平成 13 年度

学士学位論文

階層型 SOM による
質問応答システムのログ可視化

Logs Visualization of Question-Answering Systems by the Hierarchical Growing SOM

1020271　　小原 和裕

指導教員　　Ruck Thawonmas

2002 年 2 月 8 日

高知工科大学 情報システム工学科

要 旨

階層型 SOM による
質問応答システムのログ可視化

小原 和裕

本研究では，質問応答システムのログを対象に階層型 SOM による可視化を行った．まず，ログを HTML マップ上に表示することで，一覧にして観察するよりも見やすくなり，ログの全体像の把握が容易になった．また，階層化することにより，ログの数が増えてもコンパクトに管理することが可能になった．これにより，保守者の負担軽減を可能にし，質問応答システムの改良に役立つインタフェースを提供できた．

キーワード　可視化，階層化，SOM，GHSOM，インタフェース

Abstract

Logs Visualization of Question-Answering Systems by the Hierarchical Growing SOM

Kazuhiro Ohara

In this paper, I give my discussions on logs visualization of question-answering systems by the Hierarchical-Growing SOM. First, it makes logs display on HTML map. Then it realizes easy to observe the logs better than the logs are listed and easy to understand all the logs. Second, even if logs quantity has increased, it makes HTML map compact and it can easy to manage for system operator. Therefore, it can reduce system operator's burden and it can offer the useful interface to improvement of question-answering system.

key words Visualization, Hierarchical, SOM, GHSOM, Interface

目次

図目次

表目次

第 1 章

はじめに

質問応答システムにとって最も重要なのは，ユーザからの問い合わせに確実に回答できることである．多くのユーザの要望に的確に対応するためには，システムの中枢である知識ベースの充実が欠かせない．このためにはユーザからの声であるログを整理し，その解析を行うことが重要になる．しかし，ログはシステムが稼働している限り増え続け，大量のデータとして蓄積されていく．そのような大量のログデータをシステム保守者が手動で整理するのは非常に困難で，人為的ミスも起こりやすく，合理的とはいえない．このため，自動でログを整理し，保守者の負担を軽減する方法が必要となる．

そこで，本研究ではログに含まれるデータの内，ユーザからの質問文を対象とし，SOM を用いた可視化及びその階層化を行うことで，ログの自動整理を行うと同時に見やすいインタフェースを提供し，システム保守者によるログ解析を容易にするツールを提案する．

第 2 章

質問応答システムと SOM

2.1 質問応答システムとは

現在, インターネット上で一般的に使用されている情報検索システムは, キーワードを利用者が指定する方式である. この方式は, 検索サービスの構築が容易, ソフトウェアの原理が簡単などの理由で広く行われている. しかし, このキーワード型の情報検索方式は, 利用者の使い勝手を考慮しているとは言い難い. たとえば, ひとつだけのキーワードでは検索がうまく行えない場合, キーワードを複数指定して検索することになる. このとき, 利用者には AND 条件／ OR 条件などといったソフトウェア的な知識が必要となる. この方式では, 素早く・的確に知りたい情報を取り出すという利用者の要求を十分に満たしているとは言い難い. そこで近年, 質問応答システムに注目が集まっている.

質問応答システムとは, よく聞かれる質問とその回答をデータベース化し知識ベースとして蓄え, コンピュータが自動的に質問を受け付け, 回答する情報検索システムの一種である. 本学では, 2000 年 9 月から高知工科大学のホームページ[*1]にて, 質問応答システムの応用である高知工科大学ヘルプシステムの試験運用を開始している. 同ヘルプシステムは, 自然言語で書かれた質問に対して回答できることを大きな特徴としている. これにより, 利用者はまるで人間に対して質問をする感覚で使用することができ, 知りたい情報を正確に特定できる. また, 言い回しの違う質問や類似語による質問に対しても同様に回答できる.

本研究では同ヘルプシステムのログを対象に実験を行う.

[*1] http://www.kochi-tech.ac.jp/

2.2　質問応答システムへの SOM の適用

2.2.1　SOM とは

SOM(Self-Organization Map) は自己組織化マップ [1] と呼ばれる可視化手法の一つである，教師なし競合強化学習および近傍学習により，ある分布に従う多次元のデータに対してその分布を近似した特徴マップを生成する．特徴マップは二次元平面に表示され，同じような性質をもつ入力データはマップ上の近い位置に出力される．そのため，SOM はデータの類似性をマップから確認できるという特徴を持っている．SOM により生成されるマップの例を図 2.1 に示す．例では，動物を SOM によって分類している．比較的近い性質を持つ動物がマップ上の近い位置に配置されていることが確認できるはずである．

