VII-3-3. K-means 法による非階層的クラスター分析

VII-3-3-1. K-means 法の考え方

階層的クラスター分析は距離や類似度の近い要素を結合して、下から階層構造を作り上げていくクラスター分析です。この分析法の欠点はサンプルサイズが大きくなると幾何級数的に計算量が増えて解析に時間がかかることです。非階層的クラスター分析ではクラスターの結合の結果、クラスター内の要素の分布の中心（重心）が決まります。非階層クラスター分析では、あらかじめいくつのクラスターに分けるかを決めておき、クラスの分布を代表する中心を仮説的に与えます。それぞれのデータの所属するクラスは、その点から最も近いデータの代表点のクラスとします。クラスに所属するデータが決まればその重心を求めることが出来ます。この重心を新たにクラスの分布の代表点に更新します。すると、更新された代表点と所属するデータ点の距離の二乗総和（SS）は、更新される前の代表点とデータ点の距離の２乗総和よりも小さくなるはずです。それぞれのクラスの代表点が動いたので、全体として、クラスの代表点とすべてのデータ点の距離が変わります。ここまでが一つのステップです。代表点とデータの距離が変わったので、各データ点から各代表点までの距離を計算し、最も近い代表点のクラスをデータ点が所属するクラスに更新し、クラスに所属するデータ点の重心を求め、再びこれを更新された各クラスの代表点とします。更新された代表点と所属するデータ点の距離の二乗総和（SS）は、更新される前の代表点とデータ点の距離の２乗総和よりも小さくなります。これが次のステップです。このステップを重心が移動しなくなれば、最も距離の分散が小さいクラス分けになるはずです。この方法を K-means 法と言います。データにはランダムに起きるデータの粗密がありますから、この方法だと、部分的な極小に落ち込んで、最小の分散にならない場合があり得ます。これは初期の代表値の与え方によります。初期値の与え方など K-means 法には様々な改良が加えられており、部分最適に陥ることは少なくなっていますが、初期値を変えて K-means 法をしたり、ランダムに抽出したいくつかのサンプルで安定した結果が得られることを確認する必要あります。

VII-3-3-2. K-means 法の計算

次のようにP次元のデータがあります。

$$\boldsymbol{x} = \begin{pmatrix} x_{11} & \cdots & x_{1P} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NP} \end{pmatrix}$$

$$\boldsymbol{x}_n = \begin{pmatrix} x_{n1} & \cdots & x_{nP} \end{pmatrix}$$

N: *サンプルサイズ*

ここに初期値、$\boldsymbol{\mu}$を与えます。

$$\boldsymbol{\mu}_0 = \begin{pmatrix} \mu_{011} & \cdots & \mu_{01} \\ \vdots & \ddots & \vdots \\ \mu_{0K1} & \cdots & \mu_{0KP} \end{pmatrix}$$

$$\boldsymbol{\mu}_{0k} = \begin{pmatrix} \mu_{0k1} & \cdots & \mu_{0kP} \end{pmatrix}$$

K: *クラスの数*

各サンプルからクラスの代表点までの距離の二乗は

$$\|\boldsymbol{x}_n - \boldsymbol{\mu}_{0k}\|^2 = \sum_{p=1}^{P} \left(x_{np} - \mu_{0kp}\right)^2$$

一方、それぞれのデータ点と代表点の距離の 2 乗が出たので、それぞれのデータ点について、各クラスの代表点までの距離を比較し、最小の距離となるクラスをそのデータ数が所属するクラスとして、その結果を One-hot エンコーディングで表します。例えば k=1 のクラスに属するのであれば、

$$\boldsymbol{r}_{1n} = (1 \quad 0 \quad \cdots \quad 0)$$

このエンコーディング全体を行列で表すと、

$$\boldsymbol{R}_1 = \begin{pmatrix} \boldsymbol{r}_{11} \\ \vdots \\ \boldsymbol{r}_{1n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & 0 \end{pmatrix}_{n \times k}$$

