ベースモデルを分類可能な
球面自己組織化マップの開発に関する研究

2016年3月

佐賀大学大学院工学系研究科

システム創成科学専攻

新名　玄

目 次

1　緒論

1.1　研究の背景

近年, 様々なものがデジタル情報化されデータとして蓄積できるようになり, 人はそれらの蓄積されたデータを利用して, そこから新たな価値を創出しようとしている．たとえば顧客データや購買データといった企業が独自で保持している大規模なデータなどがその例である．それらのデータから購買予測につながる知見を発見したり, 新商品の開発に役立つ情報を得たりというように, データのマーケティングへの活用が期待されている．データを分析する際には, それらのデータを生成するモデルがあると考え, そのようなモデルを推定することで, データに内在する傾向や特徴を知ることができる．このようなモデルをベースモデルと呼ぶ．このとき, 得られたデータが一つのモデルから生成されている場合は, 得られたデータ全てをモデルの推定に用いることができるが, 複数のモデルから生成されたデータである場合は, データをモデルに基づいて分類する必要がある．データの分類には主成分分析や因子分析,k-means 法などが用いられるが, データが確率モデルによって生成されている場合や, データの空間的トポロジーが入り組んでいる場合は適切な分類を行うことができない．したがって, このような確率モデルに基づいたデータの分類を可能にするための手法や, データのトポロジーに基づいた分類の手法を確立させることは重要なことである.

1.2 問題点と課題

前節でも述べたように, 確率モデルから生成されたデータや, 複雑なトポロジーを持つデータにおける分類手法の確立は未だ十分とは言いがたい．確率モデルから生成されるデータについては直接データを用いるのではなく, その背景にある確率モデルに基づいてデータを分類する必要があるが, 実際にはこのようなモデルは直接観測することができず, そのモデルから出力されたデータしか観測できない場合が多い．例えば株価の変動や, タンパク質間相互作用, 音声データなどの場合である．このような確率モデルを分類する手法はこれまで多くの研究がなされ, 特に確率モデルのなかでも音声認識, 文脈情報処理, 画像認識, ゲノム解析などといった分野で幅広く活用され有用な成果を得てきた隠れマルコフモデルは, 現在でも音声認識の「Siri」などに用いられ, その実用性は高い．しかし, 隠れマルコフモデルの分類においては, 内部状態のトポロジーやパラメータをテストデータから推測し, モデルをいくつか作成した後で, どのモデルに実データがあてはまるかによって分類する手法がとられ, モデルがいくつに分かれるかを事前に決定する必要があり, モデルの適切なクラスタ数をどのように選択するかが問題となっていた [1][2].

また, 確率モデルでなくとも複雑なトポロジーをもつデータをクラスタリングすることは容易ではなく, データの空間的なつながりを考慮した分類手法が必要不可欠である．この問題に対し, データのトポロジーにもとづいた分類を可能にする自己組織化マップ [3] と呼ばれる手法が考案されたが, 実際のデータにおいては, 同じ要素間に異なる関係性をもつ混合モデルや, それぞれの要素間に異なる関係性を持つような混合モデルが存在し, 自己組織化マップを用いてもこの手の混合モデルを分類することは難しく, このような分類手法を確立することは非常に重要なことである．データを分析する際に, 因果関係のない不要な要素を含んだデータを分析しても, そのノイズに分析結果が悪影響を受けることがあり, データから価値ある知見を得ることが出来なくなる恐れがある．したがって, 不要な要素を単に除けば済むことではあるが, このときに, 要素に因果関係があるかどうかを人が経験や常識

に捉われて要素を限定しすぎると, データから得られる新たな知見の可能性を限定してしまう事にもなりかねないため, データ分析においては, まず初めに因果関係がある要素を主観に捉われず適切に抽出する必要がある．分析対象となるモデルが決まっているときは, 分析対象でないモデルと分析対象のモデルとの間の要素の違いを解析すればよいが, 対象となるモデルが決定しておらず, 異なるモデルが混在する可能性がある場合は, 要素間の特徴を解析することは極めて困難になる [4]. このような問題に対して因子化情報量基準を用いた手法が考案され, 前者の同じ要素間に異なる関係性をもつデータの分類手法は確立されつつある [5]．しかし後者のようなそれぞれの要素間に異なる関係性を持つ混合モデルの分類は未だ確立されていない．混合モデルのデータ分類に関する研究は今後, 必要不可欠である.

1.3 本研究の目的

本研究においては, 確率モデルの分類を実現するために, 教師なしでデータを分類することが可能で, 分類結果を視覚的に提示することができる自己組織化マップと, 確率モデルの内部パラメータが直接わからないシステムをモデル化することが可能な隠れマルコフモデルを用いた分類手法を提案することを第 1 の目的とする. また, 要素間に異なる関係性を持つ混合モデルの分類が分類可能なアルゴリズムを開発することを第 2 の目的としている. 前者の手法は, 自己組織化マップのノードに隠れマルコフモデルを用いることで確率モデルの分類結果をマップ上に視覚的に提示することが可能なシステムの開発を目指す. 自己組織化マップと隠れマルコフモデルを組み合わせた確率モデルの分類手法 [6][7] はすでに存在しているが, モデルの状態遷移に制約があったり, 分類が適切に行えていないなどの課題が残り, 改良が必要であり, これらの手法を本研究では発展させることを検討する. また, 後者の混合モデルの分類においては, 自己組織化マップを用いて因果関係のある要素を抽出することで, モデルの分類が可能なアルゴリズムの開発を行う. ここでは, 3 次元のデータに対して分類が可能なアルゴリズムを考案し, そのアルゴリズムを発展させることを検討するために, 異なるモデルが混合した 3 次元データをモデルに基づいて分類できるアルゴリズムの開発を目的とする.

2　自己組織化マップ（Self-Organizing Map:SOM）

2.1　SOM とは

自己組織化マップとは, コホネン (T．Kohonen) により提案されたニューラルネットワークの一つで，人の大脳皮質の視覚野をモデル化した教師なし学習モデルである．SOM の学習アルゴリズムによって学習されたデータは, 空間的トポロジーを保ちながら低次元のマップに写像することすることができ, 基本的には, マップに写像されたデータの距離関係はデータ同士の類似度が低ければ低いほど遠くに, 高ければ高いほど近くにマッピングされる．例えば, 図 1 のように, マップ上で距離の近い 2 つのデータ（data 2, data3）は, データの性質が近く離れた 2 つのデータ (data1, data2) は, 互いに類似度の低いデータであることを意味する．このように, 単に数理的計算によってデータ間の関係性を求めるだけではなく, 二次元のマップなどに写像することで, 視覚的に類似したデータを発見することの出来る手法でもある．

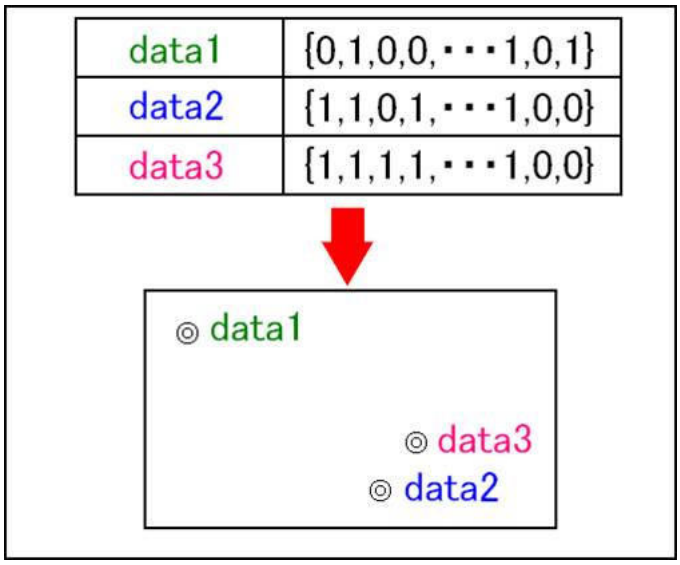


図 1: SOM におけるデータのマッピング

2.2 SOM の原理

ここでは, 例 (図 2) を用いて SOM の学習アルゴリズムについて説明する．まず, ここでは分かりやすいように, データの性質の近さを色の近さで置き換えて考える．例えば,data2 と data3 のデータは類似したデータであるから,data2 を暗い緑,data3 を明るい緑とする．それに対して,data1 は先ほどの 2 つのデータとは異なるデー

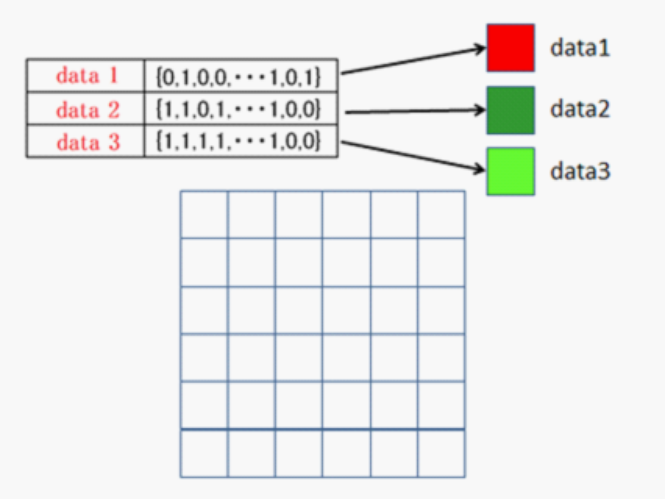


図 2: SOM に用いるデータの例と 2 次元の SOM マップ

タであり,data1 は赤色として考えることにする．また, 図 2 の下側に,SOM の学習によって data1〜data3 の高次元データを低次元に写像するための二次元平面マップを示す．ここでは説明のため, このマップは $6 \times 6 = 36$ のマスをもつとし, それぞれのデータをマッピングするための領域および, データ間のトポロジーを学習するための領域として用いる．このような領域を SOM においてはノードと呼び, 基本的にこれらは均一にマップ上に配置される．この説明においては,1 つのマスを 1 つのノードとしてとらえているが, 各格子点を各ノードとして説明されることもある．

ノードには, 入力データと同じ次元をもったベクトルが内包されており, 初期化により, 各要素のパラメータは初めランダムな値に設定される (図 3)．マップの各マスの色は, ノードに内包されているベクトルの性質を疑似的に色に置き換えたも

のであり, 各マスのランダムな色は, 各マスに内包されたベクトルの各要素が乱数で初期化されていることを示している.