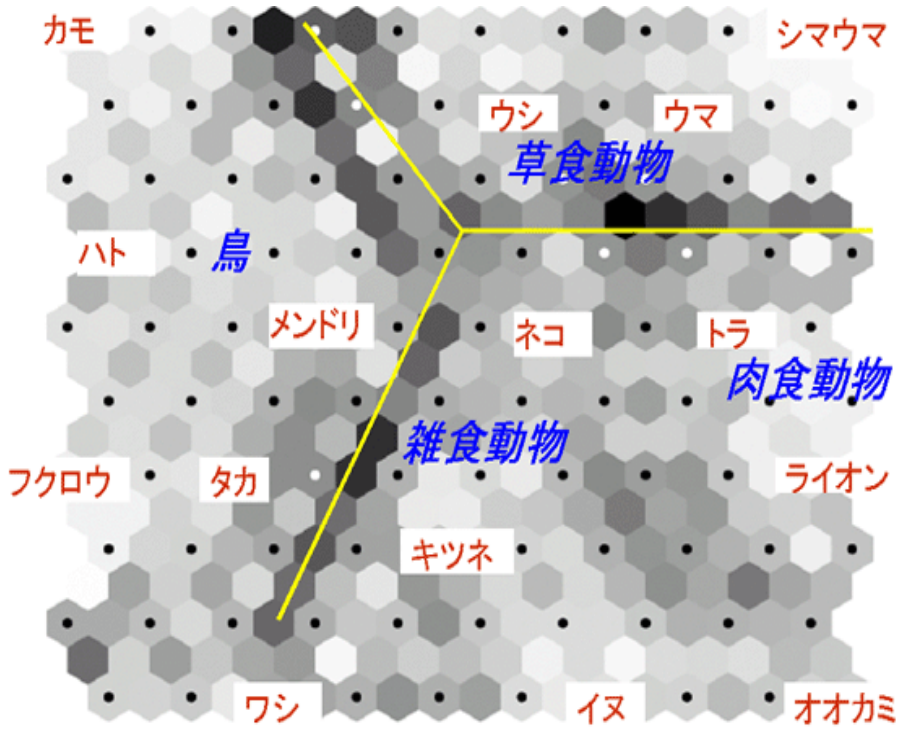


図 2.1　SOM マップの例

2.2.2　可視化の目的

本研究では，質問応答システムのログを対象に，上記の様な特徴を持つ SOM による可視化を行う．ここでは，可視化を行うことでどのような利点があるかを述べる．

ログデータを一覧にして見るだけではその全体像をしっかり把握することが難しく，また，ある種のデータに偏りがあるなどの発見が難しい．可視化により，全データを二次元のマップ上に配置することで，データ全体，あるいはデータ全体から見た特異点などが直感的に理解できるようになると考えられる．また，保守者による人為的なミスを大きく減らすことができる．つまり，可視化を行う利点は単に視覚的に分かりやすいというだけでなく，短時間で効率の良い解析を行うことにつながるのである．

第 3 章

一層の SOM

SOM マップは本来階層構造を有しない．そのため，階層型 SOM に適応する学習アルゴリズムはまだ提案されていない．よって本研究では階層型 SOM の実装においても，各層における一層の SOM マップ形成には，従来型 SOM 学習アルゴリズムを使用する．

そこで，本章では従来型 SOM のアルゴリズム及びマップ化に必要なファイルの生成手順を紹介する．なお，マップ化ツールには T.Kohonen らにより実装された SOM プログラムである，som_pak[3] を使用する．また，本研究における実験は全て，高知工科大学ヘルプシステムのログの中からランダムに選び出した 100 個のログを用いた．

3.1 SOM アルゴリズム

SOM は，入力信号の持つ位相を，教師信号を用いることなく競合層上へ写像することが出来るニューラルネットワークの一種である．SOM は入力層と競合層の 2 層からなり，各層間は完全結合で結ばれている．層間の結合には重みが与えられており，学習前に乱数で初期化される．SOM を以下に述べる競合層学習で学習させることにより，入力信号の位相を写像したトポグラフィックマップを得ることができる．

ある入力信号$\boldsymbol{x}$が与えられたとき，競合層のユニット i は自分が持つ重み $\boldsymbol{w_i}$ と入力信号間の距離を計算する．多くの場合はユークリッド距離が用いられるが，この距離が最小の競合層ユニット c を勝利ユニット (ウィナー) とし，(3.1) の式で表す．

$$c = argmin_i\{||\boldsymbol{x} - \boldsymbol{w_i}||\}$$

$$||\boldsymbol{x} - \boldsymbol{w_c}|| = min_i\{||\boldsymbol{x} - \boldsymbol{w_i}||\} \tag{3.1}$$

競合層上で c の近傍に存在するユニットは，c からの距離に反比例した強度で反応し，重みを式 (3.2) により，更新する．

$$\boldsymbol{w_i}(t+1) = \boldsymbol{w_i}(t) + h_{ci}(t)(\boldsymbol{x}(t) - \boldsymbol{w_i}(t)) \tag{3.2}$$

ここで $t = 0, 1, 2, ...$ は離散時間座標である．更新過程で $h_{ci}(t)$ は近傍関数と呼ばれ，式 (3.3) によって定義される．

$$h_{ci}(t) = h(||\boldsymbol{r_c} - \boldsymbol{r_i}||, t) \tag{3.3}$$

ここで$\boldsymbol{r_c}$,$\boldsymbol{r_i}$はそれぞれ競合層におけるユニット c と i の位置ベクトルであり，$||\boldsymbol{r_c} - \boldsymbol{r_i}||$ が増加するにつれて $h_{ci} \to 0$ とする．

必要な数，十分な時間の入力信号$\boldsymbol{x}$を用いて競合学習を行うことにより，競合層の各ユニットが持つ重みを参照ベクトルとするボロノイ・モザイク領域が形成される．図 3.1 に 200 個の乱数を学習パターンとして，1000 回の学習を行ったときのトポグラフィックマップの生成過程を示す．図 3.1 中のマップに見られる頂点は，競合層ユニットの重みを 2 次元上にプロットしたものであり，ユニット間の空間的隣接関係を，各頂点を結ぶ線で表している．重み空間に一様に分布する乱数を用いて SOM の学習を行った場合，競合層ユニットの重みは重み空間に一様に広がり，最終的に格子状のトポグラフィックマップが得られる．図 3.1(a) に示すように，学習初期では重みはランダムに初期化されているため，重み空間における重みの分布とユニットの空間的な配置には規則性が見られない．しかし，(b) → (c) と学習が進むにつれて，重みは重み空間に一様に拡散していき，同時に各ユニットの空間的隣接関係を表すトポグラフィックマップが形成され始める．1000 回学習後では，図 3.1(d) のようなトポグラフィックマップが得られる．

