# 二次元データに対する k-means クラスタリングと Local Outlier Factor の C# 実装

大阪大学 工学部 電子情報学科 3 年
情報システム工学コース
08D23091 辻 孝弥

2025 年 4 月 29 日

## 目次

**概要**

最大 200 点の二次元データを入力とし，Local Outlier Factor (LOF) で外れ値を検出した後に k-means クラスタリングを連続実行する C# プログラムを作成した。本稿ではその実装要点と実験結果を報告する。加えて，入出力仕様，実行ログの要約，将来の改良方針を示す。

# 1 背景と目的

外れ値検出とクラスタリングはデータ解析の基盤技術であり，相互に強く影響しあう。本プログラムは，まず LOF で外れ値を排除し，その後 k-means を適用することでクラスタリング結果の安定性を高めることを目的とする。

# 2 手法

## 2.1 処理フロー

**入力** 二次元座標を格納した CSV ファイル（書式：`x,y`）。

**処理手順** 1. CSV の読み込み

2. LOF の計算（近傍数 $k = 10$）
3. LOF 値が 1.2 を超えた点を外れ値としてフラグ付け
4. k-means クラスタリング（クラスタ数はコマンドライン引数，既定値 3）
5. 結果を CSV に出力

**出力** `x, y, cluster_id, is_outlier`

## 2.2 Local Outlier Factor (LOF)

点 $A$ の LOF は次式で定義される：

$$\mathrm{LOF}_k(A) = \frac{\dfrac{1}{|N_k(A)|}\displaystyle\sum_{B \in N_k(A)} \mathrm{lrd}_k(B)}{\mathrm{lrd}_k(A)}, \tag{1}$$

ここで $N_k(A)$ は $A$ の $k$ 近傍集合，$|N_k(A)| = k$，$\mathrm{lrd}_k(\cdot)$ は局所到達可能密度 (Local Reachability Density) である。

## 2.3 k-means クラスタリング

初期重心はランダムに選択し，ユークリッド距離で各点を割り当て，重心を再計算する。この更新を

$$\max_j \left\| \mathbf{c}_j^{(t+1)} - \mathbf{c}_j^{(t)} \right\| < 10^{-8}$$

が成り立つか，または 1000 反復に到達するまで繰り返す。$\mathbf{c}_j^{(t)}$ は時刻 $t$ におけるクラスタ $j$ の重心であり，実装では配列 `centroid[j,*]` に対応する。**外れ値フラグが立った点は重心計算に含めない。**

**アルゴリズム 1** 外れ値除外付き重心計算（抜粋）

```
for i ← 0 to N − 1 do
  if label[i] == k and is_outlier[i] == false then
    sum_x += data[i, 0];
    sum_y += data[i, 1];
    count += 1;
  end if
end for
```

# 3 実装要点 (C#)

- 2 次元データは `double[,]` 配列で保持し，最大 200 行を想定。
- LOF と k-means は同一ソースファイルに実装し，外部ライブラリは不使用。
- 外れ値を除外して重心を再計算するコード断片をアルゴリズム 1 に示す。

# 4 実験設定

## 4.1 データセット

5 種の人工データセット（各 200 点）を使用した：`moon`，`crater`，`square`，`three_island`，`two_island`。

## 4.2 パラメータ

LOF の近傍数は $k = 10$，閾値は 1.2。k-means のクラスタ数は 3（任意の正の数: コマンドライン引数で指定）とし，収束判定の上限を 1000 反復に設定した。

# 5 結果

## 5.1 外れ値検出（moon データセット）

式 (1) に基づき LOF を計算した結果，値が 1.2 を超えた点は 7 つであった（表 1）。

## 5.2 k-means の収束状況

`moon` データセットでは，初期重心をランダムに設定した結果，13 反復で収束した。各反復で前述の 7 点は重心計算から除外された。

## 5.3 可視化結果

図 1，2，3 は人工データセットのクラスタリング結果である。これに関する考察は 6 節で行う。

表 1　外れ値一覧（LOF $> 1.2$）

| No. | 座標 $(x, y)$ | LOF 値 |
|---|---|---|
| 1 | (0.2082, 0.9957) | 1.3159 |
| 2 | (0.1641, 1.0172) | 1.2992 |
| 3 | (0.3043, 1.0289) | 1.2795 |
| 4 | (2.2251, 0.9604) | 1.2248 |
| 5 | (0.2126, 0.9937) | 1.3193 |
| 6 | (3.1527, 1.5360) | 1.2000 |
| 7 | (1.1774, 1.6019) | 1.2411 |

