付録 C　クラスター分析について

C. 1　はじめに

類推マトリックスを分析する際、今回はクラスター分析を利用した。この付録では、この解析手法を使ってどのようにマトリックスを料理したかについて説明する。

C. 1. 1　類推マトリックスと非類推度マトリックス

今回利用したマトリックスには、類推度と定義した値が格納されている。それは値が大きければ大きいほど類推しやすいというものである。

階層的クラスター分析では、非類似度という指標を用いて分析を行うが、今回、マトリックスに収められた類推度の逆数が非類似度に相当しているので、まずはマトリックスの要素個々に対し、逆数を求めるところから分析は始まる。ただし、本モデルでは類推不可能である場合で負になる可能性を抱えているので、確実に負にならない様よう値を換算してから、逆数をとっている。

こうして用意した**非類似度マトリックス**をクラスター分析に掛けるわけだが、今回はその分析手法うちの最短距離法、郡平均法を利用したので、それらふたつの手法について続いて説明する。

C. 2　クラスター分析

クラスター分析では、デンドログラムと呼ばれる樹状の分類構造を構成することを目的とする。

以下では、本研究で利用した最短距離法と群平均法によるデンドログラムの作りかたを簡単に説明する。

C. 2. 1　最短距離法

あるクラスタuとそれとは別のクラスタvを統合して、新しいクラスタwを作るとする。

このとき、新しく統合してできたクラスタwとは別の任意のクラスタtとのあいだの非類似度D_{wt}を、統合する前のクラスタu、vとクラスタtとの非類似度D_{ut}、D_{vt}を用いて、

$$D_{wt} = \min(D_{ut}, D_{vt}) \quad (C.1)$$

と定義する。この定義を逐次適用すると、2つのクラスタ間の非類似度は、それぞれのクラスタに含まれる対象の対の中で、最も類似度の高い対の間での非類似度になる。このことから、この方法は最短距離法と呼ばれる。この方法では1つでも近い対象があるクラスタは、どんどん統合されてい行くので、長い帯状のクラスタができやすい。（近代科学社「多変量解析入門」木下栄蔵　著より）

C. 2. 2　群平均法

最短距離法（ならびに最長距離法で）は、各クラスタ間の非類似度は、それらに含まれる対象の対の非類似度の中の極端な値（最大あるいは最小）により決められた。一方、各クラスタ間の非類似度を、それらに含まれる対象間の非類似度の平均的な値で決め付けようとする考え方もある。そのような考え方に基づいた方法のひとつに群平均法がある。それは、クラスタuとクラスタvを統合してできた新しいクラスタwに含まれる対象と、別の任意のクラスタtに含まれる対象とにおいて、総ての可能な対の非類似度の平均により、このふたつのクラスタ間の非類似度を決めるものである。

クラスタu、クラスタv、クラスタtに含まれる対象の数をそれぞれn_u、n_v、n_tとすると、クラスタwとクラスタtの非類似度D_{wt}は

$$D_{wt} = \frac{n_u D_{ut} + n_v D_{vt}}{n_u + n_v} \quad (C.2)$$

となる（近代科学社「多変量解析入門」木下栄蔵　著より）。

C. 2. 3　実装例

実装するにあたっては、組み合わせ的方法（「パソコン統計解析ハンドブック　多変量解析篇」参考）と呼ばれる手法を通して行った。これは以下の式で非類似度を更新するものである。

$$D_{wt} = \alpha_u D_{ut} + \alpha_v D_{vt} + \beta D_{uv} + \gamma \left| D_{ut} - D_{vt} \right| \quad (C.3)$$

式C．3にあるα_u、α_v、β、γは下の表C．1のように、方法に応じて切り替える。

方　法	組み合わせ的方法のパラメータ			
	α_u	α_v	β	γ
最短距離法	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
群平均法	$\frac{n_u}{n_t}$	$\frac{n_v}{n_t}$	0	0