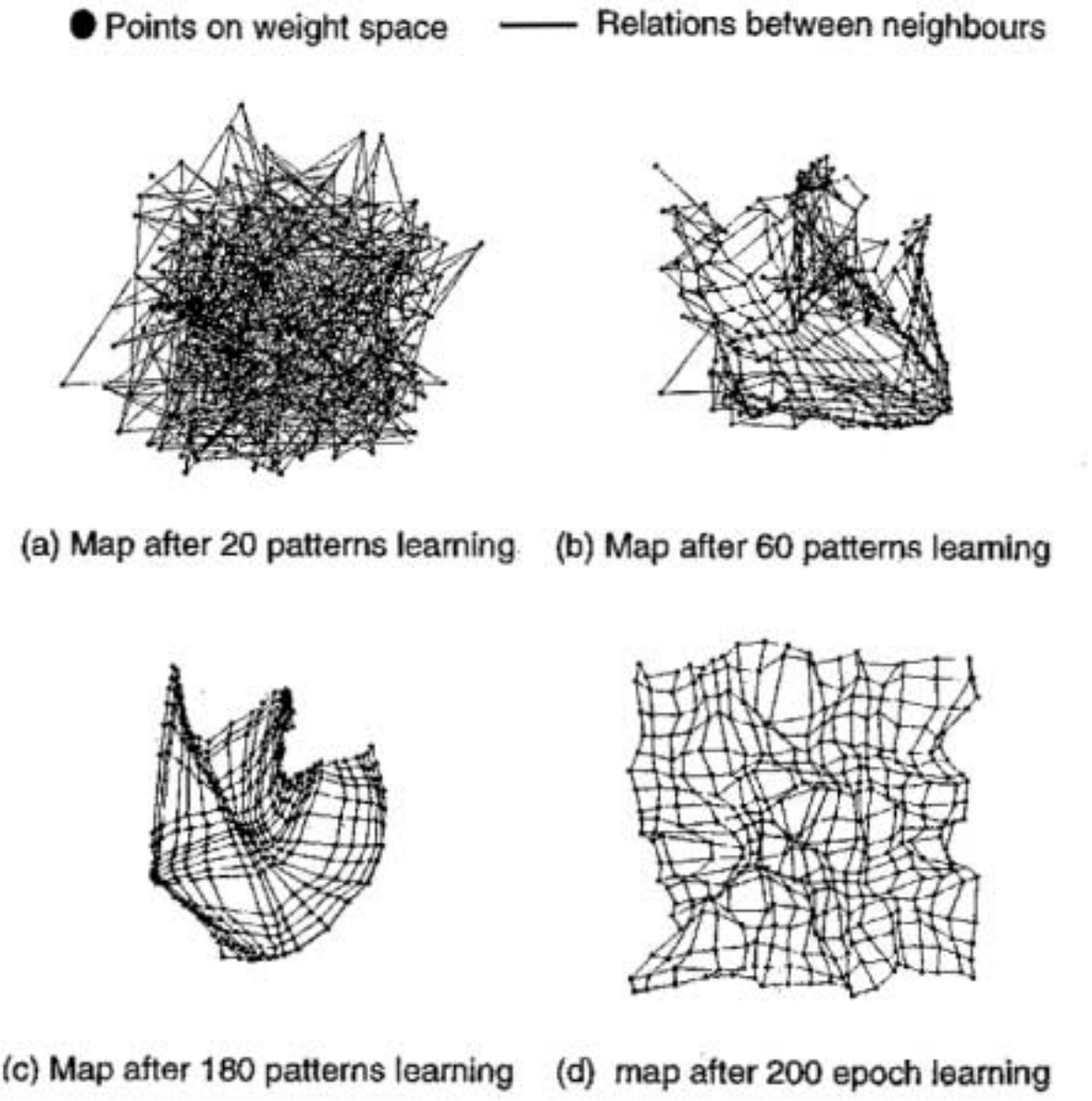


図 3.1 トポグラフィックマップの生成過程

3.1.1 近傍

勝利ユニットの周りには近傍領域が定義される．近傍サイズは最初大きくとっておき，時間軸とともに減少させていく．ノード c の周りのユニットを近傍集合とし N_c と定義する (これによって時間の関数としての $N_c = N_c(t)$ が定義できる)．もし，$i \in N_c$(i が N_c 内の

ノード) なら $h_{ci} = \alpha(t)$，$i \notin N_c$(i が N_c 外のノード) なら，$h_{ci} = 0$ である．この時，$\alpha(t)$ の値を学習率係数と言う $(0 < \alpha(t) < 1)$．$\alpha(t)$ は普通，時間軸上において単調減少させる（式 3.4）．α_0 は，α の初期値であり，一般的に 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．

$$\alpha(t) = \alpha_0(1 - \frac{t}{T}) \tag{3.4}$$

また，近傍領域 $N_c = N_c(t)$ についても同様に，

$$N_c(t) = N_c(0)(1 - \frac{t}{T}) \tag{3.5}$$

と表すことが出来る．$N_c(0)$ は初期値である．

3.2 数値化

SOM マップを生成するにはログ，つまり本研究では質問文を数値化することが必要になる．数値化ファイルを入力データとし，上記の学習アルゴリズムにしたがって学習を繰り返すことで，マップが形成されて行くのである．数値化の具体的な流れを以下に述べる．

1. 質問文を形態素解析によりすべて形態素に切り分け，同じ形態素が入らないように単語ファイルに書き込む

2. 単語ファイルと対応するように，各ログに対しその形態素を含むかどうかの判定を行い，含んでいれば 1 含んでいなければ 0 を数値をファイルに書き出す

3. 数値化ファイルにはデータの次元数情報と，マップ化した際目印として利用するラベル情報を付加する

図 3.2 に数値化の具体例を示す.

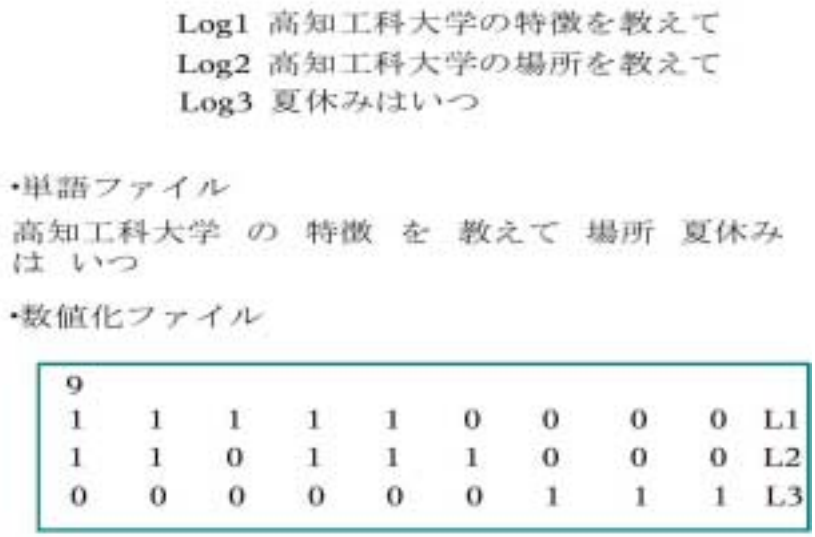


図 3.2　数値化

単語ファイルには形態素解析によって切り分けられた形態素が記入されている．数値化ファイルは単語ファイルとログを対応させ，各形態素を含むかどうかを 1 もしくは 0 で表現している．本研究ではさらに詳細なデータとするために，数値化ファイルおよび単語ファイルに対し以下の 4 つの処理を行う.

