hw02

November 20, 2024

[6]:

```python
import numpy as np

# Set seed for reproducibility
np.random.seed(42)

# Generate matrices A and B
A = np.random.normal(loc=-1, scale=1, size=(3, 4))  # N(-1, 1)
B = np.random.normal(loc=1, scale=1, size=(4, 3))   # N(1, 1)

# (a) Calculate 2A + 3B'
B_transpose = B.T
result_a = 2 * A + 3 * B_transpose
print("Result of 2A + 3B':\n", result_a)
```

```
Result of 2A + 3B':
 [[ 2.71931512 -0.96339119 -0.42869515  3.36873081]
 [-5.20814748 -2.50676727 -0.07848547  2.73745407]
 [-5.11370227  3.02786208  4.47011092 -4.20570407]]
```

[7]:

```python
# (b) Calculate C = AB
C = np.dot(A, B)
print("\nMatrix C (AB):\n", C)
```

```

Matrix C (AB):
 [[-0.75075658  1.17785467 -2.22195224]
 [-2.19975298  0.65588168  0.79961563]
 [-3.29466083  0.38657607 -2.52164903]]
```

[8]:

```python
# (c) Calculate |C| (Frobenius norm)
frobenius_norm = np.linalg.norm(C, ord='fro')
print("\nFrobenius norm of C (|C|):", frobenius_norm)
```

```

Frobenius norm of C (|C|): 5.491764087367032
```

[9]:

```python
# (d) Generate M = B(B'B)^-1 B' and verify M^2 - M is a zero matrix
B_transpose_B = np.dot(B.T, B)
```

```
B_transpose_B_inv = np.linalg.inv(B_transpose_B)
M = np.dot(B, np.dot(B_transpose_B_inv, B.T))

# Check if M^2 - M is a zero matrix
M_squared = np.dot(M, M)
is_zero_matrix = np.allclose(M_squared - M, 0)

print("\nMatrix M:\n", M)
print("\nIs M^2 - M a zero matrix?:", is_zero_matrix)
```

```
Matrix M:
 [[ 0.99228056  0.07179888 -0.044489   -0.02292373]
 [ 0.07179888  0.33219501  0.41379446  0.21321479]
 [-0.044489    0.41379446  0.74359901 -0.13211507]
 [-0.02292373  0.21321479 -0.13211507  0.93192542]]

Is M^2 - M a zero matrix?: True
```

[10]:

```
# Set seed for reproducibility
np.random.seed(42)

# Generate a 25x4 matrix X with elements from N(0, 1)
X = np.random.normal(loc=0, scale=1, size=(25, 4))

# (a) Calculate the correlation matrix R of X
R = np.corrcoef(X, rowvar=False)  # rowvar=False ensures columns are considered␣
 ↪variables
print("Correlation matrix R:\n", R)
```

```
Correlation matrix R:
 [[ 1.         -0.04903877  0.03853747 -0.13937451]
 [-0.04903877  1.         -0.01328408  0.18062347]
 [ 0.03853747 -0.01328408  1.          0.08800749]
 [-0.13937451  0.18062347  0.08800749  1.        ]]
```

[11]:

```
# (b) Obtain the eigenvalues and eigenvectors of R
eigenvalues, eigenvectors = np.linalg.eig(R)
print("\nEigenvalues of R:\n", eigenvalues)
print("\nEigenvectors of R:\n", eigenvectors)
```

```
Eigenvalues of R:
 [1.25789659 0.76143656 0.94498159 1.03568526]

Eigenvectors of R:
 [[-0.45410028  0.37327316  0.6908857   0.42087651]
 [ 0.55695101 -0.4815761   0.67241916 -0.07577932]
 [ 0.13617182 -0.34965804 -0.26304069  0.88882287]
 [ 0.68194996  0.71168171 -0.03659273  0.1646644 ]]
```

[12]:

```
# (c) Verify trace(R) equals the summation of eigenvalues
trace_R = np.trace(R)
sum_eigenvalues = np.sum(eigenvalues)
print("\nTrace of R:", trace_R)
print("Sum of eigenvalues:", sum_eigenvalues)
print("Is trace(R) equal to the sum of eigenvalues?:", np.isclose(trace_R,␣
  ↪sum_eigenvalues))
```

```

Trace of R: 3.9999999999999999
Sum of eigenvalues: 4.0
Is trace(R) equal to the sum of eigenvalues?: True
```