

## ✓ Võ Đình Khôi

24c15046

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

# Hàm sigmoid
def sigmoid(x):
    """
    Tính giá trị sigmoid cho x.
    x: Mảng đầu vào.
    Trả về: Giá trị sigmoid của x.
    """
    return 1 / (1 + np.exp(-x))

# Hàm tính hệ số đa thức bằng NumPy
def fit_polynomial_numpy(x, y, degree):
    """
    Tính hệ số của đa thức bậc 'degree' bằng phương pháp bình phương tối thiểu.
    x: Mảng 1D của các giá trị đầu vào.
    y: Mảng 1D của các giá trị đầu ra (sigmoid hoặc dữ liệu có nhiễu).
    degree: Bậc của đa thức cần xấp xỉ.
    Trả về: Vector hệ số của đa thức.
    """
    # Tạo ma trận đặc trưng X_poly với các cột là [1, x, x^2, ..., x^degree]
    X_poly = np.hstack([x**i for i in range(degree + 1)])

    # Giải bài toán bình phương tối thiểu: coeffs = (X.T @ X)^-1 @ X.T @ y
    coeffs = np.linalg.inv(X_poly.T @ X_poly) @ X_poly.T @ y
    return coeffs

# Hàm dự đoán đầu ra từ đa thức
def predict_polynomial_numpy(x, coeffs):
    """
    Tính giá trị dự đoán y từ các hệ số đa thức.
    x: Mảng 1D của các giá trị đầu vào.
    coeffs: Vector chứa các hệ số của đa thức.
    Trả về: Mảng 1D của giá trị dự đoán y.
    """
    # Tạo lại ma trận đặc trưng dựa trên số lượng hệ số (degree + 1)
    X_poly = np.hstack([x**i for i in range(len(coeffs))])

    # Nhân ma trận đặc trưng với vector hệ số để tính giá trị dự đoán
    return X_poly @ coeffs

# Hàm vẽ đồ thị xấp xỉ
def fit_and_plot_numpy(x, y, degree, ax):
    """
    Thực hiện hồi quy đa thức và vẽ đồ thị so sánh.
    x: Mảng 1D của các giá trị đầu vào.
    y: Mảng 1D của giá trị sigmoid (hoặc có nhiễu).
    degree: Bậc của đa thức cần xấp xỉ.
    ax: Đối tượng trục (axis) để vẽ đồ thị.
    Trả về: Vector hệ số của đa thức.
    """
    # Tính các hệ số của đa thức bậc 'degree'
    coeffs = fit_polynomial_numpy(x, y, degree)

    # Tính giá trị dự đoán dựa trên các hệ số vừa tính được
    y_pred = predict_polynomial_numpy(x, coeffs)

    # Vẽ đồ thị
    ax.plot(x, y, label="Sigmoid (Gốc)", color="blue") # Hàm sigmoid gốc
    ax.plot(x, y_pred, label=f"Đa thức (Bậc {degree})", color="red") # Đường đa thức xấp xỉ
    ax.set_title(f"Xấp xỉ Đa thức Bậc {degree}")
    ax.legend()
    ax.grid()
    return coeffs

# Tạo dữ liệu
x = np.linspace(-5, 5, 500).reshape(-1, 1) # Dữ liệu đầu vào nằm trong khoảng [-5, 5]
```

```

y = sigmoid(x).flatten() # Dữ liệu đầu ra sigmoid (y)

# Thêm nhiễu vào dữ liệu
np.random.seed(42) # Đặt seed để kết quả nhiễu có thể tái tạo
noise = np.random.normal(loc=0, scale=0.02, size=y.shape) # Nhiễu Gaussian với độ lệch chuẩn 0.02
y_noisy = y + noise # Thêm nhiễu vào dữ liệu sigmoid

# Vẽ đồ thị xấp xỉ với dữ liệu nhiễu
fig, axs = plt.subplots(1, 2, figsize=(12, 6)) # Tạo 2 đồ thị cạnh nhau
coeffs_deg3_noisy = fit_and_plot_numpy(x, y_noisy, degree=3, ax=axs[0]) # Xấp xỉ bậc 3
coeffs_deg5_noisy = fit_and_plot_numpy(x, y_noisy, degree=5, ax=axs[1]) # Xấp xỉ bậc 5
plt.tight_layout()
plt.show()

# Hiển thị hệ số của các đa thức với dữ liệu nhiễu
# Làm tròn các hệ số đến 5 chữ số thập phân
coeffs_deg3_noisy_rounded = np.round(coeffs_deg3_noisy, 5)
coeffs_deg5_noisy_rounded = np.round(coeffs_deg5_noisy, 5)

# In hệ số đã làm tròn
coeffs_deg3_str = ', '.join([f"{c:.5f}" for c in coeffs_deg3_noisy_rounded])
coeffs_deg5_str = ', '.join([f"{c:.5f}" for c in coeffs_deg5_noisy_rounded])
print("Hệ số của đa thức bậc 3 (có nhiễu):", coeffs_deg3_str)
print("Hệ số của đa thức bậc 5 (có nhiễu):", coeffs_deg5_str)

```