3.2.1　重み付け

重み付け処理を行うことで，単に形態素を含むかどうかだけでなく，各形態素が持つ，データ全体およびその質問文中における各形態素の比重をあらわすことができる．各形態素の重みは以下の式で決定する.

$$\text{重み} = \log \frac{\text{質問文の数}}{\text{出現質問数}} \times \text{参照数} \tag{3.6}$$

質問文の数：ログの総数

出現質問数：各形態素を含むログの数

参照数：各ログに同じ形態素がいくつ含まれるか

この処理を行った場合，多くの質問文に使われている形態素ほど重みの値が高くなる．本研究では重みが少ないほど重要な形態素と位置付けている．つまり，頻繁に出現しない形態素ほど重要な意味を持っていると考える．例えば，高知工科大学ヘルプシステムにおいて，「高知工科大学」という形態素が多く使われることは容易に想像できる．しかし，重要なことは，高知工科大学の「何」について知りたいかなので,「高知工科大学」の重みは高くなり，あまり重要な形態素ではないと判断すべきなのである．

3.2.2　正規化

SOM では多くの場合，ユークリッド距離によって類似度を計算する．ユークリッド距離とは，単純にいうと点と点との距離であり，計算が簡単であるという利点がある．その計算式を以下に示す．

$$D_{xy} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.7}$$

SOM では入力データを多次元ベクトルで表現し，SOM マップ上の各ユニットの参照ベクトルとのユークリッド距離により，入力データと各ユニットとの類似度を計算している．正規化はベクトルの長さを均一にする処理である．正規化によりベクトル長を均一にするこ

とで，以下の図 3.3 に示すようなユークリッド距離式の欠点を解消することができる．

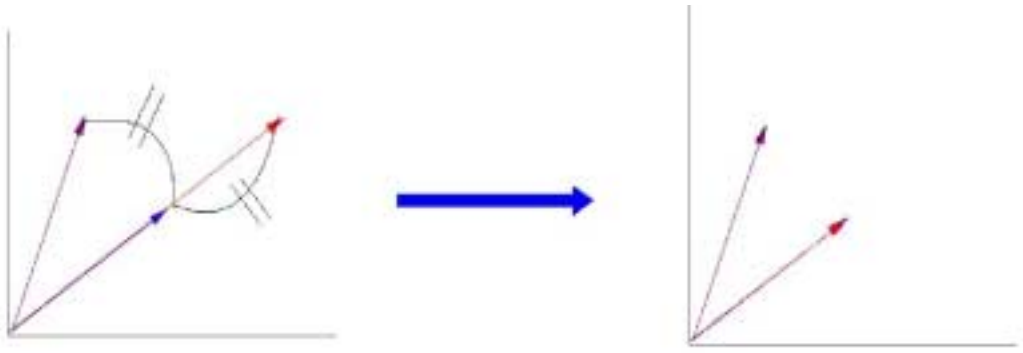


図 3.3　正規化

図 3.3 では，長さが違う同じ方向のベクトルと，違う方向のベクトルが同じ類似度であると判断されてしまてしまうミスを，正規化によりベクトルの長さを統一し，ベクトルの向きのみで距離を計算できるようにし，解決している．本研究では全入力データの全次元のデータに対し，正規化を行う．正規化式を式 (3.8) に示す．

$$d_{ij} = \frac{x_i j}{\sqrt{x_i j^2 + {x_{ij+1}}^2 + {x_{ij+2}}^2 + ... + {x_{ij+d}}^2}} \tag{3.8}$$

3.2.3　類似語変換

人間が見て同じ意味を持つ形態素でも，コンピュータはそれを理解できずに違う形態素だと判断してしまう．つまり，単語ファイルには同じ意味を持つ形態素がいくつも出て来てしまう．これでは，本来入力データが持つべき類似性が失われ，マップ生成に支障をきたす．本研究では，類似語変換処理を行うことによりこの問題を解決する．なお，変換には手動により作成した辞書を使用する．

3.2.4 パターン変換

質問はその質問内容から，いくつかのパターンに分類できる．

What 型：内容を問うもの．

高知工科大学の誇れる所を教えてください．

試験科目は何ですか．

Can 型：可能・不可能を問うもの．

ドミトリーや大学構内は携帯電話の電波は入りますか．

留学はできますか．

There 型：存在を問うもの．

近くにアルバイトをする場所はありますか．

奨学金制度はあるのか．

When 型：時を問うもの．

入試はいつですか．

夏休みはいつからですか．

Where 型：場所を問うもの

インターンシップの受け入れ先はどこですか．

授業で使う教科書はどこで買うのですか．

本研究では質問文の語尾に着目し，質問文を形態素解析により形態素に切り分ける際にその変換を行うことで，言い回しの違いによる分類ミスをなくしている．変換には類似語変換と同じく，手動により作成した辞書を使用する．図 3.4 に類似語変換およびパターン変換の例を示す．

図 3.4　類似語変換&パターン変換

3.2.5　ひらがな文字削除

単語ファイルを生成する際，ひらがなのみで構成される形態素を単語ファイルに含まないようにする．例えば,「お宅の学校の専門分野での実績を教えて下さい」という質問があった場合,「の」や「での」や「を」は，分類においてほとんど意味を成さない．にも関わらず,自然言語で質問を受け付ける高知工科大学ヘルプシステムに,「の」や「を」などの形態素が多く使われることは容易に想像できる．このような形態素を削除しておけば，データ分類の際に，意味のない形態素で類似度が変化することを防ぐことができる．また,「がっか」などの重要な単語の場合，類似語変換処理の際に「学科」に置き換えるため，削除されることは

ない．

3.3　一層型 SOM の問題点

一層の SOM で質問文を表現するだけでも，単にデータを並べて見る方法に比べると視覚的に格段に分かり易くなる．しかし，大量のデータを扱う場合には，その利点が大きく失われてしまう．原因は，マップサイズの巨大化である．データの分類をある程度正確にするためには，データ数に見合ったマップサイズを経験測から見付けなければならない．最低でデータ数と同じ数のユニット数が必要になる場合が多い．当然データ量が増大すると，それに対応してマップサイズも巨大化してしまう．マップサイズが大きくなりすぎると見にくくなるばかりか，学習における計算量も指数関数的に増大する．

このような一層の SOM の問題点を本研究では，階層型 SOM を適用することによって解決する．

第 4 章

階層化手法

本章では，SOM の階層化を行う目的，及び階層型 SOM アルゴリズムである GHSOM[2] について述べる．

4.1 階層化の目的

本研究では大量のデータを一度に扱えることが重要な要素である．しかし大量のデータを扱う場合，全データをそのまま二次元マップ上に視覚化するだけでは，サイズなどの問題から保守者にとって扱い易いインタフェースではなくなってしまう．また，サイズの巨大化による，計算量の爆発的な増加も見過すことはできない．

