Partial Distance Fuzzy C-Means

Koki Kitamori

2022 年 11 月 20 日

1　欠測値を含むデータへの FCM 法の適用

部分的距離戦略を用いて，欠測値を含むデータに対して FCM 法を適用した．
最小化する目的関数は次の式で表現される．

$$J_{fcm} = \sum_{c=1}^{C}\sum_{i=1}^{n}\sum_{j=1}^{m} h_{ij} u_{ci}^{\theta}(x_{ij} - b_{cj})^2 \tag{1}$$

ここで，h_{ij} は観測値 x_{ij} の有無を表す 2 値変量で観測された場合に 1，そうでない場合に 0 の値をとる．クラスタ中心とファジィメンバシップ値の更新式は次のようになる．$\circ$ はアダマール積を表す．

$$\boldsymbol{b}_c = \frac{\sum_{i=1}^{n} u_{ci}^{\theta}(\boldsymbol{x}_i \circ \boldsymbol{h}_i)}{\sum_{i=1}^{n} u_{ci}^{\theta}} \tag{2}$$

$$u_{ci} = \left[\sum_{l=1}^{c}\left(\frac{\|\boldsymbol{x}_i \circ \boldsymbol{h}_i - \boldsymbol{b}_c\|^2}{\|\boldsymbol{x}_i \circ \boldsymbol{h}_i - \boldsymbol{b}_l\|^2}\right)^{\frac{1}{\theta-1}}\right]^{-1} \tag{3}$$

[部分的距離戦略を用いた FCM 法のアルゴリズム]

1. 要素が h_{ij} の行列 H を作る．
2. ファジィメンバシップ値 u_{ci} をランダムに定める．
3. クラスタ中心の更新とファジィメンバシップ値の更新を収束するまで繰り返す．

クラスタリング結果は以下の 2 つの指標を用いて比較した．

1. Bezdek の partition coeffcient (V_{PC})

$$V_{PC} = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^2 \tag{4}$$

2. Dave の modified partition coeffcient (V_{MPC})

$$V_{MPC} = 1 - \frac{c}{c-1}(1 - V_{PC}) \tag{5}$$

2　数値実験

UCI Machine Learning Repository から 3 クラス 13 変量，178 個体からなる wine データセットを用いた．欠測値を含むデータはランダムに 20 個の観測値を欠落させて作成した．このレポートでは，欠測値を含まないデータでのクラスタリング結果と部分的距離戦略を用いた FCM 法でのクラスタリング結果の間で妥当性尺度の値にどのような差が出るかを実験した．ファジィ度は 2.0，クラスタ数は 2〜6 とした．

3　実験結果

[欠測値がない場合]

```
    |     V_PC          |   V_MPC
 C=2| 0.8760833757248748| 0.7521667514497496
 C=3| 0.7909398594655069| 0.6864097891982603
 C=4| 0.7829819980280049| 0.71064266403734
 C=5| 0.7440197072466368| 0.680024634058296
 C=6| 0.7464268182671943| 0.6957121819206331
```

[欠測値を処理した場合]

```
    |     V_PC          |   V_MPC
 C=2| 0.8760395425126468| 0.7520790850252936
 C=3| 0.7907883399189741| 0.6861825098784612
 C=4| 0.782683214526157 | 0.710244286034876
 C=5| 0.7434561878650163| 0.6793202348312704
 C=6| 0.7455635445927241| 0.6946762535112689
```

上記の実験結果から結束値をないデータでクラスタリングした場合と，欠測値 20 個含むデータに対して部分的距離戦略を用いた FCM 法でクラスタリングした場合ほとんど指標の低下がみられなかった．よってこの方法は欠測値を含む場合でのファジィクラスタリングに有効であったと考えられる．

4　ソースコード

ソースコード 1: PartialDistanceFCM.jl

```
1  using CSV, DataFrames
2  using Clustering, StatsBase, Distances
3
4  # 欠測値がない場合
5  # wine.csvは欠測値がないデータ
6  data = Matrix(CSV.read("wine.csv", header=false, DataFrame))[:, 1:end-1]
7  # クラスタ数Cを2から6としてFCM法を実行
8  # 評価指標としてBezdekのpartition coefficient（V_PC）とDaveのmodification of the V_PC(
       V_MPC)を用いた.
9  for C in 2:6
10     result = fuzzy_cmeans(data', C, 2.0, tol=1e-5, maxiter=1000)
11     w = result.weights
12     vpc = sum(w .^ 2) / length(data[:, end]) # V_PC = \frac{1}{n}\sum_{i=1}^{c}\sum_{j
           =1}^{n}u_{ij}^2
13     vmpc = 1 - (C / (C - 1)) * (1 - vpc) # 1-\frac{c}{c-1}(1-V_PC)
14     C == 2 && println("[欠測値がない場合]\n    |     V_PC          ,   V_MPC")
15     println("C=$C| $vpc, $vmpc")
16 end
17
18 # 欠測値がある場合
19 function update_centers!(centers, data, weights, fuzziness, H)
20     nrows, ncols = size(weights)
21     T = eltype(centers)
22     for j in 1:ncols
23         num = zeros(T, size(data, 1))
24         den = zero(T)
25         for i in 1:nrows
26             δm = weights[i, j]^fuzziness
27             num += δm * (data[:, i] .* H[:, i])
28             den += δm
29         end
30         centers[:, j] = num / den
31     end
```

```
end

function update_weights!(weights, data, centers, fuzziness, dist_metric, H)
    pow = 2.0 / (fuzziness - 1)
    nrows, ncols = size(weights)
    dists = pairwise(dist_metric, data .* H, centers, dims=2)
    for i in 1:nrows
        for j in 1:ncols
            den = 0.0
            for k in 1:ncols
                den += (dists[i, j] / dists[i, k])^pow
            end
            weights[i, j] = 1.0 / den
        end
    end
end

# wine2.csvは欠測値を20個含むデータ
data = Matrix(CSV.read("wine2.csv", header=false, DataFrame))[:, 1:end-1]

H = similar(data)
for j in eachindex(data)
    ismissing(data[j]) ? H[j] = 0.0 : H[j] = 1.0
end

#  この値を使うことはないけど適当な数値を入れとかないとエラーになるのでなんか入れておく
replace!(data, missing => 99999.9)

fuzziness = 2.0
maxiter = 1000
tol = 1e-5

for C in 2:6
    δ = Inf
    nrows, ncols = size(data')
    iter = 0

    # 初期化
    weights = rand(Float64, ncols, C)
    weights ./= sum(weights, dims=2)
    centers = zeros(nrows, C)
    prev_centers = identity.(centers)

    while iter < maxiter && δ > tol
        update_centers!(centers, data', weights, fuzziness, H')
        update_weights!(weights, data', centers, fuzziness, Euclidean(), H')
        δ = maximum(colwise(Euclidean(), prev_centers, centers))
        copyto!(prev_centers, centers)
        iter += 1
    end
    # V_PC = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n}u_{ij}^2
    vpc = sum(weights .^ 2) / length(data[:, end])

    # 1-\frac{c}{c-1}(1-V_PC)
    vmpc = 1 - (C / (C - 1)) * (1 - vpc)
    C == 2 && println("\n[欠測値を処理した場合]\n   |     V_PC          |   V_MPC")
    println("C=$C| $vpc, $vmpc")
end
```