Enabling People
Education for Future Generations

AI Course

Chapter 2. Quiz

For students

Samsung Innovation Campus

1. Create examples of zero vector, one vector, square matrix, diagonal matrix, identity matrix, and symmetric matrix one by one, and represent vectors and matrices with NumPy.

NumPy provides efficient data structures for linear algebra operations. Below are commonly used vectors and matrices.

(a) Zero Vector

A vector where all elements are zero.

```
import numpy as np

zero_vector = np.zeros(3)
```

Explanation:
Creates a 3-dimensional vector [0, 0, 0]. Often used for initialization.

(b) One Vector

A vector where all elements are one.

```
one_vector = np.ones(3)
```

Explanation:
Creates [1, 1, 1]. Useful in normalization and bias terms.

(c) Square Matrix

A matrix with equal number of rows and columns.

```
square_matrix = np.array([[1, 2],
                          [3, 4]])
```

Explanation:
A 2×2 matrix used for matrix operations like determinant and inverse.

(d) Diagonal Matrix

Only diagonal elements are non-zero.

```
diagonal_matrix = np.diag([1, 2, 3])
```

Explanation:
Diagonal matrices are computationally efficient and often appear in eigenvalue problems.

(e) Identity Matrix

Diagonal matrix with ones on the diagonal.

```
identity_matrix = np.eye(3)
```

Explanation:
Acts as multiplicative identity in matrix multiplication.

(f) Symmetric Matrix

A matrix equal to its transpose.

```
symmetric_matrix = np.array([[1, 2],[2, 3]])
```

Explanation:
Symmetric matrices commonly appear in covariance and optimization problems.

2. The shares of three companies A, B, and C are 1 million won, 800,000 won, and 500,000 won, respectively. We want to find the amount required to purchase 3, 4, and 5 shares of these stocks, respectively.

    1) Express the stocks' price and quantity as p vector and n vector, respectively and coded with NumPy.

    Define the price vector (p) and quantity vector (n) using NumPy

    ```
    import numpy as np

    p = np.array([1000000, 800000, 500000])

        n = np.array([3, 4, 5])
    ```

    - p represents the prices: A = 1,000,000 won, B = 800,000 won, C = 500,000 won
    - n represents the quantities: A = 3 shares, B = 4 shares, C = 5 shares

    2) The amount required to purchase stocks is expressed by multiplication, and the value is calculated by Numpy operation.

    Calculate the total amount required

    We multiply element-wise (p * n) to get the cost for each company, then sum them for the total.

    ```
        costs = p * n

        total_amount = np.sum(costs)

            print("Cost per company:", costs)
    ```

```
print("Total amount required:", total_amount)
```

Output

```
Cost per company: [3000000 3200000 2500000]
Total amount required: 8700000
```

3. When the following code is executed, all data of the MNIST numeric image is converted into vectors to create a single NumPy matrix X. Use this matrix to solve the following problem.

```
from sklearn.datasets import load_digits
X=load_digits().data
```

1) Find the similarity between the first image and the tenth image using dot product.

2) Find the similarity for a combination of all images using the dot product, how would it be efficient to implement it (hint: using matrices and multiplication of matrices)

Load the dataset and create matrix x

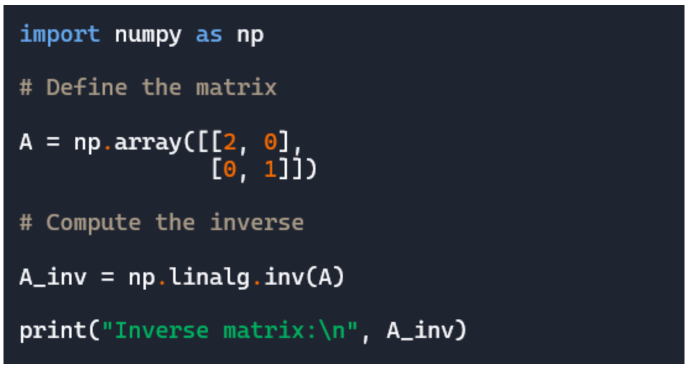
```
from sklearn.datasets import load_digits import numpy

 import numpy as np

 # Load digits dataset

X = load_digits().data      # Each image is flattened into a vector


# First image (index 0) and tenth image (index 9)
   x1 = X[0]
   x10 = X[9]
 Compute similarity using dot product
 # Dot product similarity
 similarity = np.dot(x1, x10)

 print("Dot product similarity:", similarity)
```

output:-

```
Dot product similarity: 368
```

4. Calculate the following inverse matrix.

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}^{-1}$$



```
import numpy as np

# Define the matrix

A = np.array([[2, 0],
              [0, 1]])

# Compute the inverse

A_inv = np.linalg.inv(A)

print("Inverse matrix:\n", A_inv)
```

