

Rと自己組織化マップ

1. 自己組織化マップとは

自己組織化マップ (SOM: Self-Organizing Map) は、コホネン(T. Kohonen)[1]により提案された教師なしのニューラルネットワークアルゴリズムで、高次元データを2次元平面上へ非線形写像するデータ解析方法である。

自己組織化マップは、入力層と出力層により構成された2層のニューラルネットワークである。出力層は競合層とも呼ばれている。

今、入力層には分析対象となる個体 j の特徴ベクトルを $\mathbf{x}_j(x_{j1}, x_{j2}, \dots, x_{jn})$ 、出力層には $k(i=1, 2, \dots, k)$ 個のユニットがあるとする。図1の(a)で示すように、出力層における任意の1つのユニットは、入力層における特徴ベクトルのすべての変数とリンクしている。初期段階では乱数により各変数との間に図1の(b)に示すように重み $\mathbf{m}_i(m_{i1}, m_{i2}, \dots, m_{in})$ が付けられている。

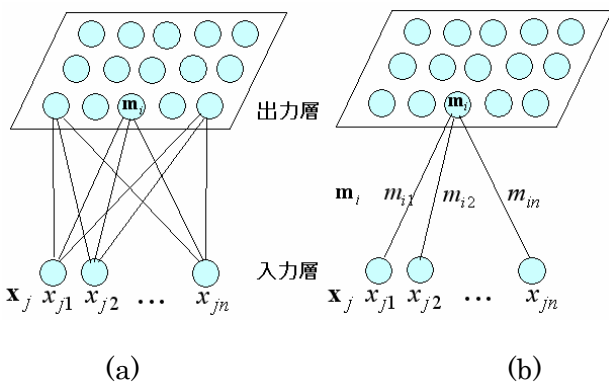


図1 自己組織化マップの基本構造

SOM のアルゴリズム :

- (1) 入力 \mathbf{x}_j と出力層におけるすべてのユニットの中から、最も類似しているユニット

\mathbf{m}_c を探し出し、そのユニットを勝者とする。

$$\|\mathbf{x}_j - \mathbf{m}_c\| = \min_i \{ \|\mathbf{x}_j - \mathbf{m}_i\| \}$$

- (2) 勝者のユニットおよびその近傍のユニットの重みベクトル \mathbf{m}_i を更新する。更新は下記の式によって行う。

$$\mathbf{m}_i(t+1) = \begin{cases} \mathbf{m}_i(t) + h_{ci}(t) [\mathbf{x}_j(t) - \mathbf{m}_i(t)] & i \in N_c \\ \mathbf{m}_i(t) & i \notin N_c \end{cases}$$

$$h_{ci}(t) = \alpha(t) \exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right)$$

式の中の $h_{ci}(t)$ は近傍関数で、ユニット c とその近傍のユニット i の近さによって \mathbf{x}_j の影響を調整する。式 $h_{ci}(t)$ の中の $\alpha(t)$ は学習率係数で、 r_c と r_i はユニット c と i の2次元上の座標ベクトルである。 $\sigma^2(t)$ はユニット c の近傍領域 N_c の半径を調整する関数である。 $\alpha(t)$ 、 $\sigma^2(t)$ は学習回数(あるいは時間)を変数とする単調減少関数である。学習回数を変数とする最も簡単な単調減少関数は $1 - \frac{t}{T}$ である。この t は学習回数(あるいは時間、1、2、3、 \dots 、 T)、 T は事前に設定した学習の総回数である。

- (3) すべての入力特徴ベクトル ($j=1, 2, \dots, m$) に対して(1)~(2)を繰り返す。

SOM は上記のアルゴリズムにより、多次元の空間上の分類対象を2次元平面に射影する。SOMの結果は、出力層の画面に図示される。個体の出力画面のユニットは、格子状(正方形)、蜂の

巣状(六边形)などが提案されているが、蜂の巣状が多く用いられている。蜂の巣状というのは、文字どおり蜂の巣のように、正六角形のユニットを並べ、出力層の画面を構成する。出力層の画面は、上述のアルゴリズムにより、似ているもの同士を同じユニット、あるいはその近辺のユニットに配置する。各ユニットの特徴の図示は、星図、エラーバー付きの折れ線、色彩濃淡などソフトによって異なる。

