平成２４年度　特別研究報告書

SOM 特徴マップの初期値と クラスタリングの明確化に関する研究

龍谷大学大学院　理工学研究科　情報メディア学専攻

学籍番号　T11M090　百井 伸次

指導教員　三好 力 教授

内容梗概

本研究では，SOM の出力結果の視覚的安定性の向上と，連続データの視覚的理解のためのクラスタリング能力向上を図る．一般的な SOM では，出力結果の基となる特徴マップの初期値を乱数で決定しており，その場合同じ入力データでも毎回違う出力結果が表れてしまい，視覚的安定性が失われてしまっている．その問題点に対する SOM の出力結果の視覚的安定性向上の先行研究があるが，データが高次元になるほど視覚的安定性が失われるという問題点がある．そこで本研究は，特徴マップと学習データに強い傾向を持たせることにより，高次元なデータに対しても視覚的安定性を保つことが出来るようになる手法を提案した．そして既存手法との出力結果の比較実験により，提案手法の方が高次元データに対する視覚的安定性が高いことを確認した．また，一般的な SOM にクラスタ性の弱いデータを学習させた場合，出力結果でデータ同士の距離が一定に見え，データの分類がし難いような結果が表れることがある．この問題点に対して，本研究では SOM に Ward 法によるクラスタリング手法を組み合わせ，郡内平方和の増加量を基に自動で適切な個数にクラスタを分割する手法を提案した．そして視覚的に分類可能なデータと分類困難なデータを提案手法によりクラスタリングした結果，出力結果の視覚的理解において適切なクラスタ数を得られる事を確認した．

Abstract

In this study, our purposes are to improve of visual stability of SOM’s output results and to improve clustering capabilities for visual understanding of continuous data. In typical SOM, the initial value of the feature map has been set by the random number. In that case, a different output appears every time even if we applied the same input data. As a result, the visual stability has been lost. There are previous studies to improve of visual stability of SOM’s output results, however, previous study has a problem that visual stability has been lost if the data are high-dimensional. Then in this study, we proposed the method that will be able to maintain visual stability even for high-dimensional data, we give a tendency to the learning data and the feature map. Then, we found the proposed method is highly visual stability for high-dimensional data by the comparative experiments to the output results of conventional method and the one of proposed method.

In addition there are some cases that, the output results of typical SOM appears to be one cluster because of continuous data. In this study, we also have proposed a method for its problems, combines the clustering method (Ward method) to SOM, to divide the cluster into appropriate number automatically based on the amount of the sum of squares in the county. As a result, we have obtained the appropriate number of clusters by the proposed method, in both data that can be classified and that can't be classified.

目次

第１章　はじめに

データマイニングの一手法として健康診断や事業戦略管理, 検索システムなどに使われている自己組織化マップ（以下 SOM）という技術が存在する.

世の中に複雑な情報が数多く存在しているが, 人間はそのような高次元の情報を瞬時に分析，判断することはできない．SOM は多次元データ間の類似度を二次元特徴マップ上での距離で表現することによって, そのような多くの要素を持ったデータ同士の関係について，人間が視覚的に判りやすくなるよう表す技術である.

SOM は他にもデータの分類・要約分析を得意とする．データの分類に関しては，出力マップ上の入力データ内で似た特徴を持つものは近くに, 全く異なる特徴を持つものは遠くにマッピングされるため，データが似ているかどうか判断しやすいためである．データの要約分析に関しては, 多量のデータの類似度を視覚化することによって視覚化前では気付くことのできなかったデータ間の情報を得ることができるためである.

SOM によって学習を行う際，その特徴マップが持つ各ベクトルの初期値を乱数で決定しているが，それでは結果出力のたびに違う結果が表れてしまう．同じ入力データでも違う出力結果が表れてしまうよりは同じ入力データなら似た出力結果が表れる, つまり出力結果に安定性がある方が，SOM の目的の一つである人間の視覚的理解の助けに繋がると考えられる.

その問題を解決するための先行研究として，特徴マップの初期化の際に，学習データの最大値から最小値の範囲で乱数を生成し, その値の平均値をソートしたものを特徴マップに与え，初期化する方法で特徴マップに傾向を持たせるという手法 1) がある．この手法により, 同一データの学習においてある程度出力結果に安定性を持たせることに成功しているが, 学習データの次元数が多くなった場合に対応しきれないという問題点がある.

そこで本研究では，次元数が増えた場合でも対応が出来るように，特徴マップに与える傾向を上記初期化手法以上に強め，さらに学習データにも傾向を与えるという初期化手法を提案する．そして上記既存手法と提案手法で実験を行い，両手法の出力結果の差について比較・検討を行う.

また，上記手法では触れられていなかった問題として，データ同士の距離が一定に見える出力結果, つまりデータ空間内でデータのグループ化が弱くデータ同士の連続性が高い場合，SOM の原理上視覚的に分類し難い出力結果が出てしまい，結果の視認性に影響が及ぼされるということがある．

本研究では，そのような結果に対し Ward 法を用いたクラスタリングを行い，さらにクラスタ形成時の郡内平方和の増加量を基にクラスタ数を自動で決定するという手法を提案し，データ同士の関係性についてさらなる視認性の向上を図る．

第２章　基本事項

2.1　SOM とは

自己組織化マップ（Self Organization Mapping：通称 SOM）とは，競合学習型ニューラルネットワークの一種であり，入力層と出力競合層の 2 層から成っている．SOM は 1980 年代にコホーネンによって開発され，多次元データの分類，解析に効果的な技術として知られている．

SOM の特徴は，n 次元のベクトル集団を学習することにより 2 次元のマップにそれらのベクトルの関係を写像することができることである．似ているベクトルは 2 次元マップ上の近い位置に配置され，似ていないベクトルは遠い位置に配置される．汎用性に優れており，認識，予測，分類など様々な応用が可能であることから，今後の活躍が期待できる．

2.2　SOM の基本的なアルゴリズム

コホーネンは生物の神経細胞，主に脳の情報処理の仕方を以下の簡単な式に整理した．

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)]. \tag{2.1}$$

この式は，次のような意味である．いま神経細胞（ノード）i が時刻 t で処理している情報処理能力を$m_i(t)$とするとき，外部から入力信号$\mathrm{x}(t)$が入ってきたとする．細胞はこの入力信号を学習して次の時刻には入力信号により近い情報処理能力$m_i(t+1)$を持つようになる．このとき$\mathrm{x}(t)$が n 次元の入力ベクトルである場合，参照ベクトルとも呼ばれる$m_i(t)$もまた同じ n 次元の要素を持つ．そして，$h_{ci}(t)$は学習率係数を含めた近傍関数を意味する．なお，$t = 0,1,2$ ...は離散時間座標である．出力競合層のベクトルは参照ベクトル$m_i(t)$で表され，入力層の次元に合わせて n 個の要素を持っている．図 2.1 に示すように，視覚的に出力を見るために，普通は 2 次元に配列されている．

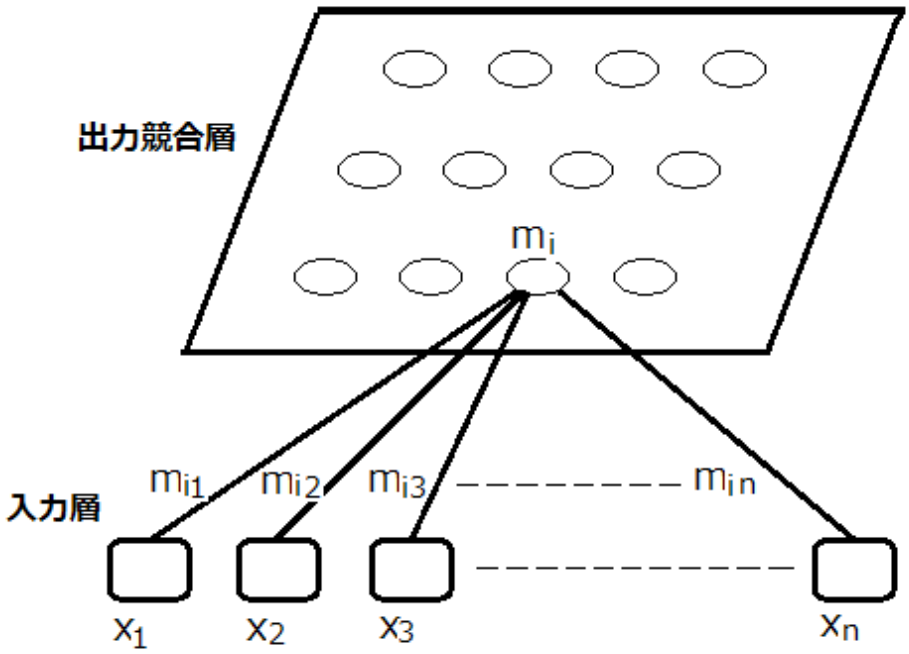

図 2.1　SOM の構造

SOM の学習は次のように行われる．入力ベクトル$\mathrm{x}(t)$はある測度，例えばユークリッド距離$|\mathrm{x}(t)-m_i(t)|$を最小にするノード i を探し，それに添え字 C をつける．

$$|\mathrm{x}(t)-m_c(t)| = min|\mathrm{x}(t)-m_i(t)|. \tag{2.2}$$

式 (2.2)で決められた参照ベクトルを持つノードを勝者ノードと呼ぶ．式(2.1)および式(2.2)での学習の状態について下図２で簡単に説明する．まず入力ベクトルが提示されると，その入力ベクトルに一番近いノードが勝者ノードとなる．そして勝者ノードの周りに囲った正方形の図では，円になっている領域を近傍領域と定義する．その形は正方形でもよい．基本のノードの配列を六角形の形にとれば六角形近傍となる．近傍内のすべてのノードは入力ベクトルを学習し，式(2.1)に従って入力ベクトルの方向へ少し近づく．この学習を繰り返し行うが，このとき近傍領域の範囲は最初広くとっておき，学習とともにその領域を狭めていく．

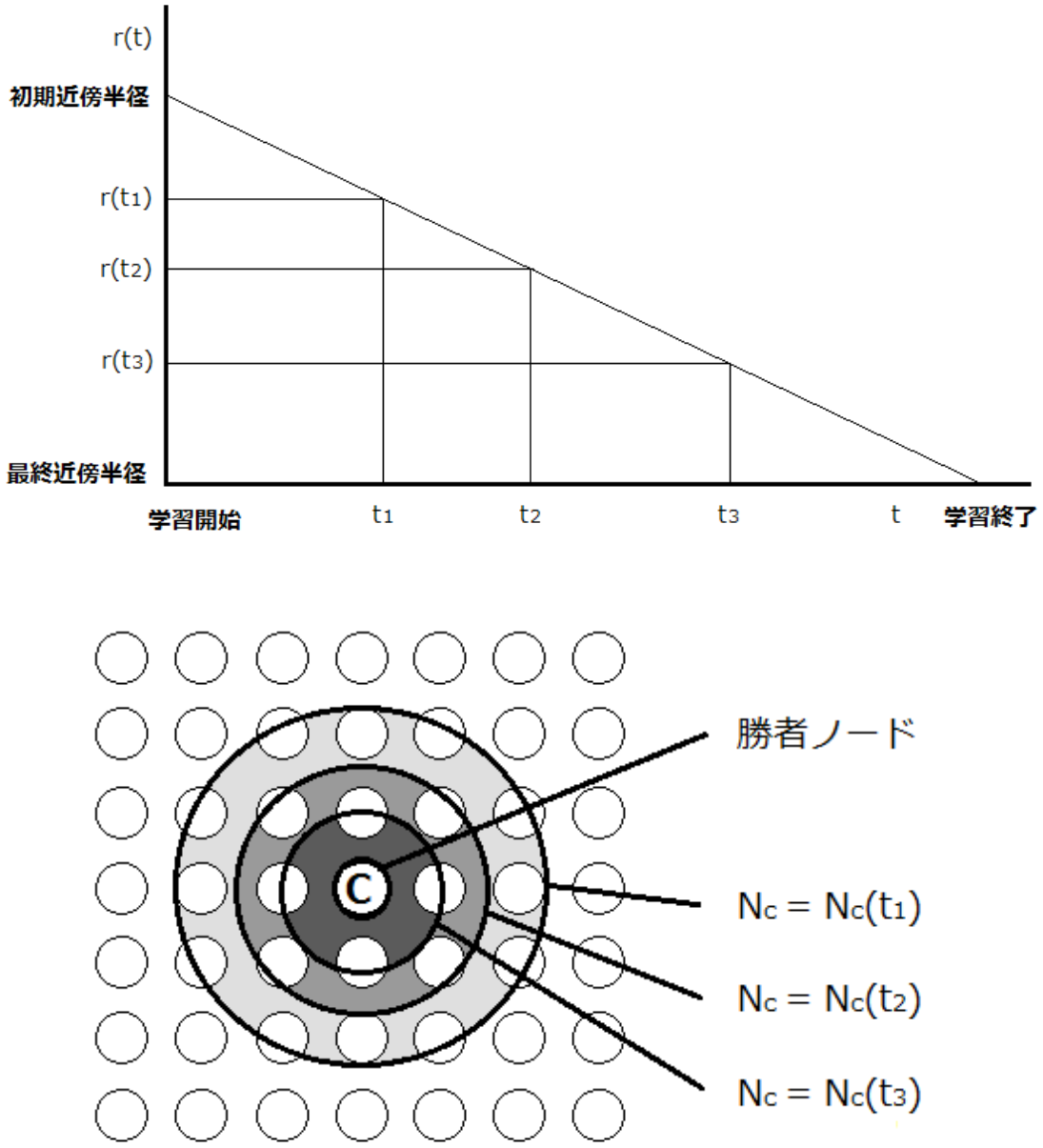

図 2.2　SOM の学習の繰り返しにおける近傍領域の変化

式(2.1)において，$h_{ci}(t)$は，学習率係数$\alpha(t)$と近傍関数$h(d,t)$により次のように表現できる．

$$h_{ci}(t) = \alpha(t) * h(d,t). \tag{2.3}$$

近傍関数は，学習される近傍領域を指定する関数である．近傍領域は，出力競合層の勝利ノードの近傍を意味し，学習によって学習ベクトルが更新される領域である．学習を

始めるときには，近傍領域をその範囲を大きくとり，学習が進むにつれて徐々にその領域を狭める．近傍領域を減少させる関数には，線形型とステップ型とがある．

ここでは，よく使用されている，ガウス型近傍関数について説明する．

$$h(d,t) = exp\left(-d^2/k\big(r(t)\big)^2\right). \tag{2.4}$$

ここで，d は式(1.5)で表せる勝者ノードから参照ベクトルまでの距離，$k\big(r(t)\big)$は学習時間 t のときの近傍領域の最大距離で，ガウス関数の波幅の係数である．

$$d^2 = (x-a_i)^2+(y-b_i)^2. \tag{2.5}$$

ここで，(a_i, b_i)は勝者ノードの位置，(x, y)は半径$\mathrm{r}(t)$により成る円形領域の内側にあるノードの位置を示す．

近傍半径$\mathrm{r}(t)$は，学習のはじめの段階では大きく，学習が進むにつれて小さくなってゆく．

半径$\mathrm{r}(t)$の円領域に含まれる範囲のノード，参照ベクトルは，指数関数で決まる重みを持って学習される．すなわち，勝者ノードに近いほど，その類似性が大きくなるように学習され，勝者ノードから遠ざかるほど，その類似性が小さくなるように学習される．そして半径$\mathrm{r}(t)$の範囲外では学習されないことになる．

したがって，式(2.1)によって学習している間，近傍領域N_c内のノードに関しては，$h_{ci}(t) = \alpha(t) * h(d,t)$で，$N_c$外のノードに関しては，$h_{ci}(t) = 0$である．つまり，近傍の外側の領域については学習されない．

以上により，近傍関数は次式で表される．

$$h_{ci}(t) = \alpha(t) * h(d,t) \quad (i \subset N_c).$$

$$h_{ci}(t) = 0 \qquad \big(\textit{それ以外}\big). \tag{2.6}$$

このとき，$\alpha(t)$の値を学習率係数と呼び，$0<\alpha(t)<1$の値を持つ．$\alpha(t)$とN_cの大きさは両方とも学習時間が経つにつれて普通，単調減少させる．$\alpha(t)$は例えば次の式で定義しても良い．

$$\alpha(t) = a_0(1-t/T). \tag{2.7}$$

ここで，a_0はαの初期値であり，普通 0.2～0.5 の値を選ぶ．Tは行われるべき学習での予定された全更新学習回数である．ただし，式(2.1)中，比例係数の$\alpha(t)$は，学習のはじめでは大きな値をとるようにし，学習が進んでくるとだんだん小さい値になるようにする．また，近傍領域$N_c = N_c(t)$も式(2.6)と同様に減らしていってもよい．つまり，

$$N_c(t) = N_c(0)(1 - t/T). \tag{2.8}$$

ここで，$N_c(0)$は初期値である．

コホーネンの SOM アルゴリズムを整理すると，以下のようになる．

1. 出力層にノードを配置し，それぞれの持つ参照ベクトルを乱数で初期化する．
2. 入力ベクトルに最も近い（似ている）競合層での参照ベクトルを探し，これを勝者ノードとする．
3. この勝者ノード及び近傍内のノードを式(2.1)に従って更新する．また，学習が進むにつれて近傍領域を狭め，学習率係数の値も例えば式(2.4)のようにして減らしていく．

SOM の学習を行うために学習パラメータの設定をする必要がある．ここに主要な学習パラメータとその説明を示す．

学習回数　　：競合層に対して学習を何回行うかを指定する．

学習係数　　：1 回の学習でノードをどれだけ更新するかを指定する．

近傍領域　　：どれだけの範囲に学習の影響を与えるか初期範囲を指定する．

マップサイズ：出力競合層のノードを X,Y 軸にいくつ並べるかを指定する．

第３章　既存技術

この章では，基礎技術の問題点について触れ，出力結果の安定性を高めるための既存手法について説明し，今提案手法の比較対象となる既存手法についても詳しく説明する．

3.1　参照ベクトルの初期値について

第一章でも述べたように，特徴マップの各ベクトル（以下参照ベクトル）が持つ値は，基礎技術では乱数で初期化されている．しかし，ただ単に乱数で初期化した場合では，SOM の学習時，主に初回の学習において大きな影響を及ぼす．以下に単に乱数で初期化した場合で起こりうる問題点の例を挙げる．

・同じ参照データでも勝利ノードの座標が大きく変わる

参照ベクトルが持つ値を乱数で決定すると，学習の度に各参照ベクトルが持つ値は変わる．そうなるとある学習時に参照した入力ベクトルと，別の学習時に参照した入力ベクトルに対する勝利ノードの座標が大きく違ってくる場合が多い．その例を図 3.1 に示す．

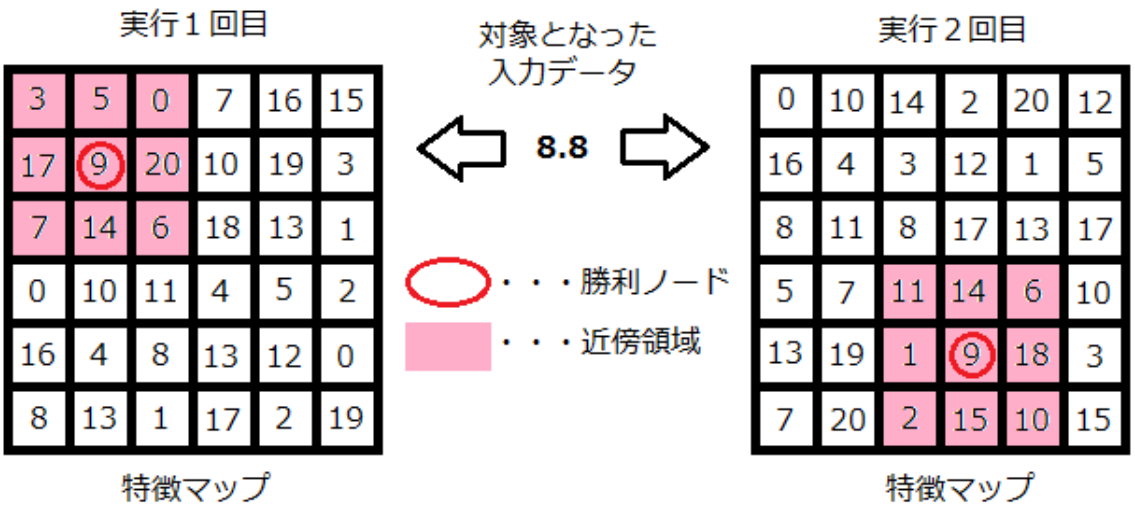

図 3.1　乱数初期化における勝利ノード座標のばらつきの例

例では特徴マップが 6×6, 入力ベクトルが 8.8, 近傍領域が周囲 1 マスとなっている．このように参照ベクトルの値が完全にランダムであると，特徴マップ上での入力データの扱いがたとえ同じ値であったとしても SOM を実行する度に同様に完全にランダムになってしまう．また，勝利ノードの周辺のベクトル，つまり近傍領域内のベクトルも値

