product = np.dot(matrix, identity_matrix)
mp = matrix @ identity_matrix

element_wise_product, matrix_product, mp
```

1.3.3 c. Transpose of a Matrix

[ ]:
```
# Transpose the matrix
transpose_matrix = matrix.T
transpose_matrix
```

1.3.4 d. Determinant and Inverse

[ ]:
```
# Calculate the determinant
determinant = np.linalg.det(matrix)

# Calculate the inverse (if the matrix is invertible)
inverse_matrix = np.linalg.inv(matrix) if determinant != 0 else None

determinant, inverse_matrix
```

1.4 4. Extracting Rows and Columns

1.4.1 a. Extracting Rows

[ ]:
```
# Extract the first row (index 0)
first_row = matrix[0, :]
first_row
```

1.4.2 b. Extracting Columns

[ ]:
```
# Extract the second column (index 1)
second_column = matrix[:, 1]

# Extract the last column (index -1)
last_column  = matrix[:,-1]

second_column, last_column
```

1.4.3 c. Extracting Submatrices

[ ]:
```
# Extract a 2x2 submatrix from the top-left corner
submatrix = matrix[0:2, 0:2]
submatrix
```

1.5 5. Row and Column Operations

1.5.1 a. Sum, Mean, and Other Operations on Rows/Columns

[ ]:
```
# Sum of each row
row_sum = np.sum(matrix, axis=1)

# Mean of each column
column_mean = np.mean(matrix, axis=0)

row_sum, column_mean
```

1.5.2 b. Adding Rows/Columns

[ ]:
```
# Add a row (shape should match)
new_matrix_row = np.vstack([matrix, [10, 11, 12]])

# Add a column (shape should match)
new_matrix_column = np.hstack([matrix, [[10], [11], [12]]])

new_matrix_row, new_matrix_column
```

1.5.3 c. Deleting Rows/Columns

[ ]:
```
# Delete the first row (index 0)
matrix_without_first_row = np.delete(matrix, 0, axis=0)

# Delete the second column (index 1)
matrix_without_second_column = np.delete(matrix, 1, axis=1)

matrix_without_first_row, matrix_without_second_column
```

1.5.4 d. Replacing Rows/Columns

[ ]:
```
# Replace the first row
matrix[0, :] = [10, 11, 12]

# Replace the third column
matrix[:, 2] = [13, 14, 15]

matrix
```