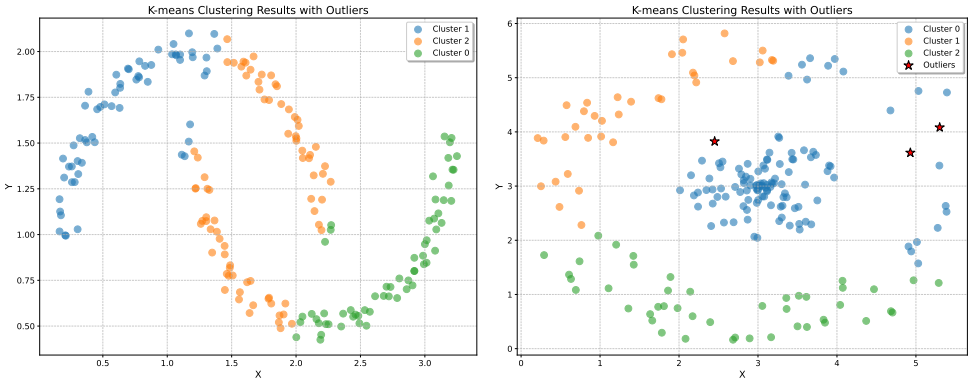


図 1　`moon`（左）と `crater`（右）のクラスタリング結果

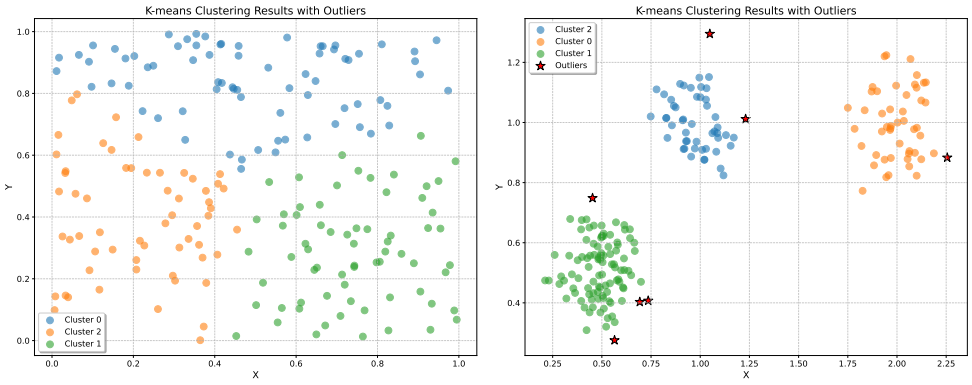


図 2　`square`（左）と `three_island`（右）のクラスタリング結果

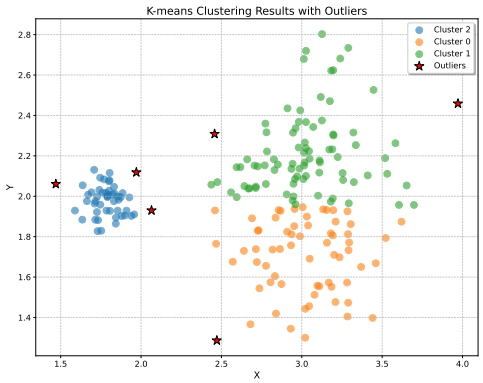


図 3　two_island データセットのクラスタリング結果

# 6　考察

## 6.1　外れ値除去の効果

### 6.1.1　結論

LOF により外れ値を事前に取り除くことで，各クラスタの重心が極端な点に引きずられず，分割が安定した。特に square, three_island, two_island のような単純形状に対しては，再現性の高いクラスタリングが得られた。

### 6.1.2　効果の検証

外れ値除去の有無による差を確認するため，two_island データに人工外れ値を追加した。外れ値を除外せずにクラスタリングした結果を図 4 に示す。初期重心の選択により 2 通りの分割結果が生じ，いずれも正しいクラスタ構造を再現できていない。

