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

CSV 形式で与えられた最大 200 点の二次元データに対し，Local Outlier Factor (LOF) による外れ値検出と k-means クラスタリングを単一の C# プログラムで逐次実行した手順と結果を報告する．本稿では実装の要点，入力・出力仕様，および実験ログの要約を整理し，将来の改善方針を示す．

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根幹であり，両者は相互に影響を及ぼす．本プログラムは LOF によって外れ値を除いた後に k-means を適用することで，クラスタリング結果の安定化を図ることを目的としている．

2 手法

2.1 処理フロー

入力 二次元座標を格納した CSV（形式：`x,y`）

処理手順 1. CSV 読み込み

2. LOF の計算 ($k = 10$)
3. 閾値 1.2 を超える点を外れ値とフラグ付け
4. k-means クラスタリング（クラスタ数 k はコマンドライン引数，既定値 3）
5. 結果を CSV へ出力

出力 `x, y, cluster_id, is_outlier`

2.2 Local Outlier Factor (LOF)

スクリーンショットで示された定義に合わせ，点 A の LOF は次式で与えられる：

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

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抽出し，ユークリッド距離による割り当てと重心再計算を繰り返す．収束判定は

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

または 1000 反復に達した時点とした．ここで $\mathbf{c}_j^{(t)}$ は時刻 t におけるクラスタ j の重心 (centroid) であり，プログラム中の変数 `centroid[j, *]` に対応する。外れ値は重心計算から除外する．

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．最大 200 行を想定．
- LOF 計算と k-means は同一ファイルで実装し，外部依存ライブラリは使用していない．
- 外れ値を除外して重心を計算する際のコード片をアルゴリズム 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 (各 200 点) を使用した：moon，crater，square，three_island，two_island．

4.2 パラメータ

LOF は $k = 10$，閾値 1.2，k-means はクラスタ数 $k = 3$ とし，最大 1000 反復で収束を判定した．

5 結果

5.1 外れ値検出結果 (moon データセット)

式 (1) に従い計算した LOF が 1.2 を超えた点は 7 個であり，表 1 に示す．

5.2 k-means 収束状況

初期重心をランダムに選択した場合，moon データセットでは 13 反復で収束した．各反復で外れ値 7 点は重心計算から除外された．

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

5.3　可視化結果

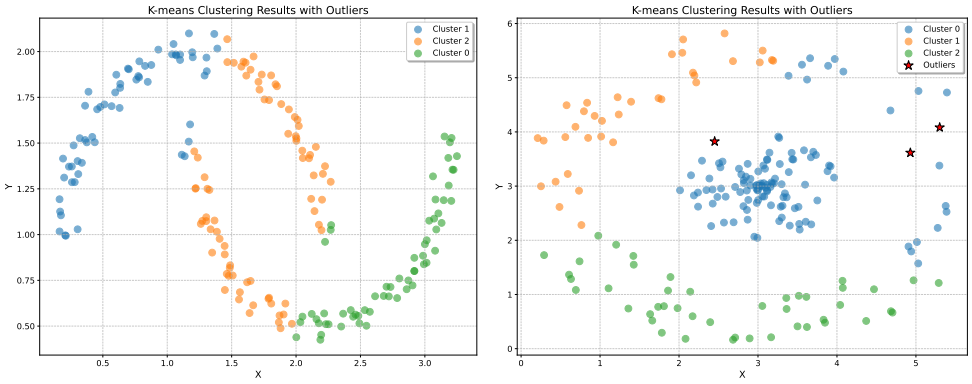

図 1　moon（左）と crater（右）のクラスタリング結果

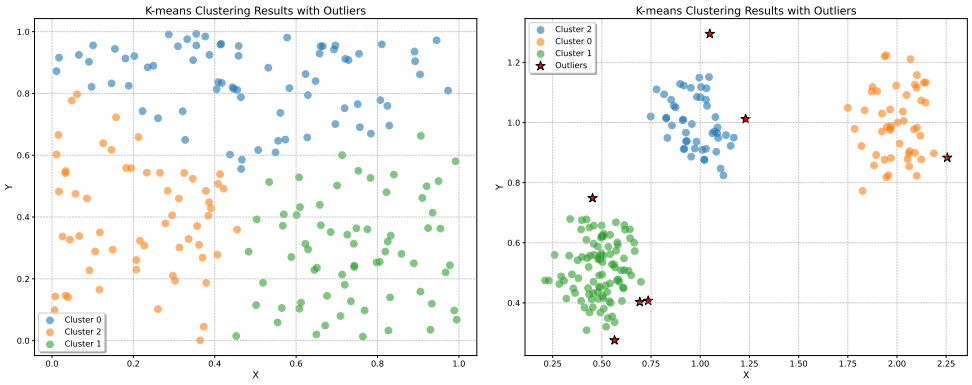

図 2　square（左）と three_island（右）のクラスタリング結果

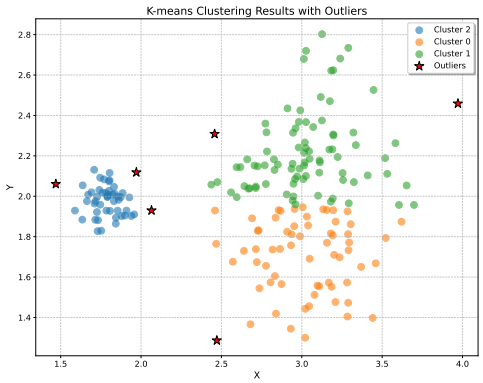

図 3　two_island データセットのクラスタリング結果

6　考察

外れ値除去の効果

LOF で外れ値を先に取り除くことで，各クラスタの重心が極端な点に引っ張られず，分割が安定した。特に `square` や `three_island` のような単純形状では，クラスタの境界が滑らかになり，再現性が向上した。

k-means がうまく働く場合

境界がおおむね直線的なデータ（`square` など）では，k-means が高速かつ正確にクラスタを分けられた。実装が簡単で計算量も小さいため，データ数が多いときは依然として有力な選択肢になる。

k-means が苦手な場合

`moon` や `crater` のように，点が曲線状に並ぶデータでは精度が落ちた。これは，重心計算がユークリッド距離の平均だけに依存し，曲線をまたぐ点を同じクラスタに入れてしまうためである。

手法選択基準の提案

- **データの形が直感的に分かる** ⇒ k-means が手軽で速い。
- **形が複雑／非線形** ⇒ DBSCAN など密度ベース手法が安全。
- 外れ値が多い場合は，どの手法でも事前に LOF などで除去しておくと結果が安定する。

今後の改善点

曲線形状への対策としては，k-means++ による初期値改良や，距離関数をユークリッド以外に変える方法がある。また，DBSCAN と k-means を組み合わせたハイブリッド法なども検討すると面白いのではないかと思う。

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 反復で収束したことを確認した。外れ値除去付き重心計算（変数 extttcentroid[j,*] が対応）がクラスタ分割の安定化に寄与することを示した。

今後は 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) の比較実装などを課題とする。