のような形になります。クラスkに属するデータ点のサンプルサイズは

$$N_{1k} = \sum_{n=1}^{N} r_{1n}$$

クラスkに属するデータ点の重心は、

$$\boldsymbol{\mu}_{1k} = (\mu_{1k1} \quad \cdots \quad \mu_{1kP})$$

$$= \frac{1}{N_{1k}} \sum_{n=1}^{N} r_{1nk} \boldsymbol{x}_n = \frac{1}{N_{1k}} \left(\sum_{n=1}^{N} r_{1n} \; x_{n1} \quad \cdots \quad \sum_{n=1}^{N} r_{1nk} x_{nP} \right)$$

つまり、個々の座標について細かく書くと

$$\mu_{1kp} = \frac{1}{N_{1k}} \sum_{n=1}^{N} r_{1nk} x_{np}$$

となります。

これで、データ$\boldsymbol{x}_n$について更新された代表点（重心）までの距離が決まります。このすべての距離の二乗和の和を歪み尺度(distortion measure)と言います。

$$J_1 = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{1kn} \|\boldsymbol{x}_n - \boldsymbol{\mu}_{1k}\|^2 = \sum_{k=1}^{K} \sum_{n=1}^{N} \sum_{p=1}^{P} r_{1kn} \left(x_{np} - \mu_{1kp}\right)^2$$

初めに仮説的に代表点を与えた時の歪み尺度J_0は

$$J_0 = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{1kn} \|\boldsymbol{x}_n - \boldsymbol{\mu}_{0k}\|^2 = \sum_{k=1}^{K} \sum_{n=1}^{N} \sum_{p=1}^{P} r_{1kn} \left(x_{np} - \mu_{0kp}\right)^2$$

となりますが、$\boldsymbol{\mu}_{1k}$はクラスkの重心ですから、

$$J_1 \leq J_0$$

です。歪み尺度はこの分析の目的変数で、J を最小化することが K-means 法によるクラスター分析のゴールです。これで一つのサイクルが終わり、次のサイクルで**$\boldsymbol{\mu}_1$を各クラスの代表点としてすべてのデータ**との距離を計算し、それぞれのクラスに含まれるデータを選び直して、$\boldsymbol{R}_2$を更新します。つまり、一つのサイクルに$\boldsymbol{\mu}_k$で$\boldsymbol{R}_{k+1}$を更新するステップと$\boldsymbol{R}_{k+1}$で$\boldsymbol{\mu}_{k+1}$を更新するステップがあり、そのステップでJ_{k+1}更新し、次のステップでは $\boldsymbol{\mu}_{k+1} \rightarrow \boldsymbol{\mu}_k$となるということです、こうして、$\boldsymbol{R}_{k+1}$, $\boldsymbol{\mu}_{k+1}$が動かなくなり、J_{k+1}が最小化されたところで、$\boldsymbol{R}_{k+1}$を採取的なクラス分けとして確定します。手順が長いので複雑に思うかもしれませんが、極めた単純な発想です。部分的な変化を積み重ねているだけですから、当然、初期条件の与え方によっては、部分最適に落ち込んでしまうこともありますし、初期条件の与え方によって、結果が不安定になることもあります。

VII-3-3-3. python, scikit-learn.k-means を使ったプログラム

実際に非階層的なクラスター分析をするときは、R か Python か何らかの言語を使ってコンピュータで計算することになります。ここでは、Python を使った計算プログラムのコードリストの例を挙げておきます。分析準備、データ読み込み、主成分分析は階層的クラスター分析で示したコードリスト VII-3-2-i を使うことにして、もし必要ならば VII-3-2-ii を使って training data と test data に分けてください。データの分布が知りたければ、VII-3-2-iii または VII-3-2-iv を使って分布図を確認してください。ここでは、k-means 法の実行の部分だけについて説明します。このコードリストを、このブログのカテゴリー「やさしい水産学（自習室）」の「参考資料」-「python」に「VII-3-3.非階層的クラスター分析（Kmeans 法）」としてあげておきました。コピーして動かしてみてください。