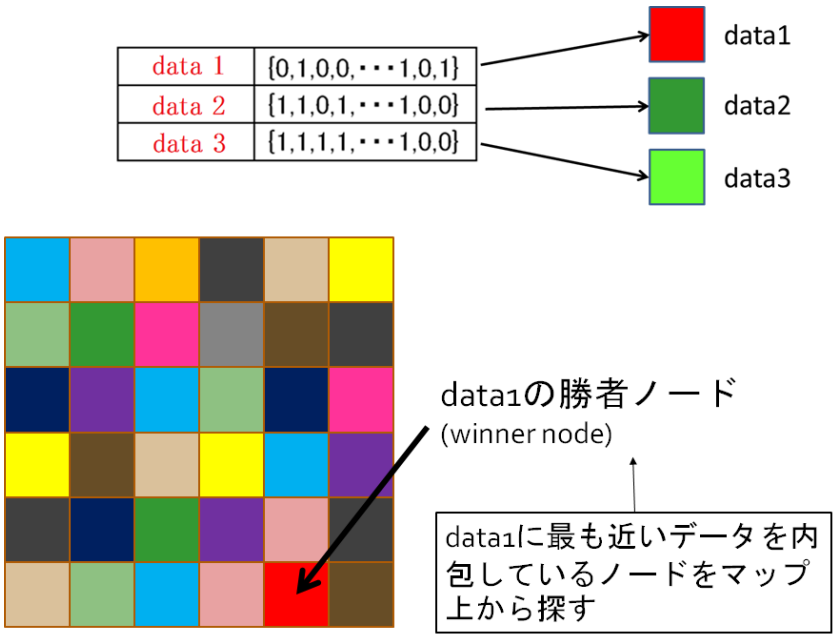


図 3: SOM におけるマップの初期化の例

次に,data1 に最も近いデータをマップ上から探す. このとき,data1 に類似したパラメータをもつノードの位置は図 3 の矢印で示した位置になる. このような, 入力データに最も類似したパラメータをもつノードの位置は勝者ノード (winner node) と呼ばれる.

次に、この勝者ノードとその近傍のノードのパラメータを data1 のデータに近づくように更新する. このとき, 図 4 のように勝者ノードから離れれば離れるほど近づける度合いが弱まるように更新する. これは, 勝者ノードから遠い距離にあるノードはほとんど更新されないことを意味し, 一方で勝者ノードに近いノードは, 強く更新されることを意味している.

したがって, 近傍のノードの更新後は data1 の勝者ノードに近いノードほどマップ上におけるノードのパラメータは data1 に類似しているといえる.

data1 と同様に,data2 に近いデータを持つノードをマップ上から探し, その位置

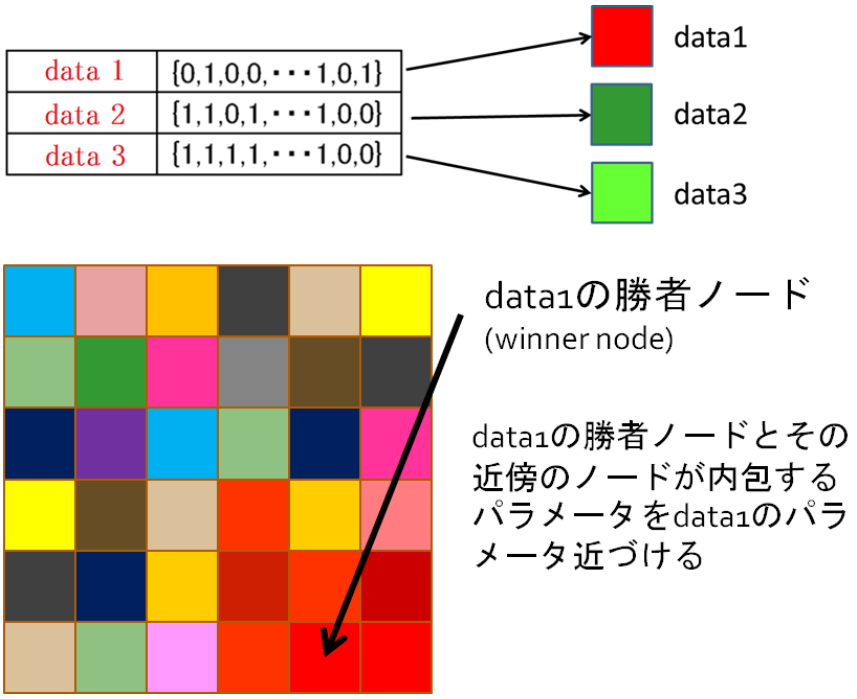


図 4: data1 におけるマップ上のノードの更新

を data2 の勝者ノードとし, 近傍を更新する．この勝者ノードは先ほどの更新によって data1 の近傍から見つかる可能性は低くなる（図 5）.

したがって, data2 の勝者ノードの近傍は更新によって data2 に類似したノードになるため, data2 に類似している data3 の勝者ノードは data2 の勝者ノードの近くで発見される可能性が高くなる（図 6）.

このように, それぞれのデータに対し勝者ノードを求め, マップの更新終わった状態を 1 回目の学習と呼ぶ.

上記のような更新（学習）によって,data3 は data2 の近くで,data2 は data1 から遠い位置というように, それぞれの勝者ノードの位置が, データの類似関係にもとづいてマップ上に写像される確率は高いが, 一回の学習だけでは適切なマップを得ることは難しい．通常はこのような更新を何度も繰り返すことで適切なマップを得る．マップの初期化の仕方や, 更新の仕方によっては, 一回の学習によってもとのデータの類似関係をマップ上に写像できるという報告もあるが, 特殊なケースで

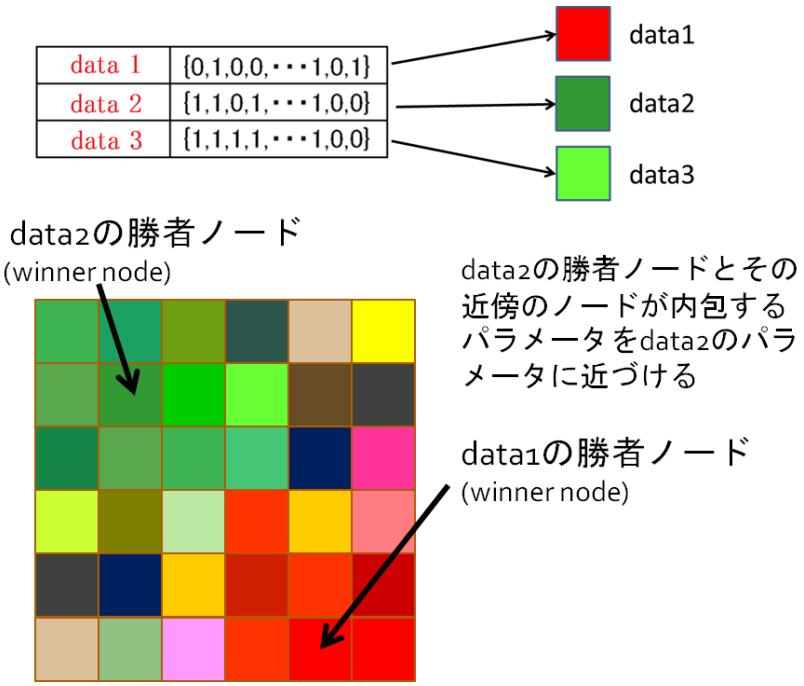


図 5: data2 におけるマップ上のノードの更新

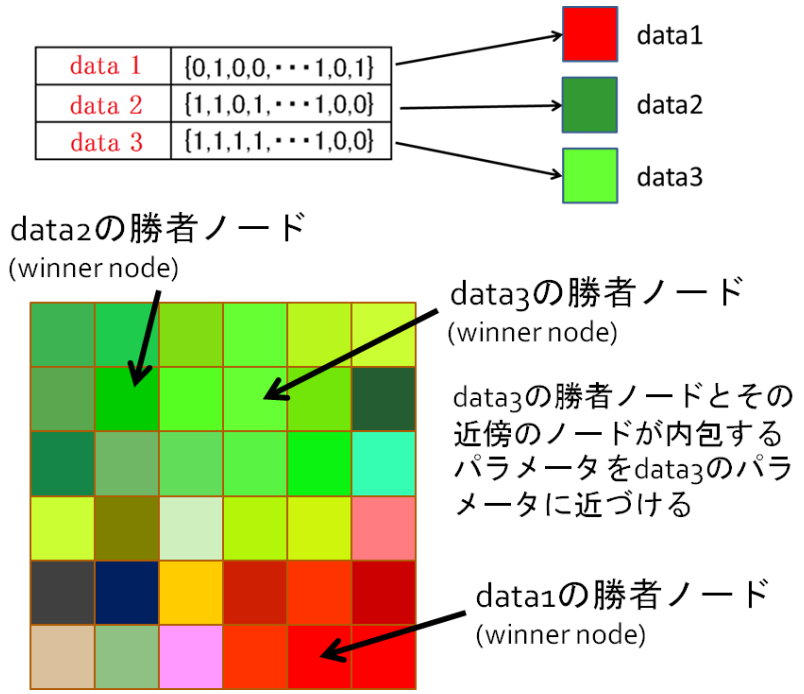


図 6: data3 におけるマップ上のノードの更新

あり, 一般的にSOMの学習アルゴリズムは, マップの初期化の後, 次に示す学習ステップを学習回分繰り返すことになる．ノード自身がマップ上を動くのではなく, ノードが内包する値が学習によって変化するため入力データと最も類似したデータをもつノード（勝者ノード）の場所が学習のたびに変化する.

以下にSOMの学習アルゴリズムを示す.

2.2.1 SOMの学習アルゴリズム

STEP 1: データの読み込み

ある入力ベクトル(データ)$\boldsymbol{v_j}$を選択し，学習データとしてSOMに読み込む.

