TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: HỌC MÁY

ĐỀ TÀI: Sử dụng phương pháp Phân cụm K-means để phân cụm các nhóm người mắc bệnh tim, hỗ trợ trong việc đưa ra phác đồ điều trị hiệu quả

Giáo viên hướng dẫn: TS. Nguyễn Thị Kim Ngân

Sinh viên/nhóm sinh viên thực hiện:

1. Vũ Anh Xuân, lớp 63TTNT
2. Đào Thị Trang, lớp 63TTNT
3. Khổng Quốc Trung, lớp 63TTNT
4. Nguyễn Quang Việt, lớp 63TTNT

Hà Nội, tháng 1 năm 2024

Phần 1: Lý thuyết

Trình bày lý thuyết của các phương pháp: K-means

- **Đầu vào (Input)**: Dữ liệu X và số lượng cluster cần tìm K

- **Đầu ra (Output)**: Các center M và label vector cho từng điểm dữ liệu Y

- **Cách thực hiện (Method)**:

1. Chọn K điểm bất kỳ làm các center ban đầu
2. Phân mỗi điểm dữ liệu vào cluster có center gần nó nhất
3. Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước nó thì thuật toán dừng
4. Cập nhật center cho từng cluster bằng cách lấy trung bình cộng của tất các các điểm dữ liệu đã được gán vào cluster đó sau bước 2
5. Quay lại bước 2

- **Hàm mất mát:**

+ Giả sử, m_k là center (hoặc representative) của mỗi cluster
  - Ước lượng tất cả các điểm được phân vào cluster này bởi m_k
  - Mỗi điểm dữ liệu x_i được phân vào cluster k sẽ có sai số là $|x_i - m_k|$
+ Chúng ta mong muốn sai số $|x_i - m_k|$ nhỏ nhất nên đại lượng dưới đây cần có giá trị nhỏ nhất:

$$||x_i - m_k||_2^2$$

+ Vì x_i được phân vào cluster k nên y_{ik}=1, y_{ij} =0, ∀j ≠ k. Do đó chúng ta có:

$$\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = y_{ik}\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

+ Như vậy, sai số trung bình cho toàn bộ dữ liệu là:

$$\mathcal{L}(\mathbf{Y}, \mathbf{M}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Trong đó, M = $[m_1, m_2, ..., m_k] \in R^{dxK}$ là ma trận tạo bởi K centroid.

Hàm mất mát trong bài toán là $L(Y, M)$ với ràng buộc như được nêu trong.

- Bài toán cần tối ưu là:

$$\mathbf{Y}, \mathbf{M} = \underset{\mathbf{Y}, \mathbf{M}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

$$\text{thoả mãn: } y_{ij} \in \{0, 1\}, \ \forall i, j; \quad \sum_{j=1}^{K} y_{ij} = 1, \ \forall i$$

Phần 2: Thực nghiệm

1. Mô tả bài toán

- Tên bài toán :Sử dụng phương pháp Phân cụm K-means để phân cụm các nhóm người mắc bệnh tim, hỗ trợ trong việc đưa ra phác đồ điều trị hiệu quả

2. Mô tả tập dữ liệu của bài toán

- Dữ liệu gồm 303 vecto dữ liệu chứa các thông tin:

1. **Id**: Mã bệnh nhân
2. **Age**: Tuổi của bệnh nhân
3. **Sex**: Giới tính của bệnh nhân
4. **Cp**: Kiểu đau ngực
5. **Trestbps**: Huyết áp lúc nghỉ (mm Hg khi nhập viện)
6. **Chol**: Cholesterol huyết thanh tính bằng mg/dl
7. **Fbs**: Đường huyết lúc đói > 120 mg/dl (1= đúng; 0=sai)
8. **Restecg**: Kết quả điện tâm đồ lúc nghỉ
9. **Thalach**: Nhịp tim tối đa đạt được
10. **Exang**: Đau thắt ngực do gắng sức
11. **Oldpeak**: ST trầm cảm gây ra bởi tập thể dục liên quan đến nghỉ ngơi

12. **Slope**: Độ dốc của đoạn ST bài tập cao điểm

- Mô tả dữ liệu ma trận:

+ Ma trận dữ liệu X [Age, Sex, Cp, Trestbps, Chol, Fbs, Restecg, Thalach, Exang, Oldpeak, Slope]

- Chia tập dữ liệu thành 2 phần: 90% mẫu dùng để huấn luyện mô hình, mô hình tìm được sẽ dự đoán nhãn của 10% mẫu còn lại.

3. Xây dựng mô hình

- Các bước tiền xử lý dữ liệu:

=> Xóa bỏ cột id vì không ảnh hưởng đến việc phân cụm.