VII-3-3-i. Scikit-learn を使った非階層的クラスター分析（元データと主成分得点）

```
#scikit-learn を使って、k-means 法で非階層的クラスター分析をする。
#[A]必要な library の読み込み
from scipy import stats
from sklearn.cluster import KMeans
#[B]クラスの数を決める
C=5
#散布図に使う変数を決める
x=1
y=2
x0=x-1
y0=y-1
#グラフの範囲を決める
x_range=[-2,2] #項目 1 の範囲
y_range=[-2,2] #項目 2 の範囲
#[C]実行
#[C1]元データでクラスター分析
sklearn.cluster.KMeans(n_clusters=C)
pred = KMeans(n_clusters=C).fit_predict(X)
N,nn=X.shape
TE=np.zeros((N,1))
```

```
for n in range (N):
    TE[n]=pred[n]+1
#[C2]主成分得点でクラスター分析
pred = KMeans(n_clusters=C).fit_predict(PC)
N,nn=PC.shape
TM=np.zeros((N,1))
for n in range (N):
    TM[n]=pred[n]+1
plt.figure(1,figsize=(8,3.7))
plt.subplot(1,2,1)
show_data1(X,TE)
plt.xlim(x_range)
plt.ylim(y_range)
plt.xlabel("X"+str(x))
plt.ylabel("X"+str(y))
plt.title('original data ')
plt.subplot(1,2,2)
show_data1(X,TM)
plt.xlim(x_range)
plt.ylim(y_range)
plt.xlabel("X"+str(x))
plt.ylabel("X"+str(y))
plt.title('Principle component')
plt.show()
```

[

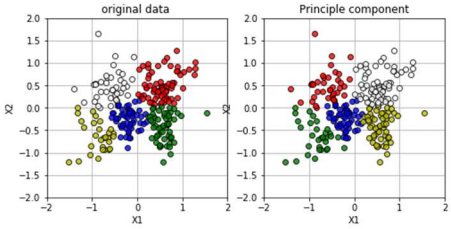

図 146.Sample10 を Kneans 法で非階層的クラスター分析した結果

*A]*必要な library の読み込み

[B]クラスの数、散布図を描く変数、グラフの範囲を決定

[C]計算の実行。[C1]元データでクラスター分析。[C2]主成分得点でクラスター分析。

sklearn.cluster.KMeans(n_clusters=C)で、クラスター数を決定していますが、この時、初期値の与え方な素も指定できます。デフォルトは kmeans を改良した kmeans++で初期値を与えています。Kmeans 法でやるならば、init="random"と指定したり、初期値を座標で指定することもできます。Sample10 を Kneans 法で非階層的クラスター分析した結果が図 145 です。階層的クラスター分析で予想したように、中央に存在するクラス明瞭に捉えています。

VII-3-3-4. scikit-learn を使った k-means 法の問題点

sklearn.cluster.Kmeans の kmeans を使うと、極めて短時間に非階層的クラスター分析が出来ます。しかも、簡単にプログラムが書けます。これは強力な分析ツールです。計算速度を考えると K-means の手法は魅力的です。コンピュータを使った計算では、最適化しようと

するいくつかの変数が複雑に絡み合って変数分離できず微分方程式が解けない場合に、しばしばEMアルゴリズムという計算法が使われます。複数の係数を変数とするある関数を、仮想的に与えた変数から期待値として求め（E:expectation)、その期待値を使って元の係数を最適化します（M:maximization)。これは繰り返せば、期待値として求めた変数もやがて最適化されるというのが、EMアルゴリズムです。EMアルゴリズムのM stepでは微分方程式を解くということが行われます。ですから、時には長い計算ステップになって計算時間がかかります。実はK-means もEMアルゴリズムを使っているのですが、K-means法がEMアルゴリズムのように見えないのは、この作業をクラスに属する点の重心を求めるところでやってしまっているからです。微分方程式を解くという作業がいらないのです。これはユークリッド距離の持つ性質と言えます。