このような理由から，本研究では GHSOM を用いた SOM の階層化を行い，階層型 SOM を実装する．階層型 SOM では，入力データが大量の場合にも各マップサイズは適度に保たれる．これにより，学習時間の短縮につながる効果も期待できる．

4.2 GHSOM とは

GHSOM(Growing-Hierarchical SOM) は，Dittenbach らが提案した SOM 階層化手法である．従来型 SOM には学習前にマップサイズを決定しておかなければならないという欠点がある．したがって，従来型 SOM の場合，人間の経験測によりマップサイズを決定しなければならない．GHSOM は単に階層化を行うためのアルゴリズムではなく，マップの成長をも助ける．GHSOM では最上位層以外の各マップは，均一なサイズからスタートし，ある判定条件を元にその条件を満たしている間は階層化をおこなわずユニットを挿入し，マッ

プの成長を行う．この手法であれば，適切なサイズまでマップサイズは拡大する．つまり，自動的にマップサイズの最適化を行えるのである．

また，階層化においても判定条件を元に，条件を満たすユニットのみを階層化して行く．これにより，無駄に階層化を行うこともない．

次に GHSOM のアルゴリズムを述べる．

4.3 アルゴリズム

GHSOM は最上位層である単一のユニットからなる第 0 層からスタートする．第 0 層では，全ての入力データの平均偏差を求め，第 1 層以降の層における成長および階層化を制御するパラメータとして利用する．第 0 層では成長は行われない．第 1 層以降は，2×2 のマップサイズから始まり，必要に応じて成長しながら，階層化を進めて行く．また，各層の各マップは独立したニューラルネットワークから構成され，学習も完全に独立で行われる．それでは，実際のマップの成長及び階層化の手順について説明する．

まず、式 (4.1) に従い最上位層マップの偏差の平均である MQE_0 を求める．

$$MQE_0 = 1/d \cdot ||m_0 - x|| \tag{4.1}$$

偏差とは，そのユニットが担当するデータの平均値と各データの差である．その各データの偏差の平均値が平均偏差となる．この値が大きなユニットは，そのユニットが担当するデータの類似性が低いことを示している．つまり，成長はデータ分類を正確にするためのもので，データ数に見合ったユニット数に自動的に調整することといえる．最上位層マップは単一のユニットしか持たないため，そのユニットが全てのデータの勝利ユニットとなる．つまり，MQE_0 は全ての入力データの平均偏差ということになる．この値は，一層目以降の全てのマップにおいて，階層化を行うかどうかの判定に用いる．最上位層マップは，このパラメータを取得するために存在し，実際のマップとしては使用しない仮想のマップである．

同様の式で子マップの各ユニットの平均偏差である mqe_i を求める．この場合は各ユニットが担当するデータの平均偏差となる．

$$mqe_i = ||m_i - x|| \tag{4.2}$$

続いて，各マップの平均偏差の平均である MQE_m を求める．u はユニット数であるが，これはマップ中の全てのユニット数ではなく，入力データを担当する勝利ユニット数である．

$$MQE_m = 1/u \cdot \Sigma\ mqe_i \tag{4.3}$$

同様の式で一つ上の階層のマップの偏差の平均

$$MQE_{m-1} = 1/u \cdot \Sigma\ mqe_i \tag{4.4}$$

を求める。

これらのパラメータは成長および階層化を行うかどうかの判定に用いる．以下には成長および階層化の規則を述べる．

4.3.1 成長規則

- 成長判定

  各マップは以下の式を満たす場合にユニットの挿入を行い成長する．T_m は閾値を表し，この値の増減で成長の度合いを操作する．

  $$MQE_m >= Tm \cdot MQE_{m-1} \tag{4.5}$$

  上記の式は，そのマップの MQE が，一つ上の階層，つまり親マップというべきマップの MQE より大きい場合に成長を行うことを示す．

続いて成長規則について説明する．まず、各マップ中最も偏差の大きいユニットを特定し errorunit とする．次にそのユニットの近傍の内，最も偏差の小さいユニットを dissimilarunit とし，errorunit と dissimilarunit の間に必要数ユニットを挿入する．成長の様子を図に示す.

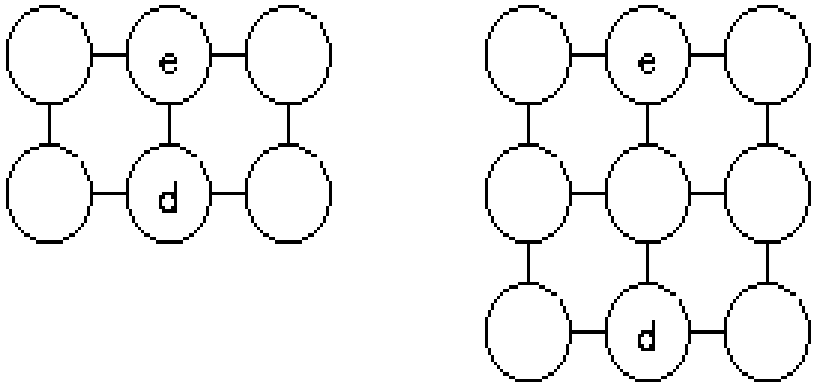


図 4.1 ユニットの挿入

挿入する各ユニットの初期値は，挟み込む 2 つのユニットの平均値とする．そして各ユニットの重みはそのままで再び従来型 SOM アルゴリズムによる学習を行い，適切なマップサイズになるまでこれを繰り返す．マップサイズを拡大し，再学習を行うことでデータの分類が再度行われる．このときユニット数の増加により，元のマップに比べ，正しい分類が行われている可能性が高い．なぜなら，ユニット数がデータ量に対してふさわしい数になるにつれ，類似性の低いデータであるのに同じユニットが勝利ユニットに選ばれてしまう可能性が少しずつ減少するからである．故に偏差は徐々に減少していき，適切なマップサイズを形成したところで成長は止まる．各マップは成長が終った時点で，階層化を行うかどうかの判定に入る.

4.3.2 階層化規則

- 階層化判定

  各ユニットは以下の式を満たす場合に階層化を行う.

$$mqe_i > Tu \cdot MQE_0 \tag{4.6}$$

上記の式は各マップ中のユニットの内，第 0 層の平均偏差より，大きな偏差を持つユニットに対して階層化を行うことを示している．T_u は閾値を表し，この値の増減で階層化の度合いを操作する．すなわち，この値を大きくすればあまり階層化を行わないマップとなる．あまり極端な閾値に設定するとデータ分類にも支障をきたすため，適切な値に設定することが必要となる．これは成長判定にも同様のことがいえる．

SOM 階層化のイメージを図 4.2 に示す.