はランダムなため，影響を受けるベクトルも SOM を実行する度に大きく変わってしまう．

さらに，一回入力データに対する勝利ノードが決定してしまうと，再度その入力データが参照された場合やその入力データに似た入力データが参照された場合でも，その勝利ノードやその近辺のベクトルが勝利ノードに選ばれる可能性が高くなる．したがって，初期値が SOM 実行毎に大きく変わってくると，出力結果も大きく変わっていくので，初期値が結果に及ぼす影響は強いと考えられる．

3.2 既存手法の問題点

3.2.1 入力データを分析したものを参考にした場合について

初期値を完全に乱数で決定する方法に対して，初期値を入力データから計算した値で決定する方法が提案されている．例えば，Mu-Chun su らによって提案された手法 2)では，入力データ内での距離が最も遠い 4 つを抽出し，それを特徴マップの四隅に割り当て重みとし，さらに両端の重みから最も遠いデータを対辺の重みとして割り当て，それをマップが埋まるまで時計回りに繰り返した結果できる特徴マップを初期値とする方法が提案されている．この例のように，入力データを解析して初期値を決定する方法について，以下のような問題点が挙げられる．

・自己組織化マップとの目的の類似

本来の目的として，入力データについてどのような関連性・性質があるかどうかを分析するために，SOM が用いられる．参照ベクトルの初期値を決めるために本来分析の対象となる入力ベクトルを別の手法で先に分析してしまうということは，程度によっては本末転倒になりかねない．

・対象となる入力データの値の数に計算量が依存する

対象となる入力データの数，及び次元数が多ければ多いほど，分析のための計算量や計算回数が多くなっていく．さらに分析した値を特徴マップに割り当てる際も，並び順などを考慮したアルゴリズムを別途組む必要があり，入力データの分析と特徴マップの割り当てとで多くのステップが必要になってしまう．

3.2.2　入力データの分析を抑えて初期化を行う既存手法

三好らの研究 3)では SOM の学習能力を十分利用するため，多くの計算コストをかけることなく，初期値をランダムに決定する方法を改善することによって学習速度の高速化を行う方法が提案されている．この方法は，初期化時点で学習するデータを参考に特徴マップのノード交換を行うことによって，入力ベクトル空間と特徴マップ上の位置関連付けを行うものである．この方法を導入することで全ノードの平均移動距離が短縮され，学習速度の高速化が確認されている．しかし，SOM の学習能力を生かす点と初期化手法に手を加える点については本研究と合致はしているが，出力結果の安定性を求める目的については十分検討されているとは言えない．

3.2.3　学習データに傾向を持たせて初期化を行う既存手法

アルゴリズムを後述するが，この手法では学習データの分析を少なく抑えるために，特徴マップ上のベクトルを生成する際に基準となる値を学習データに対して大掛かりな計算をせずに決定している．しかし，単にこれだけでは安定性の向上については見込めないため，特徴マップの値の決定後に各ノードのソートを行うことによって特徴マップに傾向を持たせることで安定性の向上を図っている．この手法は以下に示す二点を特徴としている．

・乱数の範囲の限定

学習データ内の最小値・最大値のみ取得し，その範囲で乱数を生成するということを考える．そしてその生成した乱数を参照ベクトルの初期値として暫定的に割り振る．単に乱数で参照ベクトルを決定するより，最小値・最大値を取得することで，参照ベクトルの値の範囲が学習データの持つ値の範囲内全てに対応したものになるという利点がある．また最小値・最大値を取得するだけでよいため，学習データに対して大掛かりな計算をする必要はない．

・参照ベクトルのソート

次に，暫定的に初期値として割り振った参照ベクトルが持つ値の平均値を計算し，その平均値を基に特徴マップをソートする．これを行うことで参照ベクトルが昇順に並ぶことになり，特徴マップに傾向が表れる．特徴マップに傾向を持たせることで，同じデータに対する勝利ノードの位置の振れ幅を小さく出来，結果的に最終出力結果も安定したものになると考えられる．図 3.2 に，傾向をもった特徴マップのイメージを示す．

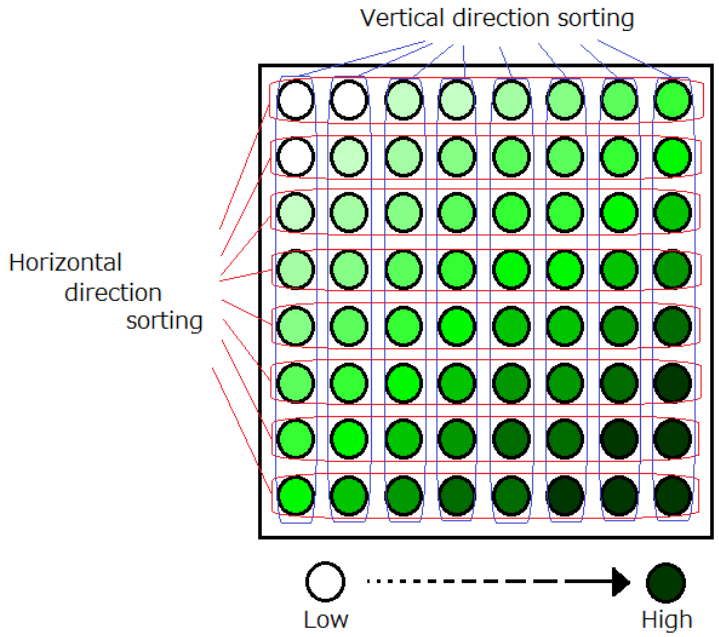

図 3.2　傾向をもった特徴マップのイメージ

・アルゴリズム

以下に 3.4.1 及び 3.4.2 までの手法のアルゴリズムを示す.

1. 入力データ全てを参照する

2. 参照した中から最小値・最大値を取得する

3. 最小値・最大値の範囲で乱数を生成する

4. 生成した乱数を参照ベクトルに暫定的に初期値として割り当てる

5. 参照ベクトルの次元数分の値の平均値を求める

6. 平均値を基に，特徴マップの縦方向に対して参照ベクトルを昇順に，クイックソートを行う

7. 平均値を基に，特徴マップの横方向に対して参照ベクトルを昇順に，クイックソートを行う

8. ソートによって出来た特徴マップを初期値とし，学習を行う

しかし上記の学習データに傾向を持たせて初期化を行う既存手法では，単純なデータに対しては従来法と比較して安定した出力結果を残すことに成功しているが，次元数が増えれば増えるほど安定性が失われ，従来法の結果に近づいていってしまうという問題点がある．

第 4 章　提案手法（1）

4.1　提案手法（1）

本研究では，次元数が増えても安定性を保つことができるように，3.2.3 の手法に改良を加えた以下の手法を提案する．

4.1.1　辞書的な特徴マップの初期値のソート

学習データに傾向を持たせて初期化を行う既存手法では，ソートの際に平均値が同じベクトルが存在した場合の順序が考慮されていない．こうした場合の順序が考慮されていないと，学習データの値の範囲が狭い場合に，平均値が同じベクトルが多数表れることが考えられるため，実質的にソートを出来きれていない場合が起こりうる．そのような問題を防ぐために，今提案手法では，平均値が同じベクトルが存在する場合は，ベクトルの最初の次元の値から比較して，ベクトルは昇順に，つまり辞書的なソートをする．図 4.1 に，そのソート方法の違いを表す．

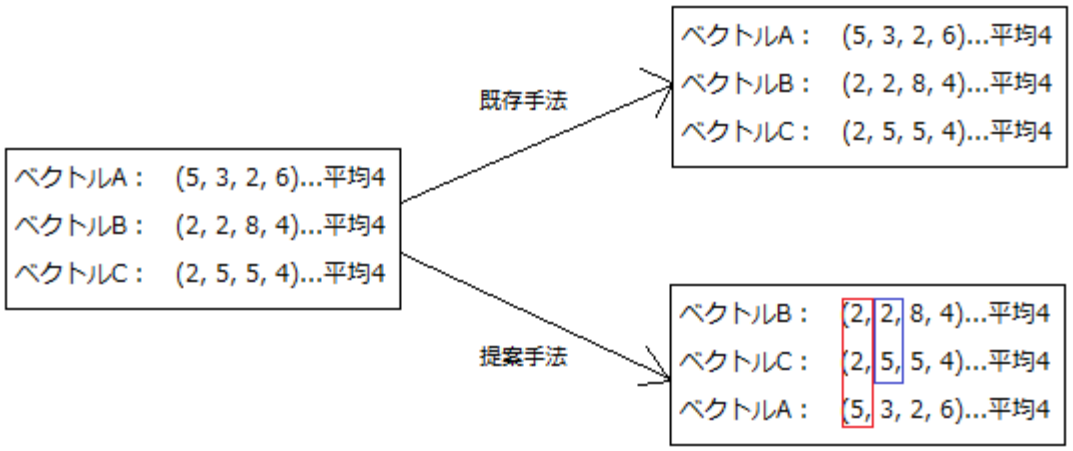

図 4.1　ソート方法の違い

図 4.1 左のようなソート対象があった場合に，既存手法では全てのベクトルの平均値が同じなため，ベクトルはソートされずそのままになってしまう．しかし今提案手法では，ベクトルの平均値が同じな場合，一次元目の値から順にソートのキーになるため，平均値が同じ場合でもそのベクトル同士の中でまた傾向を生み出すことが出来る．こう

して学習データの値の範囲が狭い場合でも特徴マップに傾向を与えることが出来ると考えられる.

4.1.2 辞書的な学習データのソート

学習データに傾向を持たせて初期化を行う既存手法では, 学習過程において学習データはランダムに抽出されていた．SOM の学習結果において，初期化が大きく影響を及ぼしていることについては説明しているが, それ以外にも学習データの抽出順も学習結果に大きな影響を与えている．SOM の学習過程では，初期近傍を広くとることが一般的であるため，最初に抽出された学習データは特徴マップ内に広く影響を及ぼし，それにより以降の学習結果にも影響を及ぼす．したがって，学習データをランダムに抽出するのではなく，今提案手法では，学習データは辞書的にソートされた順に抽出をする．これにより学習過程にも傾向を与えられ, 結果として出力結果の安定性が増すと考えられる.

4.1.3 提案手法（1）のアルゴリズム

以下に 4.1.1 及び 4.1.2 までの手法のアルゴリズムを示す.

1. 入力データ全てを参照する
2. 参照した中から最小値・最大値を取得する
3. 最小値・最大値の範囲で乱数を生成する
4. 生成した乱数を参照ベクトルに暫定的に初期値として割り当てる
5. 参照ベクトルの次元数分の値の平均値を求める
6. 平均値を基に，特徴マップの縦方向に対して参照ベクトルを昇順ソートする
   平均値が同じベクトルがある場合，1 次元目から比較し値の低いものを優先する
7. 平均値を基に，特徴マップの横方向に対して参照ベクトルを昇順ソートする
   平均値が同じベクトルがある場合，1 次元目から比較し値の低いものを優先する
8. ソートによって出来た特徴マップを初期値とし，辞書的にソートされた学習データを基に学習を行う

4.2　実験方法

特徴マップの初期ベクトルを平均値でソートしているが学習データはソートしない既存手法と提案手法のプログラムを作成し，同一の学習データに対して繰り返し学習実験を行った．評価方法は，まず比較用の特定の学習データが 4000 回の学習回数の内に何回抽出されたかを数える．そして抽出されるたびにその特定の学習データに対して選ばれた勝利ノードの特徴マップ上での座標と座標の選出回数も記録する．次に 100 回の実行回数の中で最も選出回数の多かった座標を調べ，グラフ化した．また，得られる出力結果がわかりやすいように，値が二極化した入力データを用いる．図 4.2 に，二極化した入力データのイメージを示す．

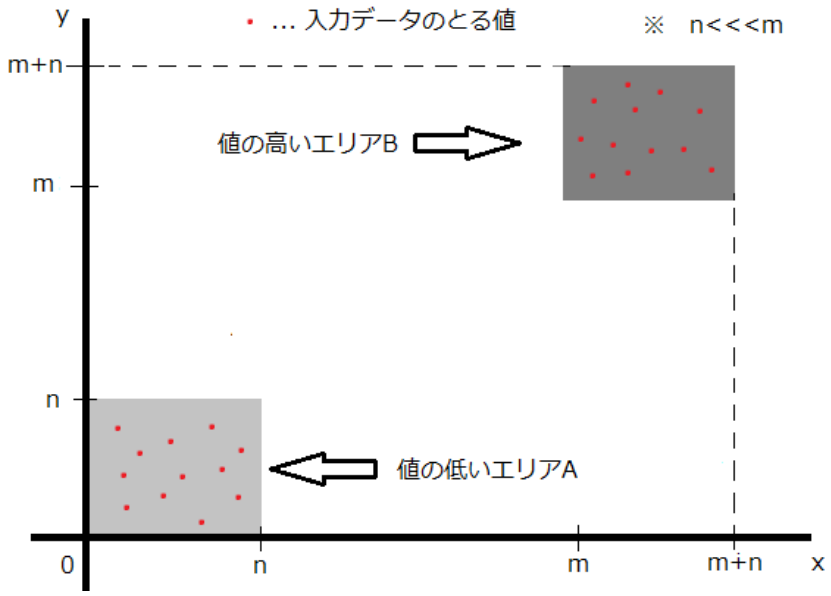

図 4.2　二極化データのイメージ

4.2.1　実験パラメータ

以下のパラメータで SOM を実行する．

・実験回数　　：100 回
・入力データ　：8 次元×24 個のデータ，値は二極化されている
・特徴マップ　：30×30 マス
・学習回数　　：4000 回
・近傍領域　　：20（学習回数 50 毎に 1 減少）
・学習率係数　：0.01×近傍領域

4.3　実験結果

図 4.3 に既存手法と提案手法の実験結果を示す．上のグラフは既存手法を，下のグラフは提案手法を示す．青点は x 軸座標，赤点は y 座標位置を表し，縦軸は座標で，横軸は実験番号を表す．各色の点が固まっていれば座標位置は安定傾向にあり，ばらついていれば安定傾向に無いといえる．

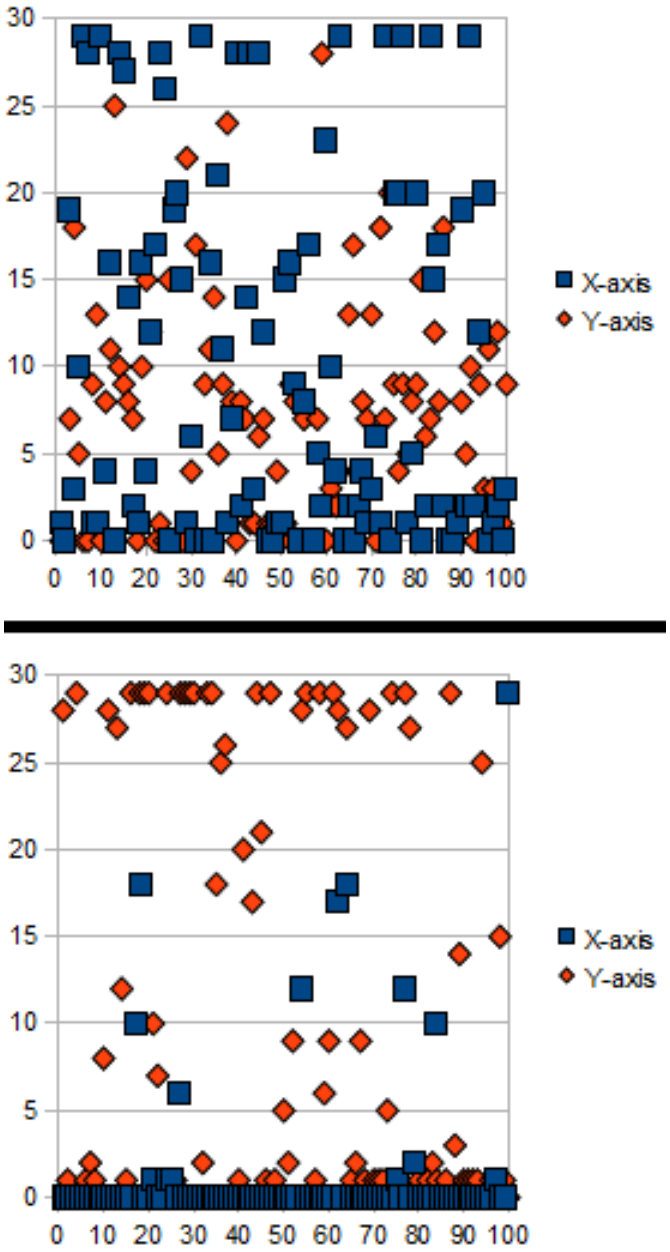

図 4.3　実験結果

4.4 考察

図 4.3 の実験結果を基に考察を行う．既存手法について，x 座標はある程度 0 付近に集まりを見せているものの，全体的に分散していることがわかる．また，y 座標も同様に全体的に分散していることがわかる．一方，提案手法では殆どの x 座標が 0 付近に集まりを見せ，y 座標は 0 と 30 付近に分かれはしたものの分散傾向にはないことがわかる．これらのことから，8 次元の学習データに対して提案手法の方が勝利ノード位置のばらつきが少なく，出力結果が安定していることが言える．

なお，低次元の学習データにおいても同様の実験を行ったが，座標位置のばらつきに差は見られなかった．このことから，今提案手法は既存手法の低次元での安定性を保ちつつ，それ以上の次元でもその安定性を保てていると言える．

第５章　別視点での安定性の問題

前章までに，データ空間内でデータがグループ化されている多極的なデータでは，既存技術より提案手法の方が安定性の高い結果を得ることが出来たことを示した．しかしデータ空間内でデータのグループ化が弱く連続性の高い場合，SOM の原理上視覚的に分類し難い出力結果が出る問題については触れていなかった．そこでこの章では，そのような結果が出た場合について説明し，その問題点を示す．

5.1　連続性のあるデータに対する出力結果図の問題点

先ほど触れたように，通常 SOM では多次元データ間の類似度を二次元特徴マップ上での距離で表現している．ここでは 2 次元特徴マップ上でデータがどのように表れ，その問題点について示す．

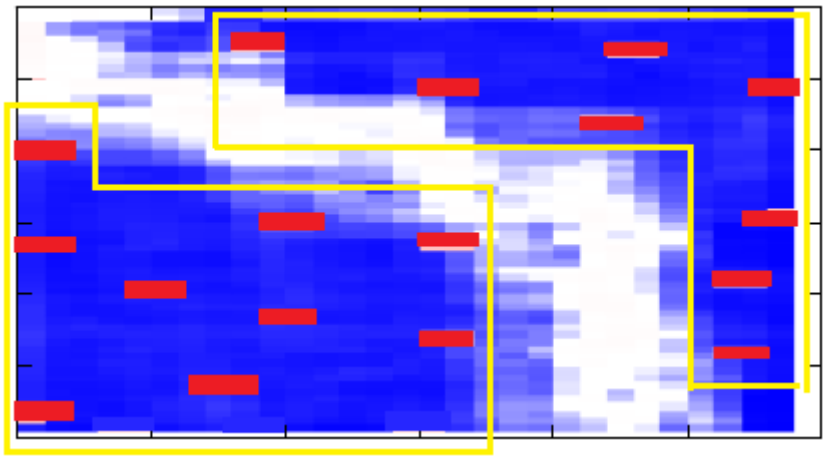

図 5.1　SOM 出力結果（二極化）

図 5.1 は SOM の出力結果の一例である．平面図内に青い領域と白い領域があるが，色が濃くなればデータ距離が近く，薄くなればデータ距離が遠くなる．そして赤点がデータ位置を示す．図８のような出力結果でデータ集合を比較した場合に，白い領域が境界線の役割を果たしているので，黄色で囲った部分でクラスタ分類が視覚的にも容易に出来ることがわかる．

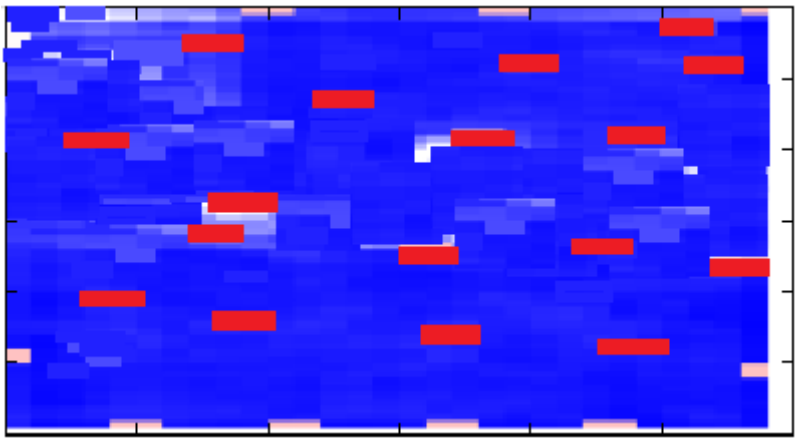

図 5.2　SOM 出力結果（連続性）

図 5.2 は図 5.1 と違い，領域が曖昧でかつデータ同士の距離も均整のとれた SOM の出力結果の一例である. このような出力結果図のことを便宜上「連続性のある出力結果」と呼ぶ.