2. パッケージ class

(1) 関数 SOM

パッケージ class の中には自己組織化マップ関数 **SOM** がある。関数の書き方を次に示す。

```
SOM(data, grid = somgrid(), rlen = 10000,
alpha, radii, init)
```

引数 data は解析対象となるデータセットで、ユークリッド距離の計算が可能なデータマトリクスあるいはデータフレームである。

引数 grid では、関数 **somgrid** の書き式に従って出力層の指定を行う。関数 **somgrid** を用いて事前出力層の構造を設定した場合は、関数 **SOM** で重複設定する必要はない。

初心者としては、必ずしも指定しなくてもよい引数 **rlen**、**alpha**、**radii**、**init** がある。引数 **rlen** は学習回数で、デフォルトは 10000 回になっている。引数 **alpha** は学習率係数で、学習回数に依存する単調減少関数である。引数 **radii** は学習回数に依存し、近傍領域の半径の更新をコントロールする。引数 **init** では初期の参考ベクトルを指定するが、指定しない場合はデータセットからランダムに用られる。

関数 **SOM** が返す主な結果は、**somgrid** で設定した出力層のユニットの座標(**\$pts**)、各ユニ

ットに配置されるデータの特徴ベクトル(**\$codes**)である。

SOM を実行する前に出力層の設定を行う関数 **somgrid** の書き方を次に示す。

```
somgrid(xdim = 8, ydim = 6, topo =
c("rectangular", "hexagonal"))
```

関数 **somgrid** では、ユニットの配置方法と出力層のサイズとなる列数(**xdim**)と行数(**ydim**)を指定する。ユニットの配置方法は引数 **topo** に **"rectangular"**、**"hexagonal"**の中から1つ選択して指定する。**"rectangular"**は矩形(ユニットを格子状に並ぶ)、**"hexagonal"**は六边形(偶数行のユニットを、ユニットの半分のサイズ分ずつ右にずらして並べる)に適した配置方法である。

SOM の結果の要約は関数 **summary**、結果の図示は関数 **plot** を用いる。関数 **plot** を用いて作成したグラフは、各ユニットに配属される個体の特徴ベクトルの星図である。

(2) データと解析

ここでも使い慣れたデータ **iris** のクラス情報を取り除いて用いることにする。関数 **SOM** を用いるためには、まず関数 **somgrid** を用いて出力層のユニットを設定する。

```
> gr <- somgrid(topo = "hexagonal", xdim=5,
ydim=4)
> out.som <- SOM(iris[,1:4], gr)
> out.som
```

```

$grid
$pts
  x      y
1  1.5 0.8660254
2  2.5 0.8660254
3  3.5 0.8660254
4  4.5 0.8660254
5  5.5 0.8660254
6  1.0 1.7320508
7  2.0 1.7320508
8  3.0 1.7320508
9  4.0 1.7320508
10 5.0 1.7320508
11 1.5 2.5980762
12 2.5 2.5980762
13 3.5 2.5980762
14 4.5 2.5980762
15 5.5 2.5980762
16 1.0 3.4641016
17 2.0 3.4641016
18 3.0 3.4641016
19 4.0 3.4641016
20 5.0 3.4641016

```

```

$xdim
[1] 5

```

```

$ydim
[1] 4

```

```

$topo
[1] "hexagonal"

```

```

attr(,"class")
[1] "somgrid"

```

```

$codes
Sepal.Length Sepal.Width Petal.Length Petal.Width
[1,] 5.859923 3.039938 3.832252 1.250781
[2,] 5.859923 3.039938 3.832252 1.250781
[3,] 5.875102 3.060884 3.735983 1.165968
[4,] 5.884237 2.982233 4.050466 1.334687
[5,] 5.130000 3.760000 1.635000 0.345000
[6,] 5.859923 3.039938 3.832252 1.250781
[7,] 5.859923 3.039938 3.832252 1.250781
[8,] 5.885676 3.029559 3.831273 1.235349
[9,] 5.702956 3.070928 3.646416 1.154233
[10,] 5.850645 3.095562 3.627105 1.120681
[11,] 5.885676 3.029559 3.831273 1.235349
[12,] 5.875102 3.060884 3.735983 1.165968
[13,] 5.702956 3.070928 3.646416 1.154233
[14,] 5.982873 3.085580 4.072671 1.343924
[15,] 6.200000 2.200000 4.500000 1.500000
[16,] 5.702956 3.070928 3.646416 1.154233
[17,] 5.702956 3.070928 3.646416 1.154233
[18,] 5.884237 2.982233 4.050466 1.334687
[19,] 5.850645 3.095562 3.627105 1.120681
[20,] 5.700000 3.000000 4.200000 1.200000