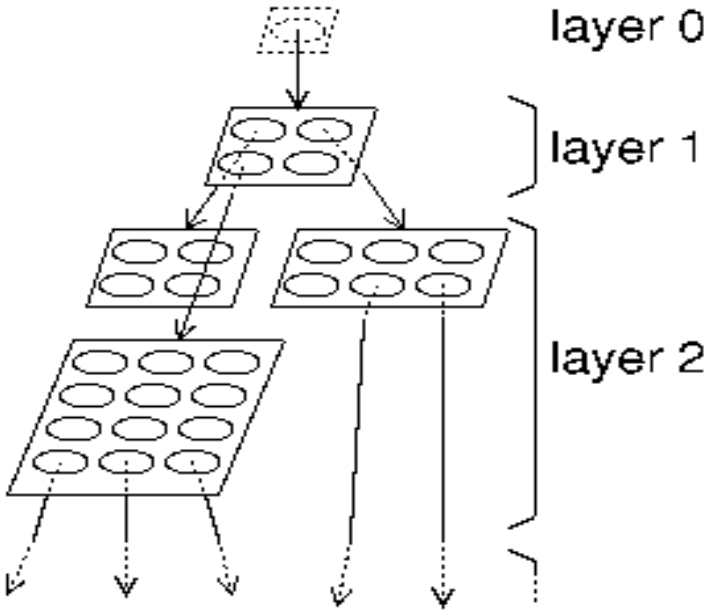


図 4.2 階層構造

図 4.2 に示されるように，各マップのサイズは一定ではない．また，最上位層は仮想マップとして扱われるため，表示には使用しない．

GHSOM は以上のような成長判定および階層化判定に基づき，成長と階層化を行っていく．最終的にどのユニットも階層化を行わなくなった時点でマップは完成となる．

第 5 章

結果の表示

5.1 表示方法

表示部分は本研究において最も重要な要素である．いかに見やすく，扱い易いかに重点を置いている．これにより，保守者の負担及び人為的ミスの軽減などの効果が期待できる．

5.1.1 インタフェース

SOM は学習を複数回繰り返すことで二次元の特徴マップを生成する．しかし，SOM により出力されるマップは，多くの場合マップを表示するのみでそれ以上の操作はできない．これでは，たとえ階層化をおこなって，あるユニットの下位層を見たいと思っても，その都度一定の操作によりマップを表示させなければならない．これは単に扱いづらいだけではなく，ファイルの指定ミスなどの誤りを引き起こす可能性がある．

そこで，本研究は表示の際のインタフェースとして，HTML を使用する．HTML のリンク機能を使用すれば，各ユニットに相当する部分をクリックするだけで下位層のマップに瞬時にアクセス可能である．つまり，下位層と上位層を物理的につなげることができる．また，「戻る」などの操作も行えるため，HTML は本研究に適した，非常に扱い易いインタフェースといえる．また，サウンド、ビデオ、Java アプレット、および JavaScript や VB Script などのスクリプト言語によって補強できるため，機能の拡張性において非常に有利であるともいえる．

さらに，表示色や表示サイズなどを自由に変更できることも HTML の大きな利点といえる．この利点を生かし，本研究では階層化されているユニット，つまりクリックにより下位

層へアクセスできるユニットと，最下位層のユニット，つまりそれ以上階層化されないユニットを，テーブルの背景色により区別できるようにしている．さらに最下位層ユニットをクリックすると元の質問文が表示され，そのラベルには質問文の個数を付加する．これらは，単純なことだが実際に使ってみると非常に効果が大きいことが分かる．

5.1.2 ラベル決定手法

SOM では各クラスタ毎に，本研究では各ユニット毎に，ラベルというそのデータ群が持つある種の特徴を表現する適切な単語や数値を与える必要がある．通常，SOM ではラベル付けを手動で行う．つまり，人間がマップ中のデータの偏りなどを見て，適当だと思われるラベルを記述していくのである．この方法は正確ではあるが，やはり，マップサイズの巨大化により，手動でのラベル付けは困難になる．本研究では，このラベルを自動的に付加する機能を提供する．その手法を以下に述べる．方法は，まず各ユニット毎に最も高い重みを持つ次元を特定する．次に単語ファイルにおける同次元に対応する単語をラベルに決定する．ユニットの各次元と，単語ファイルの各次元は対応状態にあるので，各ユニットの内最も高い重みを持つ次元数に対応する単語が，そのユニットの状態を最も良く表現しているといえるのである．

この手法を用いれば，一層のマップに限れば，手動に比べ劣りはするものの，かなり正確に最適なラベルを付加することが可能である．しかし，階層型マップではそうは行かない．階層化された各ユニットは，その内容に必ず木構造，つまり親子関係が発生する．しかし，単語ファイルにはその親子関係を表現する単語を含むことができない．故に親マップと子マップのラベルが同列に扱われてしまうのである．そこで，ラベルの親子関係を表現するために，単語ファイルと対応させ，ラベル付け専用ファイル (以下，ラベルファイル) を記述する．ラベルファイルには単語ファイルに含まれる単語の内，大きく括ることができる単語を，親マップのラベル用に変換した単語を記述する．手動で辞書を作成する手間がかかるがこれにより，階層化されたユニットの親子関係をラベルによって表現でき，より正確なマップを作ることができる．具体例を図 5.1 に示す．