STEP 2: 勝者ノードの決定

$\boldsymbol{v_j}$ベクトルに最も近い重みベクトル$\boldsymbol{w}$を持った勝者ノード$N^j_{k^*}$を探す$N^j_{k^*}$は以下の式に従う.

$$N^j_{k^*} = \min_k f_j(k) \tag{1}$$

$$\text{ただし, } f_j(k) = \sqrt{\sum_{t=1}^{n}(a_{jt} - w_{kt})^2} \tag{2}$$

STEP 3: ノードの更新

勝者ノード$N^j_{k^*}$とその近傍のノードの重みベクトル$\boldsymbol{w_k}$を以下の式に従って$\boldsymbol{v_j}$ベクトルに近づける.

$$h(P_{jx^*}, P_{jx}) = exp(\frac{-|P_{jx^*} - P_{jx}|^2}{\sigma^2}) \tag{3}$$

$$\Delta \boldsymbol{w_k} = \beta \cdot h(P_{jx^*}, P_{jx}) f_j(k) \tag{4}$$

$$\boldsymbol{w_k^{new}} = \boldsymbol{w_k} + \Delta \boldsymbol{w_k} \tag{5}$$

ただし，

P_{jx^*} は勝者ノード $N^j_{k^*}$ のマップ上における位置，P_{jx} はその近傍ノードの位置を表し $|P_{jx^*} - P_{jx}|$ は勝者ノードと近傍ノードのマップ上での距離である．また，σ は初期近傍範囲，β は学習係数と呼ばれ学習回数に応じて減衰する関数であり，近傍範囲は学習とともに小さくなる（図 7）．

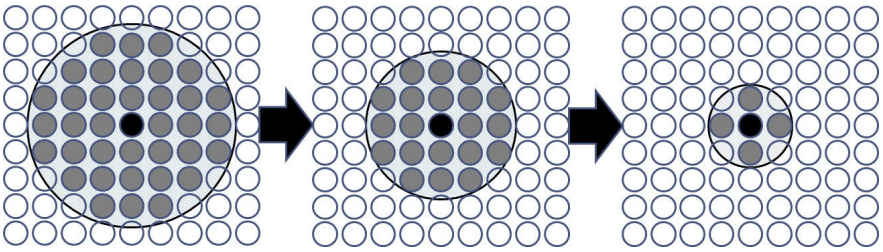


図 7: 近傍範囲の変化

このようにして 2〜4 の手順をすべての入力ベクトルに対して行い重みベクトル $\boldsymbol{w}$ を更新していく．これを学習回数分繰り返し，最終的な各入力ベクトルに対する勝者ノードの位置関係が入力データの類似関係として反映される．

2.3 SOM における問題点とその解決法

2.3.1 マップの端による不均一学習

図 8 は Data1〜Data5 までの色情報を各要素が 0〜255 の 4 ビットの三原色 R,G,B であらわし，これを入力ベクトルとし SOM の学習に用いた結果を示したものである．これは，分類として近いデータであるはずの Data2 と Data5 がマップの左上角と右下角の離れた位置にお互いがマッピングされてしまった例である．類似度が近いもの同士がマップ上で距離的に近い位置に配置されるのが理想であるが, マップの初期値の与え方や, データを学習させる順番によって, このような例外も起こり得る．この原因としてはマップの端に勝者ニューロンがきて近傍を更新する際，学習をする範囲がマップの外に及ぶ場合ノードが存在しないためこの範囲

においては学習ができず，他の入力データと競合できるノード数が減ることによって起こるものである．これまで, 例にあげてきたSOMは平面SOM（Plane-SOM）と呼ばれるもので、このように端のノードにおいては図に赤く示した領域を学習できないため, 中央にあるノードに対し, 更新できるノードが４分の３少なくなる．更新できないノードが少ないことは, その周りのノードが類似する勝者ノードとして選ばれない確率が高くなることを意味する．つまりノードが場所によっては同じ条件で学習できない恐れがありマップの端の影響をなくす必要がある．実際にそのための, 手法がいくつかありそのうちの一つがマップの上下左右をつなげることでマップの端を無くしたトーラス状のSOMである (図9:上２つ)．しかし, トーラス状のSOMにおいては, データをマッピングした際，視覚的に分類が理解しにくくなる問題が生じるため, 球面状のSOMが考案された (図9:下)．このような端をなくす手法を用いることで, 不均一学習を防ぐことができる．

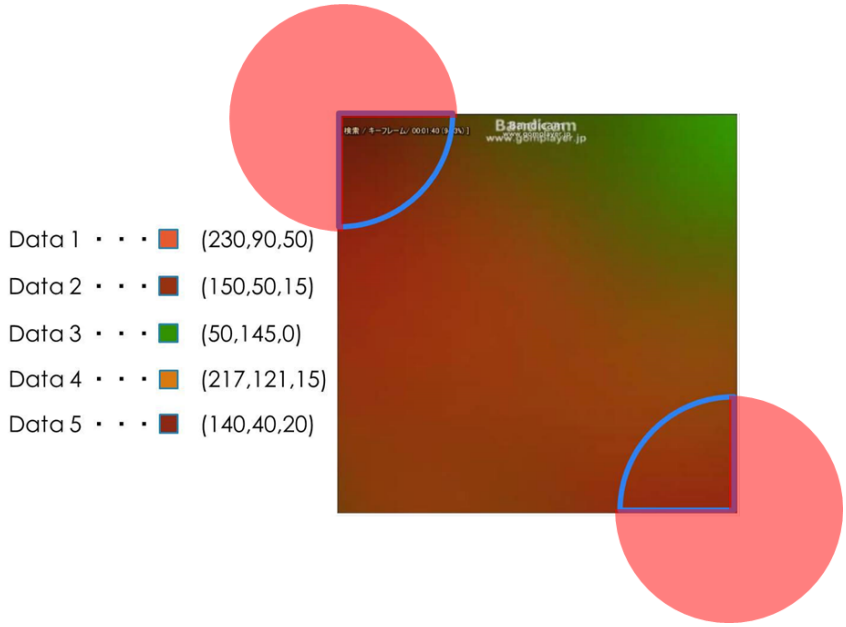


図 8: SOM における色情報の学習結果

2.4 球面自己組織化マップ (S-SOM)

SOM におけるデータの分類において端をなくし，かつマッピングの結果を理解しやすくする方法として球面状のマップを用いた SOM が提案されている [8][9][10][11]. しかし，球面を 2 次元の平面として表示すると，球面の裏側のマッピングが見えず全体の分類を見ることができない．したがって，球面のマップを表示する際，人が球面を回転させ，マップの任意の位置を見ることができるシステムの作成が必要となり，これには三次元の描画は必要不可欠である．本研究では文献 [9] で用いられている OpenGL による球面描画より，より高速に描画でき，Windows アプリケーション作成時に適した Windows の標準開発環境を提供する API 群の一つである DirectX を用いて描画する [12][13].

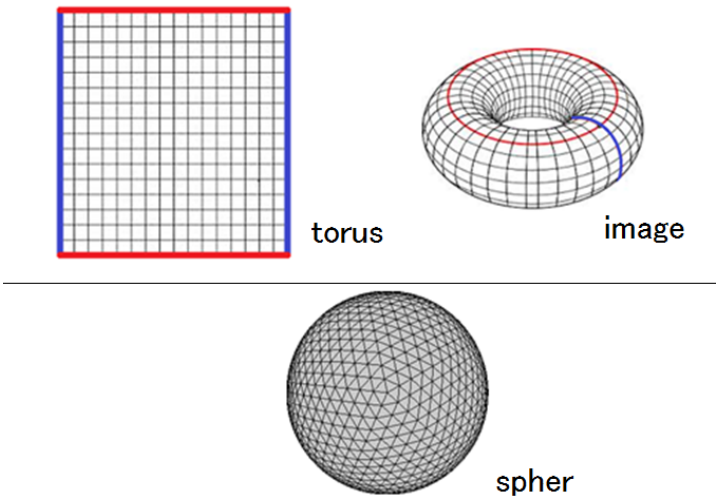


図 9: SOM のマップの端をなくすための一例

2.4.1 S-SOM におけるシステム開発について

DirectX における 3 D 描画では面の描画の際，三角形の面を張り合わせていくことで任意の面を構成し仮想空間に形を作る．この物体をどこから見たときの描画を行うかを，仮想カメラの位置と向きを指定することで，このカメラに映る視野範囲を描画できる．この時，プログラム内では任意の面を構成する三角形の頂点座標を格納する頂点配列を用意し，それぞれの三角形について反時計回りに頂点を格納する（図 10）．反時計回りでなく時計回りで格納してもよいが, このシステムにおいてはカメラに対して反時計回りに格納されたポリゴンのみを描画し，時計回りのポリゴンは描画しないようにしている．これはカメラに対して表向きと裏向きを考える概念であり，カメラからは見えない裏向きの面を描画しないこと（これをカリングという）で描画する際の負担を減らし，より高速な描画を可能にするものである．

球面を作成する際は，ポリゴンの張り合わせによってこの球面を作る必要があり，その作成方法は正二十面体の面である三角形それぞれについてその辺の中点

図 10: ポリゴンの頂点座標の格納

を求め，正二十面体に外接する球面上に写像することによってできる点をその三角形からできる新たな頂点とし結ぶことで，一つの三角形でできた面を，４つの面に増やすことができる．この操作をすべての面に対しおこなうことで正二十面体の分割をする.（図 11）．この工程を繰り返すことで，正二十面体から球面を得ることができる.

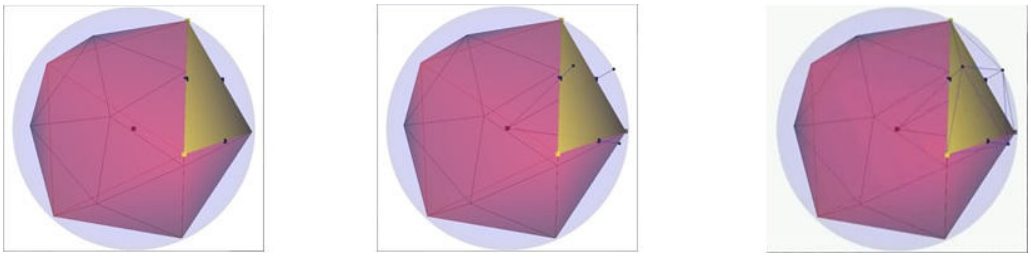


図 11: 球面の作成方法

このように分割してできたひとつひとつの三角形が球面を作るためのポリゴンとなり，頂点配列に反時計回りのカリングを考慮してそれぞれの三角形の頂点座標を格納する．このときこの頂点配列から，それぞれ頂点座標を得られるが，ポリゴンによる球面を考えるため，いくつもの同じ頂点座標をこの頂点配列が含むことになり，この頂点配列をそのまま S-SOM におけるノードの位置情報として扱うことができない．このため，この頂点配列から SOM に必要な頂点のみを別の配列に格納する必要があり，これらの頂点を見つけるための時間が分割数が多けれ

ば多いほどかかることになる．また，SOM の学習後にこの球面上への分類結果を反映する際にも SOM 用の頂点情報を含む配列から球面を描画するためのポリゴン用の配列に変換する計算が必要で，S-SOM においてこの計算過程はシステムに負担をかける．

したがって，本研究ではポリゴン用頂点配列と SOM 用頂点配列との関連情報をテーブル化し保存しておき，描画に必要なポリゴン情報と SOM の頂点座標の相互関係を参照する必要がある場合は，このテーブルを用いることで処理の高速化を実現した．また，処理同士が重ならないものについてはプログラムをスレッド化し並列計算をさせることでシステムの高速化を実現している（図 12）．