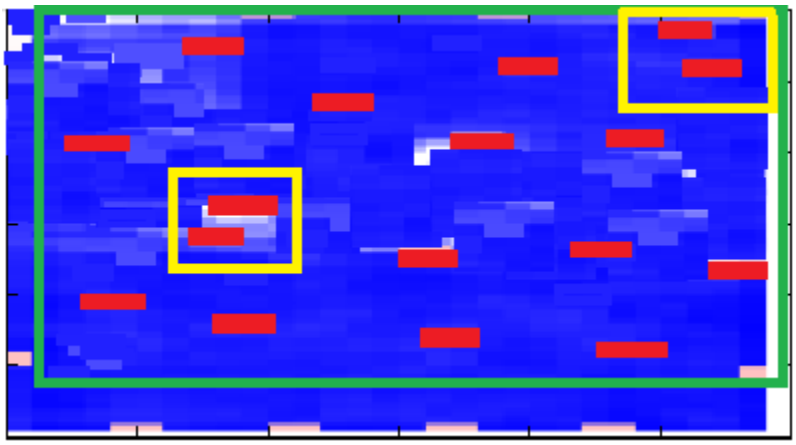

図 5.3　SOM 出力結果（連続性）の分類

連続性のある出力結果の場合，図 5.3 内の黄色で囲った一部のデータに関しては隣接データの距離の類似度は視覚的にも判別出来るものの，大部分のデータの類似性やデータ集合単位で見た場合に緑で囲った中のクラスタ分類は非常に困難になる.

本研究では SOM にクラスタリング手法を組み合わせ，連続性のあるデータに対して SOM による学習では表現されないデータ間の違いを手掛かりにクラスタリングを自動で行い，理解しやすい程度の数に分類する手法を考えた．それによってデータの関係性についての視覚的理解の向上を図る．

5.2 クラスタリングとは

クラスタリングとは，分類集合の対象を類似度（非類似度）や距離を用いて，データをいくつかの部分集合に分割する方法である．これにより分割された部分集合を，クラスタと呼ぶ．

クラスタリング手法には大きく，階層的クラスタリングと非階層的クラスタリングに分けられる．階層的クラスタリングには，最長距離法， Ward 法などがあり，非階層的クラスタリングには k-means 法などがある．階層的クラスタリングではクラスタ生成の際に基準となる距離に，ユークリッド距離やマハラノビス距離などがあるが，一般的にはユークリッド距離が用いられることが多い．そしてクラスタ生成法は各手法によって異なる．非階層的クラスタリングでは，それぞれの手法の評価関数によってクラスタが生成される．

5.3 階層的クラスタリング

階層的クラスタリングの基本としては，まず分類集合の対象の各データを一つのクラスタとし，類似のクラスタを順に合わせていき，クラスタを拡大させていく．出来上がったクラスタは，デンドログラムという樹形図で表すことが出来（下図 5.4 参照），クラスタ数はそのデンドログラムにより決定される．クラスタ生成法は各手法により異なるため，生成されるデンドログラムは手法によって大きく異なる場合がある．

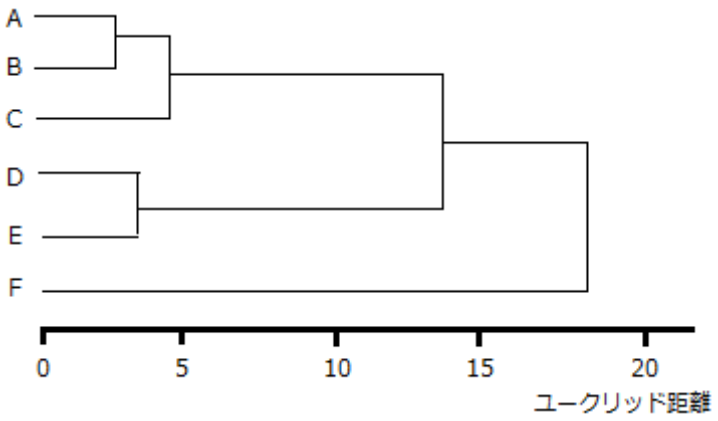

図 5.4　デンドログラムの例

5.3.1　最長距離法

最長距離法とは，各クラスタ間の距離における最長距離を，クラスタ間の距離とする方法である．この方法は，一つのクラスタが極端に大きくなるのを抑えられ，大きさのそろったクラスタを得ることができる．

$$D(C_1, C_2) = \max_{x_1 \in C_1, x_2 \in C_2} D(x_1, x_2). \tag{5.1}$$

ここでx_1, x_2は二つのデータを表し，C_1, C_2は各クラスタを表す．そして$D(x_1, x_2)$は二つのデータ間の距離（類似度）を表す．また，データとクラスタ間の距離及びクラスタとクラスタ間の距離もDで表す．

5.3.2　Ward 法

Ward 法は，二つのクラスタを結合する際に，「群内平方和の増加量」が最小になる二つのクラスタを一つに結合するという手法である．すべてのクラスタ内の偏差平方和の和をできるだけ小さくするように組み合わせていくので，比較的まとまりのあるクラスタを得ることが出来る．

$x_{k_i}^A$をクラスタ A に属する i 番目の対象（クラスタ A 内に n 個）の第 k 変数（全部で p 個）についての値とすればクラスタ A 内の平方和は

$$S_A = \sum_{k=1}^{p} \sum_{i=1}^{n_A} \left(x_{k_i}^A - \overline{x}_k^A \right)^2 . \tag{5.2}$$

と表される．ただし，

$$\overline{x}_k^A = \frac{1}{n_A}\sum_{i=1}^{n_A} x_{k_i}^A. \tag{5.3}$$

クラスタ B も同様に,

$$S_B = \sum_{k=1}^{p}\sum_{i=1}^{n_B}\left(x_{k_i}^B - \overline{x}_k^B\right)^2. \tag{5.4}$$

と表される．ただし,

$$\overline{x}_k^B = \frac{1}{n_B}\sum_{i=1}^{n_B} x_{k_i}^B. \tag{5.5}$$

クラスタ A とクラスタ B を結合して新しくクラスタ C を作ったとすると，新しいクラスタ C 内の平方和は

$$S_C = \sum_{k=1}^{p}\sum_{i=1}^{n_C}\left(x_{k_i}^C - \overline{x}_k^C\right)^2 = \sum_{k=1}^{p}\left[\sum_{i=1}^{n_A}\left(x_{k_i}^A - \overline{x}_k^C\right)^2 + \sum_{i=1}^{n_B}\left(x_{k_i}^B - \overline{x}_k^C\right)^2\right]. \tag{5.6}$$

ここに,

$$\overline{x}_k^C = \frac{1}{n_C}\sum_{i=1}^{n_C} x_{k_i}^C = \frac{1}{n_A+n_B}\left[\sum_{i=1}^{n_A} x_{k_i}^A + \sum_{i=1}^{n_B} x_{k_i}^B\right] = \frac{n_A}{n_A+n_B}\overline{x}_k^A + \frac{n_B}{n_A+n_B}\overline{x}_k^B. \tag{5.7}$$

これより,

$$S_C = \sum_{k=1}^{p}\sum_{i=1}^{n_A}\left(x_{k_i}^A - \overline{x}_k^A\right)^2 + \sum_{k=1}^{p}\sum_{i=1}^{n_B}\left(x_{k_i}^B - \overline{x}_k^B\right)^2 + \frac{n_A n_B}{n_A+n_B}\sum_{k=1}^{p}\left(\overline{x}_k^A - \overline{x}_k^B\right)^2. \tag{5.8}$$

つまり,

$$S_C = S_A + S_B + \Delta S_{AB} \tag{5.9}$$

$$\Delta S_{AB} = \frac{n_A n_B}{n_A+n_B}\sum_{k=1}^{p}\left(\overline{x}_k^A - \overline{x}_k^B\right)^2 \tag{5.10}$$

となる．したがって，ΔS_{AB}はクラスタ A とクラスタ B を統合して新しくクラスタ C を作った時，その平方和の増分であることを意味する．つまり，各段階でΔS_{AB}が最小になるように，クラスタ対を選んで統合すればよい.

5.4　非階層的クラスタリング

非階層的クラスタリングは，あらかじめいくつのクラスタ数にするかを決めておき，その数に従ってデータを振り分けていくというものである. 決められたクラスタ数に対して，できるだけクラスタ間の距離は大きく，各クラスタのデータ間の距離は小さくなるように新たなデータを振り分けていく方法であるので, クラスタ数を変えて分析を行うと生成されたクラスタ間に包含関係がないことが多い. 非階層的クラスタリングはその特徴から計算量が膨大なため，処理時間が長くなることが欠点である.

5.4.1　k-means 法

k-means 法は，あらかじめクラスタ数を決めておき，各データを振り分けていく方法である．クラスタに含まれるデータとそのクラスタの重心点の距離が，他のどのクラスタの重心点よりも小さくなるように求める．各クラスタ間の距離は大きく，クラスタ内の距離は小さくなるように分割されている．つまり，計算方法としては，分割のよさの評価関数を定め，その評価関数を最適にする分割を探し当てることになる.

評価関数は，次のように与えられる.

$$\sum_{i=1}^{m} \sum_{x \in C_1} D(x, C_i)^2 . \tag{5.11}$$

k-means 法は，クラスタの重心点をクラスタの代表点とし，評価関数を，最小化する m 個のクラスタに分割する．最適解は，対象点のクラスタへの割り当てと，代表点の再計算を交互に繰り返し行って探す．この方法は，局所最適解しか求められないため，ランダムに初期値を変更して，評価関数を最小する結果を選択するのが一般的である.

第６章　提案手法（２）

6.1　クラスタリング手法の選択

ここでは SOM にクラスタリングを導入するにあたって，適切な手法は何かを比較・検討する．

まず先ほど, クラスタリングには階層的クラスタリングと非階層クラスタリングがあることを述べた. 非階層クラスタリングを導入した場合，クラスタ数を予め設定する必要があるが，多種多様なデータを取り扱う SOM にとって，最適なクラスタ数を学習前からある程度把握しておくのは容易ではない. そこで決められたクラスタ数は主観によるところが大部分を占め，SOM の学習結果に客観性が失われることとなり，SOM の特徴の一つであるデータの潜在的な関連性を発見することに大きな影響を及ぼす. よって本研究では k-means 法のような非階層的クラスタリング手法ではなく，階層的クラスタリング手法を用いることとする．

次に，分類の感度，つまりクラスタのまとまりに着目する．最短距離法は分類感度が低く，正確性に欠ける部分がある．また，メディアン法や重心法はクラスタ間の距離が逆転する現象が起こってしまうことがある．先ほど説明した最長距離法と Ward 法は分類感度が高く，目的に適しているが，本研究ではその中でも比較的分類感度が高くまとまりのあるクラスタを得易い Ward 法を使用する．

6.2　デンドログラムの切断点

階層的クラスタリングではデンドログラムを生成し，クラスタ分類を行うが，最終的に切断点を決めてクラスタ数を決めなければならない．一般的には，図 5.1 のようなデータの分類が視覚的に分かる場合や，扱うデータ集合の性質などを考慮して，主観に基づいて切断することがあるが， SOM では客観性が出来るだけ得られることが望ましいので，本研究では切断点を決めるための指標を定める．

6.2.1 切断点の指標

今提案手法ではクラスタ個数を決定する指標として，Ward 法による階層的クラスタ形成時における郡内平方和の増加量を用いる．本来，郡内平方和の増加量は，クラスタ併合結果と元クラスタとの距離を取るものであり，それぞれのクラスタにおいて求めた増加量の比較を行い最小のクラスタ対を結合するという，クラスタ併合の指標に使用されるものである．

階層化が進むにつれ，郡内平方和の増加量が最小の値は徐々に大きくなっていく傾向はあるが，階層化をする過程で隣接クラスタとの距離が極端に遠くなる場合があり，その時の郡内平方和の増加量は，前結合と比べ極端に大きくなる．本研究ではその点に着目し，クラスタの全階層化の中で最小として選択された郡内平方和が極端に大きくなる階層を切断点とする指標を提案する．

・アルゴリズム

以下に今提案手法のアルゴリズムを示す．

1. 入力データを Ward 法により階層化クラスタリングを行い，クラスタを結合する
2. 結合されたクラスタの内，郡内平方和の増加量が最小のものを記録する
3. クラスタが一つに形成されるまで，1，2 を繰り返す
4. クラスタが一つに形成された時点で，記録した増加量が極端に大きくなる階層を切断点とする

6.3　視覚的に分類可能なデータに対する実験

視覚的に 3 個のクラスタに分類できる SOM の学習結果，視覚的に 5 個のクラスタに分類できる SOM の学習結果を用いて，分類実験を行った．以下に実験に使用したそれぞれの入力データの例を示す．

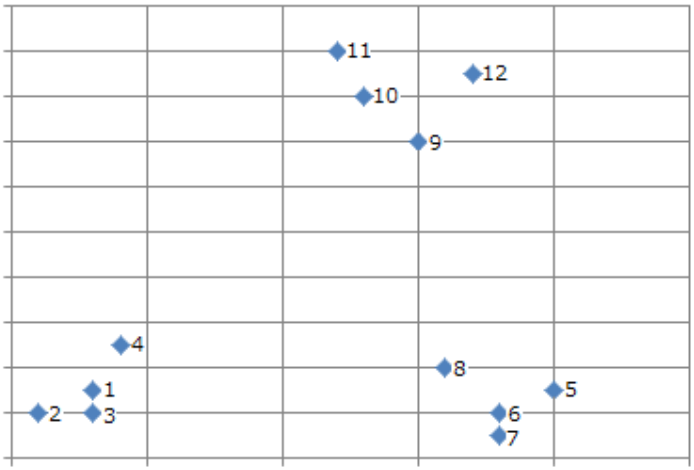

図 6.1　視覚的に 3 個のクラスタに分類できる SOM の学習結果（入力データ 1）

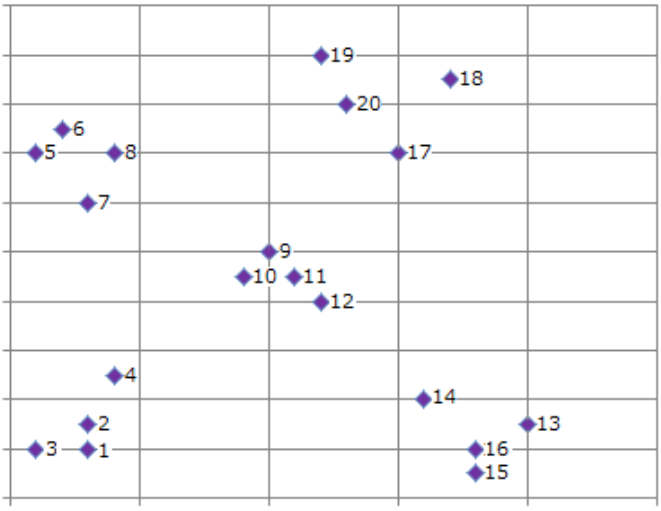

図 6.2　視覚的に 5 個のクラスタに分類できる SOM の学習結果（入力データ 2）

図 6.1 及び図 6.2 は，SOM の実験結果の例を表す．各色点が二次元平面上でのデータ位置となり，その右にはデータ番号を示している．

図 6.1 及び図 6.2 を視覚的にクラスタリングした場合，下図のようになる．

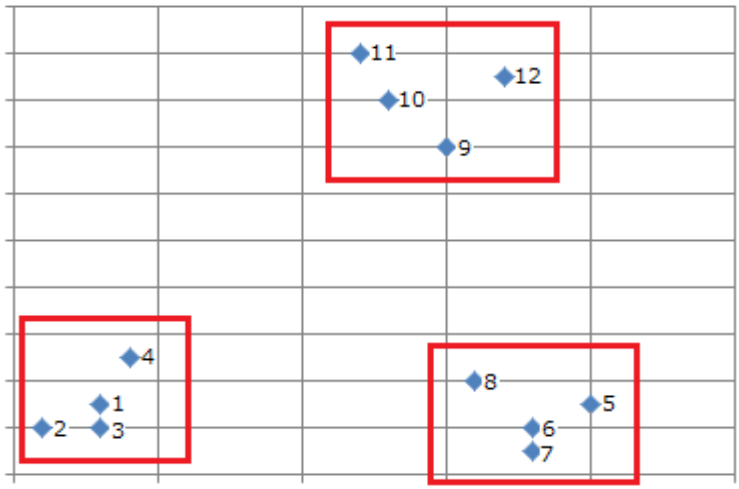

図 6.3　入力データ 1 の視覚的なクラスタリング

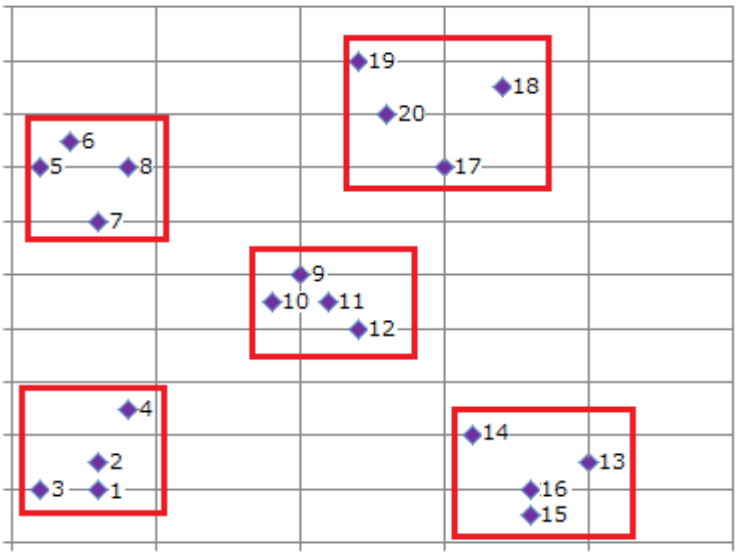

図 6.4　入力データ 2 の視覚的なクラスタリング

今実験では，図 6.1 及び図 6.2 のようなデータを提案手法によりクラスタリングし，その結果が図 6.3 及び図 6.4 のような視覚的なクラスタリングと同様な結果になることを確認する。

今実験の入力データとして，視覚的に 3 つのクラスタに分類可能な SOM の学習結果と，視覚的に 5 つのクラスタに分類可能な SOM の学習結果をそれぞれ 100 個用意し，それぞれのデータで 100 回ずつ計 200 回の反復実験を行う．入力データは X 値，Y 値の 2 次元の値を持ち，それぞれ 0〜25 の値を持つ．また，視覚的に 3 つのクラスタに分類可能な SOM の学習結果は 12 個の変数を持ち，視覚的に 4 個ずつ 3 つのクラスタを形成している．視覚的に 5 つのクラスタに分類可能な SOM の学習結果は 20 個の変数を持ち，視覚的に 4 個ずつ 5 つのクラスタを形成している．

実験結果は，自動クラスタリングによって形成されたクラスタ数を調べ，その平均及び標準偏差で表す．また，典型的な結果例として，図 6.1 及び図 6.2 のデータの実行結果を切断階層に点線を入れたデンドログラムで表す. デンドログラムの番号は各図のデータ番号に対応している．このような条件でクラスタリングをし，図 6.3，図 6.4 のような視覚的なクラスタリングと同様なクラスタリングができているのかを調べる．

6.4　視覚的に分類可能なデータに対する実験結果

以下の表に, それぞれの 100 回ずつの実験結果におけるクラスタ数の平均と標準偏差を示す.