```

```

attr(,"class")
[1] "SOM"

```

このように結果は\$pts,\$xdim,\$ydim,\$topo,\$codesの項目に分かれている。\$ptsには指定した20(xdim=5、ydim=4)の出力層の各ユニットの中心座標値、\$xdim、\$ydim、\$topoにはsomgridで指定した情報、\$codesには各ユニットに配属される個体の特徴となるコード情報が格納される。このSOMの結果を関数plotで図を作成することができる。

```

> plot(out.som)

```

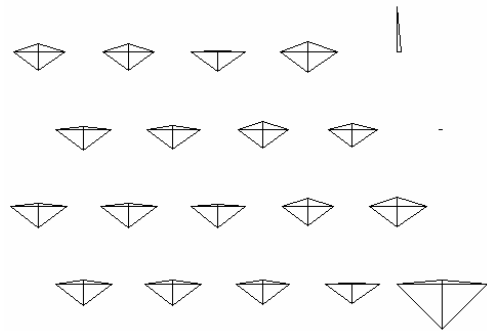


図2 irisデータのSOM図1

この図2は、out.som\$ptsの座標に作成したout.som\$codesの星グラフ(stars)である。この図では、各ユニットに配属される個体の特徴の情報が読み取れる。図の中の星図は、最下段の左から右の方向にout.som\$codesの行のデータを昇順に配置する。この星図の4つの軸(横右、縦上、横左、縦下)は反時計回りに、それぞれirisの4つの変数、ガク弁の長さ(Sepal.Length)、ガク弁の幅(Sepal.Width)、花弁の長さ(Petal.Length)、花弁の幅(Petal.Width)に対応している。

個体を見やすく、SOMの出力画面に配置させるためには若干の手間が必要である。次のように関数symbolsを用いて、各ユニットを円で図示し(六辺形の代わり)、個体を配置することもできる。

```

> symbols(out.som$grid$pts[,1:2], circles =
c(rep(0.5, 20)), inches = FALSE, add = TRUE)

```

個体がどのユニットに配置されるかは解析したSOMのコードを基準とし判別分析を行わなければならない。ここでは関数knnを用いて判別を行うことにする。

```

> haizoku<-as.numeric(knn(out.som$codes,
iris[,1:4], 1:20))

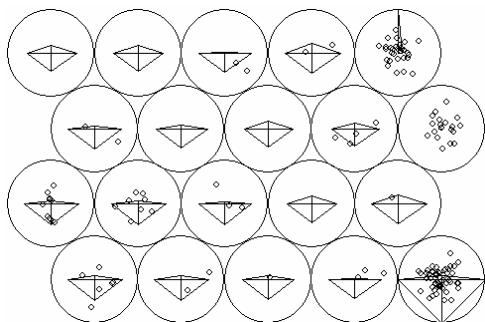
```

上記のコマンドで得られている結果 `haizoku` には、150 個の個体がそれぞれのユニットに配属されるかに関して `knn` 分類法を用いて判別した結果が格納されている。このデータを直接ユニットに配置すればよいが、値がユニットの番号になっているので、同じユニットに配属される点がすべて重なってしまう。そこで各点の座標値に小さい正規乱数を加え、同じユニットに配属される個体が重ならないように調整する。

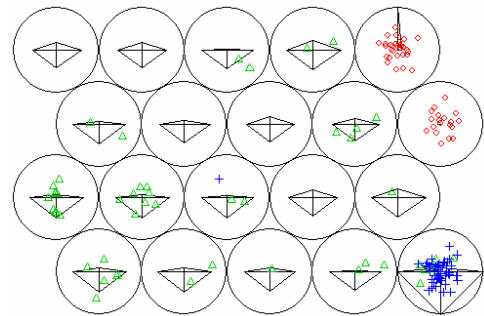
```
> ransu<- cbind(rnorm(nrow(iris), 0, 0.15),
rnorm(nrow(iris), 0, 0.15))
> out.new<-out.som$grid$pts[haizoku,] +ransu
> points(out.new)
```