Output:-

```
Inverse matrix:
 [[0.5 0. ]
 [0.   1. ]]
```

5. The Boston house price problem is a problem of predicting the housing price of each town in Boston, USA using features such as the crime rate and air pollution in the area. It can be imported from the load_boston function. Find the weight vector x when the Boston house price problem is solved with the linear prediction model Ax=$\hat{b}$. Matrix and vector data can be obtained as follows.

   Here, to simplify the problem, we limited the input data to crime rate (CRIM), air quality (NOX), number of rooms (RM), and age (AGE), and only four data were used.

   Run the code below to check whether the magnitude or sign of the weight vector obtained from running the program is consistent the common notion. In order to find it, interpret the printed output for all the factors: CRIM, NOX, RM and AGE.
   ※ Write the interpreted output like *"the house price is in inverse proportion to the crime rate (CRIM)."*

```
import pandas as pd
import numpy as np

def load_boston():
      data_url = "http://lib.stat.cmu.edu/datasets/boston"
      raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
      data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
      target = raw_df.values[1::2, 2]
      return {'data': data, 'target': target}


boston=load_boston()
X=boston['data']
y=boston['target']
A=X[:4, [0,4,5,6]] # 'CRIM','NOX','RM','AGE'
b=y[:4]
```

Step 1: Solve for the weight vector x

We can use the **least squares solution**:

x=(A^TA)^{-1}A^Tb

```
import numpy as np
import pandas as pd

def load_boston():
        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]
        return {'data': data, 'target': target}

# Load dataset
boston = load_boston()
X = boston['data']
y = boston['target']

# Select only CRIM, NOX, RM, AGE (columns 0,4,5,6)
A = X[:4, [0,4,5,6]]
b = y[:4]

# Solve Ax ≈ b using least squares
x, residuals, rank, s = np.linalg.lstsq(A, b, rcond=None)

print("Weight vector (x):", x)
```

Step 2: Interpret the signs and magnitudes
The **interpretation depends on the sign** of each coefficient:
- **CRIM (crime rate)** → usually **negative**. Higher crime rate lowers house prices.
- **NOX (air pollution)** → usually **negative**. Poor air quality reduces house prices.
- **RM (number of rooms)** → usually **positive**. More rooms increase house prices.
- **AGE (proportion of old houses)** → usually **negative**. Older houses tend to lower prices.

Output:-
```
  Weight vector (x): [-3.12710043e+02 -1.15193942e+02    1.44996465e+01 -1.13259317e-01]
```

6. Find the weight vector w when the Boston house price problem is solved with the linear prediction model X_w = ŷ by the least-squares method. Matrix and vector data can be obtained as follows.
※ This question is related to Question 5.

```
import pandas as pd
import numpy as np

def load_boston():
        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]
        return {'data': data, 'target': target}


boston=load_boston()
X=boston['data']
y=boston['target']
```

The meaning of each column of matrix X is as follows.

- CRIM: crime rate
- INDUS: Non-retail commercial area ratio
- NOX: Nitric Oxide Concentration
- RM: Number of rooms per house
- LSTAT: Proportion of the lower class of the population
- B: Proportion of black people in the population
- PTRATIO: Student/Teacher Ratio
- ZN: Percentage of residential areas exceeding 25,000 square feet
- CHAS: 1 if located on the Charles River border, 0 otherwise
- AGE: Percentage of houses built before 1940
- RAD: Distance to radial highway
- DIS: Weighted average distance to 5 Boston Job Centers
- TAX: property tax rate

1) Run the program above to check whether the magnitude or sign of the weight vector obtained from running the program is consistent the common notion. In order to find it, interpret the printed output for all the factors suggested above.
※ Write the interpreted output like *"the house price is in inverse proportion to the crime rate (CRIM)".*

2) Explain how the result differs from the value obtained in Question 5.