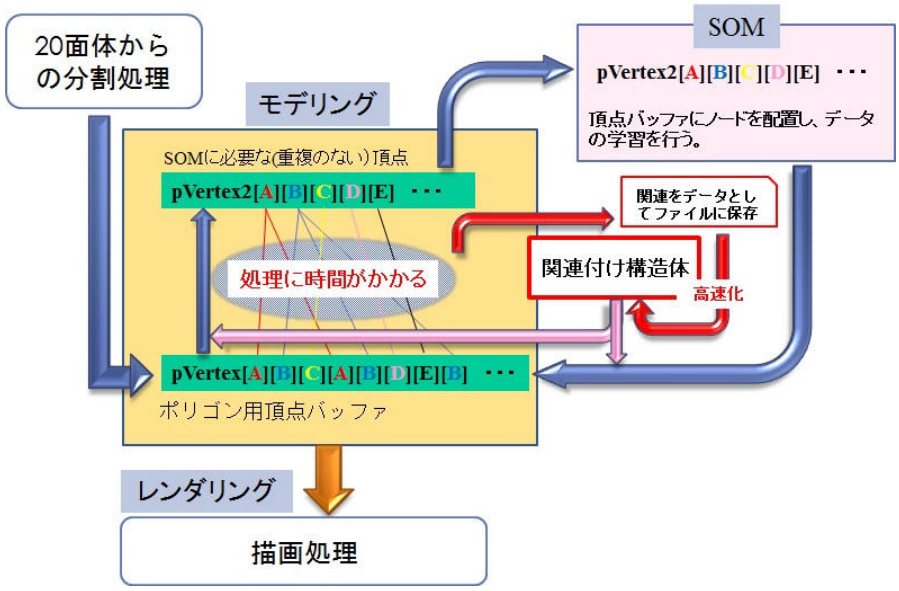


図 12: S-SOM の面描画システム概要図

2.4.2 S-SOM の学習アルゴリズム

n 次元で m 個の入力データ $\boldsymbol{v_j} = \{a_{j1}, a_{j2}, a_{j3}, ..., a_{jn}\}(j = 1, 2, 3, ..., m)$ を S-SOM に学習させることを考える．このとき，マップ上における l 個すべてのノード $N_k(k = 1, 2, 3, ..., l)$ は入力データと同じ次元の $\boldsymbol{w_k} = \{w_{k1}, w_{k2}, w_{k3}, ..., w_{kn}\}$ なる重みベクトルを持つ．

以下にS-SOMの学習ステップを示す

STEP 1: 重みベクトル w の初期化

マップ上のすべてのノードがもつ重みベクトル $\boldsymbol{w}$ を乱数で初期化する.

STEP 2: 入力ベクトル v の読み込み

ある入力ベクトル $\boldsymbol{v_j}$ を選択し，学習データとしてSOMに読み込む.

STEP 3: 勝者ノードの決定

$\boldsymbol{v_j}$ ベクトルに最も近い重みベクトル $\boldsymbol{w}$ を持った勝者ノード $N^j_{k^*}$ を探す $N^j_{k^*}$ は以下の式に従う.

$$N^j_{k^*} = \min_k f_j(k) \tag{6}$$

ただし,

$$f_j(k) = \sqrt{\sum_{t=1}^{n}(a_{jt} - w_{kt})^2} \tag{7}$$

STEP 4: ノードの更新

勝者ノード $N^j_{k^*}$ とその近傍のノードの重みベクトル $\boldsymbol{w_k}$ を以下の式に従って $\boldsymbol{v_j}$ ベクトルに近づける.

$$h(P_{jx^*}, P_{jx}) = exp(\frac{-|P_{jx^*} - P_{jx}|}{\theta^2}) \tag{8}$$

$$\Delta \boldsymbol{w_k} = \beta \cdot h(P_{jx^*}, P_{jx}) f_j(k) \tag{9}$$

$$\boldsymbol{w_k^{new}} = \boldsymbol{w_k} + \Delta \boldsymbol{w_k} \tag{10}$$

ただし，P_{jx^*} は勝者ノード $N^j_{k^*}$ のマップ上における位置，P_{jx} はその近傍ノードの位置を表し $|P_{jx^*} - P_{jx}|$ は球の中心に対する勝者ノードと近傍ノードのマップ上での立体角である．θ は初期近傍範囲の広さを表す立体角，β は学

習係数．θ の近傍は学習回数とともに小さくなるように設定し（図 13）β は減衰関数を用いる．

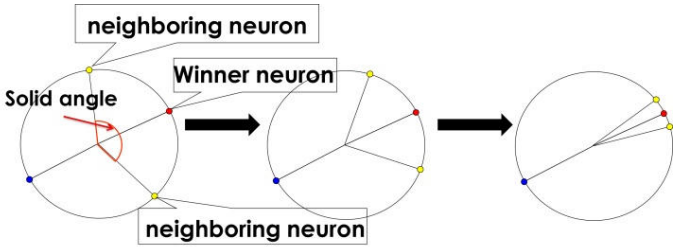
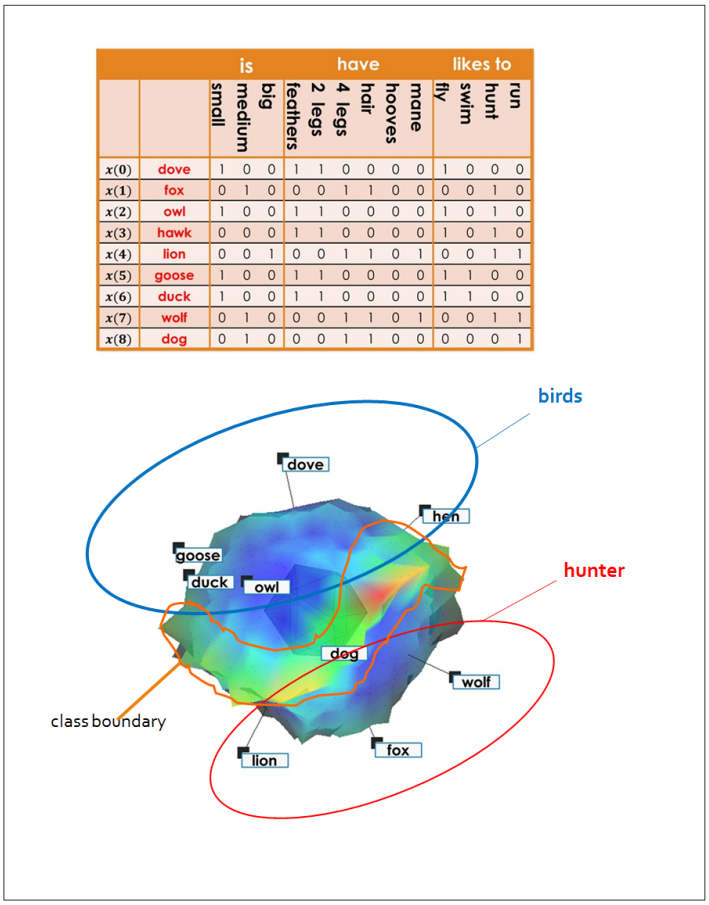


図 13: S-SOM における近傍範囲の変化

2.4.3 S-SOM における U-matrix

よく SOM の例で用いられる動物データを本研究で作成した S-SOM 学習させた結果を図 14 に示す．学習させる入力ベクトルはそれぞれの動物の特徴を要素に分け，当てはまる要素には 1 それ以外には 0 の値を設定した．



		is			have						likes to			
		small	medium	big	feathers	2 legs	4 legs	hair	hooves	mane	fly	swim	hunt	run
$x(0)$	dove	1	0	0	1	1	0	0	0	0	1	0	0	0
$x(1)$	fox	0	1	0	0	0	1	1	0	0	0	0	1	0
$x(2)$	owl	1	0	0	1	1	0	0	0	0	1	0	1	0
$x(3)$	hawk	0	0	0	1	1	0	0	0	0	1	0	1	0
$x(4)$	lion	0	0	1	0	0	1	1	0	1	0	0	1	1
$x(5)$	goose	1	0	0	1	1	0	0	0	0	1	1	0	0
$x(6)$	duck	1	0	0	1	1	0	0	0	0	1	1	0	0
$x(7)$	wolf	0	1	0	0	0	1	1	0	1	0	0	1	1
$x(8)$	dog	0	1	0	0	0	1	1	0	0	0	0	0	1

図 14: S-SOM による動物マップ

このマッピング結果は大きく分けて上が鳥類、下が四足動物というように中央をはしる青以外の山によって境界線を境に分けることができる．このように，マップにおける分類をわかりやすくするために，類似度の遠いデータ同士の間に山をつくるなどし，分類をより視覚的に理解できるようにした手法を U-matrix という．本研究においてはカラーリングも行っている．

以下に S-SOM における U-matrix について記述する．

S-SOM の球面が正二十面体を分割数を n 分割して作成されたものであれば，n-1 分割目に作られる点を S-SOM におけるノードとし，n 分割目によって各ノードの間にできる新たな頂点を各ノードが内包する重みベクトルの差を記憶するユニットとして扱う．図 16 は図 15 における面の一部で，隣り合う 6 面からなる領域を球の中心方向にみたものである．この六角形の角と中心がノードの位置で，三角形それぞれの辺の中央が U-matrix のユニットとなる．

ユニット T_g には各ノード間の差を保存する変数 U^g_{ab} を内包している．ただし、U^g_{ab} は隣接するノード N_a とノード N_b における重みベクトルの差を意味し $U^g_{ab} = U^g_{ba}$ が成り立ち U^g_{ab}, U^g_{ba} をまとめて U^g と定義する．ここで，マップ上におけるノード $N_k(k = 1, 2, 3, ..., l)$ が重みベクトル $\boldsymbol{w_k} = \{w_{k1}, w_{k2}, w_{k3}, ..., w_{kn}\}$ を内包しているとすれば，SOM の学習後，U^g_{ab} は以下の式で求めることができる．

$$U^g_{ab} = \sqrt{\sum_{j=1}^{n}(w_{aj} - w_{bj})^2} \tag{11}$$

このとき，ユニット T_g，ノード N_a を持つそれぞれの頂点の球の中心からの距離を V_a, V_g とするとそれぞれの頂点の位置は，球の半径 L を用いて次式に従い更新される．

$$V_g = L \cdot (\alpha U^g + \frac{\max_h(U^h) - M \cdot \min_h(U^h)}{\max_h(U^h) - \min_h(U^h)}) \tag{12}$$

$$V_a = L \cdot \frac{1}{6}\sum_j (\alpha U^g_{aj} + \frac{\max_h(U^h) - M \cdot \min_h(U^h)}{\max_h(U^h) - \min_h(U^h)}) \tag{13}$$

ただし，

$$\alpha = \frac{M - 1}{\max_h(U^h) - \min_h(U^h)} \tag{14}$$

V_g は $\min_h(U^h)$ を $0, \max_h(U^h)$ を M に正規化した際の U^g の値に L をかけたもの

で，球面の頂点位置は最大でも球の中心から $\mathrm{M} \cdot L$ 離れることを意味する．一方，V_a については，ノード N_a に隣接する 6 つのユニットが持つ変数 U_a を正規化した値の平均値と L との積になる．