多次元のユークリッド距離について考えてみます。

点$X(x_1 \quad \cdots \quad x_p)$と点$M(m_1 \quad \cdots \quad m_p)$の距離$d\langle A,M\rangle$は、

$$d\langle X,M\rangle = \sqrt{(x_1-m_1)^2+\cdots+\left(x_p-m_p\right)^2}$$

$$d\langle X,M\rangle^2 = (x_1-m_1)^2+\cdots+\left(x_p-m_p\right)^2$$

同様に、

$$d\langle X_i,M\rangle = \sqrt{(x_{1i}-m_1)^2+\cdots+\left(x_{pi}-m_p\right)^2}$$

$$d\langle X_i,M\rangle^2 = (x_{1i}-m_1)^2+\cdots+\left(x_{pi}-m_p\right)^2$$

$$i=1,\cdots N$$

として、距離の2乗総和を求めると$SS=\sum_{i=1}^{n} d\langle X_i,M\rangle^2$ですから、

標本集団の分散は

$$\sigma^2 = \frac{SS}{N} = \frac{1}{N}\sum_{i=1}^{n} d\langle X_i,M\rangle^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{p}\left(x_{ji}-m_j\right)^2$$

となります、σ^2を最小化するMを求めるために、SSの極値を与えるMを求めます。

$$\frac{\partial SS}{\partial m_i} = -2\sum_{i=1}^{N}\left(x_{ij}-m_i\right) = -2\left(\sum_{i=1}^{N} x_{ij}+Nm_i\right) = 0$$

$$\sum_{i=1}^{N} x_{ij}+Nm_i = 0$$

$$m_i = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$$

したがって、最小の分散を与える中心点は平均値（重心）であるということになります。つまり重心を移動すれば、分散は小さくなって、確率が最大化されるのです。

線形結合によって、軸を変換した場合も同じです。

$$X = (x_1 \quad \cdots \quad x_p)$$
$$\bar{X} = (t_1 x_1 \quad \cdots \quad t_p x_p)$$
$$d\langle \bar{X}, \bar{M} \rangle = \sqrt{(t_1 x_1 - \overline{m_1})^2 + \cdots + \left(t_p x_p - \overline{m_p}\right)^2}$$
$$d\langle \bar{X}, \bar{M} \rangle^2 = (t_1 x_1 - \overline{m_1})^2 + \cdots + \left(t_p x_p - \overline{m_p}\right)$$

同様に、

$$d\langle \bar{X}_\iota, \bar{M} \rangle = \sqrt{(t_1 x_{1i} - \overline{m_1})^2 + \cdots + \left(t_p x_{pi} - \overline{m_p}\right)}$$
$$d\langle \bar{X}_\iota, \bar{M} \rangle^2 = (t_1 x_{1i} - \overline{m_1})^2 + \cdots + \left(t_p x_{pi} - \overline{m_p}\right)$$
$$i = 1, \cdots N$$

として、距離の 2 乗総和を求めると$SS = \sum_{i=1}^{n} d\langle \bar{X}_\iota, \bar{M} \rangle^2$

標本集団の分散は

$$\sigma^2 = \frac{SS}{N} = \frac{1}{N}\sum_{i=1}^{n} d\langle \bar{X}_\iota, \bar{M} \rangle^2$$
$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{p}\left(t_j x_{ji} - \overline{m_j}\right)^2$$

SS を最小化する M を求めるために、極値を与える M を求める。

$$\frac{\partial SS}{\partial m_j} = -2\sum_{i=1}^{N}\left(t_j x_{ij} - \overline{m_j}\right) = -2\left(\sum_{i=1}^{N} t_j x_{ij} + N\overline{m_j}\right) = 0$$
$$\sum_{i=1}^{N} t_j x_{ij} + N\overline{m_j} = 0$$
$$\overline{m_j} = t_j \frac{1}{N}\sum_{i=1}^{N} t_j x_{ij} = t_j m_j = t_j m_j$$

マハラノビスの距離はユークリッド距離の線形結合による変形だから、マハラノビスの距離を取った場合にも中心点を重心にとることは分散の最小化であり確率の最大化になります。せめて、マハラノビスの距離ぐらいには使えるようにしてもらいたいという気がしますが、sklearn.cluster.Kmeans で扱える距離(非類似度)はユークリッド距離だけなのです。クラスター分析は教師なし学習の典型であらかじめ正解がわかっているわけではありません。様々な角度からクラスター分析をして、潜在的なクラスを見つけだして、その意味を考えるためにやっているのです。ですから、様々な視点で類似性を考える必要があります。ユークリッド距離しか使えないというのは致命的な欠点だと思います。