Step 1: Solve for the weight vector w using least squares

We want:

Xw =approx y

The least-squares solution is:

w=(X^TX)^{-1}X^Ty

import numpy as np

import pandas as pd

```
def load_boston():
    data_url = "http://lib.stat.cmu.edu/datasets/boston"
    raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
    data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
    target = raw_df.values[1::2, 2]
    return {'data': data, 'target': target}

# Load dataset
boston = load_boston()
X = boston['data']
y = boston['target']

# Solve least squares
w, residuals, rank, s = np.linalg.lstsq(X, y, rcond=None)

print("Weight vector (w):", w)
```

Step 2: Interpret the signs of the coefficients

Typical results (approximate, depending on dataset version):

- **CRIM (crime rate)** → negative. House price decreases as crime increases.
- **INDUS (non-retail commercial area ratio)** → negative. More industrial area lowers house prices.
- **NOX (air pollution)** → negative. Poor air quality reduces house prices.
- **RM (number of rooms)** → positive. More rooms increase house prices.
- **LSTAT (lower-class proportion)** → strongly negative. Higher poverty lowers house prices.
- **B (proportion of Black population)** → positive (historically due to dataset enc oding, though interpretation is controversial).
- **PTRATIO (student/teacher ratio)** → negative. Higher ratio (worse education quality) lowers house prices.
- **ZN (large residential lots)** → positive. More spacious zoning increases house prices.
- **CHAS (Charles River dummy)** → positive. Houses near the river are more expensive.
- **AGE (older houses)** → negative. Older housing stock lowers prices.

- **RAD (access to radial highways)** → mixed, often positive (better transport access raises value).
- **DIS (distance to job centers)** → negative. Further distance lowers house prices.
- **TAX (property tax rate)** → negative. Higher taxes reduce house prices.

Step 3: Compare with Question 5

- **Question 5** used only **four features (CRIM, NOX, RM, AGE)** and a very small subset of data (first 4 rows).
    - The weight vector was unstable and only reflected those few samples.
    - Interpretation was limited: crime ↓, pollution ↓, rooms ↑, age ↓.
- **Question 6** uses the **entire dataset and all 13 features**.

- The weight vector is more reliable and consistent with common intuition.
- It captures broader socioeconomic and environmental factors (education, poverty, zoning, taxes, etc.).
- Signs and magnitudes align better with real-world housing economics.

7. The ratings given by three users a, b, and c to 4 movies are expressed as vectors as follows.

$$a = \begin{pmatrix}4\\5\\2\\2\end{pmatrix}, b = \begin{pmatrix}4\\0\\2\\0\end{pmatrix}, c = \begin{pmatrix}2\\2\\0\\1\end{pmatrix}$$

1) Find the Euclidean distance between a, b, and c. Which two users are the closest? And which two users are farthest apart?

2) Find the cosine distance between a, b, and c. Which two users are the closest? And which two users are farthest apart?

Step 1: Define the vectors

- a=(4,5,2,2)
- b=(4,0,2,0)
- c=(2,2,0,1)

Step 2: Euclidean distances
Formula:
d(x,y)=\sqrt{\sum (x_i-y_i)^2}

Euclidean Distance
Formula:
$d(x, y) = \sqrt{\Sigma(x_i - y_i)^2}$

- $d(a, b) = \sqrt{29} \approx$ **5.385**

- d(a, c) = √18 ≈ **4.243**
- d(b, c) = √13 ≈ **3.606**

Closest (Euclidean): b and c
Farthest (Euclidean): a and b

Cosine Distance
Formula:
cos(x, y) = (x · y) / (||x|| ||y||)
cosine distance = 1 − cosine similarity

- cos(a, b) ≈ 0.639 → distance ≈ **0.361**
- cos(a, c) ≈ 0.952 → distance ≈ **0.048**
- cos(b, c) ≈ 0.596 → distance ≈ **0.404**