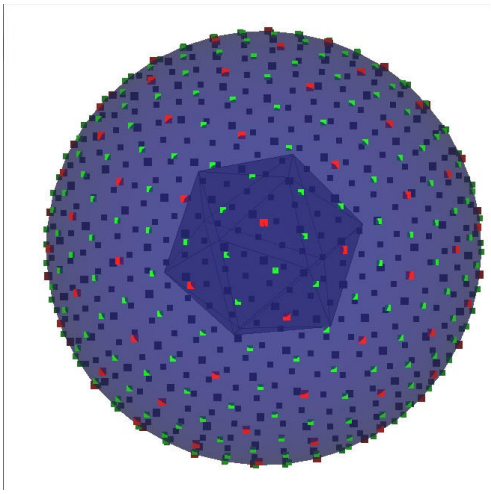


図 15: S-SOM における U-matrix

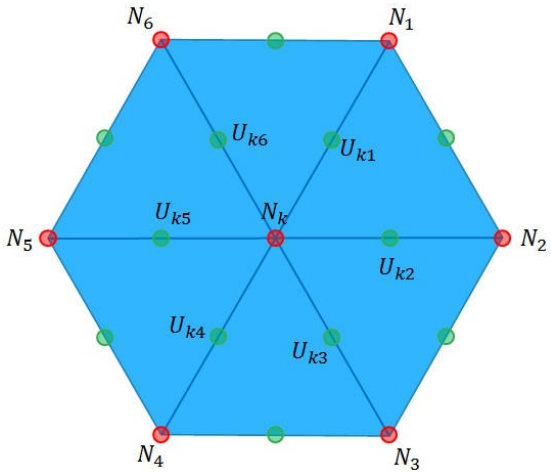


図 16: S-SOM における U-matrix の概念図

3　隠れマルコフモデル

3.1　隠れマルコフモデル (HMM) とは

隠れマルコフモデルとは初期状態から始まり，確率的に状態遷移していきながら各状態において確率的にシンボルを１つ出力するもので，現在の状態のみによって次の状態の遷移が確率的に決まるマルコフ性をもつモデルである．このように状態遷移を繰り返しながら，最終状態で遷移を停止する．観測することができるのは各状態から出力されるシンボル出力のみで，これらのシンボルがどの状態を遷移しながら出力されたものであるかは観測することができない．しかし，Baum-Welch algorithm を用いることで，観測されたシンボル系列からモデルのパラメータを推測することができる．また，モデルが内包するこれらのパラメータから，観測されるシンボルが出力される尤もらしい経路を推定することができる Viterbi algorithm とよばれる手法がある．これらのアルゴリズムについては後に記述する．

3.2　HMM におけるアルゴリズム

HMM における各パラメータを以下のように定義する．

■ $Q = \{q_1, q_2, q_3, ..., q_k\}$: 状態 q の集合

■ $a_{i,j}$: 状態 q_i から状態 q_j に遷移する状態遷移確率．ただし，$\sum_j a_{i,j} = 1$

■ $s_i(x)$: 状態 i からシンボル x を出力するシンボル出力確率．ただし，最終状態ではシンボルは出力しない．それ以外では $\sum_x s_i(x) = 1$ が成り立つ．

■ $w[t]$: 遷移 t 回目において出力されるシンボル．

■ Θ : HMM に内包される状態遷移確率およびシンボル出力確率のパラメータの集合.

3.2.1 Forward algorithm

Forward algorithm とは HMM においてシンボル出力系列 $\boldsymbol{w}$ が観測されたとき，シンボル出力系列 $\boldsymbol{w}$ が出力される確率を求めるアルゴリズムである.

シンボルが N 個観測されるときはその経路 R は N 個の状態を遷移したことになる．ただし，経路 R は一つとは限らない．経路 R が M 個あるとすれば，経路 R は以下の式で定義できる.

$$R_m(W) = \{r_m[0], r_m[1], r_m[2], ..., r_m[N]\} \qquad (m = 1, 2, 3, ..., M) \tag{15}$$

ここで，$r_m[t]$ は経路 $R_m(W)$ において、遷移回数 t 回目における遷移した状態の番号を表し，シンボル系列 S が観測される確率（尤度は）L_w は次のようになる.

$$L_w = \sum_{m=1}^{M} \sum_{t=1}^{N} s_{g(m,t-1)}(w[t]) \cdot a_{g(m,t-1),j} \qquad \text{ただし } g(m,t) = r_m[t] \tag{16}$$

具体的には，図 17 に示す状態数 4，シンボル出力を A, B, C の 3 種類とした HMM において観測されるシンボル系列 $\boldsymbol{w}$ が $AABC$ であるとすると，このモデルがシンボル系列 $\boldsymbol{w}$ を出力できる状態遷移の経路は全部で 3 通りあることになる（表 1）.

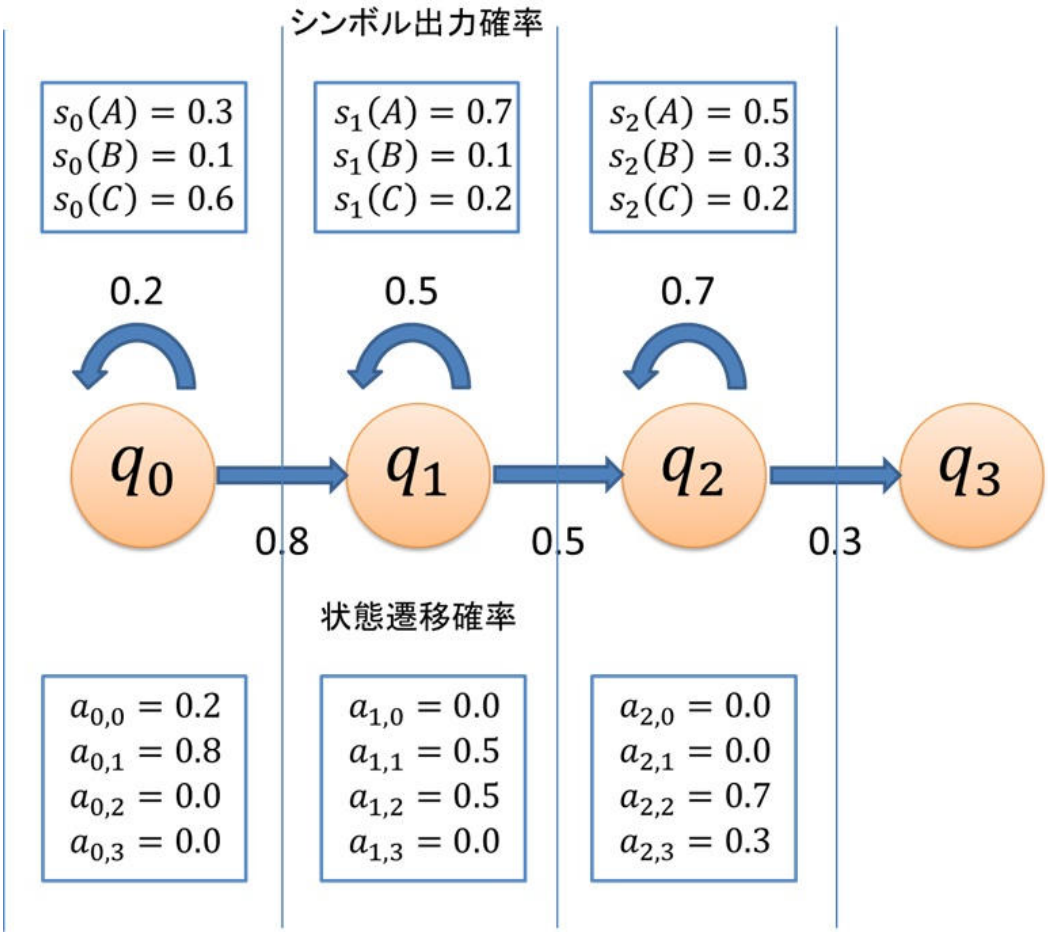

シンボル出力確率
$s_0(A) = 0.3$
$s_0(B) = 0.1$
$s_0(C) = 0.6$
$s_1(A) = 0.7$
$s_1(B) = 0.1$
$s_1(C) = 0.2$
$s_2(A) = 0.5$
$s_2(B) = 0.3$
$s_2(C) = 0.2$
0.2
0.5
0.7
q_0
q_1
q_2
q_3
0.8
0.5
0.3
状態遷移確率
$a_{0,0} = 0.2$
$a_{0,1} = 0.8$
$a_{0,2} = 0.0$
$a_{0,3} = 0.0$
$a_{1,0} = 0.0$
$a_{1,1} = 0.5$
$a_{1,2} = 0.5$
$a_{1,3} = 0.0$
$a_{2,0} = 0.0$
$a_{2,1} = 0.0$
$a_{2,2} = 0.7$
$a_{2,3} = 0.3$

図 17: モデルの例

表 1: シンボル系列 $\boldsymbol{w}$ を出力できる経路

遷移回数 t	0	1	2	3	4
出力シンボル $w[t]$		A	A	B	C
経路$R_1(\boldsymbol{w})$	0	0	1	2	3
経路$R_2(\boldsymbol{w})$	0	1	1	2	3
経路$R_3(\boldsymbol{w})$	0	1	2	2	3

このとき，それぞれの状態遷移での経路 R_1〜R_3 におけるシンボル系列 $\boldsymbol{w}$ を出力する確率は以下の通りである．

経路 R_1:

$$\begin{aligned} & a_{0,0} \cdot s_0(w[1]) \cdot a_{0,1} \cdot s_0(w[2]) \cdot a_{1,2} \cdot s_1(w[3]) \cdot a_{2,3} \cdot s_2(w[4]) \\ = \; & a_{0,0} \cdot s_0(A) \cdot a_{0,1} \cdot s_0(A) \cdot a_{1,2} \cdot s_1(B) \cdot a_{2,3} \cdot s_2(C) \\ = \; & 0.2 \cdot 0.3 \cdot 0.8 \cdot 0.3 \cdot 0.5 \cdot 0.1 \cdot 0.3 \cdot 0.2 \\ = \; & 0.0000432 \end{aligned} \tag{17}$$

経路 R_2:

$$\begin{aligned} & a_{0,1} \cdot s_0(w[1]) \cdot a_{1,1} \cdot s_1(w[2]) \cdot a_{1,2} \cdot s_1(w[3]) \cdot a_{2,3} \cdot s_2(w[4]) \\ = \; & a_{0,1} \cdot s_0(A) \cdot a_{1,1} \cdot s_1(A) \cdot a_{1,2} \cdot s_1(B) \cdot a_{2,3} \cdot s_2(C) \\ = \; & 0.8 \cdot 0.3 \cdot 0.5 \cdot 0.7 \cdot 0.5 \cdot 0.1 \cdot 0.3 \cdot 0.2 \\ = \; & 0.0002520 \end{aligned} \tag{18}$$

