Lab1

Ruoxin Jia
rj3u25@soton.ac.uk

October 11, 2025

1 Question 1

Question: What do you observe for the last command above (i.e. `print(np.dot(U[:,0], U[:,1]))`)? Can you formally prove that this is the result you would expect for the specific structure in the matrix B?

1.1 Observation

The dot product between different eigenvectors is numerically zero (within machine precision), indicating they are orthogonal.

1.2 Formal Proof

Given:

- Matrix B is real symmetric: $B = B^T$
- U contains eigenvectors of B as columns
- D contains corresponding eigenvalues
- Let $\mathbf{v}_1 = U[:, 0]$ be the eigenvector corresponding to eigenvalue λ_1
- Let $\mathbf{v}_2 = U[:, 1]$ be the eigenvector corresponding to eigenvalue λ_2
- $\lambda_1 \neq \lambda_2$ (distinct eigenvalues)

From the eigen definition:

$$B\mathbf{v}_1 = \lambda_1\mathbf{v}_1 \tag{1}$$
$$B\mathbf{v}_2 = \lambda_2\mathbf{v}_2 \tag{2}$$

Now consider $\mathbf{v}_2^T B\mathbf{v}_1$:

$$\mathbf{v}_2^T B\mathbf{v}_1 = \mathbf{v}_2^T(\lambda_1\mathbf{v}_1) = \lambda_1(\mathbf{v}_2^T\mathbf{v}_1) \tag{3}$$

Since B is symmetric ($B = B^T$):

$$\mathbf{v}_2^T B\mathbf{v}_1 = (B\mathbf{v}_2)^T\mathbf{v}_1 = (\lambda_2\mathbf{v}_2)^T\mathbf{v}_1 = \lambda_2(\mathbf{v}_2^T\mathbf{v}_1) \tag{4}$$

Therefore:

$$\lambda_1(\mathbf{v}_2^T\mathbf{v}_1) = \lambda_2(\mathbf{v}_2^T\mathbf{v}_1) \tag{5}$$
$$(\lambda_1 - \lambda_2)(\mathbf{v}_2^T\mathbf{v}_1) = 0 \tag{6}$$

Since $\lambda_1 \neq \lambda_2$, we must have:

$$\mathbf{v}_2^T \mathbf{v}_1 = 0 \tag{7}$$

This proves that the eigenvectors corresponding to distinct eigenvalues of a real symmetric matrix are orthogonal.

2 Question 2

2.1 Why doesn't the histogram appear flat even though the data is from a uniform distribution?

- **Sampling Variability:** Any finite sample from a perfect uniform distribution will exhibit random fluctuations
- **Law of Large Numbers Limitations:** 1000 data points is insufficient for a perfectly smooth distribution
- **Bin Count Effect:**
  - With 4 bins: Appears relatively flat because each bin contains ∼250 points
  - With 40 bins: Shows significant fluctuations because each bin contains only ∼25 points

2.2 Why does the histogram look slightly different every time you run it?

- **Random Number Generation:** `np.random.rand()` generates different sequences each execution
- **Sample Variability:** Different samples from the same distribution have different statistical properties
- **Stochastic Nature:** The histogram represents one realization of a random process

2.3 How would these observations change with more data?

With more data, the histogram becomes flatter and more closely approximates the ideal uniform distribution. This is due to the Law of Large Numbers - as sample size increases, the sample distribution converges to the theoretical distribution.

```python
import numpy as np
import matplotlib.pyplot as plt

# Test different sample sizes
sample_sizes = [1000, 10000, 100000]
bin_configs = [4, 40]

plt.figure(figsize=(15, 10))

for i, n in enumerate(sample_sizes):
    data = np.random.rand(n)

    for j, bins in enumerate(bin_configs):
        plt.subplot(len(sample_sizes), len(bin_configs), i*len(bin_configs) + j + 1)
        plt.hist(data, bins=bins, alpha=0.7, edgecolor='black')
        plt.title(f'n = {n:,}, bins = {bins}')
        plt.ylim(0, n/bins * 1.5)
        plt.ylabel('Count')
        plt.xlabel('Value')

plt.tight_layout()
plt.show()
```