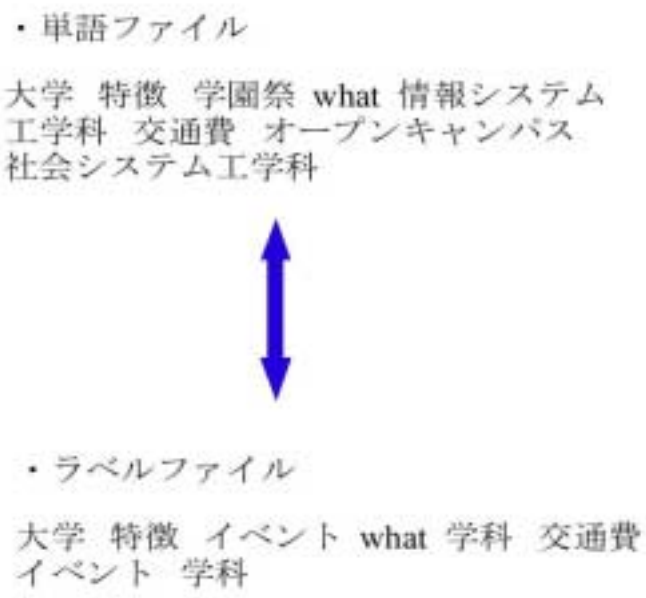
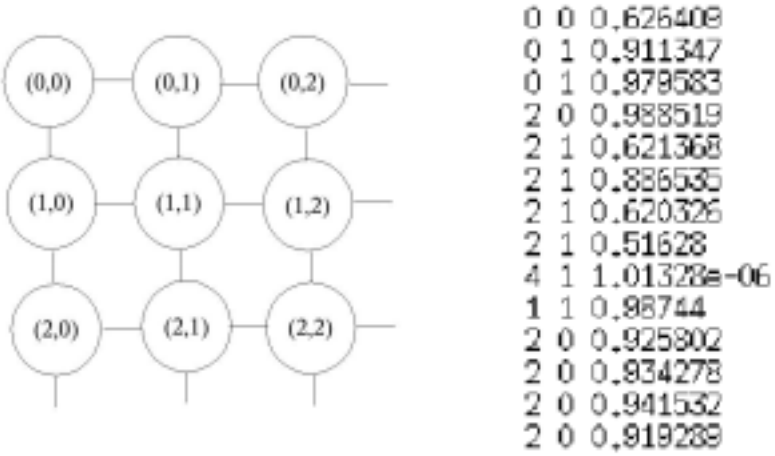


図 5.1 ラベルファイル

5.2 SOM マップから HTML への変換

生成された SOM マップを参考に，手動で HTML へ変換するのは非常に困難である．そこで本研究では，SOM マップから HTML マップへの変換を自動的に行う．

今回使用した SOM プログラムである som_pak は，入力データを与えると，マップを生成すると同時に，様々なファイルを生成する．その中には各データの座標情報を示すファイルや，各ユニットの重みを記述したファイルがある．本手法では，これらのファイルを利用し SOM マップから HTML マップへの変換を行う．また，som_pak におけるトポロジーには「Rectangular」と「Hexagonal」の 2 種が存在する．この 2 種のトポロジーは，ユニット接合方式の相違による近傍範囲選択が違う．通常は「Hexagonal」を選択する．som_pak マニュアルにおいても「Hexagonal」が推奨されている．本手法では HTML マップと SOM マップの整合性を保つため，トポロジーに「Rectangular」を使用した．図 5.2(a) に Rectangular トポロジー及び図 5.2(b) に座標情報を示すファイルの内容を示す．



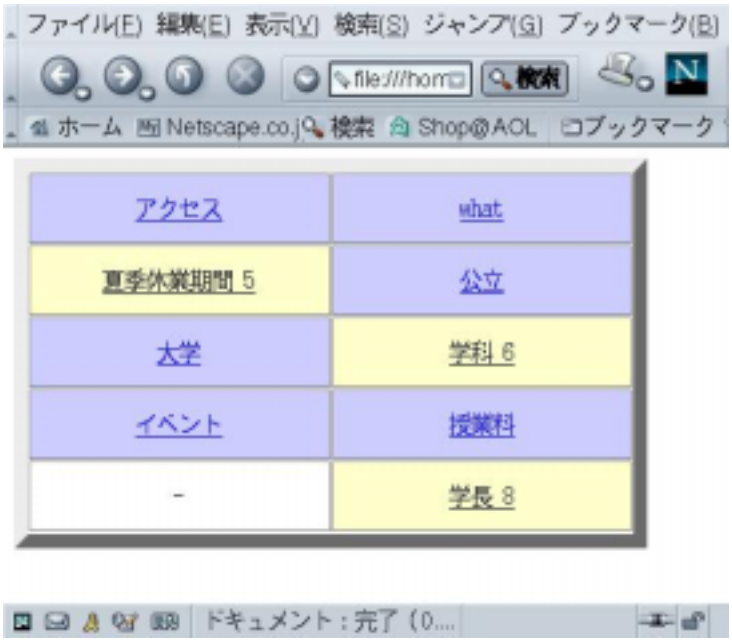
(a) トポロジー　　(b) 座標ファイル

図 5.2　座標情報

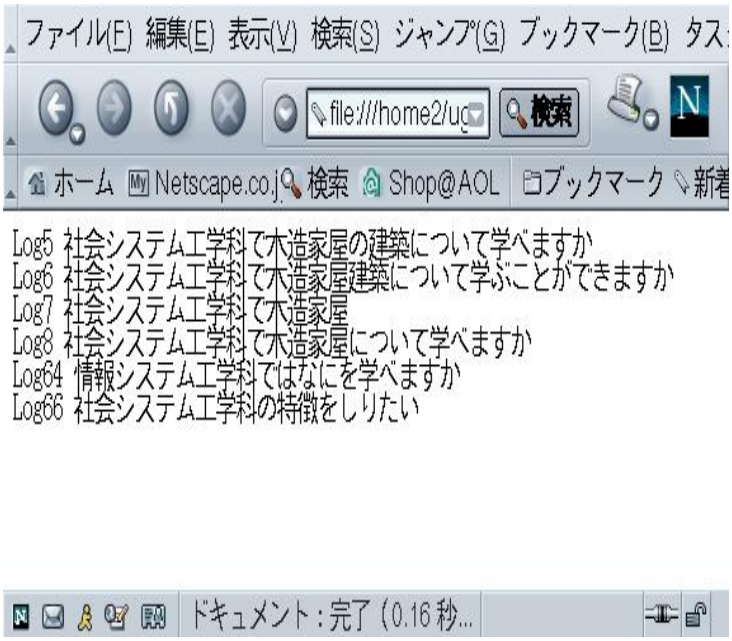
座標ファイルは入力データ，つまり，数値化ファイルと対応している．つまり，このファイルには，座標で表された各ユニットがどのデータの勝利ユニットとなったかが示されていることになる．本研究では，このファイルおよび各ユニットの重みベクトルを示すファイルを Java プログラムによって処理し，学習後の SOM マップを自動的に HTML に変換している．

5.3　階層型 SOM マップ

以下に本研究で実装した階層型 SOM マップの表示結果を示す．表示面の工夫として，HTML の利点を利用し，テーブルの背景色により下位階層を持つユニットと持たないユニットを区別できるようになっている．図 5.3(a) は第一層であり，図 5.3(b) はその中の「学科」をマウスによりクリックすると表示される最下位層である．また，図 5.4(a) は図 5.3(a) の「イベント」をクリックし，表示された第二層目のマップである．