表 6.1　視覚的に分類可能な SOM の学習結果に対するクラスタ数平均とその標準偏差

クラスタ数	平均	標準偏差
視覚的に 3 つのクラスタに分類可能なデータ	3.0	0.0
視覚的に 5 つのクラスタに分類可能なデータ	5.0	0.0

表 6.1 から，視覚的に 3 つのクラスタに分類可能な SOM の学習結果では，クラスタ数の平均が 3.0，標準偏差が 0.0 と，全てのデータに対して視覚的にクラスタリングを行った場合と同様の結果が得られたことがわかる. また視覚的に 5 つのクラスタに分類可能な SOM の学習結果でも同様に，クラスタ数の平均が 5.0，標準偏差が 0.0 と，全てのデータに対して視覚的にクラスタリングを行った場合と同様の結果が得られたことがわかる.

次に，典型的な結果例として，図 6.1 の入力データ 1 の実行時に得られたデンドログラムを図 6.5 に，図 6.2 の入力データ 2 の実行時に得られたデンドログラムを図 6.6 に示す. 数字はデータ番号を表し，図 6.5 は図 6.1 の番号，図 6.6 は図 6.2 の番号とリンクしている. 赤点線は切断点を表し，その高さ以下のクラスタ数に分割される.

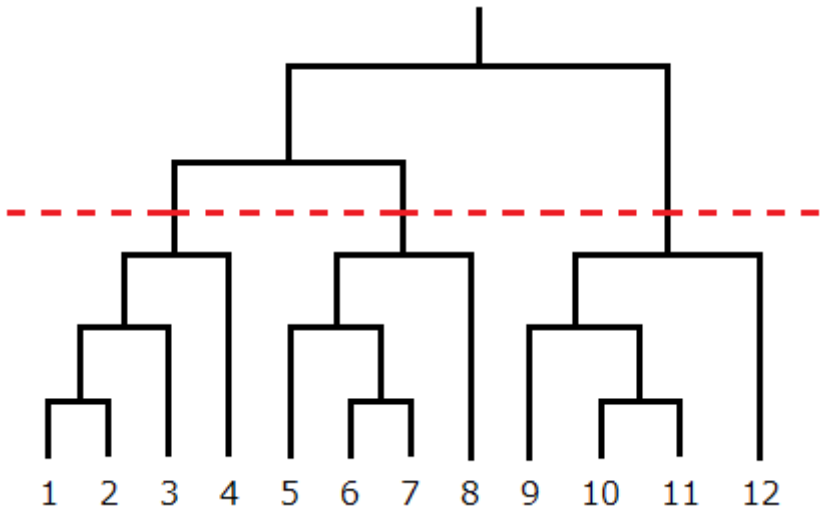

図 6.5　入力データ 1 の実行時に得られたデンドログラム

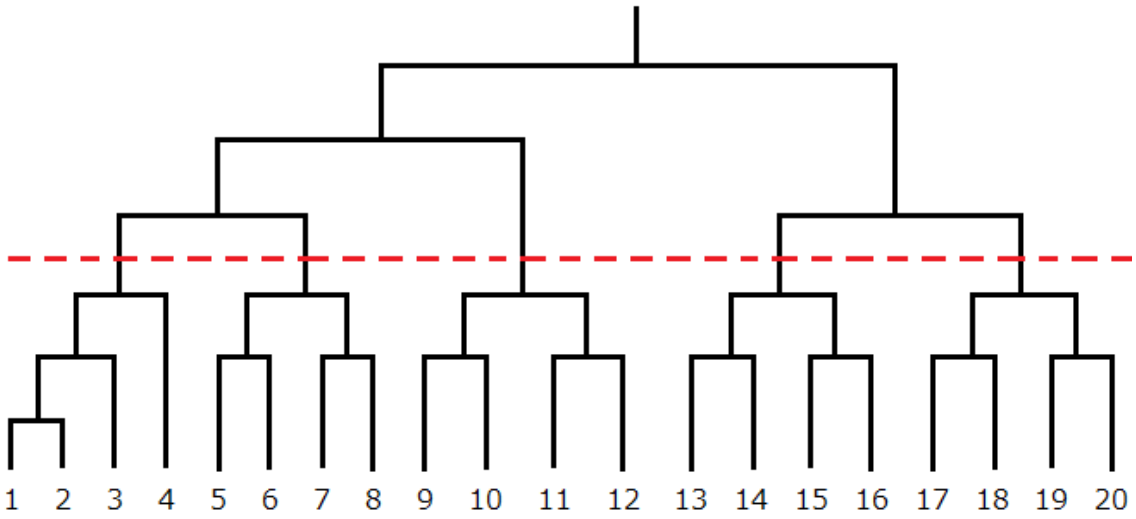

図 6.6　入力データ２の実行時に得られたデンドログラム

図 6.5 より，入力データ１に対するクラスタはデータ番号１～４，５～８，９～１２の 3 つに分かれており，図 6.3 の視覚的なクラスタリングと同様の結果が得られたことがわかる．同様に図 6.6 より，入力データ２に対するクラスタはデータ番号１～４，５～８，９～１２，１３～１６，１７～２０の 5 つに分かれており，図 6.4 の視覚的なクラスタリングと同様の結果が得られたことがわかる．これらのことから，クラスタリングの自動化が十分できているといえる．

6.5　視覚的に分類困難なデータに対する実験

次に，視覚的に分類が困難な SOM の学習結果を用いて，分類実験を行った．以下に実験に使用した入力データ例として，SOM の学習結果を示す．各色点が二次元平面上でのデータ位置となり，その右にはデータ番号を示している．

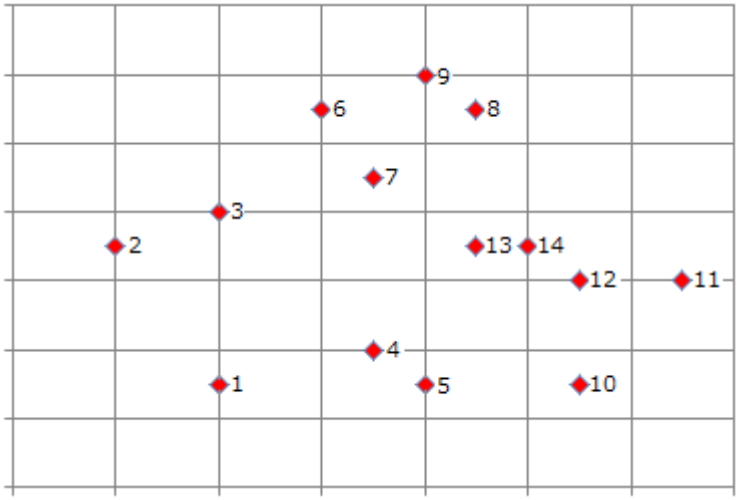

図 6.7　視覚的に分類が困難な SOM の学習結果（入力データ３）

今実験では，図 6.7 のような視覚的に分類が困難なデータを提案手法によりクラスタリングし，SOM の出力結果の視認性を向上させられるようなクラスタ数になるかを調べる。

今実験の入力データとして, 視覚的に分類が困難な SOM の学習結果を 100 個用意し，100 回の反復実験を行う．入力データは X 値，Y 値の 2 次元の値を持ち，14 個の変数がそれぞれ 0～25 の値を持つ.

実験結果は, 6.4 と同様に自動クラスタリングによって形成されたクラスタ数を調べ，その平均及び標準偏差で表す．また，典型的な結果例として，図 6.7 のデータの実行結果を切断階層に点線を入れたデンドログラムで表す．デンドログラムの番号は図 6.7 のデータ番号に対応している．

6.6　視覚的に分類困難なデータに対する実験結果

以下の表に，100 回の実験結果におけるクラスタ数の平均と標準偏差を示す.

表 6.2　視覚的に分類困難な SOM の学習結果に対するクラスタ数平均とその標準偏差

クラスタ数	平均	標準偏差
視覚的に分類困難なデータ	4.1	0.7

表 6.2 から，視覚的に分類困難な SOM の学習結果では，クラスタ数の平均が 4.1 となり，視覚的な分類がし易いクラスタ数になったと言える．また，標準偏差が 0.7 と，全てのデータにおいてクラスタ数にばらつきが少ないと言える.

次に，典型的な結果例として，図 6.8 に図 6.7 の入力データ 3 の実行時に得られたデンドログラムを示す．先ほどと同様に，数字はデータ番号を表し，図 6.7 の番号とリンクしている．赤点線は切断点を表し，その高さ以下のクラスタ数に分割される．また，図 6.9 に，図 6.8 の結果で分割した場合の二次元平面図を示す.

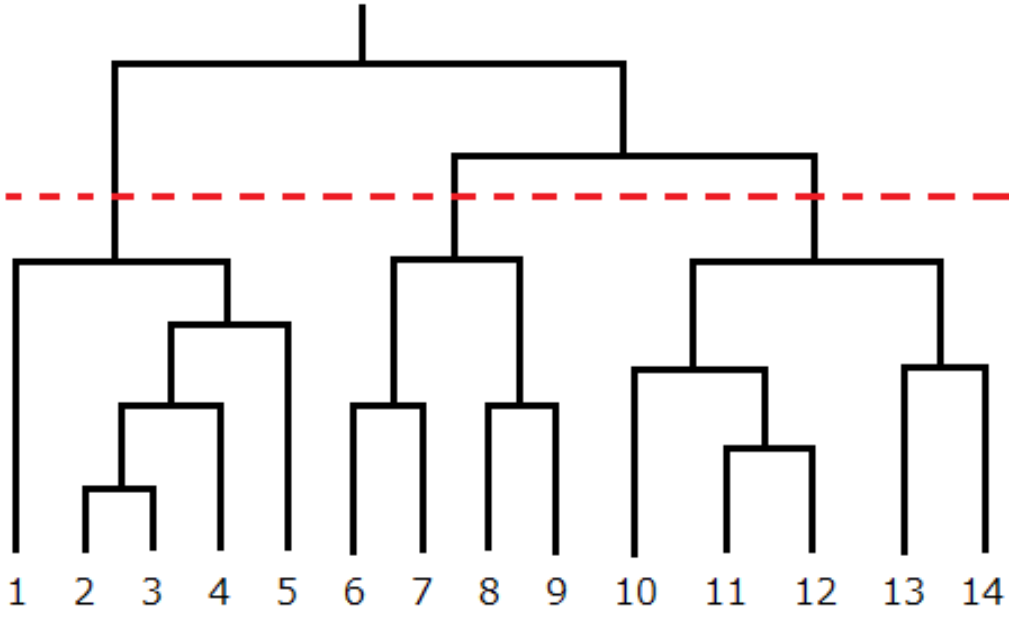

図 6.8　入力データ 3 の実行時に得られたデンドログラム

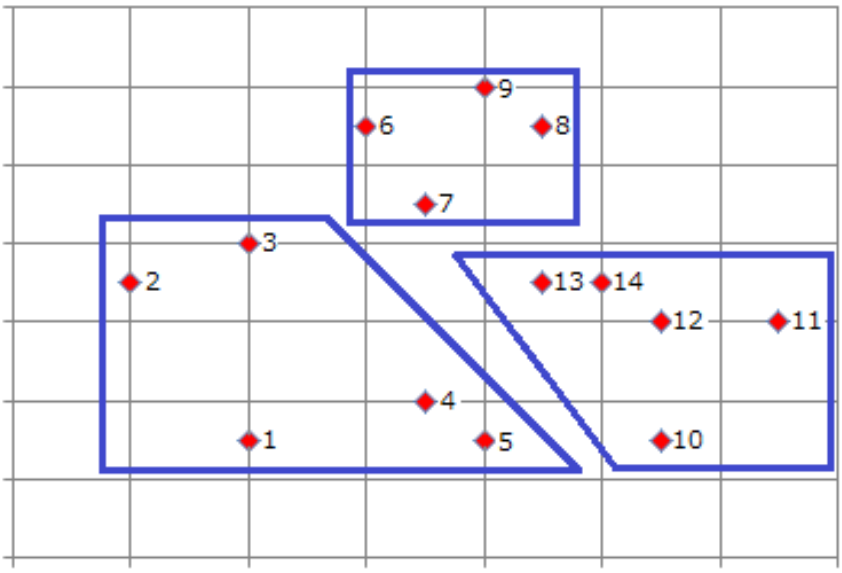

図 6.9　二次元平面上での分割結果

図 6.8 から，今回用いた入力データ３では，データ番号１～５，６～９，１０～１４の 3 つのクラスタに分割された．図 6.9 を見てみると，データの極端なばらつきは見られず，比較的まとまりのあるクラスタに分類されたと考えられる．

6.7 考察

6.4 及び 6.6 の実験結果を基に，考察を行う．視覚的に分類可能なクラスタリングでは，それぞれ 100 回の実験結果よりクラスタ数 3，クラスタ数 5 の両方の全てのデータに対して視覚的にクラスタリングを行った場合と同様の結果が得られたことがわかる．また典型的な実験結果例として示した入力データ 1 と入力データ 2 のクラスタリングでは，クラスタ内データ個数も，視覚的なクラスタリングと同様な結果が表れた．このことから，今提案手法は視覚的に分類可能であれば同様の結果を残すことが出来るといえる．

次に，視覚的に分類困難なクラスタリングでは，100 回の実験結果よりクラスタ数平均が 4.1，標準偏差が 0.7 と，視覚的理解がし易いと言えるクラスタ数になり，そのクラスタ数のばらつきも少ないと言える結果となった．また典型的な実験結果例として示した入力データ 3 では，クラスタ数が 3 という結果になった．入力データ 3 の場合，各データ間の距離に極端な差がなかったため，クラスタ数がデータ個数に近くなるほど多くなる可能性があったが，クラスタ数は 3 であった為，視覚的にも理解しやすい分類になったと考えられる．また，クラスタの形状について，Ward 法のまとまりのあるクラスタを形成しやすいという特徴から，極端な形になることなくクラスタが形成できていると考えられる．これらのことから，クラスタリングが困難な入力データの場合でも，視覚的な理解の支援になるようなクラスタリングが出来ているといえる．

第７章　おわりに

本研究は SOM の目的のひとつである多次元データの視覚的理解の向上を目的にし，SOM の出力結果の視覚的安定性の向上と，連続データの視覚的理解のためのクラスタリング能力向上を図った．

まず，SOM の出力結果の視覚的安定性向上の先行研究として，基礎手法の乱数による初期化の出力結果へのばらつきの影響を改善するための初期化範囲指定と特徴マップのソートを導入した手法があり，その問題点である高次元データの出力安定性向上を図った．ここでは学習データと特徴マップの両方を辞書式にソートすることで学習過程における傾向を強めるという手法を提案した．既存手法と提案手法の出力結果比較実験の考察より，既存手法に比べ提案手法の方が高次元データに対しても出力結果の安定性を保てていることが言える．

次に，データ同士の距離が一定に見える出力結果のようにデータの分類がし難いような結果に対し，SOM にクラスタリング手法を組み合わせた．SOM による学習では表現されないデータ間の違いを手掛かりに Ward 法によるクラスタリングを行い，郡内平方和の増加量を基に自動で適当な個数にクラスタを分割する手法を提案した．実験は視覚的クラスタリングが可能な入力データと困難な入力データ両方に対し提案手法によるクラスタリングを行い，クラスタの形成状況を確認した．考察より，視覚的にクラスタリングが可能な入力データに対しては同様のクラスタを形成することが言え，視覚的クラスタリングが困難な入力データに対しても視認性向上に適した結果を得ることが出来た．

これらの結果から，SOM の出力結果の視覚的安定性の向上と，連続データの視覚的理解のためのクラスタリング能力向上が出来たと言え，本研究の目的である SOM の多次元データの視覚的理解の向上を達成したと言える．

今後の展望として，本研究の手法を取り入れたインタラクティブなインタフェースによって，さらに多次元データに対する視覚的理解の向上に繋がることが期待される．

謝辞

本研究を進めるにあたり，様々なご指導を頂きました三好力教授に深謝いたします.
また発表を通じて多くの示唆を頂いた三好研究室の皆様に感謝します.

参考文献

1) Momoi Shinji, Miyoshi Tsutomu, “Improvement Method for Visual Stability of SOM’s Feature Map by Initial Value Assignment”, (IFSA-AFSS2011, OY-201, 2011).

2) Mu-Chun Su, Hsiao-Te Chang, “Fast Self-Organizing Feature Map Algorithm”, (IEEE Transactions on Neural Networks, 11(3):721--733, MAY 2000).

3) 三好力, “学習データによる初期ノード交換を用いた SOM の特徴マップ初期化”, (日本知能情報ファジィ学会誌 Vol.19 No.2 pp.167-175, 2007).

4) 志津綾香, 松田眞一, “クラスター分析におけるクラスター数自動決定法の比較”, (アカデミア 情報理工学編 Vol11 pp.17-34,2011).

5) 徳高平蔵, 大北正昭, 藤村喜久郎, “自己組織化マップとその応用”, (シュプリンガー・ジャパン株式会社, 2007).

6) 大北正昭, 徳高平蔵, 藤村喜久郎, 権田英功, “自己組織化マップとツール”, (シュプリンガー・ジャパン株式会社, 2008).

7) クラスタリングとは（クラスター分析とは）｜Toshihiro Kamishima http://www.kamishima.net/jp/clustering/

8) 階層的クラスター分析（Hiearchical Cluster Analysis) http://nakaikemi.com/clusterexp.htm

発表履歴

- Momoi Shinji, Miyoshi Tsutomu,
  “Improvement Method for Visual Stability of SOM’s Feature Map by Initial Value Assignment”,
  IFSA-AFSS 2011, OY-201, (2011).
- Shinji Momoi, Tsutomu Miyoshi,
  “Visual Stability Improvement of SOM’s Feature Map by Initial Value Assignment”
  FUZZ-IEEE 2011, #502, (2011).
- Shinji Momoi, Tsutomu Miyoshi,
  “Improvement of SOM Visual Stability by Adjusting Feature Maps and Sorting of Leaning Data”
  SCIS-ISIS 2012, W2-46-7, (2012)

Improvement method for visual stability of SOM feature map by initial value assignment

MOMOI Shinji
Department of Media Informatics,
Graduate School of Science and Technology,
Ryukoku University

MIYOSHI Tsutomu
Department of Media Informatics,
Faculty of Science and Technology,
Ryukoku University
mijosxi@i.ryukoku.ac.jp

Abstract: The location of the node or the distance between nodes on SOM feature map is important factor to determine feature of individual data. In many applications the judgment is made by a person depending on a location on the feature map. In conventional method, initial value of feature map has been decided at random, so it can be said that it lacks visual stability. In this paper, we focused on visual stability of SOM feature map, and we proposed new initialization method with less processing or less analysis of leaning data. By two types of experiments, we revealed that, proposed method is visually stable than conventional method in the point of feature map location, the computational complexity, and the standard deviation of winning node location.

1 INTRODUCTION

Kohonen's Self Organizing Map (SOM) involves neural networks, for which an algorithm learns the feature of input data through unsupervised, competitive neighborhood learning. The SOM is applied in many fields and has been widely studied. Based on the conventional SOM learning algorithm, SOM learning is influenced by the sequence of learning data and the initial feature map. The location of the node or the distance between nodes on feature map is important factor to determine feature of individual data. For example, in data detection of hematopoietic tumors, given data is detected by location of the feature map. If given data are near to the area of physically unimpaired person then the data consider to normal, and if the data are near to the area of tumors then the data consider to abnormal. As well as this example, in many applications the judgment is made by a person depending on a location on the feature map.

In conventional method, initial value of feature map has been decided at random, so a different mapping result appears even if using same input data. Consequently, it can be said that it lacks visual stability. In SOM application of medical diagnosis assistance, disease or health will be diagnosed with a visual impression of SOM feature map. If the learning results are visually unstable, maps must become different impressions in each learning, and maps of different impressions could be increased to the same data in different diagnosis.

To solve this visually unstable problem, there are steadies, that euclidean distance of learning data is pre-calculated and vector of feature map permute based on the result of pre-calclate. Because it is necessary to compare distance of all vectors, however, the computational complexity becomes very high, and a lot of time will be spent in analysis of learning data. As a result, generalization ability of SOM is not fully utilized. Moreover, the method focused on high speed leaning, so, there are not enough discussion about visual stability.

In this paper, we focused on visual stability of SOM feature map, and we proposed new initialization method with less processing or less analysis of leaning data.

2 SELF-ORGANIZING MAP

Kohonen's [1] self-organizing map (SOM) involves of neural networks, that learn the features of input data through unsupervised, competitive neighborhood learning. The SOM is mapping from high to low dimensional space, usually as a two-dimensional (2D) map. It provides a feature map that arranges similar classes nearer to one another to visualize high-dimensional information in a 2D feature map. Map representation makes it easier to understand relations between data. (Fig.1)

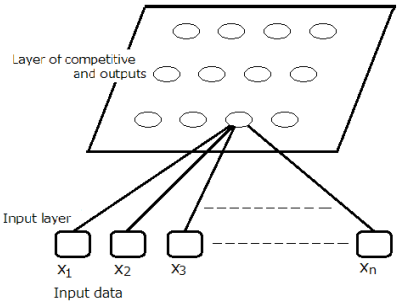

Fig.1 : SOM overview

The SOM generally has two layers, an input layer and an output layer. Output layer nodes usually form a 2D grid and input layer nodes are fully connected with those at output layer nodes. Each connection has a connection weight, so all output layer nodes have patterns or vectors to be learned.