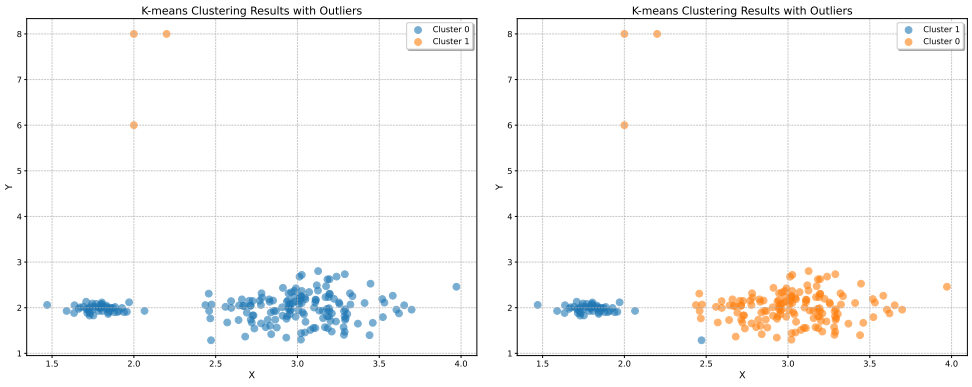


図 4　外れ値を除外しない場合の two_island クラスタリング例（2 パターン）

一方，外れ値を除外してからクラスタリングした結果を図 5 に示す。こちらでは 2 島構造が正確に分離されており，LOF による前処理の有効性が視覚的に確認できる。

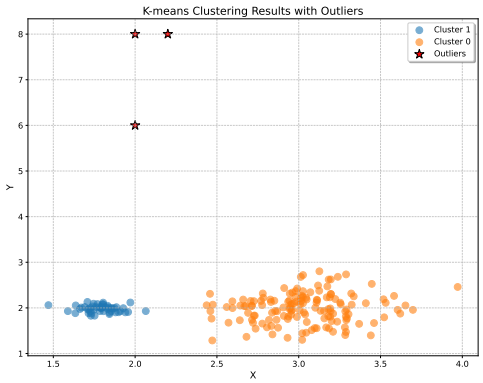


図 5　外れ値を除外した場合の `two_island` クラスタリング結果

#### 6.1.3　結果が分岐した要因

外れ値を除去しない場合に 2 通りの結果が生じたのは，初期重心が外れ値に設定されるか否かによる。ログを解析したところ，外れ値 3 点のいずれかが初期重心に選ばれたケースでは誤った分割が生じることが判明した。

### 6.2　k-means の性質と適用範囲

#### 6.2.1　k-means が適する場合

境界がほぼ直線的なデータ（例：`square`）では，k-means は高速かつ高精度にクラスタを分割できる。実装が簡易で計算コストも低いため，大規模データにおいて依然有力な選択肢である。

#### 6.2.2　k-means が苦手な場合

曲線状に点が分布するデータ（例：`moon`，`crater`）では精度が低下する。重心計算がユークリッド距離の平均に依存し，曲線をまたぐ点が同一クラスタに取り込まれるためである。

#### 6.2.3　手法選択の指針

- **形状が単純で直感的**: k-means を優先。
- **形状が複雑または非線形**: DBSCAN など密度ベース手法を検討。
- 外れ値が多い場合は，どの手法でも LOF などで事前に除去すると結果が安定する。

### 6.3　今後の改善点

曲線形状への対策として k-means++ による初期値改良や，ユークリッド以外の距離関数の採用が考えられる。また，DBSCAN と k-means を組み合わせたハイブリッド手法の検討も一案である。

## 7　まとめ

本稿で示した C# プログラムは，LOF による外れ値検出と k-means クラスタリングを連続実行し，200 点のデータに対して 13 反復で収束した。外れ値除去付き重心計算（配列 `centroid[j,*]`）がクラスタ分割の安定化に寄与することを確認した。

今後の課題として，LOF 閾値の最適化，自動初期化手法（k-means++），および非線形クラスタリング手法（DBSCAN）の比較実装を挙げる。