上記の操作で、図 3(a)が作成される。次のように個体が所属するグループ情報を用いると図 3(b)のように SOM の出力結果の理解と解釈を助ける。グループ情報がない場合は個体の番号を用いることもできる。

```
> points(out.new, col=c(2:4)[unclass(
iris[, 5])], pch=c(1:3)[unclass(iris[, 5])])
```



(a) クラス情報を用いていない



(b) クラス情報を用いる

図 3 iris データの SOM 図 2

3. パッケージ som

(1) 関数 som

パッケージ `som` は、自己組織化マップの専用パッケージで、CRAN ミラーサイトからダウンロードできる。自己組織化マップのメイン関数は `som` である。その書き方を次に示す。この `som` は小文字である。

```
som(data, xdim, ydim, ...)
```

関数 `som` には幾つかの引数を用意されているが、ここでは必ず指定しなければならない引数のみを示している。引数 `data` は、解析の対象となるデータセットである。データによっては、パッケージに用意されている標準化関数 `normalize` を用いて標準化した方がよい。引数 `xdim`、`ydim` は出力画面のユニットの列数と行数である。

(2) データと解析

前節と同じくデータ `iris` を用いた関数 `som` の使用例を示す。

```
> library(som)
> iris.n <- normalize(iris[, 1:4])
> iris.som<-som(iris.n, xdim=5, ydim=4)
```

```
> plot(iris.som)
```

関数 `plot` のグラフは、エラーバー付きの折れ線でユニットの特徴ベクトルを図示する。各個体をユニットへプロットするためには、前節と類似の手順を踏む。ただし、関数 `som` は、各個体が属するユニット番号を `$visual` に記録しているので、前節のように個体の所属を判別するために判別関数を用いる必要がない。実行コマンドおよびその結果を次に示す(図 4)。

```
> ransu<-cbind(rnorm(nrow(iris),0, 0.13),
  rnorm(nrow(iris), 0, 0.13))
> out.new<-iris.som$visual[,1:2]+0.5+ransu
> points(out.new[,1:2],col=c(2:4)[unclass(iris[,5])],
  pch=c(1:3)[unclass(iris[,5])])
```

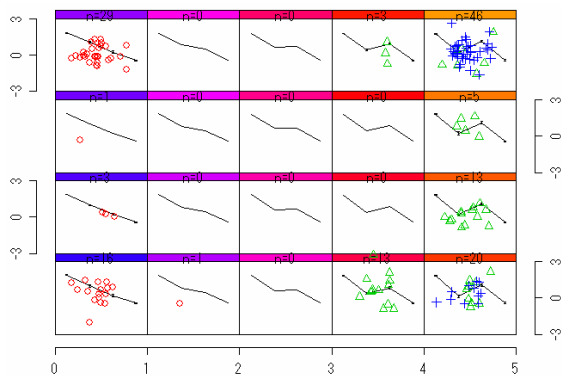


図 4 関数 `som` による図示

次のように `out.new[,1:2]` データを用いてユニットの特徴を表示しない散布図を作成することもできる。

```
>plot (out.new[,1:2],col=c(2:4)[unclass(
  iris [,5])],pch=c(1:3)[unclass(iris[,5])])
```

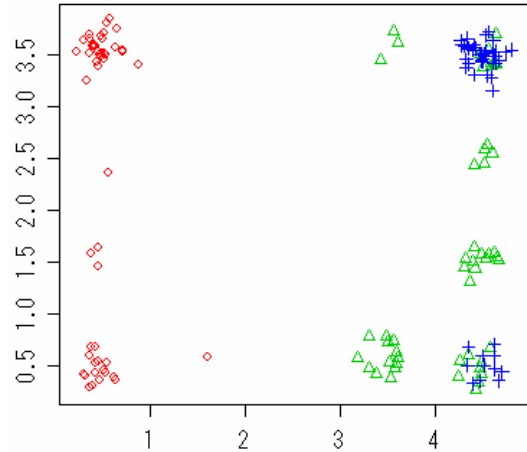


図 5 iris の som 散布図(xdim=5、 ydim=4)