経路 R_3:

$$
\begin{aligned}
& a_{0,1} \cdot s_0(w[1]) \cdot a_{1,2} \cdot s_1(w[2]) \cdot a_{2,2} \cdot s_2(w[3]) \cdot a_{2,3} \cdot s_2(w[4]) \\
= \; & a_{0,1} \cdot s_0(A) \cdot a_{1,2} \cdot s_1(A) \cdot a_{2,2} \cdot s_2(B) \cdot a_{2,3} \cdot s_2(C) \\
= \; & 0.8 \cdot 0.3 \cdot 0.5 \cdot 0.7 \cdot 0.7 \cdot 0.3 \cdot 0.3 \cdot 0.2 \\
= \; & 0.0010584 \qquad (19)
\end{aligned}
$$

よって，このモデルにおいてシンボル系列 $\boldsymbol{w}$ を出力する確率，つまり尤度 L_w は式 17〜式 19 によって求めたそれぞれの確率の和であり以下の計算で求まる．

$$
\begin{aligned}
L_w &= 0.0000432 + 0.0002520 + 0.0010584 \\
&= 0.0013536 \qquad (20)
\end{aligned}
$$

しかし，式 17〜式 19 の計算には重複している部分があり，再計算をするため計算効率が悪い．特に，状態数が増えたりシンボル出力系列の長さが長くなると，取り得る状態遷移の経路は急激に増え，この方法では計算の重複する箇所も増えるため計算に時間がかかる．したがって尤度をもとめる際には，計算途中を記憶しておくことで計算の重複を避けて尤度を求めることが可能な次の再帰式にもとづく動的計画法アルゴリズムを用いる．このアルゴリズムは Forward algorithm と呼ばれている．

$$
f_j(t) = s_i(w[t]) \sum_{q_i \in Q} f_i(t-1) a_{i,j} \qquad (t \geq 1) \qquad (21)
$$

ただし $f_0(0) = 1$ とする．

$f_j(t)$ は、遷移回数 t 回目において最後の状態が q_j という条件のもとでシンボル系列 $\boldsymbol{w} = \{w[1], w[2], ..., w[t]\}$ が出力される確率である．

Forward algorithm によって, 尤度 L_w をもとめる場合の計算は次のように行う.

$$
\begin{aligned}
f_0(0) &= 1.0 \\
f_0(1) &= f_0(0) \cdot a_{0,0} \cdot s_0(w[1]) = f_0(0) \cdot a_{0,0} \cdot s_0(A) \\
&= 1.0 \cdot 0.2 \cdot 0.3 = 0.06 \\
f_1(1) &= f_0(0) \cdot a_{0,1} \cdot s_0(w[1]) = f_0(0) \cdot a_{0,1} \cdot s_0(A) \\
&= 1.0 \cdot 0.8 \cdot 0.3 = 0.24 \\
f_1(2) &= f_0(1) \cdot a_{0,1} \cdot s_0(w[2]) + f_1(1) \cdot a_{1,1} \cdot s_1(w[2]) \\
&= f_0(1) \cdot a_{0,1} \cdot s_0(A) + f_1(1) \cdot a_{1,1} \cdot s_1(A) \\
&= 0.06 \cdot 0.8 \cdot 0.3 + 0.24 \cdot 0.5 \cdot 0.7 \\
&= 0.0984 \\
f_2(2) &= f_1(1) \cdot a_{1,2} \cdot s_1(w[2]) = f_1(1) \cdot a_{1,2} \cdot s_1(A) \\
&= 0.24 \cdot 0.5 \cdot 0.7 = 0.084 \\
f_2(3) &= f_1(2) \cdot a_{1,2} \cdot s_1(w[3]) + f_2(2) \cdot a_{2,2} \cdot s_2(w[3]) \\
&= f_1(2) \cdot a_{1,2} \cdot s_1(B) + f_2(2) \cdot a_{2,2} \cdot s_2(B) \\
&= 0.0984 \cdot 0.5 \cdot 0.1 + 0.084 \cdot 0.7 \cdot 0.3 = 0.02256 \\
f_3(4) &= f_2(3) \cdot a_{2,3} \cdot s_2(w[4]) = f_2(3) \cdot a_{2,3} \cdot s_2(C) \\
&= 0.02256 \cdot 0.3 \cdot 0.2 = 0.0013536
\end{aligned}
$$

これより尤度 $L_w = f_3(4) = 0.0013536$ となり，この値は式 20 に等しい.

3.2.2 Backward algorithm

Forward algorithm ではシンボル系列 $\boldsymbol{w}$ が与えられたとき，初期状態から q_j で終わる経路を考えたが，Backward algorithm においては後ろ向きに，最終状態から q_j で終わる経路において，$\boldsymbol{w} = \{w[N], w[N-1], ..., w[t+1]\}$ を出力する確率を求めるものであり，その確率を $b_j(t)$ は以下の式で定義される．

$$b_i(t) = \sum_{q_j \in Q} b_j(t+1) a_{i,j} \cdot s_i(w[t+1]) \tag{22}$$

図 17 におけるモデルに対しての観測シンボル系列 $AABC$ の尤度 L_w を Backward algorithm を用いて求める場合は次のように計算する．

$$
\begin{aligned}
b_3(4) &= 1.0 \\
b_2(3) &= b_3(4) \cdot a_{2,3} \cdot s_2(w[4]) = b_3(4) \cdot a_{2,3} \cdot s_2(C) \\
&= 1.0 \cdot 0.3 \cdot 0.2 = 0.06 \\
b_2(2) &= b_2(3) \cdot a_{2,2} \cdot s_2(w[3]) = b_2(3) \cdot a_{2,2} \cdot s_2(B) \\
&= 0.06 \cdot 0.7 \cdot 0.3 = 0.0126 \\
b_1(2) &= b_2(3) \cdot a_{1,2} \cdot s_1(w[3]) = b_2(3) \cdot a_{1,2} \cdot s_1(B) \\
&= 0.06 \cdot 0.5 \cdot 0.1 = 0.003 \\
b_1(1) &= b_2(2) \cdot a_{1,2} \cdot s_1(w[2]) + b_1(2) \cdot a_{1,1} \cdot s_1(w[2]) \\
&= b_2(2) \cdot a_{1,2} \cdot s_1(A) + b_1(2) \cdot a_{1,1} \cdot s_1(A) \\
&= 0.0126 \cdot 0.5 \cdot 0.7 + 0.003 \cdot 0.5 \cdot 0.7 = 0.00546 \\
b_0(1) &= b_1(2) \cdot a_{0,1} \cdot s_0(w[2]) = b_1(2) \cdot a_{0,1} \cdot s_0(A) \\
&= 0.003 \cdot 0.8 \cdot 0.3 = 0.00072 \\
b_0(0) &= b_1(1) \cdot a_{0,1} \cdot s_0(w[1]) + b_0(1) \cdot a_{0,0} \cdot s_0(w[1]) \\
&= b_1(1) \cdot a_{0,1} \cdot s_0(A) + b_0(1) \cdot a_{0,0} \cdot s_0(A) \\
&= 0.00546 \cdot 0.8 \cdot 0.3 + 0.00072 \cdot 0.2 \cdot 0.3 = 0.0013536
\end{aligned}
$$

よって $L_w = b_0(0) = 0.0013536$

したがって，Backward algorithm を用いてもとめた $b_0(0)$ は Forward algorithm を用いてもとめた $f_3(4)$ と等しくなり，以下の式が成り立つ.

$$b_0(0) = f_3(4)$$

3.2.3 Viterbi algorithm

Forward algorithm において，$f_j(t) = s_i(w[t]) \sum_{q_i \in Q} f_i(t-1)a_{i,j}$ と再帰式によりシンボル系列 $\boldsymbol{w}$ が出力されるすべての経路を求めるが，Viterbi algorithm では，それらの経路のうち，シンボル系列 W を出力する尤度が最大になる経路 $R^*_m(\boldsymbol{w})$ を求める．

$$L^*_w = arg\max \sum_{t=0}^{N} s_{g(m,t)}(w[t]) \cdot a_{g(m,t),j} \qquad \text{ただし } g(m,t) = r_m[t] \tag{23}$$

この経路は，動的計画法アルゴリズムの次式

$$f_j(t) = s_i(w[t]) \max_{q \in Q} f_i(t-1)a_{i,j} \tag{24}$$

を計算することで求めることができる．

3.2.4 Baum-Welch algorithm

HMM で観測される出力系列から，どの経路を通ったかを Viterbi algorithm において求めることができた．しかし，これは HMM に内包しているパラメータ Θ が既知であることが前提であり，隠れマルコフモデルにおいては観測できるのはパラメータ Θ ではなく，出力されるシンボル系列 $\boldsymbol{w}$ のみであった，したがって，Viterbi algorithm のみではシンボル系列 $\boldsymbol{w}$ を出力する尤もらしいパラメータ Θ を推定できない，

そこで，Baum-Welch algorithm では未知のパラメータ Θ を仮定して考え，このパラメータを内包する HMM においてシンボル系列 $\boldsymbol{w}$ が出力されるとき，各状態遷移および各状態からシンボルが出力される回数の期待値を求めることによってこれらの期待値から尤もらしいパラメータを求め，仮定したパラメータ Θ を新しく求まったパラメータに置き換える．この過程を繰り返し，パラメータを更新していくことで未知のパラメータ Θ を推定する．

このアルゴリズムによって更新を繰り返していくことで Forward algorithm によって求まる尤度 L_w は極大値に達するまで増大していくものであり，最大値ではない.（図 18）この極大値に関しては初期パラメータの与え方によっては最適なパラメータに収束するとは限らない．ただし，状態遷移において次の遷移が自身に戻ることを除いて，以前の状態には遷移しない left-to-light のモデルにおいては収束性がよいことが知られている．

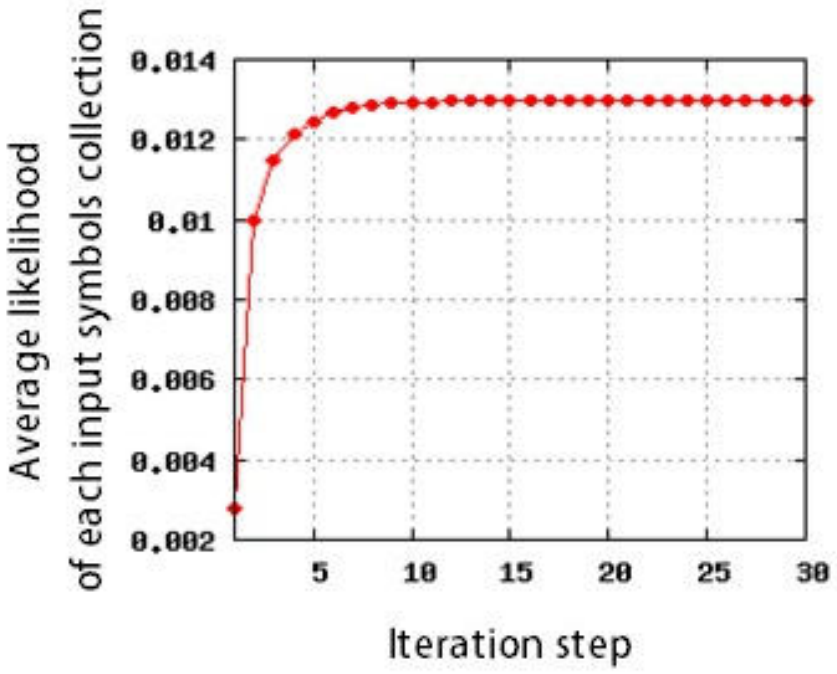


図 18: Baum-Welch algorithm による尤度の変化

以下に Baum-Welch algorithm にけるパラメータの更新について記述する．変数を次のように定義する．

■ $O_{i,j}$: シンボル列集合 $\mathbb{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_l\}$ を与えたとき，状態 q_i から状態 q_j に遷移が起こる回数の期待値.