Listing 1: Testing different sample sizes and bin configurations

3 Question 3

3.1 What Do You Observe?

- Approximately normal distribution
- The distribution is centered around zero

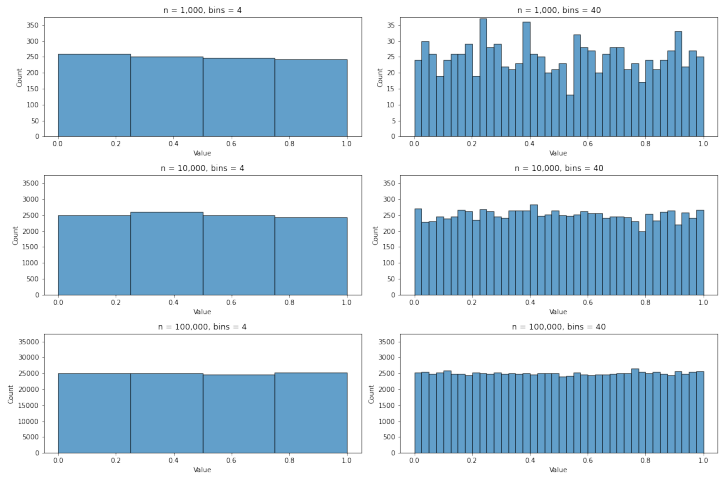

Figure 1: Histogram comparison for different sample sizes and bin configurations

3.2 How the Histogram Changes When Changing the Number of Uniform Random Numbers

The histogram becomes more normally distributed as the number of uniform random numbers increases.

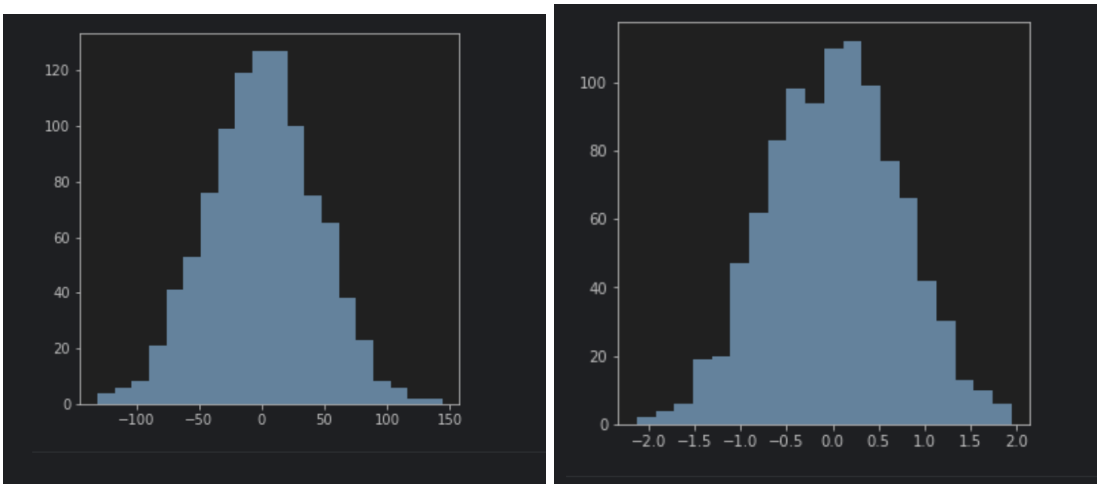

Figure 2: Demonstration of Central Limit Theorem with increasing sample sizes

3.3 Is there a theory that explains your observation?

- **The Theory: Central Limit Theorem (CLT)**
- The Central Limit Theorem states that the sum of a large number of independent, identically distributed random variables with finite mean and variance will approximate a normal distribution, regardless of the original distribution's shape.

4 Question 4: Multivariate Gaussian Distribution Contours

4.1 Objective

Draw contours of the following multivariate Gaussian distributions:

- $N\left(\begin{bmatrix}2.4\\3.2\end{bmatrix}, \begin{bmatrix}2 & -1\\-1 & 2\end{bmatrix}\right)$
- $N\left(\begin{bmatrix}1.2\\0.2\end{bmatrix}, \begin{bmatrix}2 & 0\\0 & 4\end{bmatrix}\right)$

- $N\left(\begin{bmatrix}2.4\\3.2\end{bmatrix}, \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}\right)$