出力層のユニットの数は、結果を大きく左右する。次に `iris` データについてユニット数を 1000 個 (`xdim=100`、 `ydim=100`) にした結果を示す。

```
> iris.som2<-som(iris.n, xdim=100, ydim=100)
> out.new2<-iris.som100$visual[,1:2]+0.5+ransu
> plot(out.new2[,1:2],pch=c(1:3)[unclass(
  iris[,5])],col=c(2:4)[unclass(iris[,5])])
```

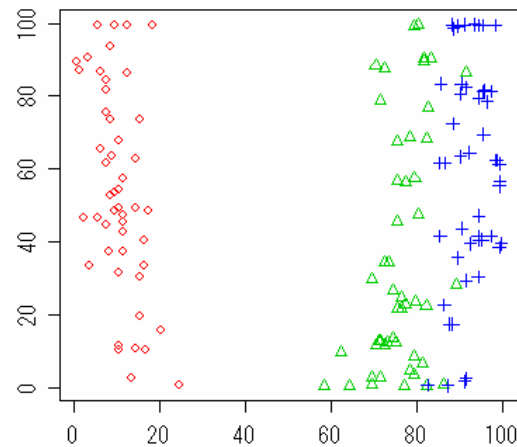


図 6 iris の som 散布図(xdim=100, ydim=100)

図 5 と図 6 を比較すると、図 6 における個体がより明確に 3 種類ごとに分類されていることが分かる。

4. その他

パッケージ `klaR` の中の作図関数 `shardsplot` を用いると次のような図が作成される。

```
> library(klaR)
> iris.som3 <- som(iris[,1:4], xdim = 14,
  ydim = 6)
> library(klaR); opar<- par(xpd = NA)
> shardsplot(iris.som3, data.or = iris,
  label = TRUE)
> legend(3.5, 14.3, col = rainbow(3), xjust =
  0.5, yjust = 0, legend = levels(iris[, 5]),
  pch = 16, horiz = TRUE)
> par(opar)
```

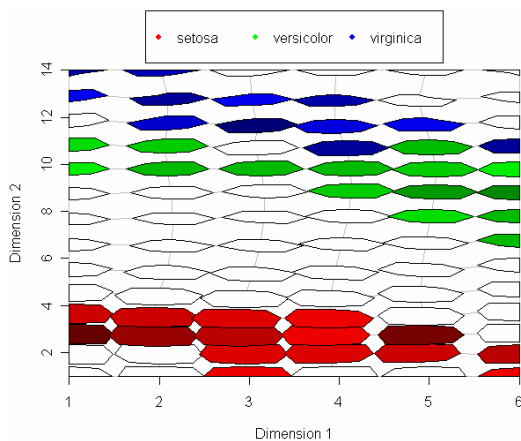


図7 関数 `shardsplot` の SOM 図

SOM は学習データを持たないニューラルネットワークのデータ解析方法である。近年 SOM を扱う本も増えている。初心者には [2]、[3]、より深く追求したい方には [1]、応用に関しては [4] が参考になるであろう。参考文献 [4] では、事業戦略管理・マーケティング、経済分析、建設分野、日本語意味マップ、顔画像認証、データベース/情報検索分野、健康診断などの分野での SOM の応用事例が紹介されている。芥川龍之介、菊池寛、夏目漱石、島崎藤村の作品における助詞を用いた SOM による文体分析の研究事例

としては [5] がある。

参考文献

- [1] T. コホネン著, 徳高 平蔵・その他共訳 (2005): 自己組織化マップ、シュプリンガー・フェアラーク東京
- [2] 豊田 秀樹 (2000): 金鉱を掘り当てる統計学—データマイニング入門、講談社
- [3] 山口 和範、高橋 淳一、竹内 光悦 (2004): 図解入門 よくわかる多変量解析の基本と仕組み—巨大データベースの分析手法入門、秀和システム
- [4] 徳高 平蔵、山川 烈、藤村 喜久郎 (2002): 自己組織化マップ応用事例集—SOM による可視化情報処理、海文堂出版
- [5] 金明哲 (2003): 自己組織化マップと助詞分布を用いた書き手の同定及びその特徴分析、計量国語学、23 巻 8 号、369-386.