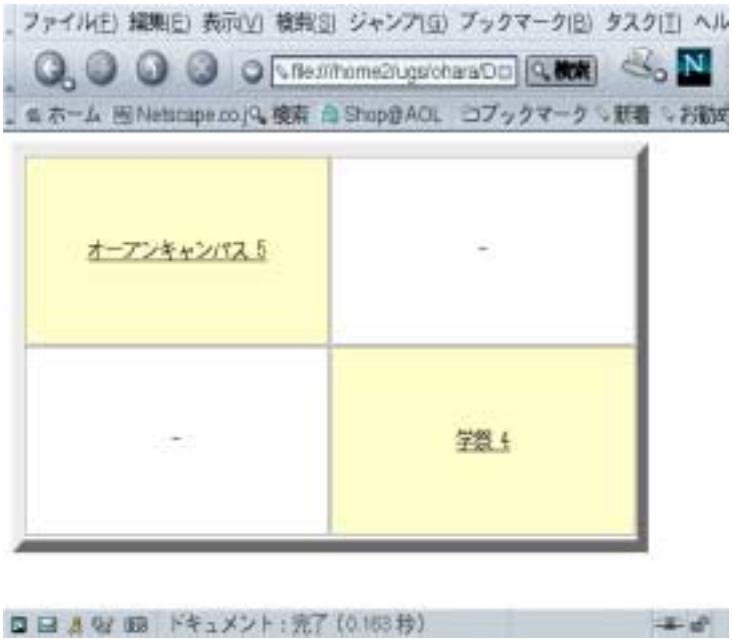


(a) 第一層

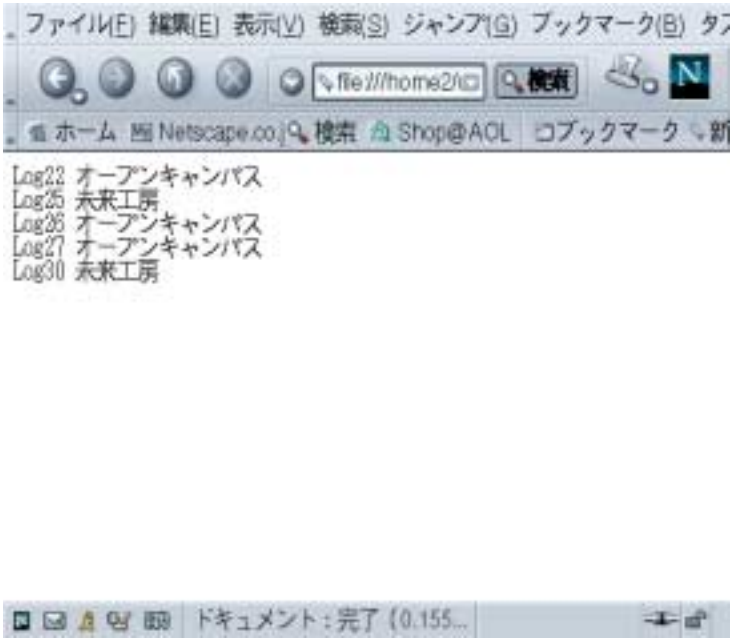


(b) 最下位層

図 5.3　学科をクリックした場合



(a) 第二層



(b) 最下位層

図 5.4　オープンキャンパスをクリックした場合

5.4 考察

一層型 SOM マップと比較して，階層型 SOM マップは扱うデータが大量の場合にもマップのコンパクトさが保てる．さらに表示部分に HTML を使用することで，扱いやすいインタフェースとなり，視覚的にも分かりやすいマップとなった．しかし，図 5.4 に示すように，人間が見て明かに違うデータが同じユニットに配置されてしまうというように，SOM によるデータ分類の正確性が欠如してしまっていた．これは一層型 SOM の場合に既に起こってしまう問題のため，階層型 SOM でもこれをひきずった形となっている．これは様々な要因が考えられ，例えば，機械的に見れば近いデータといえるのかも知れない．しかし，本研究ではログ解析に利用できるツールの作成を目的としている．つまり，システム保守者が見て，納得できる分類でなければログ解析ツールとしては意味を成さないことになる．

しかし，目的の一つであった扱いやすいインタフェースの作成は達成できた．今後，分類の精度が高くなるようなアルゴリズムを実装すれば，質問応答システムのログ解析ツールとして有効なものになると考えられる．

第 6 章

まとめ

本研究では質問応答システムのログを対象に，階層型 SOM による可視化を行い，システム保守者の負担軽減及び解析の助けとなるツールの作成を試みた．

結果として，二次元マップ上にログデータを配置することで，その内容の把握が容易となった．そして，階層化により，各マップは適切なサイズにとどまり，データ量に関わらずコンパクトにまとまったマップが形成できた．また，表示部分に HTML を使用することで，従来の SOM マップよりも自由度の高いマップとなり，システム保守者の解析効率を上げる効果が見込まれる．

しかし，従来型 SOM の学習アルゴリズムでは，入力するデータの種類によってはデータの分類がうまく出来ないという欠点がある．本研究では階層型 SOM を実装しているが，各マップ毎の学習には従来型 SOM の学習アルゴリズムを利用していたため，SOM の特徴である，マップ上の距離によるデータの類似性や分類の信頼性が高いとはいえない結果になってしまった．このため，一応の分類はできたものの，正確な解析が行えるものではなくなってしまった．

これを解決する改良型 SOM 学習アルゴリズムの提案および実装が今後の課題である．

謝辞

本研究を進めるにあたって，時にあたたかく時に厳しく見守ってくれた Ruck Thawonmas 助教授に心よりお礼申し上げます．また，労力を惜しまず手伝ってくれた人工知能研究室の皆様にも深く感謝しております．

参考文献

[1] T・コホネン，” 自己組織化マップ”pp102-171．シュプリンガー・フェアラーク東京株式会社．1996.

[2] M. Dittenbach, D. Merkl and A. Rauber, ”The Growing Hierarchical Self-Organizing Map,” S. Amari and C. L. Giles and M. Gori and V. Puri, editors, Proceedings of the International Joint Conference on Neural Networks (IJCNN 2000), vol. 6, pp. 15-19, July 24-27, 2000, Como, Italy, IEEE Computer Society. 2000.

[3] http://www.cis.hut.fi/research/som_pak/

[4] http://www.forest.dnj.ynu.ac.jp/ toshy/index-j.html