■ $E_i(x)$: シンボル列集合 $\mathbb{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_l\}$ を与えたとき，シンボル x が状態 q_i から出力される回数の期待値.

■ $P(\boldsymbol{w}_k|\Theta)$: シンボル列集合 $\mathbb{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_l\}$ を与えたとき，パラメータ Θ を内包するモデルがシンボル系列 $\boldsymbol{w}_k$ を出力する尤度 L_w であり，$P(\boldsymbol{w}_k|\Theta) = L_w$ が成り立つ．ここで，モデルがシンボル列集合 $\mathbb{W}$ を出力する尤度 $P(\mathbb{W}|\Theta)$ は次のように定義できる．

$$P(\mathbb{W}|\Theta) = \frac{\sum_{k=1}^{l} L_k}{l}$$

これは尤度の平均値を意味する．

また, 期待値 $O_{i,j}$ および $E_i(x)$ は以下の式で計算できる．ただし，$t \geq 1$

$$O_{i,j} = \sum_{k=1}^{l} \frac{1}{P(\boldsymbol{w}_k|\Theta)} \sum_{t^*} f_i^k(t-1) \cdot a_{i,j} \cdot s_i(w_k[t]) \cdot b_j^k(t) \tag{25}$$

$$E_i(x) = \sum_{k=1}^{l} \frac{1}{P(\boldsymbol{w}_k|\Theta)} \sum_{t:w_k[t]=x} f_i^k(t-1) \cdot b_i^k(t-1) \tag{26}$$

上式から得られる期待値 $O_{i,j}$ および $E_i(x)$ から新しいパラメータ $\overline{a_{i,j}}$ および $\overline{s_i(x)}$ を次式で求めることができる．

$$\overline{a_{i,j}} = \frac{O_{i,j}}{\sum_j O_{i,j}} \tag{27}$$

$$\overline{s_i(x)} = \frac{E_i(x)}{\sum_{x'} E_i(x')} \tag{28}$$

具体的な例として図 17 におけるモデルからシンボル系列 $AABC$ が観測されたとした場合を考える．ただし，モデルからシンボル系列 $AABC$ が観測される尤度を L_w とする．このとき，シンボル系列は一つであるから，式は次のようになる．

$$\begin{aligned} O_{i,j} &= \sum_{k=1}^{\mathbf{1}} \frac{1}{P(\boldsymbol{w_k}|\Theta)} \sum_t f_i(t-1) \cdot a_{i,j} \cdot s_i(w_1[t]) \cdot b_j(t) \\ &= \frac{1}{P(\boldsymbol{w}_1|\Theta)} \sum_t f_i(t-1) \cdot a_{i,j} \cdot s_i(w_1[t]) \cdot b_j(t) \\ &= \sum_t \frac{f_i(t-1) \cdot a_{i,j} \cdot s_i(w_1[t]) \cdot b_j(t)}{L_w} \end{aligned}$$

ここで遷移回数 t 回目において，状態 q_i から状態 q_j に遷移する確率を $\xi_t(i,j)$ とすれば，

$$\xi_t(i,j) = \frac{f_i(t-1) \cdot a_{i,j} \cdot s_i(w_k[t]) \cdot b_j(t)}{L_w}$$

であるので，以下の式が成り立つ．

$$O_{i,j} = \sum_t \xi_t(i,j)$$

新たな値に更新する際には，次の PROCESS1〜PROCESS4 の流れでプログラム内では求めている．

PROCESS 1　期待値ステップ $\xi_t(i,j)$ を求める

$\xi_t(i,j)$ については次の計算式によって求めることができる.

$$
\begin{aligned}
\xi_1(0,0) &= \frac{f_0(0)\cdot a_{0,0}\cdot s_0(w_1[1])\cdot b_0(1)}{L_w} = \frac{f_0(0)\cdot a_{0,0}\cdot s_0(A)\cdot b_0(1)}{L_w} \\
&= \frac{1.0\cdot 0.2\cdot 0.3\cdot 0.00072}{0.00135360} = 0.03191489\cdots \\
\xi_1(0,1) &= \frac{f_0(0)\cdot a_{0,1}\cdot s_0(w_1[1])\cdot b_1(1)}{L_w} = \frac{f_0(0)\cdot a_{0,1}\cdot s_0(A)\cdot b_1(1)}{L_w} \\
&= \frac{1.0\cdot 0.8\cdot 0.3\cdot 0.00546}{0.00135360} = 0.96808510\cdots \\
\xi_2(0,1) &= \frac{f_0(1)\cdot a_{0,1}\cdot s_0(w_1[2])\cdot b_1(2)}{L_w} = \frac{f_0(1)\cdot a_{0,1}\cdot s_0(A)\cdot b_1(2)}{L_w} \\
&= \frac{0.06\cdot 0.8\cdot 0.3\cdot 0.003}{0.00135360} = 0.03191489\cdots \\
\xi_2(1,1) &= \frac{f_1(1)\cdot a_{1,1}\cdot s_1(w_1[2])\cdot b_1(2)}{L_w} = \frac{f_1(1)\cdot a_{1,1}\cdot s_1(A)\cdot b_1(2)}{L_w} \\
&= \frac{0.24\cdot 0.5\cdot 0.7\cdot 0.003}{0.00135360} = 0.18617021\cdots \\
\xi_2(1,2) &= \frac{f_1(1)\cdot a_{1,2}\cdot s_1(w_1[2])\cdot b_2(2)}{L_w} = \frac{f_1(1)\cdot a_{1,2}\cdot s_1(A)\cdot b_2(2)}{L_w} \\
&= \frac{0.24\cdot 0.5\cdot 0.7\cdot 0.0126}{0.00135360} = 0.78191489\cdots \\
\xi_3(1,2) &= \frac{f_1(2)\cdot a_{1,2}\cdot s_1(w_1[3])\cdot b_2(3)}{L_w} = \frac{f_1(2)\cdot a_{1,2}\cdot s_1(B)\cdot b_2(3)}{L_w} \\
&= \frac{0.0984\cdot 0.5\cdot 0.1\cdot 0.06}{0.00135360} = 0.21808510\cdots \\
\xi_3(2,2) &= \frac{f_2(2)\cdot a_{2,2}\cdot s_2(w_1[3])\cdot b_2(3)}{L_w} = \frac{f_2(2)\cdot a_{2,2}\cdot s_2(B)\cdot b_2(3)}{L_w} \\
&= \frac{0.084\cdot 0.7\cdot 0.3\cdot 0.06}{0.00135360} = 0.78191489\cdots \\
\xi_4(2,3) &= \frac{f_2(3)\cdot a_{2,3}\cdot s_2(w_1[4])\cdot b_3(4)}{L_w} = \frac{f_2(3)\cdot a_{2,3}\cdot s_2(C)\cdot b_3(4)}{L_w} \\
&= \frac{0.02256\cdot 0.3\cdot 0.2\cdot 1.0}{0.00135360} = 1.0\cdots
\end{aligned}
$$

これらの式において
$$\begin{cases} \xi_1(0,0)+\xi_1(0,1)=1.0 \\ \xi_2(0,1)+\xi_2(1,1)+\xi_2(1,2)=1.0 \\ \xi_3(1,2)+\xi_3(2,2)=1.0 \\ \xi_4(2,3)=1.0 \end{cases}$$
が成り立つ.

PROCESS 2　新たな状態遷移確率 $a_{i,j}$ の計算

PROCESS 1 によって $\xi_t(i,j)$ を求めたところで、新たな状態遷移確率 $a_{i,j} = \dfrac{O_{i,j}}{\sum_j O_{i,j}}$ の値を求める.

$O_{i,j}$ の値は $O_{i,j} = \sum_t \xi_t(i,j)$, $\sum_j O_{i,j} = \sum_t \sum_j \xi_t(i,j)$ であるから, $a_{i,j}$ は次の式（29）で表すことができる.

$$a_{i,j} = \frac{\sum_t \xi_t(i,j)}{\sum_t \sum_j \xi_t(i,j)} \tag{29}$$

したがって, 各 $a_{i,j}$ は次の計算式でもとまる.

$$
\begin{aligned}
a_{0,0} &= \frac{\xi_1(0,0)}{\xi_1(0,0)+\xi_1(0,1)+\xi_2(0,1)} \\
&= \frac{0.03191489}{0.03191489+0.96808510+0.03191489} \fallingdotseq 0.03092783 \\
a_{0,1} &= \frac{\xi_1(0,1)+\xi_2(0,1)}{\xi_1(0,0)+\xi_1(0,1)+\xi_2(0,1)} \\
&= \frac{0.03191489}{0.03191489+0.96808510+0.03191489} \fallingdotseq 0.96907217 \\
a_{1,1} &= \frac{\xi_2(1,1)}{\xi_2(1,1)+\xi_2(1,2)+\xi_3(1,2)} \\
&= \frac{0.18617021}{0.18617021+0.78191489+0.21808510} \fallingdotseq 0.15695067 \\
a_{1,2} &= \frac{\xi_2(1,2)+\xi_3(1,2)}{\xi_2(1,1)+\xi_2(1,2)+\xi_3(1,2)} \\
&= \frac{0.78191489+0.21808510}{0.18617021+0.78191489+0.21808510} \fallingdotseq 0.84304933 \\
a_{2,2} &= \frac{\xi_3(2,2)}{\xi_3(2,2)+\xi_4(2,3)} \\
&= \frac{0.78191489}{0.78191489+1.0} \fallingdotseq 0.43880597 \\
a_{2,3} &= \frac{\xi_4(2,3)}{\xi_3(2,2)+\xi_4(2,3)} \\
&= \frac{1.0}{0.78191489+1.0} \fallingdotseq 0.56119403
\end{aligned}
$$