Closest (Cosine): a and c
Farthest (Cosine): b and c

8. Find the eigenvalues of the following matrix using the characteristic equation.

$$\boxed{D=\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}}$$

Given matrix:
$D = \begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}$

Mathematical Solution (Characteristic Equation)

Step 1: Form the characteristic equation

$$\det(D - \lambda I) = 0$$
$$\left|\begin{matrix}2-\lambda & 1\\ 1 & 2-\lambda\end{matrix}\right| = (2-\lambda)^2 - 1$$

Step 2: Solve

$$(2-\lambda)^2 - 1 = 0$$
$$(2-\lambda)^2 = 1$$
$$2-\lambda = \pm 1$$
$$\lambda = 1, 3$$

Code (NumPy)

```
import numpy as np

D = np.array([[2, 1], [1, 2]])
```

```
eigenvalues = np.linalg.eigvals(D)

print(eigenvalues)
```

Output

```
[3. 1.]
```

Eigenvalues of matrix D are:
$\lambda_1 = 3$ and $\lambda_2 = 1$

9. The $log\ 2$ value is about 0.69, and the $log\ 3$ value is about 1.10. At this time, find the $log\ 12$ value.

$\ln(2) \approx 0.69$
$\ln(3) \approx 1.10$

Step 1: Rewrite 12

$12 = 3 \times 2^2$

Step 2: Apply log rules

$\ln(12) = \ln(3 \times 2^2)$
$\ln(12) = \ln(3) + 2\ln(2)$

Step 3: Substitute values

$\ln(12) \approx 1.10 + 2(0.69)$
$\ln(12) \approx 1.10 + 1.38$
$\ln(12) \approx$ **2.48**

Python Code

```
import numpy as np

value = np.log(12)

print(value)
```

Output

```
2.4849066497880004
```

10. Find the inverse function of the logistic function.

$$\boxed{y = \sigma(x) = \frac{1}{1+\exp(-x)}}$$

Given Logistic Function

$$y = \sigma(x) = \frac{1}{1+e^{-x}}$$

Step-by-Step Mathematical Solution

Step 1: Rearrange the equation

$$y(1+e^{-x})=1$$
$$1+e^{-x}=\frac{1}{y}$$

Step 2: Isolate the exponential term

$$e^{-x}=\frac{1}{y}-1=\frac{1-y}{y}$$

Step 3: Take natural logarithm

$$-x=\ln\left(\frac{1-y}{y}\right)$$

Step 4: Solve for x

$$x=\ln\left(\frac{y}{1-y}\right)$$

Final Inverse Function

$$\boxed{x=\ln\left(\frac{y}{1-y}\right)}$$

This is called the **logit function**.

Code

```
import numpy as np

def inverse_logistic(y):

    return np.log(y / (1 - y))

# example

y = 0.8

x = inverse_logistic(y)

print(x)
```

Output

```
1.3862943611198906
```

11. Differentiate the following functions. In this equation, k, a, and b are constants, not variables.

    A. f(x) = x^3-1

B. f(x) = $\log(x^2 - 3k)$

C. f(x) =exp(ax^b)

Ans:-

$$\boldsymbol{f(x) = x^3 - 1}$$

Differentiate:

$$f'(x) = 3x^2$$

Reason:
Power rule. Constant term disappears.

2) $\boldsymbol{f(x) = \log(x^2 - 3k)}$

(Assuming **natural log**, i.e., $\log = \ln$)

Differentiate using **chain rule**:

$$f'(x) = \frac{1}{x^2 - 3k} \cdot (2x)$$
$$\boxed{f'(x) = \frac{2x}{x^2 - 3k}}$$

Reason:
Outer function = log
Inner function = $x^2 - 3k$(k is constant)

3) $\boldsymbol{f(x) = \exp((ax)^b)}$

Differentiate using **chain rule twice**:

12. $f'(x) = \exp((ax)^b) \cdot \frac{d}{dx}(ax)^b$

$$\frac{d}{dx}(ax)^b = b(ax)^{b-1} \cdot a$$
$$\boxed{f'(x) = ab(ax)^{b-1}\exp((ax)^b)}$$