After learning, each node represents a group of individuals with similar features, the individual corresponding to the same node or to neighboring nodes. That is, the SOM configures output nodes into a topological representation of original data through a process called self-organization.

In learning process, when an input pattern or input vector is presented to the input layer as learning data, output layer nodes compete mutually for the right to be declared the winner. The winning node is the output layer node whose incoming connection weights are the closest to the input pattern in Euclidean distance. The connection weights of the winning node and its neighbor nodes are then adjusted, i.e., moved closer toward the input pattern.

As learning process progresses, the learning rate and the size of the neighbor area around the winning node decreases, so in an early stage of learning process, large numbers of output layer nodes are adjusted strongly and, in the final stage, the winning node alone is adjusted weakly.

3 Problem of Conventional Technology

3.1 Initial Value of Reference Vector

Vector of SOM feature maps (reference vector) has been initializing by random number on conventional technology as described ahead. However, in learning process of SOM, simply initializing by random number has a big influence mainly first learning. The example of problem in case of simply initializing it at random is enumerated as follows.

First of all, there are large change of coordinates of winning node even if using same leaning data, because value of reference vector is selected by random number. For each learning, the location of winning node for the same leaning data is usually different, because initial vector is randomly set its values. The example is shown in Fig. 2.

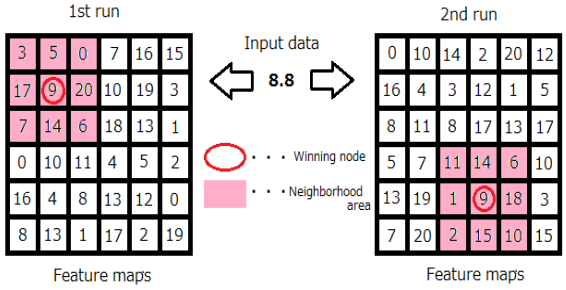

Fig. 2. Difference of coordinates of winning node by random number

In the example, feature maps is 6×6, input data is 8.8, and neighborhood area is one surrounding space. Whenever winning node becomes completely random, when value of reference vector is completely random. Vector surrounding winning node, i.e., vector in neighborhood area, also have random value, so, influence of learning is different in each leaning.

3.2 Data Pre-calucration Method

There are a method which initial value is calculated from input data. The method proposed by Mu-Chun Su [2] is enumerated as an example. (1) four vectors whose distance is the longest in input data are extracted. (2) it is allocated in the four corners of feature maps as weight. (3) data that is the furthest from both ends is allocated to opposite weight. (4) until all weights are assigned, this can be done clockwise repeatedly. As for the method of analyzing input data like this example and deciding initial value, the following problems are enumerated.

First of all, by this initialization method, leaning data are analyzed their feature, so SOM leaning is used to miner adjustment. The purpose to use SOM is to analyze what kind of relations are exist in input data. So the purpose might be lost by separately analyzing input data to decide initial value of reference vector.

Second, there is a problem that the computational complexity depends on the number of input data and number of dimensions. Therefore, a lot of procedures will be added before it learns by SOM.

Third, the method focused on high speed leaning, so, there are not enough discussion about visual stability.

3.3 Node Exchange Method

In the research of Miyoshi[3], they proposed initialization method that tried to utilize generalization ability of SOM. The method is to purpose at the speed-up study by improving the procedure of deciding initial value at random. This method relates input vector space to the position of feature maps in exchanging nodes of feature maps referring to learning data at initialization. The average moved distance of all nodes is shortened, and the speed-up of the learning speed is confirmed.

In point of view that utilize generalization ability of SOM, this method has same purpose, however, the purpose to improve visual stability of output result is not discussed enough in the research.

4 Proposed Method

The purposes of proposed method are (1) improvement of visual stability of feature map, and (2) utilization of generalization ability of SOM. To achieve second purpose, range of vector values are calculated from small number of learning data. And to achieve first purpose, reference vectors are sorted and allocated to give feature map location tendency.

4.1 Range of Values

To reduce the complexity of learning, we try to limit the range of random values. To determine the range of values, randomly selected from a small number of learning data and approximate the maximum and minimum values from these data. Then the random numbers are provisionally allocated as initial value of reference vector. The advantage of this method is shown below. By this algorithm, it is possible to make the range of reference vector correspond to the range of learning data, and the calculation cost doesn't increase so much.

4.2 Vector Sorting

To give location tendency to initial feature map, reference vector are sorted and allocated to feature map. First, the average of all dimensions value of each reference vector are calculated, and by using these averages as sorting key, reference vectors are sorted. Then, vectors are allocated to feature map in sorting order. As a result, the difference of winning node to same learning data is reduced. So, it is thought that output result is become steady.

5 Experiments with Simple Data

To compare visual stability between conventional method and proposed method, we performed following experiments. Conventional method is that of having initial value of reference vector by random numbers. The experiments are giving the same learning data to these two programs. In order to simplify the visual changes, bi-polarization data was used.

Experiment parameters are followings. Learning data set is 3*200, feature maps size is 30*30 nodes, learning repetition is 4000 times, neighborhood area is 20 (it decreased

from 20 to 1), and learning rate is 0.01*(neighborhood area size).

5.1 Leaning Results of Simple Data

The experimental results is shown in figure 4. The left hand side are results of conventional method and, the right hand side are results of proposed method.

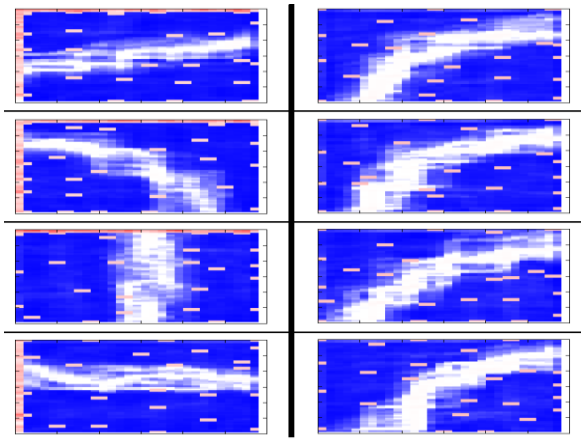

Fig. 4. Output result of existing technology and proposed technique

According to the results of conventional method, visually different feature maps are generated at every single learning even if leaning data is completely same. So, winning node of same input data is located in different part of each feature map. On the other hand, according to the results of proposed method, visually similar feature map are generated by same learning data. So, winning node of same input data is located in similar part of feature maps.

The computational complexity of data pre-calucration method is estimated to $O(n!)$, and the one of proposed method is estimated to $O(n \log n)$. So, it can be said that the computational complexity of proposed method is greatly reduced.

5.2 Stability Estimation of Simple Data

By above mentioned experiments, we revealed that proposed method is visually stable than conventional method. In this chapter, we revealed stability estimation by winning node location.

This experiment uses two data selected from 3*200 learning data that used in above experiments. Two selected data (index data) are shown as follows.

Table 1. index data

	The value of one-dimensional	The value of two-dimensional	The value of three-dimensional
Data_1	4	9	2
Data_2	43	44	43

The procedure of this experiment is shown below. While 4000 leaning, if the index data is selected as learning data, we keep record of its selection frequency and winning node location in feature map. Then, most frequently selected winning node location is detected from them. Finally, the standard deviation of the location was calculated. This experiment was done four times. Table 2 and Table 3 show the outcome of an experiment.

Table 2. Outcome of an experiment to Data_1

	Existing technology	Proposed method
1st run	(7, 3)	(6, 4)
2nd run	(21, 23)	(5, 7)
3rd run	(2, 29)	(1, 6)
4th run	(22, 1)	(8, 6)
Standard deviation	(8.22, 12.2)	(2.55, 1.09)

According to Table 2, followings are obtained. In case of proposed method, winning node locations are similar, but in conventional method, winning node locations are different in each run. The standard deviation of proposed method is smaller than the one of conventional method in both x and y axises, (8.74, 11.4) vs. (2.42, 1.10). Thereby, proposed method is shown to be steady by index data 1.

Table 3. Outcome of an experiment to Data_2

	Existing technology	Proposed method
1st run	(19, 18)	(22, 15)
2nd run	(10, 12)	(23, 16)
3rd run	(18, 1)	(24, 14)
4th run	(20, 28)	(23, 16)
Standard deviation	(4.68, 9.78)	(0.707, 0.829)

According to Table 3, followings are obtained. In case of proposed method, winning node locations are similar, but in conventional method, winning node locations are different in each run. The standard deviation of proposed method is smaller than the one of conventional method in both x and y axises, (4.68, 9.78) vs. (0.707, 0.829). Thereby, proposed method is shown to be steady by index data 2.

6 Experiments with the Iris Data

We have experimented with artificial data in Chapter 5. In this chapter we test the method on benchmark data, Iris data.

The following shows the configuration of the iris data. There are a total of 150 of data, 50 of "Set" label, 50 of "Ver" label, 50 of "Gnc" label. Each data is four-dimensional.

6.1 Learning Results of Iris Data

The experimental results is shown in figure 5. The left hand side are results of conventional method and, the right hand side are results of proposed method.

According to the results of conventional method, visually different feature maps are generated at every single learning even if leaning data is completely same. So, winning node of same input data is located in different part of each feature map. On the other hand, according to the results of proposed method, visually similar feature map are generated by same learning data. So, winning node of same input data is located in similar part of feature maps.

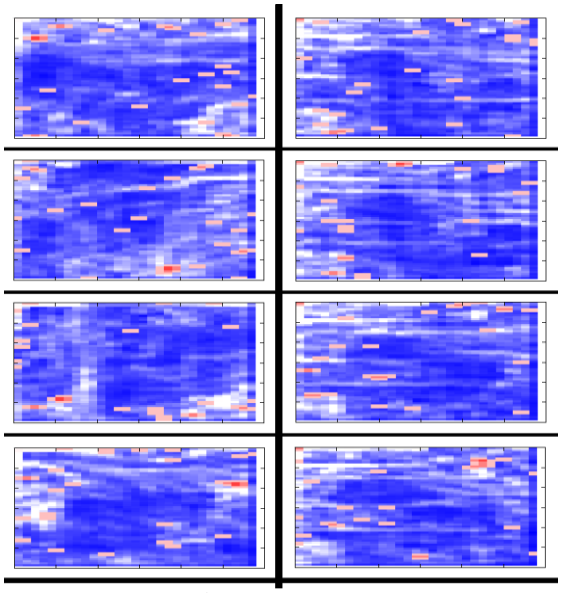

Fig. 5. Output result of existing technology and proposed technique

6.2 Stability Estimation of Iris Data

In this chapter, we revealed stability estimation by winning node location for the iris data. We select few data from each label "Set", "Ver" and "Gnc". The procedure of this experiment is same in chapter 5.2. This experiment was done four times. Table 4 Table 5 and Table 6 show the outcome of an experiment.

Table 4. Outcome of an experiment to Set

	Existing technology	Proposed method
1st run	(0, 16)	(0, 25)
2nd run	(20, 27)	(0, 22)
3rd run	(16, 1)	(0, 23)
4th run	(1, 19)	(0, 26)
Standard deviation	(8.99, 9.42)	(0.000, 1.58)

According to Table 4, in case of conventional method, winning node locations are vary widely. However, proposed method, winning node locations are similar. The standard deviation of proposed method is smaller than the one of conventional method in both x and y axises, (8.99, 9.42) vs. (0.000, 1.58).

Table 5. Outcome of an experiment to Ver

	Existing technology	Proposed method
1st run	(14, 29)	(18, 3)
2nd run	(3, 29)	(18, 3)
3rd run	(28, 14)	(18, 4)
4th run	(4, 2)	(19, 6)
Standard deviation	(10.1, 11.3)	(0.433, 1.22)

According to Table 5, The standard deviation of proposed method is lower than conventional method. Accordingly, dispersion of proposed method is few.

Table 6. Outcome of an experiment to Gnc

	Existing technology	Proposed method
1st run	(29, 20)	(29, 29)
2nd run	(0, 4)	(29, 29)
3rd run	(20, 29)	(29, 12)
4th run	(28, 0)	(29, 14)
Standard deviation	(11.6, 11.8)	(0.000, 8.03)

According to Table 6, The standard deviation of proposed method is lower than conventional method. Accordingly, dispersion of proposed method is few.

7 Conclusion

In this paper, we proposed new initialization method of SOM feature map. The purposes of proposed method are improvement of visual stability of feature map, and utilization of generalization ability of SOM. The range of reference vector values are calculated from maximum and minimum values approximated by randomly selected small number of learning data, and reference vectors are sorted and allocated to give location tendency in feature map.

By two types of experiments, followings are revealed that, proposed method is visually stable than conventional method in the point of feature map location, the computational complexity of proposed method is greatly reduced, and the standard deviation of the location in feature map of proposed method is smaller than the one of conventional method in both x and y axises. Then, it was able to be shown quantitatively that proposed method was steady. We also get the above results with using benchmark data.

As for further research, it is necessary to think about sorting method not related to number of dimensions. Sorting key which keep stability of feature map even if the dimension increases is preferable.

Reference

[1] Teuvo Kohonen, Self-Organizing Maps, Springer Verlag, 1995.

[2] Mu-Chun Su, Hsiao-Te Chang, "Fast Self-Organizing Feature Map Algorithm" IEEE Transactions on Neural Networks, Vol.11, No.3, pp.721-733, (2000-05).

[3] Tsutomu Miyoshi, "Relation Organization of SOM Initial Map by Improved Node Exchange," Journal of Computers (JCP), ISSN 1976-203X, Vol.3, No.9, pp.77-84,(2008-09).

Visual Stability Improvement of SOM's Feature Map by Initial Value Assignment

MOMOI Shinji
Department of Media Informatics,
Graduate School of Science and Technology,
Ryukoku University
t11m090@mail.ryukoku.ac.jp

MIYOSHI Tsutomu
Department of Media Informatics,
Faculty of Science and Technology,
Ryukoku University
mijosxi@i.ryukoku.ac.jp

***Abstract*— In SOM learning, learning result depends on initial value of feature map and the location of the node or the distance between nodes on feature map is important factor to determine feature of individual data. For example, in data detection of hematopoietic tumors, given data is detected by location of the feature map. In this paper, we forcused on visual stability of SOM feature map, and we proposed new initialization method of SOM feature map. The purposes of proposed method are improvement of visual stability of SOM feature map, and utilization of generalization ability of SOM. By experiments, proposed method is visually stable than conventional method in the point of feature map location, and the computational complexity of proposed method is greatly reduced.**

Keywords-component; self-organizing map; feature maps; visual stability; improvement method;

I. Introduction

Kohonen's Self Organizing Map (SOM) involves neural networks, for which an algorithm learns the feature of input data through unsupervised, competitive neighborhood learning. The SOM is applied in many fields and has been widely studied [4-7]. Based on the conventional SOM learning algorithm, SOM learning is influenced by the sequence of learning data and the initial feature map. The location of the node or the distance between nodes on feature map is important factor to determine feature of individual data.

In conventional method, initial value of feature map has set at random, so a different mapping appears even by same input data. Consequently, it can be said that it lacks visual stability. For example, in data detection of hematopoietic tumors, given data is detected by location of the feature map. If given data are near to the area of physically unimpaired person then the data consider to normal, and if the data are near to the area of tumors then the data consider to abnormal. As well as this example, in many applications the judgment is made by a person depending on a location on the feature map.

If the learning results are visually unstable, maps must become different impressions in each learning, and maps of different impressions could be increased to the same data in different diagnosis.

To solve this visually unstable problem, there are steadies, that euclidean distance of learning data is pre-calculated and vector of feature map permute based on the result of pre-calclate. Because it is necessary to compare distance of all vectors, however, the computational complexity becomes very high, and a lot of time will be spent in analysis of learning data. As a result, generalization ability of SOM is not fully utilized. Moreover, the method focused on high speed leaning, so, there are not enough discussion about visual stability.

In this paper, we focused on visual stability of SOM feature map, and we proposed new initialization method.

II. Self-Organizing Map

Kohonen's self-organizing map (SOM) [1] involves of neural networks, that learn the features of input data through unsupervised, competitive neighborhood learning. The SOM is mapping from high to low dimensional space, usually as a two-dimensional (2D) map. It provides a feature map that arranges similar classes nearer to one another to visualize high-dimensional information in a 2D feature map. Map representation makes it easier to understand relations between data. (Fig.1)

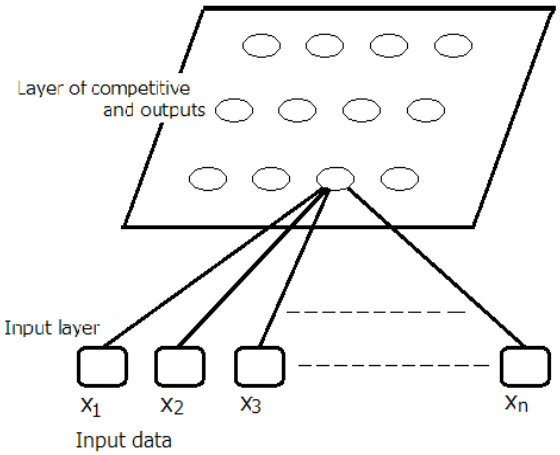

Figure 1. SOM overview

The SOM generally has two layers, an input layer and an output layer. Output layer nodes usually form a 2D grid and input layer nodes are fully connected with those at output layer nodes. Each connection has a connection weight, so all output layer nodes have patterns or vectors to be learned.

After learning, each node represents a group of individuals with similar features, the individual corresponding to the same node or to neighboring nodes. That is, the SOM configures output nodes into a topological representation of original data through a process called self-organization.

In learning process, when an input pattern or input vector is presented to the input layer as learning data, output layer nodes compete mutually for the right to be declared the winner. The winning node is the output layer node whose incoming connection weights are the closest to the input pattern in Euclidean distance. The connection weights of the winning node and its neighbor nodes are then adjusted, i.e., moved closer toward the input pattern.

As learning process progresses, the learning rate and the size of the neighbor area around the winning node decreases, so in an early stage of learning process, large numbers of output layer nodes are adjusted strongly and, in the final stage, the winning node alone is adjusted weakly.

III. Problem of Conventional Technology

A. Initial Value of Reference Vector

Vector of SOM's feature maps (reference vector) has been initializing by random number on conventional technology as described ahead. However, in learning process of SOM, simply initializing by random number has a big influence chiefly first learning. The example of possible problem in case of simply initializing it by random number is enumerated as follows.

First of all, there are large change of coordinates of winning node even if used same leaning data, because value of reference vector is selected by random number. In different learning, the location of winning node for the same leaning data is usually different, because initial vector is randomly set its values. The example is shown in figure 2.

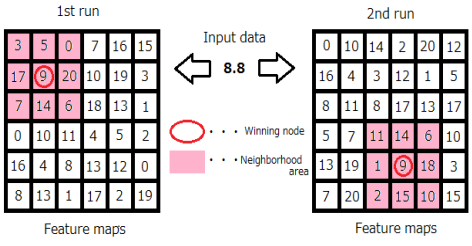

Figure 2. Difference of coordinates of winning node by random number

In the example, feature maps is 6×6, input data is 8.8, and neighborhood area is one surrounding space. Whenever winning node becomes completely random when value of reference vector is completely random like this. Vector surrounding winning node, i.e., vector in neighborhood area, also have random value, so, influence of learning is different in each leaning.

B. Data Pre-calculation Method

There are a method which initial value is calculated from input data. The method proposed by Mu-Chun Su [2] is enumerated as an example. (1) four vectors whose distance is the longest in input data are extracted. (2) it is allocated in the four corners of feature maps as weight. (3) data that is the furthest from both ends is allocated to opposite weight. (4) until all weights are assigned, this can be done clockwise repeatedly. As for the method of analyzing input data like this example and deciding initial value, the following problems are enumerated.

First of all, by this initialization method, learning data are analyzed their feature, so SOM leaning is used to miner adjustment. The purpose to use SOM is to analyze what kind of relations are exist in input data. The purpose might be lost by separately analyzing input data to decide initial value of reference vector.

Second, there is a problem that the computational complexity depends on the number of input data and number of dimensions. Therefore, a lot of procedures will be added before it learns by SOM.

