Bài tập Thực hành môn Khai phá Dữ liệu

Họ và tên: Huỳnh Nguyễn Thế Dân

MSSV: 21110256

Lớp: 21TTH1

Thuật toán Mahalanobis k-means

Định nghĩa hàm calculate_mahalanobis_distance

```
# Định nghĩa hàm
def calculate_mahalanobis_distance(point, center, covariance_matrix):
    # Tính và trả về khoảng cách Mahalanobis
    return mahalanobis(point, center, np.linalg.inv(covariance_matrix))
```

Tóm tắt thuật toán:

- **Dòng 1**: Định nghĩa hàm calculate_mahalanobis_distance với các tham số là điểm dữ liệu, tâm cụm và ma trận hiệp phương sai.
- **Dòng 2**: Sử dụng hàm mahalanobis từ scipy để tính khoảng cách Mahalanobis giữa điểm dữ liệu và tâm cụm với ma trận hiệp phương sai nghịch đảo, và trả về kết quả.

Phân tích hàm mahalanobis:

- mahalanobis(point, center, np.linalg.inv(covariance_matrix)):
  - point: Là điểm dữ liệu cần tính khoảng cách, biểu diễn dưới dạng vector.
  - center: Là tâm cụm, biểu diễn dưới dạng vector.
  - np.linalg.inv(covariance_matrix): Là nghịch đảo của ma trận hiệp phương sai Σ.
- Hàm mahalanobis sử dụng các tham số này để thực hiện phép tính:
  - Tính hiệu số giữa điểm dữ liệu và tâm cụm: x−μ.
  - Tính nghịch đảo của ma trận hiệp phương sai Σ −1 .
  - Nhân hiệu số trên với nghịch đảo của ma trận hiệp phương sai: $(x-\mu)T\Sigma$^ −1.
  - Nhân tiếp với hiệu số ban đầu và lấy căn bậc hai để có được khoảng cách Mahalanobis.

Kết quả là một giá trị khoảng cách Mahalanobis giữa điểm dữ liệu và tâm cụm, thể hiện mức độ tương tự giữa điểm dữ liệu và cụm trong không gian đa chiều có tính đến sự phân tán của dữ liệu.

Định nghĩa hàm mahalanobis_kmeans

```
def mahalanobis_kmeans(X, k, max_iters=100, tol=1e-4):
    n_samples, n_features = X.shape  # Lấy số lượng mẫu và số lượng đặc trưng từ dữ liệu X

    np.random.seed(10)  # Cố định seed để kết quả nhất quán
    centers = X[np.random.choice(n_samples, k, replace=False)]  # Chọn ngẫu nhiên k tâm cụm ban đầu

    covariance_matrices = [np.cov(X.T) for _ in range(k)]  # Tạo ma trận hiệp phương sai ban đầu cho mỗi cụm

    labels = np.zeros(n_samples)  # Khởi tạo nhãn cho mỗi mẫu dữ liệu
    for it in range(max_iters):
        new_centers = np.zeros((k, n_features))  # Khởi tạo các tâm cụm mới
        counts = np.zeros(k)  # Khởi tạo bộ đếm số lượng điểm trong mỗi cụm

        for i, x in enumerate(X):
            distances = [calculate_mahalanobis_distance(x, centers[j], covariance_matrices[j]) for j in range(k)]  # Tính khoảng cách Mahalanobis
            labels[i] = np.argmin(distances)  # Gán nhãn cho cụm gần nhất
            new_centers[int(labels[i])] += x  # Cập nhật tâm cụm mới
            counts[int(labels[i])] += 1

        for j in range(k):
            if counts[j] != 0:
                new_centers[j] /= counts[j]  # Tính trung bình các điểm dữ liệu trong cụm
            covariance_matrices[j] = np.cov(X[labels == j].T)  # Tính ma trận hiệp phương sai mới

        if np.all(np.linalg.norm(new_centers - centers, axis=1) < tol):  # Kiểm tra sự hội tụ
            break
        centers = new_centers  # Cập nhật tâm cụm

    return centers, labels  # Trả về tâm cụm và nhãn của mỗi điểm dữ liệu
```

Tóm tắt thuật toán:

1. **Khởi tạo:**
   - Lấy số lượng mẫu và số lượng đặc trưng từ dữ liệu (X).
   - Cố định seed cho bộ sinh số ngẫu nhiên để kết quả nhất quán.
   - Chọn ngẫu nhiên (k) tâm cụm ban đầu từ dữ liệu (X).
   - Tạo ma trận hiệp phương sai ban đầu cho mỗi cụm.
2. **Gán cụm và Cập nhật tâm cụm:**
   - Khởi tạo các nhãn và các biến lưu trữ tâm cụm mới và bộ đếm.

- Với mỗi điểm dữ liệu (x):
  - Tính khoảng cách Mahalanobis từ (x) đến mỗi tâm cụm.
  - Gán nhãn cho cụm có khoảng cách Mahalanobis nhỏ nhất.
  - Cập nhật tâm cụm mới và tăng bộ đếm.
- Tính lại tâm cụm bằng cách lấy trung bình các điểm dữ liệu trong cụm.
- Tính ma trận hiệp phương sai mới cho mỗi cụm.

3. **Kiểm tra hội tụ:**

- Kiểm tra xem các tâm cụm có thay đổi ít hơn ngưỡng (tol) hay không.
- Nếu hội tụ, dừng lại; nếu không, cập nhật tâm cụm và lặp lại.

4. **Trả về kết quả:**

- Trả về tâm cụm và nhãn của mỗi điểm dữ liệu.

Hết.