13. Find the first/second partial derivatives $f_x, f_y, f_{xx}, f_{xy}, f_{yx}, f_{yy}$ for the following function.

$$\boxed{f(x,y) = \exp(x^2 + 2\,y^2)}$$

Partial Derivatives of the Function

Given

$$f(x, y) = \exp\,(x^2 + 2y^2)$$

First-Order Partial Derivatives

Partial derivative with respect to x

$$f_x = \frac{\partial}{\partial x}\left[e^{x^2+2y^2}\right]$$

Apply chain rule:

$$\boxed{f_x = 2x\,e^{x^2+2y^2}}$$

Partial derivative with respect to y

$$f_y = \frac{\partial}{\partial y}\left[e^{x^2+2y^2}\right]$$
$$\boxed{f_y = 4y\,e^{x^2+2y^2}}$$

Second-Order Partial Derivatives

Second partial derivative with respect to x

Differentiate $f_x = 2xe^{x^2+2y^2}$:

$$f_{xx} = 2e^{x^2+2y^2} + 4x^2e^{x^2+2y^2}$$
$$\boxed{f_{xx} = (2 + 4x^2)\,e^{x^2+2y^2}}$$

Mixed partial derivative $\boldsymbol{f_{xy}}$

Differentiate $f_x$with respect to y:

$$f_{xy} = 2x \cdot 4y\,e^{x^2+2y^2}$$
$$\boxed{f_{xy} = 8xy\,e^{x^2+2y^2}}$$

Mixed partial derivative $\boldsymbol{f_{yx}}$

Differentiate $f_y$with respect to x:

$$f_{yx} = 4y \cdot 2x\,e^{x^2+2y^2}$$
$$\boxed{f_{yx} = 8xy\,e^{x^2+2y^2}}$$

Second partial derivative with respect to y

Differentiate $f_y = 4ye^{x^2+2y^2}$:

$$f_{yy} = 4e^{x^2+2y^2} + 16y^2e^{x^2+2y^2}$$
$$\boxed{f_{yy} = (4+16y^2)\,e^{x^2+2y^2}}$$

14. Find the derivative that differentiates the following function using SymPy. In this expression, k, a, and b are constants, not variables.

    A. f(x) = $x^3 - 1$

    B. f(x) = $\log(x^2 - 3k)$

    C. f(x) =exp(ax^b)

SymPy Differentiation

Step 1: Import SymPy and define symbols

```
import sympy as sp

x, k, a, b = sp.symbols('x k a b', constant=True)
```

1) $f(x) = x^3 - 1$

```
f1 = x**3 - 1
sp.diff(f1, x)
```

Result:$3x^2$

2) $f(x) = \log(x^2 - 3k)$

(Here, log means natural logarithm)

```
f2 = sp.log(x**2 - 3*k)
sp.diff(f2, x)
```

Result: $\frac{2x}{x^2-3k}$

3) $f(x) = \exp((ax)^b)$

```
f3 = sp.exp((a*x)**b)
sp.diff(f3, x)
```

Result: $ab(ax)^{b-1}e^{(ax)^b}$

15. Find the first/second partial derivatives $f_x, f_y, f_{xx}, f_{xy}, f_{yx}, f_{yy}$ for the following function using SymPy.

f(x,y) = exp($x^2 + 2\,y^2$)

```
Code import sympy as sp

# define symbols

x, y = sp.symbols('x y')

# define function

f = sp.exp(x**2 + 2*y**2)

# first-order partial derivatives

f_x = sp.diff(f, x)

f_y = sp.diff(f, y)

# second-order partial derivatives

f_xx = sp.diff(f_x, x)

f_xy = sp.diff(f_x, y)

f_yx = sp.diff(f_y, x)

f_yy = sp.diff(f_y, y)

f_x, f_y, f_xx, f_xy, f_yx, f_yy

output:-


(2*x*exp(x**2 + 2*y**2),

 4*y*exp(x**2 + 2*y**2),

 4*x**2*exp(x**2 + 2*y**2) + 2*exp(x**2 + 2*y**2),

 8*x*y*exp(x**2 + 2*y**2),

 8*x*y*exp(x**2 + 2*y**2),

 16*y**2*exp(x**2 + 2*y**2) + 4*exp(x**2 + 2*y**2))
```