表　C．1　組み合わせ法のパラメータ

ここに示したものは、ふたつの手法に関してのみであるが、他にも、最長距離法、重心法、メジアン法、ウォード法などに関してもパラメータを変えるだけで実現できる。

以降では、参考程度に、デンドログラムのデータを作成する関数部を示す。

尚、以下に示したソースは、実際に用いたソースよりコメントを幾分追加している。また、改行を追加してなるべく見やすいように工夫した。

プログラム C. 1

```
//この手続きは、**plot_data;*m_sequence; にデータを格納するのが目的。
//-------------------------------------------------------------------------------
//**plot_data
// デンドログラムを出力する際必要となるデータ配列で、ある要素(クラスタ)と別の要素(クラスタ)のワード番号と
// そのふたつが結合する高さの3つの値が収められている。
// この配列は結合する高さによって昇順にソートされているので、次に示す m_sequence に従って
// ワード番号を配置したうえで、plot_data を順番に読み込みこみつつ線を書いていけば、目的のデンドログラム
// が得られるようになる。
//*m_sequence
// デンドログラムを出力する際、どういう順にワード番号を並べればよいかを与える配列。

void CAgent::set_dendrogram_data(int modify_mode)
{
   int count_x;
   int count_y;

   //クラスタの数を把握しておく変数。
   int clusters = m_words;

   double ds_buf;
   int w_buf0,w_buf1;

   int data_counter;

   //結合途中にあるクラスタを一時的にためておく場所
   int **dissim_buf;

   //組み合わせ的方法で利用されるパラメータ
   double ap,aq,b,c;

   //配列の用意
   set_new_handler(0);
   dissim_buf = new int*[2];
   for(count_x = 0; count_x < 2; count_x++){
      dissim_buf[count_x] = new int[m_words];
   }
   if (m_graph == NULL) {
      cout << "メモリが確保できませんでしたので終了しました. Exit 50¥n";
      exit(1);
   }

   //まずは非類似度行列を用意する。
   m_set_dissimilarity();

   //int *m_graph_elements の初期値をセット
```

```
for(count_x = 0; count_x < m_words; count_x++)
{
   m_graph_elements[count_x] = 1;
}

//int **m_graph の初期値をセット
for(count_x = 0; count_x < m_words; count_x++)
{
   m_graph[0][count_x] = count_x;
}

//double *m_plot_start の初期値をセット
for(count_x = 0; count_x < m_words; count_x++)
{
   m_plot_start[count_x] = 0;
}

//--------------------------------------------------------------------------------
// -1  1つずつの対象を構成単位とする n 個のクラスタから出発する。m_graph と m_graph_elements
//      の初期値が、その状態を反映している。
// -2  クラスタ間の非類似度行列参照し、最も類似性の高い2つのクラスタを融合して1つの
//    クラスタを作る。
// -3  クラスタ数が1になっていれば終了。そうでなければ次のステップに進む。
// -4  -2 で作られたクラスタと他のクラスタとの非類似度を計算して、非類似度行列を更新。
//    その後-2 に戻る。
//--------------------------------------------------------------------------------

data_counter = 0;

do{
   // まず、**m_graph と*m_graph_elements を参照し、そこから得られた word 番号を利用する事により
   // **m_dissimilarity の中にある最小の組合せをゲットする。
   //(なに言ってるんだか、さっぱりわかんないっすね)
   // m_graph[0][x]の[0][0]から[0][99]を走査していく。
   // その時、m_graph[0][x]に要素が格納されているか否かを
   //m_graph_elements[x]で確認できる様になっている
   // -2を実行する為、最も類似度の高いクラスタ同士を抽出する。(最左のものだけを抽出する)
   ds_buf = 100000; //あり得ない最小数。
   for(count_x = 0; count_x < m_words; count_x++){
      if(m_graph_elements[count_x] > 0){
         for(count_y = count_x + 1; count_y < m_words; count_y++){
            if(m_graph_elements[count_y] > 0){
               if(m_dissimilarity[m_graph[0][count_x]][m_graph[0][count_y]] < ds_buf){
                  //最小値の入替え
                  ds_buf = m_dissimilarity[m_graph[0][count_x]][m_graph[0][count_y]];
                  //最小値を呈した配列の組み合わせを入れておく
                  w_buf0 = m_graph[0][count_x];
                  w_buf1 = m_graph[0][count_y];
               }
            }
         }
      }
   }
   //書式に従って double **plot_data 配列にデータ(結合するふたつの要素番号(クラスタ)と結合高さ)を収める
```

```
        plot_data[data_counter][0] = w_buf0;  //一方のクラスタの最左のワード番号
        plot_data[data_counter][1] = w_buf1;  //もう一方のクラスタの最左のワード番号
        plot_data[data_counter][2] = m_dissimilarity[w_buf0][w_buf1];  //融合した時の高さ
        data_counter++; //次のデータのためにインクリメント

        //非類似度が最小の物を取り出したら、そのおのおの行列を更新する
        // ① 高さを変える