Third, the method focused on high speed leaning, so, there are not enough discussion about visual stability.

C. Node Exchange Method

In the research of Miyoshi[3], they proposed initialization method that tried to utilize generalization ability of SOM. The method is to purpose at the speed-up study by improving the procedure of deciding initial value at random. This method relates input vector space to the position of feature maps in exchanging nodes of feature maps referring to learning data at initialization. The average moved distance of all nodes is shortened when this method is used, and the speed-up of the learning speed is confirmed.

In point of view that utilize generalization ability of SOM, this method has same purpose, however, the purpose to improve visual stability of output result is not discussed enough in the research.

IV. Proposed Method

The purposes of proposed method are (1) improvement of visual stability for output result, and (2) utilization of generalization ability of SOM. To achieve second purpose, renge of vector values are calculated from small number of learning data. And to achieve first pupose, reference vectors are sorted and allocated to give feature maps location tendency.

A. Range of Values

To reduce the complexity of learning, we try to limit the range of random values. To determine the range of values, randomly selected from a small number of learning data and approximate the maximum and minimum values for these data. Number of randomly selected data is determined by following formula:

$$n=\frac{N}{(\frac{E}{Z})^2(\frac{N-1}{P(1-P)})+1} \quad (1)$$

where, n is number of randomly selected data, N is a population, E is maximum error, Z is reliability coefficient, and P is population proportion. Using this formula, number of randomly selected data can be kept at a constant value.

Then the random numbers are provisionally allocated as initial value of reference vector. The advantage of this method is shown below. By this algorithm, it is possible to make the range of reference vector correspond to the range of learning data, and the calculation cost doesn't hang so much.

B. *Vector Sorting*

To give feature map location tendency, reference vector are sorted and allocated to feature map. First, the average of all dimentions value of each reference vector are calculated and, by using these averages as sorting key, reference vectors are sorted. Averages is caluclated by the following formula:

$$A=(m_{ij0}+m_{ij1}+m_{ij2}+\cdots+m_{ijk})/k \quad (2)$$

where, A is average, m is reference vector, i is x axis of feature map, j is y axis of feature map, and k is value of dimention.

Then, vectors are allocated to feature map in sorting order. The image of sorting is shown in figure 3.

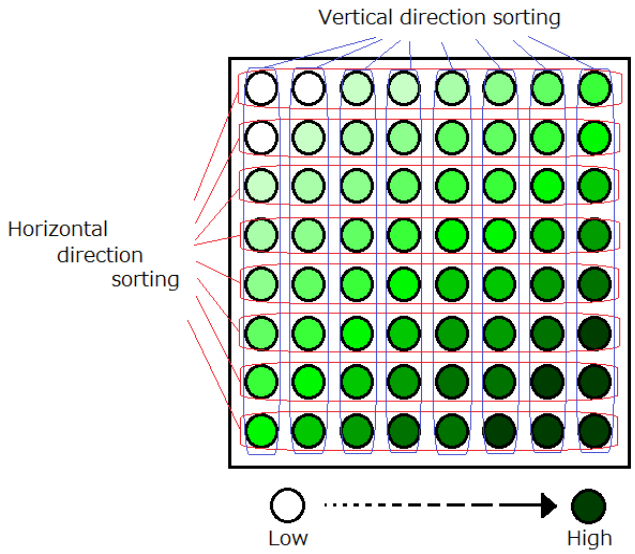

Figure 3. The image of sorting

As a result, the difference of winning node to same learning data is reduced. So, it is thought that output result is become steady.

C. *Algorithm of Feature Map Initialization*

Algorithm of Proposed Method is following.

- From the learning data, randomly select a value.
- From the selected value to determine the minimum and maximum values.
- random numbers are generated at the range of minimum and maximum values.
- Generated random number are temporarily assigned to the reference vector.
- To calculate the average value of the reference vectors.
- Based on the average, reference vector is done to quick sort in ascending order for the vertical direction feature map.
- Based on the average, reference vector is done to quick sort in ascending order for the horizontal direction feature map.
- Feature maps are made by this sorting, is initial value.

V. Experiments for Visual Stability

To compare visual stability between conventional method and proposed method, we performed experiments. Conventional method is that of having initial value of reference vector by random numbers. The experiments are giving the same learning data to these two programs. In order to simplify the visual changes, bi-polarization data was used. The image of learning data is shown in figure 4.

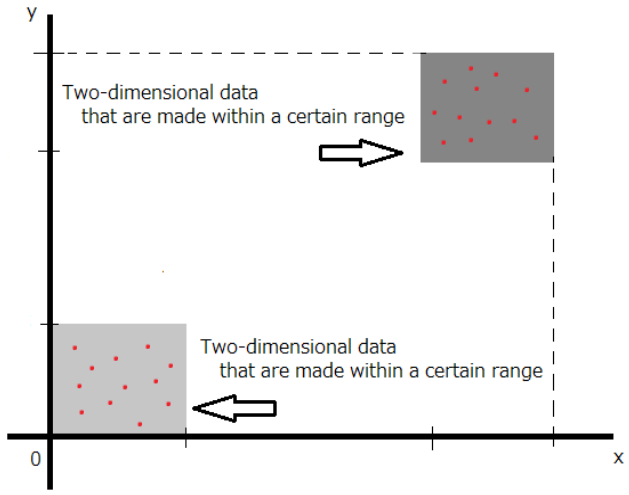

Figure 4. The image of learning data

Experiment parameters are followings. Learning data set is 3*200, feature maps size is 30*30 nodes, learning repetition is 4000 times, neighborhood area is 20 (it decreased from 20 to 1), and learning rate is 0.01*(neighborhood area size).

A. *Learning Results*

The experimental results is shown in figure 5. The left hand side are results of conventional method and, the right hand side are results of proposed method.

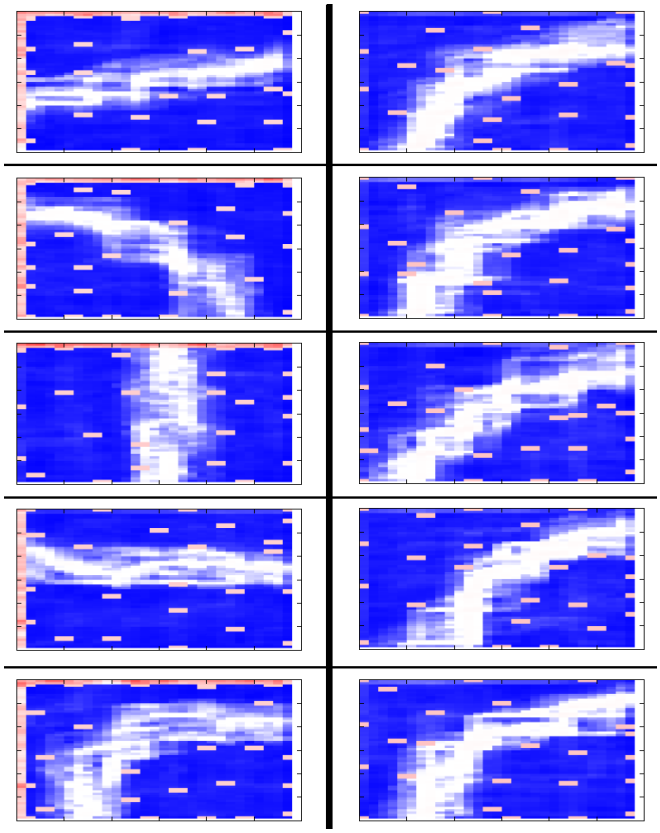

Figure 5. Output result of existing technology and proposal technique

According to the results of conventional method, visually different feature maps are generated at every single learning, even if leaning data is completely same. So, winning node of same input data is located in different part of each feature map. On the other hand, according to the results of proposed method, visually similar feature map are generated by same learning data. So, winning nodes of same input data are located in similar part of feature maps. The position of all input data showed similar tendency, typical winning nodes location are shows in figure 6.

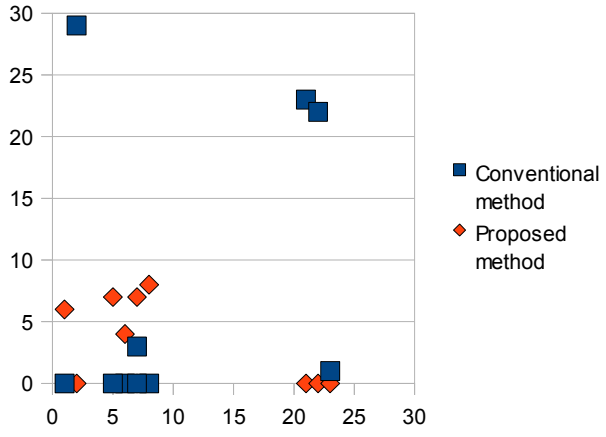

Figure 6. Feature map coordinates on a single input data

Figure 6 shows the location of winning nodes in five different feature maps for a single input data. X axis is x-coordinate and y axis is y-coordinate of feature map. According to Figure 6, winning node locations of conventional method are widely scattered even though input data is same. On the other hand, winning node locations of proposed method is almost unchanged corresponding to same input data.

The computational complexity of conventional method is estimated to $O(n!)$, and the one of proposed method is estimated to $O(n\log n)$. So, it can be said that the computational complexity of proposed method is greatly reduced.

VI. Conclusion

In this paper, we proposed new initialization method of SOM feature map. The purposes of proposed method are improvement of visual stability for output result, and utilization of generalization ability of SOM. The renge of reference vector values are calculated from maximum and minimum values approximated by randomly selected small number of learning data, and reference vectors are sorted and allocated to give location tendency in feature map.

By experiments, followings are revealed that, proposed method is visually stable than conventional method in the point of feature map location, the computational complexity of proposed method is greatly reduced.

As for further research, it is necessary to think about sorting method not related to number of dimensions. The average is sorting key in this time. However, sorting key to which stability of output result is kept even if the dimension increases is preferable. In addition, this study used a graph to show minutely the stability of the output information on input data. However, we need to think interface to output result more visually clear.

References

[1] Teuvo Kohonen, Self-Organizing Maps, Springer Verlag, 1995.

[2] Mu-Chun Su, Hsiao-Te Chang, “Fast Self-Organizing Feature Map Algorithm” IEEE Transactions on Neural Networks, Vol.11, No.3, 2000, pp.721-733.

[3] Tsutomu Miyoshi, “Relation Organization of SOM Initial Map by Improved Node Exchange,” Journal of Computers (JCP), ISSN 1976-203X, Vol.3, No.9, 2008, pp.77-84.

[4] SHIEH Shu-Ling, LIAO I-En, “A New Clustering Validity Index for Cluster Analysis Based on a Two-Level SOM” IEICE transactions on information and systems, Vol.92, No.9, 2009, pp.1668-1674.

[5] SHIEH Shu-Ling, LIAO I-En, HWANG Kuo-Feng, CHEN Heng-Yu, “An Efficient Initialization Scheme for SOM Algorithm Based on Reference Point and Filters” IEICE transactions on information and systems, Vol.92, No.3, 2009, pp.422-432.

[6] KATO Satoru, HORIUCHI Tadashi, ITOH Yshio, “A Study on Clustering Method by Self-Organizing Map and Information Criteria” Journal of Japan Society for Fuzzy Theory and Intelligent Informatics, Vol.21, No.4, 2009, pp.452-460.

[7] Jun YoungPyo, Yoon Hyunsoo, Cho JungWan, “L^* Learning: A Fast Self-Organizing Feature Map Learning Algorithm Based on Incremental Ordering” IEICE transactions on information and systems, Vol.76, No.6, 1993, pp.698-706.

Improvement of SOM visual stability by adjusting feature maps and sorting of leaning data

MOMOI Shinji
Department of Media Informatics,
Graduate School of Science and Technology,
Ryukoku University
t11m090@mail.ryukoku.ac.jp

MIYOSHI Tsutomu
Department of Media Informatics,
Faculty of Science and Technology,
Ryukoku University
mijosxi@i.ryukoku.ac.jp

***Abstract*— Based on the SOM learning algorithm, SOM learning is influenced by the sequence of learning data and the initial feature map. The location of the node or the distance between nodes on feature map is important factor to determine feature of individual data. In conventional method, initial value of feature map has set at random, so a different mapping appears even by same input data, so different impressions could be increased to the same data in different diagnosis. In this paper, we forcused on visual stability of SOM feature map, and we proposed two new initialization method of SOM feature map. The purposes of proposed method are improvement of visual stability of SOM feature map, and utilization of generalization ability of SOM. By some experiments with both artificial data and benchimark data, two proposed methods are visually stable than conventional method in the point of feature map location, and the computational complexity of proposed method is greatly reduced.**

Keywords-component; self-organizing map; feature maps; visual stability; improvement method;

I. INTRODUCTION

Kohonen's Self Organizing Map (SOM) involves neural networks, for which an algorithm learns the feature of input data through unsupervised, competitive neighborhood learning. The SOM is applied in many fields and has been widely studied [4-7]. Based on the conventional SOM learning algorithm, SOM learning is influenced by the sequence of learning data and the initial feature map. The location of the node or the distance between nodes on feature map is important factor to determine feature of individual data.

In conventional method, initial value of feature map has been set at random, so a different mapping appears even by same input data. Consequently, it can be said that it lacks visual stability. For example, in data detection of hematopoietic tumors, given data is detected by location of the feature map. If given data are near to the area of physically unimpaired person then the data consider to normal, and if the data are near to the area of tumors then the data consider to abnormal. As well as this example, in many applications the judgment is made by a person depending on a location on the feature map.

If the learning results are visually unstable, maps must become different impressions in each learning, and maps of different impressions could be increased to the same data in different diagnosis. If the initial value of the SOM is random, the learning results inevitably be visually random.

To solve this visually unstable problem, there are steadies [2], that Euclidean distance of learning data is pre-calculated and vector of feature map permute based on the result of pre-calculate. Because it is necessary to compare distance of all vectors, however, the computational complexity becomes very high, and a lot of time will be spent in analysis of learning data. As a result, generalization ability of SOM is not fully utilized. Moreover, the method focused on high speed leaning, so, there are not enough discussion about visual stability.

In this paper, we focused on visual stability of SOM feature map, and we proposed new initialization method. Then, in order to verify the proposed method, we experiment including the existing methods and comparison by using the artificial bi-polarization data and benchmark data.

II. SELF-ORGANIZING MAP

Kohonen's self-organizing map (SOM) [1] involves of neural networks, that learn the features of input data through unsupervised, competitive neighborhood learning. The SOM is mapping from high to low dimensional space, usually as a two-dimensional (2D) map. It provides a feature map that arranges similar classes nearer to one another to visualize high-dimensional information in a 2D feature map. Map representation makes it easier to understand relations between data. (figure.1)

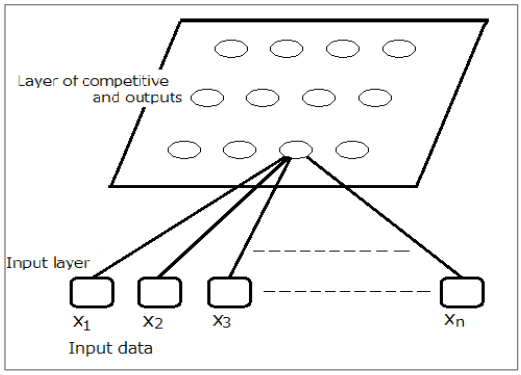

Figure 1. SOM overview

The SOM generally has two layers, an input layer and an output layer. Output layer nodes usually form a 2D grid and input layer nodes are fully connected with those at output layer nodes. Each connection has a connection weight, so all output layer nodes have patterns or vectors to be learned.

After learning, each node represents a group of individuals with similar features, the individual corresponding to the same node or to neighboring nodes. That is, the SOM configures output nodes into a topological representation of original data through a process called self-organization.

In learning process, when an input pattern or input vector is presented to the input layer as learning data, output layer nodes compete mutually for the right to be declared the winner. The winning node is the output layer node whose incoming connection weights are the closest to the input pattern in Euclidean distance. The connection weights of the winning node and its neighbor nodes are then adjusted, i.e., moved closer toward the input pattern.

As learning process progresses, the learning rate and the size of the neighbor area around the winning node decreases, so in an early stage of learning process, large numbers of output layer nodes are adjusted strongly and, in the final stage, the winning node alone is adjusted weakly.

III. Problem of Conventional Method

A. Initial Value of Reference Vector

Vector of SOM's feature maps (reference vector) has been initializing by random number on conventional method as described ahead. However, in learning process of SOM, simply initializing by random number has a big influence chiefly first learning. The example of possible problem in case of simply initializing it by random number is enumerated as follows.

First of all, there are large change of coordinates of winning node even if used same leaning data, because value of reference vector is selected by random number. In different learning, the location of winning node for the same leaning data is usually different, because initial vector is randomly set its values. The example is shown in figure 2.

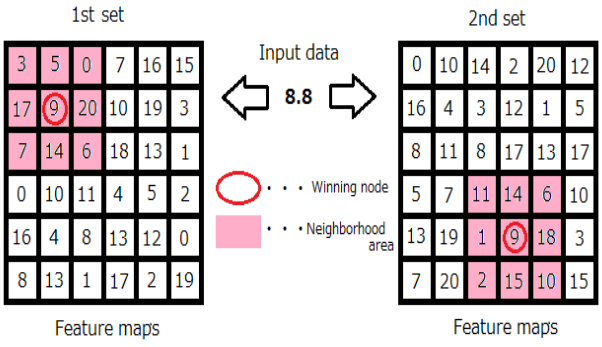

Figure 2. Difference of coordinates of winning node by random number

In the example, feature maps is 6×6, input data is 8.8, and neighborhood area is one surrounding space. Whenever winning node becomes completely random when value of reference vector is completely random like this. Vector surrounding winning node, i.e., vector in neighborhood area, also have random value, so, influence of learning is different in each leaning.

B. Data Pre-calculation Method

There are a method which initial value is calculated from input data. The method proposed by Mu-Chun Su [2] is enumerated as an example. (1) four vectors whose distance is the longest in input data are extracted. (2) it is allocated in the four corners of feature maps as weight. (3) data that is the furthest from both ends is allocated to opposite weight. (4) until all weights are assigned, this can be done clockwise repeatedly. As for the method of analyzing input data like this example and deciding initial value, the following problems are enumerated.

First of all, by this initialization method, learning data are analyzed their feature, so SOM leaning is used to minor adjustment. The purpose to use SOM is to analyze what kind of relations are exist in input data. The purpose might be lost by separately analyzing input data to decide initial value of reference vector.

Second, there is a problem that the computational complexity depends on the number of input data and number of dimensions. Therefore, a lot of procedures will be added before it learns by SOM.

Third, the method focused on high speed learning, so, there are not enough discussion about visual stability.

C. Node Exchange Method

In the research of Miyoshi[3], they proposed initialization method that tried to utilize generalization ability of SOM. The method is to purpose at the speed-up study by improving the procedure of deciding initial value at random. This method relates input vector space to the position of feature maps in exchanging nodes of feature maps referring to learning data at initialization. The average moved distance of all nodes is shortened when this method is used, and the speed-up of the learning speed is confirmed.

In point of view that utilize generalization ability of SOM, this method has same purpose, however, the purpose to improve visual stability of learning result is not discussed enough in the research.

IV. Proposed Method - 1

The purposes of proposed method are (1) improvement of visual stability of learning result, and (2) utilization of generalization ability of SOM. To achieve second purpose, renge of vector values are calculated from small number of learning data. And to achieve first purpose, reference vectors are sorted and allocated to give feature maps location tendency.

A. *Range of Values*

To reduce the complexity of learning, we try to limit the range of random values. To determine the range of values, randomly selected a small number of learning data and approximate the maximum and minimum values from these data. Number of randomly selected data is determined by following formula:

$$n = \frac{N}{\left(\frac{E}{Z}\right)^2 \left(\frac{N-1}{P(1-P)}\right) + 1} \quad (1)$$

where, n is number of randomly selected data, N is a population, E is maximum error, Z is reliability coefficient, and P is population proportion. Using this formula, number of randomly selected data can be kept at a constant value.

Then the random numbers are provisionally allocated as initial value of reference vector. The advantage of this method is shown below. By this algorithm, it is possible to make the range of reference vector correspond to the range of learning data, and the calculation cost doesn't hang so much.

B. *Vector Sorting*

To give feature map location tendency, reference vector are sorted and allocated to feature map. First, the average of all dimentions value of each reference vector are calculated and, by using these averages as sorting key, reference vectors are sorted. Averages is calculated by the following formula:

$$A = \left(m_{ij0} + m_{ij1} + m_{ij2} + \cdots + m_{ijk}\right)/k \quad (2)$$

where, A is average, m is reference vector, i is x axis of feature map, j is y axis of feature map, and k is value of dimention.