4.2 Implementation

```
nx, ny = 50, 40

m1 = np.array([2.4, 3.2])
C1 = np.array([[2, -1], [-1, 2]], dtype=np.float32)

m2 = np.array([1.2, 0.2])
C2 = np.array([[2, 0], [0, 4]], dtype=np.float32)

m3 = np.array([2.4, 3.2])
C3 = np.array([[2, 0], [0, 2]], dtype=np.float32)

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

X1, Y1, Z1 = twoDGaussianPlot(nx, ny, m1, C1)
contour1 = axes[0].contour(X1, Y1, Z1, 5)
axes[0].clabel(contour1, inline=True, fontsize=8)
axes[0].set_title('N([2.4, 3.2], [[2, -1], [-1, 2]])')
axes[0].set_xlabel('X1')
axes[0].set_ylabel('X2')
axes[0].grid(True, alpha=0.3)
axes[0].set_aspect('equal')

X2, Y2, Z2 = twoDGaussianPlot(nx, ny, m2, C2)
contour2 = axes[1].contour(X2, Y2, Z2, 5)
axes[1].clabel(contour2, inline=True, fontsize=8)
axes[1].set_title('N([1.2, 0.2], [[2, 0], [0, 4]])')
axes[1].set_xlabel('X1')
axes[1].set_ylabel('X2')
axes[1].grid(True, alpha=0.3)
axes[1].set_aspect('equal')

X3, Y3, Z3 = twoDGaussianPlot(nx, ny, m3, C3)
contour3 = axes[2].contour(X3, Y3, Z3, 5)
axes[2].clabel(contour3, inline=True, fontsize=8)
axes[2].set_title('N([2.4, 3.2], [[2, 0], [0, 2]])')
axes[2].set_xlabel('X1')
axes[2].set_ylabel('X2')
axes[2].grid(True, alpha=0.3)
axes[2].set_aspect('equal')

plt.tight_layout()
plt.savefig("three_gaussian_contours.png", dpi=300, bbox_inches='tight')
plt.show()
```

Listing 2: Variance analysis of projected data using Cholesky decomposition

4.3 Results

See Figure 3.

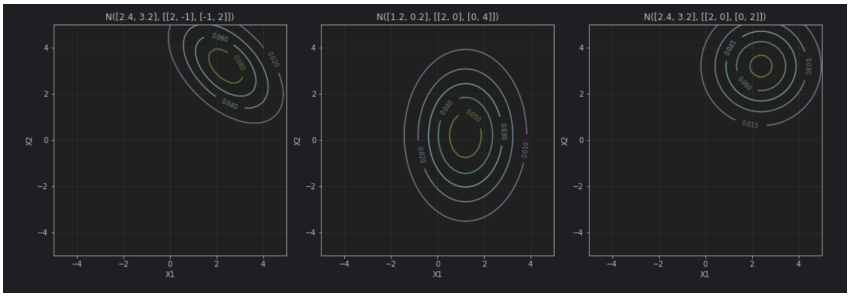

Figure 3: Contour plots of three multivariate Gaussian distributions with different mean vectors and covariance matrices

5 Question 5: Projection Variance Analysis

5.1 Problem

Analyze projection variance of multivariate Gaussian with $C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ along directions $\mathbf{u} = [\sin\theta, \cos\theta]^T$.

Extrema:

- Max variance: 3.000 at $\theta = 45.0°$
- Min variance: 1.000 at $\theta = 135.0°$

Eigen Analysis:

- Eigenvalues: $\lambda_1 = 3.000$, $\lambda_2 = 1.000$
- Eigenvectors: $\mathbf{v}_1 = [0.707, 0.707]$, $\mathbf{v}_2 = [-0.707, 0.707]$

Key Finding: Maximum/minimum projection variances equal eigenvalues and occur along eigenvectors.

5.2 Code

```
C = np.array([[2.0, 1.0], [1.0, 2.0]])
A = np.linalg.cholesky(C)
Y = np.random.randn(5000, 2) @ A.T

theta_range = np.linspace(0, 2*np.pi, 360)
variances = [np.var(Y @ [np.sin(theta), np.cos(theta)]) for theta in theta_range]

max_idx, min_idx = np.argmax(variances), np.argmin(variances)
print(f"Max: {variances[max_idx]:.3f} at {theta_range[max_idx]*180/np.pi:.1f} degrees")
print(f"Min: {variances[min_idx]:.3f} at {theta_range[min_idx]*180/np.pi:.1f} degrees")

eigenvals, eigenvecs = np.linalg.eig(C)
print(f"Eigenvalues: {eigenvals}")

plt.plot(theta_range*180/np.pi, variances)
plt.axvline(theta_range[max_idx]*180/np.pi, color='red', linestyle='--')
plt.axvline(theta_range[min_idx]*180/np.pi, color='green', linestyle='--')
plt.xlabel('Angle theta (degrees)')
plt.ylabel('Projected Variance')
plt.show()
```

Listing 3: Variance analysis of projected data using Cholesky decomposition

5.3 Plot

Analytical: $\text{Var}(\theta) = 2 + \sin(2\theta)$ confirms sinusoidal pattern with 180° period.