        m_plot_start[w_buf0] = m_plot_start[w_buf1] = m_dissimilarity[w_buf0][w_buf1];

        // ② m_dissimilarity の変更
        // - 1 まずは一時的に二つ分の配列を別の場所にコピーしておく（最左のものだけを計算する）
        for(count_x = 0; count_x < m_words; count_x++){
           dissim_buf[0][count_x] = m_dissimilarity[w_buf0][count_x];
           dissim_buf[1][count_x] = m_dissimilarity[w_buf1][count_x];
        }

        //組み合わせ的方法にたいするパラメータの指定。この関数に渡される引数がスイッチになっている。
        switch(modify_mode){
        //最短距離法
        case 1:
           ap = 0.5 ; aq = 0.5 ; b = 0 ; c = -0.5 ;
           break;
        //最長距離法
        case 2:
           ap = 0.5 ; aq = 0.5 ; b = 0 ; c = 0.5 ;
           break;
        //群平均法
        case 3:
           ap = ((double)m_graph_elements[w_buf0]/((double)m_graph_elements[w_buf0]+
              (double)m_graph_elements[w_buf1]));
           aq = ((double)m_graph_elements[w_buf1]/((double)m_graph_elements[w_buf0]+
              (double)m_graph_elements[w_buf1]));
           b = 0;c = 0;
           break;
        //重心法
        case 4:
           ap = ((double)m_graph_elements[w_buf0]/((double)m_graph_elements[w_buf0]+
              (double)m_graph_elements[w_buf1]));
           aq = ((double)m_graph_elements[w_buf1]/((double)m_graph_elements[w_buf0]+
              (double)m_graph_elements[w_buf1]));
           b = ((double)-m_graph_elements[w_buf0] * (double)m_graph_elements[w_buf1] /
            pow(((double)m_graph_elements[w_buf0] + (double)m_graph_elements[w_buf1]),2));
           c = 0;
           break;
        //メシアン法
        case 5:
           ap = 0.5; aq = 0.5; b = -0.25; c = 0;
           break;
        //(未完の)ウォード法
//      case 6:
//         ap = ; aq = ; b = ; c = ;
//         break;
        default:
           ap = 0.5 ; aq = 0.5 ; b = 0 ; c = -0.5 ;
```

```
        }

        // ③参照配列を書きかえる（最左のものだけを書きかえる）
        //  要は、組み合わせ的方法により、非類似度の書き換えを行う
        for(count_x = 0; count_x < m_words; count_x++)
        {
           if(m_graph_elements[count_x] > 0)
           {
              for(count_y = count_x + 1; count_y < m_words; count_y++)
              {
                 if(m_graph_elements[count_y] > 0)
                 {
                    //最小値の入替え
                    m_dissimilarity[w_buf0][count_x] =
                        m_dissimilarity[count_x][w_buf0] =
                       (int)((double)ap*(double)dissim_buf[0][count_x] +
                       (double)aq*(double)dissim_buf[1][count_x] +
                       (double)b*(double)dissim_buf[0][w_buf1] +
                       (double)c*(double)fabs((double)dissim_buf[0][count_x]-
                       (double)dissim_buf[1][count_x]));   //計算式
                 }
              }
           }
        }

        // - 1 グラフの要素位置を変える
        for(count_x = 0; count_x < m_graph_elements[w_buf0]; count_x++) ;
        for(count_y = 0; count_y < m_graph_elements[w_buf1]; count_y++)
        {
           m_graph[count_x][w_buf0] = m_graph[count_y][w_buf1];
           count_x++;
        }

        // - 2 要素数を変更
        m_graph_elements[w_buf0] = count_x;
        m_graph_elements[w_buf1] = 0;

        // ④クラスタ数を減らす
        clusters--;

     }while (clusters > 1); // -3

     //*m_sequence に m_graph[y][x](> 0)の配列を一行コピーする。
     //
     for(count_x = 0; count_x < m_words ; count_x++)sequence[count_x] = m_graph[count_x][0];

     delete[] dissim_buf;
}
```

参考文献

木下栄蔵(1995)『わかりやすい数学モデルによる多変量解析入門』近代科学社

田中豊　垂水共之　脇本和昌　編『パソコン統計解析ハンドブック　多変量解析篇』