Then, vectors are allocated to feature map in sorting order. The image of sorting is shown in figure 3.

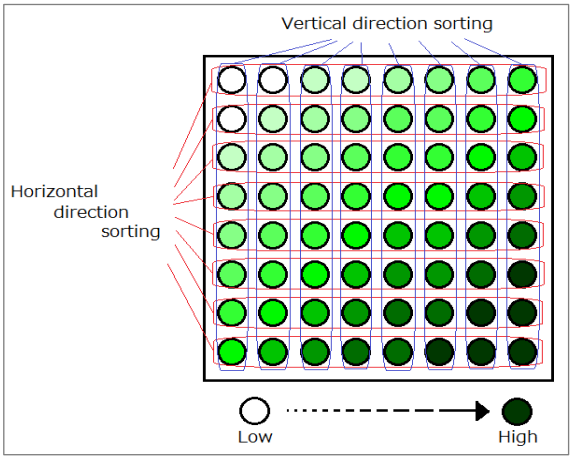

Figure 3. The image of sorting

As a result, the difference of winning node to same learning data is reduced. So, it is thought that learning result is become steady.

V. EXPERIMENTS FOR METHOD - 1

To compare visual stability between conventional method and proposed method, we performed experiments.

Conventional method is that of having initial value of reference vector by random numbers. The experiments are giving the same learning data to these two programs. In order to simplify the visual changes, bi-polarization data was used.

Experiment parameters are followings. Learning data set is 3*200, feature maps size is 30*30 nodes, learning repetition is 4000 times, neighborhood area is 20 (it decreased from 20 to 1), and learning rate is 0.01*(neighborhood area size).

A. *Computational Complexxity*

The computational complexity of data pre-calucration method is estimated to $O(n!)$, and the one of proposed method is estimated to $O(n\log n)$. So, it can be said that the computational complexity of proposed method is greatly reduced.

B. *Nodes Mapping*

The experimental results is shown in figure 4. The left hand side are results of conventional method and, the right hand side are results of proposed method.

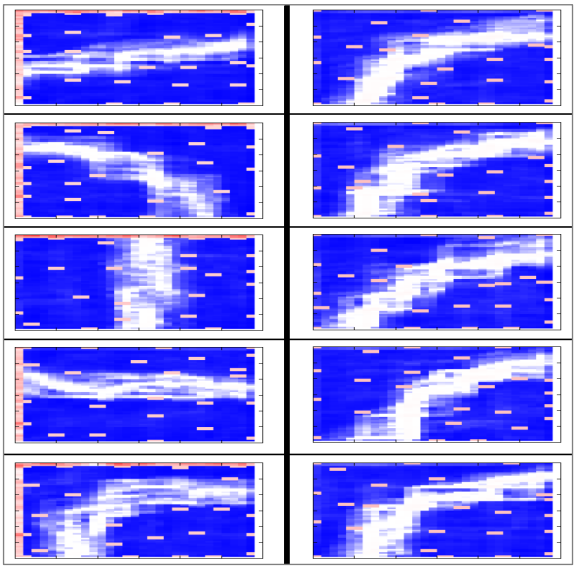

Figure 4. Learning result of conventional method and proposal method

According to the results of conventional method, visually different feature maps are generated at every single learning even if leaning data is completely same. So, winning node of same input data is located in different part of each feature map. On the other hand, according to the results of proposed method, visually similar feature map are generated by same learning data. So, winning node of same input data is located in similar part of feature maps.

C. *Stability Estimation Using Index Data*

By above mentioned experiments, we revealed that proposed method is visually stable than conventional method.

In this chapter, we revealed stability estimation by winning node location.

This experiment uses data selected from both of each data group and 3*200 learning data that used in above experiments. Selected index data are shown in table 1.

Table 1. index data

data1	(48, 46, 48)

The procedure of this experiment is shown below. While 4000 leaning, 1) if the index data is selected as learning data, we keep record of its selection frequency and winning node location in feature map. 2) Then, most frequently selected winning node location is detected from them. 3) Finally, the standard deviation of the location was calculated.

To compared with conventional and proposed methods, the most wins coordinates of index data are shown as follows.

The experimental results of the data1 is shown in figure 5. This experiment was done 100 times, and depicted the respective X and Y coordinates. The vertical axis is position, blue square points are coordinates of X, red diamond shape points are coordinates of Y, and the horizontal axis shows the number of experiments. If the points are gathering for each color, coordinate or position have the stable trend. However, if the points are distributing for each color, coordinate or position have the unstable trend. The left graph of figure 5 shows the conventional methods, the right graph shows the proposed method-1. Table 2 shows the standard deviation of each cooridinates.

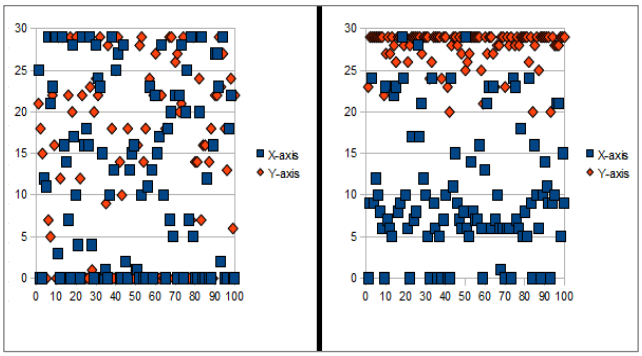

Figure 5. Comparison of data 1

Table 2. The standard deviation for the coordinates about data1

	Conventional methods	Proposed method-1
Standard deviation	(10.9, 11.1)	(7.75, 2.29)

According to figure 5 and table 2, followings are obtained. In case of proposed method, winning node locations are similar, but in conventional method, winning node locations are distributing in each run. The standard deviation of proposed method is smaller than the one of conventional method in both X and Y axes, (10.9, 11.1) vs. (7.75, 2.29). Thereby, proposed method is shown to be stable by index data1.

VI. Proposed Method-2

We showed that the output of the proposed method-1 is stable than the one of the conventional method with simple artifitial data. However, the proposed method 1 has a problem that lose the stability in inverse proportion to the number of dimentions of data. If the number of dimensions increase, the output becomes similar to the feature map that learned by conventional method. Therefore, we tried to support a large number of dimensions, we propose the following methods.

Rule **1.** Lexicographically sorting initial value of feature map

Rule **2.** Lexicographically sorting the learning data

Describe about rule 1. The proposed method-1 has not been considered for the order of the same average vectors. In this proposed method, if there are the vectors that average of each dimension value are same, we compared from the first dimension value of the vectors and the vectors are sorted in ascending order. The same average vector value often occur in a narrow range of the random number generated to give a value to the feature map, so, this approach can give a tendency to initial value of feature map even in a narrow range.

Describe about rule 2. In proposed method-1, learning data were randomly extracted in learning process. In this proposed method, learning data are extracted in order of lexicographically sorted. Trend also appears in the learning process by this method. As a result, we anticipate the stability of the output is further increased.

VII. Experiments for Method - 2

To compare stability between proposed method-1 and proposed method-2, we performed following experiments.

The experiments are giving the same learning data to these two programs. In order to simplify the visual changes, bi-polarization data was used like a section 5. But dimensions are 8.

This experiment uses data selected from 8*200 learning data that used in above experiments. Selected data (index data) are shown as follows.

Table 3. index data

data2	(50, 50, 46, 48, 45, 44, 46, 44)

The procedure of this experiment is shown below. While 4000 leaning, if the index data is selected as learning data, we keep record of its selection frequency and winning node location in feature map. Then, most frequently selected winning node location is detected from them.

Experiment parameters are followings. Learning data set is 8*200, feature maps size is 30*30 nodes, learning repetition is 4000 learning, neighborhood area is 20 (it decreased from 20 to 1), and learning rate is 0.01*(neighborhood area size).

A. *Stability Estimation*

About the learning data for the comparison of the two, the coordinates of selected nodes with the most wins are shown as follows.

The experimental results about the data2 is shown in figure 6. How to view a graph is the same as in figure 5.

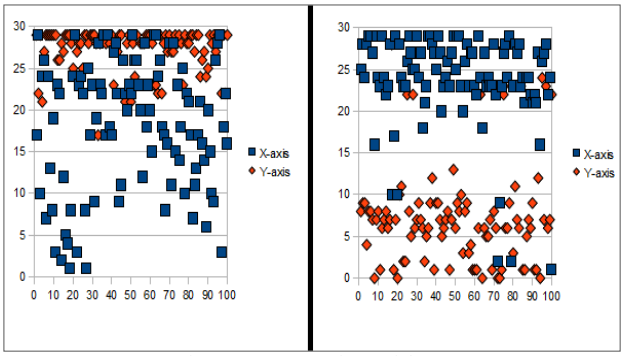

Figure. 6. Comparison of data2

According to figure 6, followings are obtained. In case of proposed method-2, winning node locations are similar, but in proposed method-1 winning node locations are distributing in each run. Thereby, proposed method-2 is shown to be stable by index data2.

VIII. EXPERIMENTS WITH BENCHMARK DATA

We were showing the effectiveness of the proposed method with artificial bi-polarization data. In order to verify the effectiveness of real-world, in this chapter we performed following experiments with the benchmark data.

To compare stability between proposed method-1 and proposed method-2, we performed following experiments. The experiments are giving the same learning data to these two programs. This time as general data, we use the benchmark data. This experiment uses data selected from four-dimensional benchmark data. Selected data (index data) are shown as follows.

Table 4. index data

data3	(0.000, 0.417, 0.017, 0.000)

We test the method on benchmark data, Iris data. The following shows the configuration of the iris data. There are total of 150 of data, 50 of "Set" label, 50 of "Ver" label, 50 of "Gnc" label. Each data is four-dimensional. Data3 of table 4 was extracted from the Set label as typical data.

The procedure of this experiment is shown below. While 4000 leaning, if the index data is selected as learning data, we keep record of its selection frequency and winning node location in feature map. Then, most frequently selected winning node location is detected from them.

Experiment parameters are followings. Learning data set is 4*150, feature maps size is 30*30 nodes, learning repetition is 4000 times, neighborhood area is 20 (it decreased from 20 to 1), and learning rate is 0.01*(neighborhood area size).

A. *Stability Estimation*

About the learning data for the comparison of the two, the coordinates of selected nodes with the most wins are shown as follows.

The experimental results about the data3 is shown in figure 7. How to view a graph is the same as in figure 5. table 5 shows the standard deviation of each cooridinates.

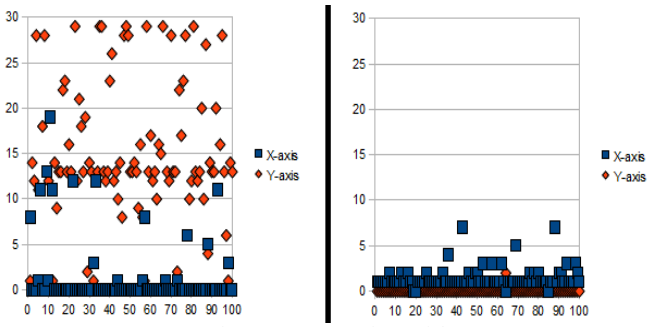

Figure. 7. Comparison of data3

Table 5. The standard deviation for the coordinates about data3

	Proposed method-1	Proposed method-2
Standard deviation	(3.51, 8.34)	(1.09, 0.24)

According to figure 7 and table 5, followings are obtained. In case of method-2, winning node locations are similar, but in method-1, winning node locations are distributing in each run. The standard deviation of proposed method is smaller than the one of conventional method in both X and Y axes, (3.51, 8.34) vs. (1.09, 0.24). Thereby, method-2 is shown to be stable by index data3.

IX. CONCLUSION

In this paper, we proposed new initialization method of SOM feature map. The purposes of proposed method are improvement of visual stability for learning result, and utilization of generalization ability of SOM. The renge of initial reference vector values are calculated from maximum and minimum values approximated by randomly selected small number of learning data, and reference vectors are sorted and allocated to give location tendency in feature map. By two types of experiments, followings are revealed that, proposed method is visually stable than conventional method in the point of feature map location, the computational complexity of proposed method is greatly reduced, and the standard deviation of the location in feature map of proposed method is smaller than the one of conventional method in both X and Y axes. Then, it was able to be shown quantitatively that proposed method was steady.

In addition, in order to improve the stability of the learning result for high-dimensional training data, we proposed a method to sort lexicographically feature map and learning data. Then, we compared the stability experiment results using the 8-dimensional training data. As a result, we show that the proposed method-2 is more stable than proposed method-1. Finally, in order to be valid our proposed methods by general

data, we experimented comparative of method-1 and method-2 using the benchmark data.

As for further research, it is necessary to consider an approach such we can be output to maintain the stability of higher-dimensional learning data.

REFERENCES

1 Teuvo Kohonen, Self-Organizing Maps, Springer Verlag, 1995.

2 Mu-Chun Su, Hsiao-Te Chang, "Fast Self-Organizing Feature Map Algorithm" IEEE Transactions on Neural Networks, Vol.11, No.3, 2000, pp.721-733.

3 Tsutomu Miyoshi, "Relation Organization of SOM Initial Map by Improved Node Exchange," Journal of Computers (JCP), ISSN 1976-203X, Vol.3, No.9, 2008, pp.77-84.

4 SHIEH Shu-Ling, LIAO I-En, "A New Clustering Validity Index for Cluster Analysis Based on a Two-Level SOM" IEICE transactions on information and systems, Vol.92, No.9, 2009, pp.1668-1674.

5 SHIEH Shu-Ling, LIAO I-En, HWANG Kuo-Feng, CHEN Heng-Yu, "An Efficient Initialization Scheme for SOM Algorithm Based on Reference Point and Filters" IEICE transactions on information and systems, Vol.92, No.3, 2009, pp.422-432.

6 KATO Satoru, HORIUCHI Tadashi, ITOH Yshio, "A Study on Clustering Method by Self-Organizing Map and Information Criteria" Journal of Japan Society for Fuzzy Theory and Intelligent Informatics, Vol.21, No.4, 2009, pp.452-460.

7 Jun YoungPyo, Yoon Hyunsoo, Cho JungWan, "L^* Learning: A Fast Self-Organizing Feature Map Learning Algorithm Based on Incremental Ordering" IEICE transactions on information and systems, Vol.76, No.6, 1993, pp.698-706.