- Cách xây dựng mô hình phân cụm: Chọn ra mô hình tốt nhất, thử lần lượt k = [2, 10]. Mô hình tốt nhất là mô hình có độ đo silhouette gần 1

Nhóm em code tay class mô hình phân cụm Kmean:

class KMean_Clustering:

+ Hàm tạo mặc định: giá trị K = 3, centroids = None

+ Hàm lấy ra K mẫu đầu tiên trong X_Train làm tâm cụm:

```
def kmeans_init_centroids(self, X):
    arr = []
    for i in range (self.K):
    arr.append(i+1)
    return X[arr]
```

+ Hàm Tính toán khoảng cách từng mẫu dữ liệu đến từng tâm cụm, phân mẫu dữ liệu về cụm có khoảng cách nhỏ nhất:

```
def kmeans_assign_labels(self, X, centroids):
    D = cdist(X, centroids)
```

```
        return np.argmin(D, axis = 1)
```

+ Hàm Kiểm tra tâm lần lặp sau có bằng tâm lần lặp trước không???:

```
    def has_converged(self, centroids, new_centroids):
        return (set([tuple(a) for a in centroids]) == set([tuple(a) for a in
new_centroids]))
```

+ Hàm Tính tâm mới bằng cách lấy trung bình cộng các điểm trong cụm:

```
    def kmeans_update_centroids(self, X, labels):
        centroids = np.zeros((self.K, X.shape[1]))
        for k in range(self.K):
            Xk = X[labels == k, :]
            centroids[k,:] = np.mean(Xk, axis = 0)
        return centroids
```

+ Hàm fit trên tập X_train:

```
        def __init__(self, K = 3):
        self.K = K
        self.centroids = None
```

+ Hàm khởi tạo tâm cụm: lấy ra K mẫu đầu tiên trong X_Train làm tâm cụm

```
    def kmeans_init_centroids(self, X):
        arr = []
        for i in range (self.K):
            arr.append(i+1)
        return X[arr]
```

+ Hàm gán nhãn cho mẫu dữ liệu: tính toán khoảng cách từng mẫu dữ liệu đến từng tâm cụm, phân mẫu dữ liệu về cụm có khoảng cách nhỏ nhất

```
def kmeans_assign_labels(self, X, centroids):
        D = cdist(X, centroids)
        return np.argmin(D, axis = 1)
```

+ Hàm điều kiện dừng thuật toán: kiểm tra tâm lần lặp sau có bằng tâm lần lặp trước

```
def has_converged(self, centroids, new_centroids):
        return (set([tuple(a) for a in centroids]) == set([tuple(a) for a in new_centroids]))
```

+ Hàm cập nhật tâm cụm: tính tâm mới bằng cách lấy trung bình cộng các điểm trong cụm

```
def kmeans_update_centroids(self, X, labels):
        centroids = np.zeros((self.K, X.shape[1]))
        for k in range(self.K):
            Xk = X[labels == k, :]
            centroids[k,:] = np.mean(Xk, axis = 0)
        return centroids
```

+ Hàm huấn luyện mô hình

```
def fit(self, X):
# Khởi tạo tâm cụm
        self.centroids = [self.kmeans_init_centroids(X)]
        labels = []
        it = 0
        while True:
# Tính toán khoảng cách các mẫu dữ liệu đến các tâm, phân về cụm có khoảng cách nhỏ nhất
```

```
                    labels.append(self.kmeans_assign_labels(X,
self.centroids[-1]))

                # Cập nhật lại các tâm

                    new_centroids = self.kmeans_update_centroids(X,
labels[-1])

                # Kiểm tra tâm mới và tâm cũ có == nhau không

                # Nếu có thì kết thúc thuật toán

                    if self.has_converged(self.centroids[-1], new_centroids):

                        break

                # Thêm tâm cụm mới vào mảng centroids

                    self.centroids.append(new_centroids)

                    it += 1

                return (self.centroids, labels, it)
```

+ Hàm dự đoán điểm dữ liệu bất kỳ: tính toán khoảng cách các mẫu dữ liệu đến các tâm, phân về cụm có khoảng cách nhỏ nhất

```
    def predict(self, X_Test):

            labels = []

            labels.append(self.kmeans_assign_labels(X_Test,
self.centroids[-1]))

        return labels
```

- Đánh giá chất lượng mô hình qua quan sát các độ đo Silhouette và Davies-Bouldin.

+ davies_boudlin: 0.672308432300016
+ silhouette: 0.5001822850641424

- Giao diện người dùng để dự đoán nhãn của một mẫu mới.

Phần 3. Kết luận

- Hiểu hơn về phương pháp phân cụm trong học không giám sát

- Thực hiện phân cụm K - means phân cụm nhóm người mắc bệnh tim, qua đó giúp hỗ trợ trong việc đưa ra phác đồ điều trị hiệu quả

****

Tài liệu tham khảo

- Sách Meachin Learning cơ bản, Vũ Hữu Tiệp
- Bài giảng Meachin Learning, TS. Nguyễn Thị Kim Ngân