付録　ソースコード

ソースコード１：　SomOutNewDic.java

```
package som;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StreamTokenizer;
import java.text.DecimalFormat;
import java.util.Random;

public class SomOutNewDic {
    static DecimalFormat df = new DecimalFormat();
    static int mapsize = 30;
    static double[][] w_average = new double[mapsize][mapsize];
    static int sort_nx;
    static double[][][] w = new double[mapsize][mapsize][1000];
    static double[][] wAve = new double[mapsize][mapsize];

    public static void main(String args[]) throws IOException{
        Random rand = new Random();
        double[][] u = new double[mapsize][mapsize];
        int winx=0, winy=0;
        double a = 0.01;
        int nearcount = 0;
        int near = 20;
        int nt;
        int learncount=4000;
        double[][] ux = new double[mapsize][mapsize];
        double[][] uy = new double[mapsize][mapsize];
        int inputsize = 150;
        int sampling_min = 0;
        int[][] wincount_min = new int[mapsize][mapsize];
        int sampling_mid = 0;
        int[][] wincount_mid = new int[mapsize][mapsize];
        int sampling_max = 0;
        int[][] wincount_max = new int[mapsize][mapsize];

        for(int i=0 ; i<mapsize ; i++){
            for(int j=0 ; j<mapsize ; j++){
                wincount_min[i][j]=0;
                wincount_max[i][j]=0;
            }
        }

        int keta[] = new int[3];
        keta[0]=0;keta[1]=0;keta[2]=0;
        ketaOutput(keta);

        double m[][]=new double[5000][100];
        int nx=0, ny=0;
        try{
            FileReader fr=new FileReader("C:/som-data-file/Iris/iris_M.data");
            StreamTokenizer st=new StreamTokenizer(fr);
            st.eolIsSignificant(true);

            while(st.nextToken()!=StreamTokenizer.TT_EOF){
                switch(st.ttype){
                case StreamTokenizer.TT_NUMBER:
```

```
                m[ny][nx]=st.nval;
                System.out.print(m[ny][nx]+ " |");
                nx++;
                break;

            case StreamTokenizer.TT_EOL:
                ny++;
                nx=0;
                System.out.println("");
                break;
            }
        }
        System.out.println("¥n¥n");
        fr.close();
}
catch(Exception e){
        System.out.println(e);
}
sort_nx=nx;
System.out.println(ny+","+nx);

double m_res[][]=new double[1000][1000];
int nx_res=0, ny_res=0;
try{
        FileReader fr=new FileReader("C:/som-data-file/Iris/iris_Input.data");
        StreamTokenizer st=new StreamTokenizer(fr);
        st.eolIsSignificant(true);

        while(st.nextToken()!=StreamTokenizer.TT_EOF){
            switch(st.ttype){
            case StreamTokenizer.TT_NUMBER:
                m_res[ny_res][nx_res]=st.nval;
                System.out.print(m_res[ny_res][nx_res]+ " |");
                nx_res++;
                break;

            case StreamTokenizer.TT_EOL:
                ny_res++;
                nx_res=0;
                System.out.println("");
                break;
            }
        }
        System.out.println("¥n¥n");
        fr.close();
}
catch(Exception e){
        System.out.println(e);
}

int large_x=0, large_y=0;
int small_x=0, small_y=0;
int check[][] = new int[ny+1][nx];
for(int i=0 ; i<ny+1 ; i++){
        for(int j=0 ; j<nx ; j++){
            check[i][j] = 0;
        }
}

int N = (ny+1)*nx;
double E = 0.05;
double Z = 1.96;
double P = 0.5;
int n;
n=(int)(N/((Math.pow(E/Z,2)*((N-1)/(P*(1-P))))+1));
```

```
int R_input_ny, R_input_nx;
int overlap=0;
for(int i=0 ; i<n ; i++){
    overlap++;
    R_input_ny = (int) Math.floor(Math.random()*(ny+1));
    R_input_nx = (int) Math.floor(Math.random()*(nx));
    switch(check[R_input_ny][R_input_nx]){
    case 0:
        if(m[large_x][large_y] < m[R_input_ny][R_input_nx]){
            large_x = R_input_ny;
            large_y = R_input_nx;
        }
        if(m[small_x][small_y] > m[R_input_ny][R_input_nx]){
            small_x = R_input_ny;
            small_y = R_input_nx;
        }
        check[R_input_ny][R_input_nx] = 1;
        break;

    case 1:
        i--;
        break;
    }
}
double r_max=m[large_x][large_y], r_min=m[small_x][small_y];

sortInputMethod(m,ny);
aveMethod(r_max, r_min);
sortXMethod();
sortYMethod();
printMethod();

int turn=0;
while(learncount>0){
    nt = turn;
    if(turn>=ny){ turn=0;}
    else{ turn++;}

    double sum=0;
    for(int i=0 ; i<mapsize ; i++){
        for(int j=0 ; j<mapsize ; j++){
            for(int k=0; k<nx; k++){
            sum = sum + Math.pow((w[i][j][k] - m[nt][k]),2);
            }
            u[i][j] = Math.sqrt(sum);
            sum=0;
            df.applyLocalizedPattern("0.00");
        }
    }


    for(int i=0 ; i<mapsize ; i++){
        for(int j=0 ; j<mapsize ; j++){
                if(u[winx][winy]>u[i][j]){
                    winx = i;
                    winy = j;
                }
        }
    }

    if(nt==13){
        sampling_min++;
        wincount_min[winx][winy]++;
    }
```

```
            for(int k=0; k<nx; k++){
                w[winx][winy][k] = w[winx][winy][k] + a*near*(m[nt][k]-w[winx][winy][k]);
            }

            for(int i=1 ; i<=near ; i++){
                if(mapsize>winx+i){
                    for(int k=0; k<nx; k++){
                        w[winx+i][winy][k] = w[winx+i][winy][k] + a*near*(m[nt][k]-w[winx+i][winy][k]);
                    }

                if(0<winx-i){
                    for(int k=0; k<nx; k++){
                        //w[winx-i][winy][k] = w[winx-i][winy][k] + (a*(1/near+1))*(m[nt][k]-w[winx-i][winy][k]);
                        w[winx-i][winy][k] = w[winx-i][winy][k] + a*near*(m[nt][k]-w[winx-i][winy][k]);
                    }
                }
                if(mapsize>winy+i){
                    for(int k=0; k<nx; k++){
                        w[winx][winy+i][k] = w[winx][winy+i][k] + a*near*(m[nt][k]-w[winx][winy+i][k]);
                    }
                }
                if(0<winy-i){
                    for(int k=0; k<nx; k++){
                        w[winx][winy-i][k] = w[winx][winy-i][k] + a*near*(m[nt][k]-w[winx][winy-i][k]);
                    }
                }
            }

            if(near>=2){
                for(int i=1 ; i<=near-1 ; i++){
                    for(int j=1 ; j<=near-1 ; j++){
                        if(mapsize>winx+i && mapsize>winy+j){
                            for(int k=0; k<nx; k++){
                                w[winx+i][winy+j][k] = w[winx+i][winy+j][k] +
a*near*(m[nt][k]-w[winx+i][winy+j][k]);
                            }
                        }
                        if(mapsize>winx+i && 0<winy-j){
                            for(int k=0; k<nx; k++){
                                w[winx+i][winy-j][k] = w[winx+i][winy-j][k] +
a*near*(m[nt][k]-w[winx+i][winy-j][k]);
                            }
                        }
                        if(0<winx-i && mapsize>winy+j){
                            for(int k=0; k<nx; k++){
                                w[winx-i][winy+j][k] = w[winx-i][winy+j][k] +
a*near*(m[nt][k]-w[winx-i][winy+j][k]);
                            }
                        }
                        if(0<winx-i && 0<winy-j){
                            for(int k=0; k<nx; k++){
                                w[winx-i][winy-j][k] = w[winx-i][winy-j][k] + a*near*(m[nt][k]-w[winx-i][winy-j][k]);
                            }
                        }
                    }
                }
            }

            learncount--;
            nearcount++;
            if(nearcount%50==0) near--;
        }

        double sumx = 0;
        for(int i=0 ; i<mapsize ; i++){
```

```
    for(int j=0 ; j<mapsize-1 ; j++){
        for(int k=0; k<nx; k++){
            sumx = sumx + Math.pow((w[i][j][k] - w[i][j+1][k]),2);
        }
        ux[i][j] = Math.sqrt(sumx);
        sumx=0;
    }
}

for(int i=0 ; i<mapsize ; i++){
    ux[i][mapsize-1]=0;
}

double sumy = 0;
for(int i=0 ; i<mapsize-1 ; i++){
    for(int j=0 ; j<mapsize-1 ; j++){
        for(int k=0; k<nx; k++){
            sumy = sumy + Math.pow((w[i][j][k] - w[i+1][j][k]),2);
        }
        uy[i][j] = Math.sqrt(sumy);
        sumy=0;
    }
}
for(int i=0 ; i<mapsize ; i++){
    uy[mapsize-1][i]=0;
}

int ix=0,iy=0,map2;
map2=mapsize*2;
double[][] uxy = new double[map2][map2];
for(int i=0 ; i<map2 ; i++){
    for(int j=0 ; j<map2 ; j++){
        if(i%2 == 0){
            if(j<mapsize){
                uxy[i][j] = ux[ix][j];
            }else{
                uxy[i][j] = 0;
            }
        }
        else if(i%2 == 1){
            if(j<mapsize){
                uxy[i][j] = uy[iy][j];
            }else{
                uxy[i][j] = 0;
            }
        }
    }
    if(i%2 == 0){
        ix++;
    }
    else if(i%2 == 1){
        iy++;
    }
}

double[][][] u_input = new double[mapsize][mapsize][inputsize];
double sum_input=0;
int piece_input=0;
while(piece_input<inputsize){
    for(int i=0 ; i<mapsize ; i++){
        for(int j=0 ; j<mapsize ; j++){
            for(int k=0; k<nx; k++){
                sum_input = sum_input + Math.pow((w[i][j][k] - m_res[piece_input][k]),2);
            }
            u_input[i][j][piece_input] = Math.sqrt(sum_input);
```

```
                    sum_input=0;
                    df.applyLocalizedPattern("0.00");
                }
            }
            piece_input++;
        }

        piece_input=0;
        winx=0;
        winy=0;
        while(piece_input<inputsize){
            for(int i=0 ; i<mapsize ; i++){
                for(int j=0 ; j<mapsize ; j++){
                    if(u_input[winx][winy][piece_input]>u_input[i][j][piece_input]){
                        winx = i;
                        winy = j;
                    }
                }
            }

            if(winx==0 && winy!=0){
                if(uxy[(winx*2)][winy-1]<=uxy[(winx*2)][winy] && uxy[(winx*2)][winy-1]<=uxy[(winx*2)+1][winy]){
                    uxy[(winx*2)][winy-1]=100;
                }else if(uxy[(winx*2)][winy]<uxy[(winx*2)][winy-1] &&
uxy[(winx*2)][winy]<=uxy[(winx*2)+1][winy]){
                    uxy[(winx*2)][winy]=100;
                }else if(uxy[(winx*2)+1][winy]<uxy[(winx*2)][winy-1] &&
uxy[(winx*2)+1][winy]<uxy[(winx*2)][winy]){
                    uxy[(winx*2)+1][winy]=100;
                }
            }else if(winy==0 && winx!=0)
                if(uxy[(winx*2)-1][winy]<=uxy[(winx*2)][winy] && uxy[(winx*2)-1][winy]<=uxy[(winx*2)+1][winy]){
                    uxy[(winx*2)-1][winy]=100;
                }else if(uxy[(winx*2)][winy]<uxy[(winx*2)-1][winy] &&
uxy[(winx*2)][winy]<=uxy[(winx*2)+1][winy]){
                    uxy[(winx*2)][winy]=100;
                }else if(uxy[(winx*2)+1][winy]<uxy[(winx*2)-1][winy] &&
uxy[(winx*2)+1][winy]<uxy[(winx*2)][winy]){
                    uxy[(winx*2)+1][winy]=100;
                }
            }else if(winx==0 && winy==0){
                if(uxy[(winx*2)][winy]<=uxy[(winx*2)+1][winy]){
                    uxy[(winx*2)][winy]=100;
                }else if(uxy[(winx*2)+1][winy]<uxy[(winx*2)][winy]){
                    uxy[(winx*2)+1][winy]=100;
                }
            }else{
                if(uxy[(winx*2)-1][winy]<=uxy[(winx*2)][winy-1] && uxy[(winx*2)-1][winy]<=uxy[(winx*2)][winy]
&& uxy[(winx*2)-1][winy]<=uxy[(winx*2)+1][winy]){
                    uxy[(winx*2)-1][winy]=100;
                }else if(uxy[(winx*2)][winy-1]<uxy[(winx*2)-1][winy] &&
uxy[(winx*2)][winy-1]<=uxy[(winx*2)][winy] && uxy[(winx*2)][winy-1]<=uxy[(winx*2)+1][winy]){
                    uxy[(winx*2)][winy-1]=100;
                }else if(uxy[(winx*2)][winy]<uxy[(winx*2)-1][winy] && uxy[(winx*2)][winy]<uxy[(winx*2)][winy-1]
&& uxy[(winx*2)][winy]<=uxy[(winx*2)+1][winy]){
                    uxy[(winx*2)][winy]=100;
                }else if(uxy[(winx*2)+1][winy]<uxy[(winx*2)-1][winy] &&
uxy[(winx*2)+1][winy]<uxy[(winx*2)][winy-1] && uxy[(winx*2)+1][winy]<uxy[(winx*2)][winy]){
                    uxy[(winx*2)+1][winy]=100;
                }
            }
            winx=0;
            winy=0;
            piece_input++;
```

```
        }

        try{
            BufferedWriter bw = new BufferedWriter(new
FileWriter("c:¥¥som-data-file¥¥output8d2kD_"+keta[2]+keta[1]+keta[0]+".data"));
            for(int i=0 ; i<map2 ; i++){
                for(int j=0 ; j<mapsize ; j++){
                    Double dNumberx = new Double(uxy[i][j]);
                    bw.write(j+" "+i+" "+dNumberx.toString());
                    bw.newLine();
                }
                bw.newLine();
            }
            bw.close();
        }
        catch(Exception e){
            System.out.println(e);
        }


        try{
            BufferedWriter bw = new BufferedWriter(new
FileWriter("c:¥¥som-data-file¥¥countIrisCD_Set_"+keta[2]+keta[1]+keta[0]+".data"));
            for(int i=0 ; i<mapsize ; i++){
                for(int j=0 ; j<mapsize ; j++){
                    bw.write(i+" "+j+" "+wincount_min[i][j]);
                    bw.newLine();
                }
            }
            bw.close();
        }
        catch(Exception e){
            System.out.println(e);
        }
    }

    static void aveMethod(double r_max, double r_min){
        for(int i=0 ; i<mapsize ; i++){
            for(int j=0 ; j<mapsize ; j++){
                System.out.print("(");
                for(int k=0 ; k<sort_nx ; k++){
                    w[i][j][k] = Math.floor(Math.random()*(r_max-r_min+1))+r_min;
                    System.out.print(w[i][j][k]+",");
                }
                System.out.print(") ");
            }
            System.out.println();
        }

        double w_sum=0;
        for(int i=0 ; i<mapsize ; i++){
            for(int j=0 ; j<mapsize ; j++){
                System.out.print("(");
                for(int k=0 ; k<sort_nx ; k++){
                    w_sum = w_sum+w[i][j][k];
                }
                wAve[i][j] = w_sum/sort_nx;
                System.out.print(wAve[i][j]+") ");
                w_sum=0;
            }
            System.out.println();
        }
    }

    static double[][] sortInputMethod(double m[][], int ny){
```

```
        double w_sum=0;
        double[] mAve = new double[ny+1];
        for(int i=0 ; i<ny+1 ; i++){
            for(int j=0 ; j<sort_nx ; j++){
                    w_sum = w_sum+m[i][j];
            }
            mAve[i] = w_sum/sort_nx;
            w_sum=0;
        }

        int min, v=0;
        double tmp,tmpAve;
        for(int i=0 ; i<ny ; i++){
            min=i;
            for(int j=i ; j<ny+1 ; j++){
                if(mAve[min]>mAve[j]){
                    min=j;
                }else if(mAve[min]==mAve[j]){
                    while(v<sort_nx){
                        if(m[min][v]<m[j][v]){
                            break;
                        }else if(m[min][v]>m[j][v]){
                            min=j;
                            break;
                        }
                        v++;
                    }
                }
                v=0;
            }
            for(int k=0 ; k<sort_nx ; k++){
                tmp=m[i][k];m[i][k]=m[min][k];m[min][k]=tmp;
            }
            tmpAve=mAve[i];mAve[i]=mAve[min];mAve[min]=tmpAve;
        }
        for(int i=0 ; i<ny+1 ; i++){
            for(int j=0 ; j<sort_nx ; j++){
                System.out.print(m[i][j]+ " |");
            }
            System.out.println();
        }

        return m;
    }

    static void sortXMethod(){
        int min=0, v=0;
        double tmp,tmpAve;
        for(int i=0 ; i<mapsize ; i++){
            for(int j=0 ; j<mapsize-1 ; j++){
                min=j;
                for(int p=j ; p<mapsize ; p++){
                    if(wAve[i][min]>wAve[i][p]){
                        min=p;
                    }else if(wAve[i][min]==wAve[i][p]){
                        while(v<sort_nx){
                            if(w[i][min][v]<w[i][p][v]){
                                break;
                            }else if(w[i][min][v]>w[i][p][v]){
                                min=p;
                                break;
                            }
                            v++;
                        }
                    }
```

```
                v=0;
            }
            for(int k=0 ; k<sort_nx ; k++){
                tmp=w[i][j][k];w[i][j][k]=w[i][min][k];w[i][min][k]=tmp;
            }
            tmpAve=wAve[i][j];wAve[i][j]=wAve[i][min];wAve[i][min]=tmpAve;
        }
    }
}

static void sortYMethod(){
    int min=0, v=0;
    double tmp,tmpAve;
    for(int i=0 ; i<mapsize ; i++){
        for(int j=0 ; j<mapsize-1 ; j++){
            min=j;
            for(int p=j ; p<mapsize ; p++){
                if(wAve[min][i]>wAve[p][i]){
                    min=p;
                }else if(wAve[min][i]==wAve[p][i]){
                    while(v<sort_nx){
                        if(w[min][i][v]<w[p][i][v]){
                            break;
                        }else if(w[min][i][v]>w[p][i][v]){
                            min=p;
                            break;
                        }
                        v++;
                    }
                }
                v=0;
            }
            for(int k=0 ; k<sort_nx ; k++){
                tmp=w[j][i][k];w[j][i][k]=w[min][i][k];w[min][i][k]=tmp;
            }
            tmpAve=wAve[j][i];wAve[j][i]=wAve[min][i];wAve[min][i]=tmpAve;
        }
    }
}

static void printMethod(){
    for(int i=0 ; i<mapsize ; i++){
        for(int j=0 ; j<mapsize ; j++){
            System.out.print("(");
            for(int k=0 ; k<sort_nx ; k++){
                System.out.print(w[i][j][k]+",");
            }
            System.out.print(") ");
        }
        System.out.println();
    }
}

static int[] ketaOutput(int k[]) throws IOException{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String a = br.readLine();
    int hiki = Integer.parseInt(a);

    k[0]=hiki%10;
    if(hiki<10){
        k[1]=0;
    }else{
        k[1]=(hiki/10)%10;
    }
    if(hiki<100){
```

```
            k[2]=0;
        }else{
            k[2]=(hiki/100)%10;
        }
        return k;
    }
}
```

ソースコード２：　Ward.java

```
package som;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StreamTokenizer;
import java.text.DecimalFormat;
import java.util.Random;

public class Ward {
    public static void main(String args[]) throws IOException{
        int NUMBER = 13;
        int z[][]=new int[100][2];
        int xy=0, data_number=0;
        try{
            FileReader fr=new FileReader("C:/som-data-file/input.data");
            StreamTokenizer st=new StreamTokenizer(fr);
            st.eolIsSignificant(true);

            while(st.nextToken()!=StreamTokenizer.TT_EOF){
                switch(st.ttype){
                case StreamTokenizer.TT_NUMBER:
                    z[data_number][xy]=(int) st.nval;
                    xy++;
                    break;

                case StreamTokenizer.TT_EOL:
                    xy=0;
                    break;
                }
            }
            fr.close();
        }
        catch(Exception e){
            System.out.println(e);
        }
        data_number=data_number+1;

        double k[][]=new double[data_number][2];
        double g[][]=new double[data_number][data_number];
        double d[][]=new double[data_number][3];
        int h[][]=new int[data_number][data_number];
        int count=0, N=2, NMAX = 0, NMAX2 = 0;
        double winBEF = 0;
        double zouka = 0, zoukaMAX = 0, zoukaMAX2 = 0;

        for(int n=0;n<data_number;n++){
            for(int m=0;m<data_number;m++){
                h[n][m]=-1;
            }
        }

        while(true){
            for(int n=count;n<data_number-1;n++){
                for(int m=count+1;m<data_number;m++){
                    k[n][0] = (double)(z[n][0]+z[m][0])/2;
                    k[n][1] = (double)(z[n][1]+z[m][1])/2;
```

```
    g[n][m]=Math.pow((z[n][0]-k[n][0]),2)+Math.pow((z[n][1]-k[n][1]),2)+Math.pow((z[m][0]-k[n][0]),2)+Math.pow((
z[m][1]-k[n][1]),2);
                }
                count++;
            }
            count=0;
            int win1=0, win2=0;
            double win = 10000;
            for(int n=count;n<data_number-1;n++){
                for(int m=count+1;m<data_number;m++){
                    if(win>g[n][m]){win=g[n][m]; win1=n; win2=m;}
                }
                count++;
            }
            winBEF = win;
            for(int i=0;i<data_number;i++){
                d[i][0]=z[i][0];
                d[i][1]=z[i][1];
                d[i][2]=0;
                h[i][0]=i;
                if(i==win1){d[i][2]=0;h[i][0]=win1;h[i][1]=win2;}
                if(i==win2){d[i][2]=1;h[i][0]=win2;h[i][1]=win1;}
            }
            for(int i=0;i<data_number;i++){
                System.out.println("data"+i+":("+d[i][0]+", "+d[i][1]+"),Count "+d[i][2]);
            }
            break;
        }
        for(int n=0;n<data_number;n++){
            for(int m=0;m<data_number;m++){
                g[n][m]=0;
            }
        }
        for(int n=0;n<data_number;n++){
            k[n][0] = 0; k[n][1] = 0;
        }

        int flag=1;
        count=0;
        int K=0, heigo_count_1=0, heigo_count_2=0, juh=0, juh2=0;
        double jNum[]=new double[2];
        double Gnai[]=new double[data_number];
        int sum_number[]=new int[data_number];
        for(int n=0;n<data_number;n++){
            Gnai[n]=0;sum_number[n]=0;
        }
        jNum[0]=0;jNum[1]=0;

        while(flag!=0){
            for(int n=count;n<data_number-1;n++){
                for(int m=count+1;m<data_number;m++){
                    if(d[n][2]<1&&d[m][2]<1){
                        while(h[n][K]!=-1){
                            sum_number[juh]=h[n][K];
                            k[n][0]=k[n][0]+d[h[n][K]][0];
                            k[n][1]=k[n][1]+d[h[n][K]][1];
                            heigo_count_1++;
                            K++;juh++;
                        }
                        jNum[0]=k[n][0]/heigo_count_1;
                        jNum[1]=k[n][1]/heigo_count_1;
                        for(int p=0;p<juh;p++){

    Gnai[0]=Gnai[0]+Math.pow((z[sum_number[p]][0]-jNum[0]),2)+Math.pow((z[sum_number[p]][1]-jNum[1]),2);
```

```
                    }K=0;juh2=juh;

                    while(h[m][K]!=-1){
                        sum_number[juh]=h[m][K];
                        k[m][0]=k[m][0]+d[h[m][K]][0];
                        k[m][1]=k[m][1]+d[h[m][K]][1];
                        heigo_count_2++;
                        K++;juh++;
                    }
                    jNum[0]=k[m][0]/heigo_count_2;
                    jNum[1]=k[m][1]/heigo_count_2;
                    for(int p=juh2;p<juh;p++){

Gnai[1]=Gnai[1]+Math.pow((z[sum_number[p]][0]-jNum[0]),2)+Math.pow((z[sum_number[p]][1]-jNum[1]),2);
                    }K=0;jNum[0]=0;jNum[1]=0;

                    for(int p=0;p<juh;p++){
                        jNum[0]=jNum[0]+z[sum_number[p]][0];
                        jNum[1]=jNum[1]+z[sum_number[p]][1];
                    }
                    jNum[0]=jNum[0]/(heigo_count_1+heigo_count_2);
                    jNum[1]=jNum[1]/(heigo_count_1+heigo_count_2);
                    for(int p=0;p<juh;p++){

g[n][m]=g[n][m]+Math.pow((z[sum_number[p]][0]-jNum[0]),2)+Math.pow((z[sum_number[p]][1]-jNum[1]),2);
                    }
                    System.out.println("g[n][m]="+g[n][m]);
                    g[n][m]=g[n][m]-Gnai[0]-Gnai[1];
                    K=0;jNum[0]=0;jNum[1]=0;juh=0;juh2=0;
                    heigo_count_1=0;heigo_count_2=0;
                    for(int p=0;p<data_number;p++){
                        Gnai[p]=0;sum_number[p]=0;
                        k[p][0] = 0; k[p][1] = 0;
                    }
                }
            }
            count++;
        }
        count=0;
        int win1=0, win2=0;
        double win = 100;//unyo
        for(int n=count;n<data_number-1;n++){
            for(int m=count+1;m<data_number;m++){
                if(g[n][m]!=0){
                    if(win>g[n][m]){win=g[n][m]; win1=n; win2=m;}
                }
            }
            count++;
        }
        zouka = win-winBEF;
        if(zoukaMAX<zouka){
            zoukaMAX2=zoukaMAX;NMAX2=NMAX;
            zoukaMAX=zouka;NMAX=N;
        }


        System.out.println("Win="+win+",("+win1+"|"+win2+")¥n");
        for(int i=0;i<data_number;i++){
            if(i==win1){
                for(int p=0;p<data_number;p++){
                    if(h[i][p]==-1){h[i][p]=win2;break;}
                }
            }
            if(i==win2){
                d[i][2]=1;
```

```
                    for(int p=0;p<data_number;p++){
                        if(h[i][p]==-1){h[i][p]=win1;break;}
                    }
                }
            }

            count=0;
            for(int n=0;n<data_number;n++){
                for(int m=0;m<data_number;m++){
                    g[n][m]=0;
                }
            }
            winBEF=win;
            for(int n=0;n<data_number;n++){
                if(d[n][2]==0)flag++;
            }
            if(N>NUMBER){flag=0;}
            N++;
        }
    }
}
```