観測されるシンボル系列が複数ある場合は，それぞれのシンボル系列に対する期待値ステップから各 $a_{i,j}$ を求め，その平均値 $\overline{a_{i,j}}$ を更新後の状態遷移確率として用いる．

PROCESS 3　新たなシンボル出力確率 $s_i(x)$ の計算

ここでは，更新後に設定するシンボル出力確率 $s_i(x)$ を求める．出力されるシンボル系列が一つの場合を考えているので，尤度 L_w を用いて $\sum_{k=1}^{l} \frac{1}{P(\boldsymbol{w}_k|\Theta)}$ は $\sum_{k=1}^{l} \frac{1}{P(\boldsymbol{w}_k|\Theta)} = \sum_{k=1}^{1} \frac{1}{P(\boldsymbol{w}_1|\Theta)} = \frac{1}{L_w}$ と表すことができる．

したがって，式（26）は

$$E_i(x) = \frac{1}{L_w} \sum_{t:w_k[t]=x} f_i(t-1) \cdot b_i(t-1) \tag{30}$$

と書き換えられる．さらに上式（30）に式（22）を代入して，

$$\begin{aligned} E_i(x) &= \frac{1}{L_w} \sum_{t:w_k[t]=x} f_i(t-1) \cdot b_i(t-1) \\ &= \frac{1}{L_w} \sum_{t:w_k[t]=x} f_i(t-1) \cdot \sum_{q_j \in Q} b_j(t) a_{i,j} \cdot s_i(w[t]) \\ &= \frac{\sum_{t:w_k[t]=x} f_i(t-1) \cdot \sum_{q_j \in Q} b_j(t) a_{i,j} \cdot s_i(w[t])}{L_w} \\ &= \sum_{t:w_k[t]=x} \sum_{q_j \in Q} \frac{f_i(t-1) \cdot b_j(t) a_{i,j} \cdot s_i(w[t])}{L_w} \end{aligned} \tag{31}$$

ここで，

$$\xi_t(i,j) = \frac{f_i(t-1) \cdot a_{i,j} \cdot s_i(w_k[t]) \cdot b_j(t)}{L_w}$$

を式（31）に代入すると，

$$E_i(x) = \sum_{t:w_k[t]=x} \sum_{q_j \in Q} \frac{f_i(t-1) \cdot b_j(t) a_{i,j} \cdot s_i(w[t])}{L_w} = \sum_{t:w_k[t]=x} \sum_{q_j \in Q} \xi_t(i,j) \tag{32}$$

よって，式（28）は次のように書き換えることができる．

$$
\begin{aligned}
s_i(x) &= \frac{E_i(x)}{\sum_{x'} E_i(x')} \\
&= \frac{\sum_{t:w_k[t]=x} \sum_{q_j \in Q} \xi_t(i,j)}{\sum_{x'} \sum_{t:w_k[t]=x} \sum_{q_j \in Q} \xi_t(i,j)} \\
&= \frac{\sum_{t:w_k[t]=x} \sum_{q_j \in Q} \xi_t(i,j)}{\sum_{t} \sum_{q_j \in Q} \xi_t(i,j)}
\end{aligned}
\tag{33}
$$

式（33）より，新たに設定する各シンボル出力確率 $s_i(x)$ 次の計算式でもとまる．

$$
\begin{aligned}
s_0(A) &= \frac{\xi_1(0,0)+\xi_1(0,1)+\xi_2(0,1)}{\xi_1(0,0)+\xi_1(0,1)+\xi_2(0,1)} = 1.0 \\
s_0(B) &= \frac{0}{\xi_1(0,0)+\xi_1(0,1)+\xi_2(0,1)} = 0.0 \\
s_0(C) &= \frac{0}{\xi_1(0,0)+\xi_1(0,1)+\xi_2(0,1)} = 0.0 \\
s_1(A) &= \frac{\xi_2(1,1)+\xi_2(1,2)}{\xi_2(1,1)+\xi_2(1,2)+\xi_3(1,2)} \\
&= \frac{0.18617021+0.78191489}{0.18617021+0.78191489+0.21808510} \fallingdotseq 0.81614350 \\
s_1(B) &= \frac{\xi_3(1,2)}{\xi_2(1,1)+\xi_2(1,2)+\xi_3(1,2)} \\
&= \frac{0.21808510}{0.18617021+0.78191489+0.21808510} \fallingdotseq 0.18385650 \\
s_1(C) &= \frac{0}{\xi_2(1,1)+\xi_2(1,2)+\xi_3(1,2)} = 0.0 \\
s_2(A) &= \frac{0}{\xi_3(2,2)+\xi_4(2,3)} = 0.0 \\
s_2(B) &= \frac{\xi_3(2,2)}{\xi_3(2,2)+\xi_4(2,3)} \\
&= \frac{0.78191489}{0.78191489+1.0} \fallingdotseq 0.43880597 \\
s_2(C) &= \frac{\xi_4(2,3)}{\xi_3(2,2)+\xi_4(2,3)} \\
&= \frac{1.0}{0.78191489+1.0} \fallingdotseq 0.56119403
\end{aligned}
$$

3.3 HMM における経路の求め方

本研究では，Forward algorithm，Backward algorithm，Baum-Welch algorithm において，各パラメータの値を求める際に経路をもとに計算することで高速に計算できるようにした．具体的な求め方については図 17 のモデルにおいてシンボル系列 $AABC$ を出力できる経路（表 1）を求めることを例にして考える．

このとき，遷移 t 回目において各状態から各状態へ遷移できるかを記憶するための行列集合 $\mathbb{G} = \{G_1, G_2, ...G_t, ...G_l\}$ を用意する．状態数が k 個の時の G_t はそれぞれの遷移回数 t に対して $k \times k$ 行列を用意したもので，

$$G_t = \begin{bmatrix} \Gamma_t(0,0) & \Gamma_t(0,1) & \cdots & \Gamma_t(0,k-1) \\ \Gamma_t(1,0) & \Gamma_t(1,1) & \cdots & \Gamma_t(1,k-1) \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma_t(k-1,0) & \Gamma_t(k-1,1) & \cdots & \Gamma_t(k-1,k-1) \end{bmatrix}$$

を意味する．

この例では，状態数は 4 であるから $k = 4$ となる行列 G_t を考える．この行列における各 $\Gamma_t(i,j)$ の値は，遷移回数 t 回目において，状態 i からシンボル $s_i(w[t])$ を出力可能な場所かつ状態 i から状態 j に遷移可能な場所は 0 をセットし，状態 i からシンボル $s_i(w[t])$ を出力可能または状態 i から状態 j に遷移不可能な場所は -1 にセットする．ここで設定する 0 や -1 の値には意味はなく，便宜上用いるものとする．

具体的には一回目の遷移 $t = 1$ の場合，出力されるシンボルは A であるから，状態 i からシンボル A が出力されない場合は，$\Gamma_1(i, \forall j) = -1$ である．また, 状態 i か

らシンボル A が出力される場合であっても状態 i から状態 j に遷移できない場合は $\Gamma_1(i,j) = -1$ となる．

ここで重要なのは，状態 i から状態 j に遷移できない場合というのはある遷移回数における，という条件のもとではない．これはモデルが内包するパラメータにのみ依存する設定であり，遷移回数は関係ない．ただし，初期状態からの遷移 $t = 1$ の場合と最終状態への遷移 $t = 4$ の場合については，それぞれ，どの状態から始まり，終わるかを考慮しなくてはならないので次の条件が必要とされる．

■ 初期状態からの遷移　：　$t = 1$ のとき

$t = 1$ のときは，初期状態 0 からの遷移のみ考えればよく，状態 $i = 0$ 以外の状態からの遷移はおきないため，行列 G_1 における $i \neq 0$ の行はすべて -1 に設定する．

■ 最終状態への遷移　：　$t = 4$ のとき

$t = 4$（ここでは例として 4 が最終状態への遷移）のときは，状態 i からの状態 $3(\because \mathrm{k} - 1 = 4 - 1 = 3)$ への遷移のみ考えればよく，$a_{i,3} \neq 0$ を満たす i 行以外の行はすべて -1 に設定する．

したがって, それぞれの値は以下のようになる．ただし初期状態では状態 0 から遷移するので状態 $i = 0$ 以外のすべての行は -1 である．

$$
\begin{aligned}
\Gamma_1(0,0) &= 0 \quad (\because \quad s_0(A) \cdot a_{0,0} \neq 0) \\
\Gamma_1(0,1) &= 0 \quad (\because \quad s_0(A) \cdot a_{0,1} \neq 0) \\
\Gamma_1(0,2) &= -1 \quad (\because \quad s_0(A) \cdot a_{0,1} = 0) \\
\Gamma_1(0,3) &= -1 \quad (\because \quad s_0(A) \cdot a_{0,1} = 0)
\end{aligned}
$$

$$
\begin{aligned}
\Gamma_1(1,0) &= -1 \quad \Gamma_1(2,0) = -1 \quad \Gamma_1(3,0) = -1 \\
\Gamma_1(1,1) &= -1 \quad \Gamma_1(2,1) = -1 \quad \Gamma_1(3,1) = -1 \\
\Gamma_1(1,2) &= -1 \quad \Gamma_1(2,2) = -1 \quad \Gamma_1(3,2) = -1 \\
\Gamma_1(1,3) &= -1 \quad \Gamma_1(2,3) = -1 \quad \Gamma_1(3,3) = -1
\end{aligned}
$$

このようにして各遷移t回目における行列G_tの値を設定した結果を図19に示す.

G_1	0	1	2	3
0	0	0	-1	-1
1	-1	-1	-1	-1
2	-1	-1	-1	-1
3	-1	-1	-1	-1

G_2	0	1	2	3
0	0	0	-1	-1
1	-1	0	0	-1
2	-1	-1	0	0
3	-1	-1	-1	-1

G_3	0	1	2	3
0	0	0	-1	-1
1	-1	0	0	-1
2	-1	-1	0	0
3	-1	-1	-1	-1

G_4	0	1	2	3
0	-1	-1	-1	-1
1	-1	-1	-1	-1
2	-1	-1	-1	0
3	-1	-1	-1	-1

図 19: 行列 G_t の設定例

こうして値を設定した行列 G_t について次のステップに示す操作を行い値を書き換える．

■ STEP　1　：前向きによる更新

遷移回数 1 回目に関する行列 G_1 の j 列がすべて -1 であれば，遷移回数 2 回目に関する行列 G_2 の j 行をすべて -1 に更新する．図 20 に示すように行列 G_1 の各列のすべてについて，それに対応する行列 G_2 の各行を更新した後，同じように行列 G_3, G_4 に関しても更新を行う．

この操作は $t = 1$ 回目から始め，行列 G_t における j 列のすべてが -1 の場合は，行列 G_{t+1} における j 行全てを -1 に設定するものである．$t = l - 1$ になるまでこの操作をおこなうことで，行列集合 $\mathbb{G}$ を更新する．

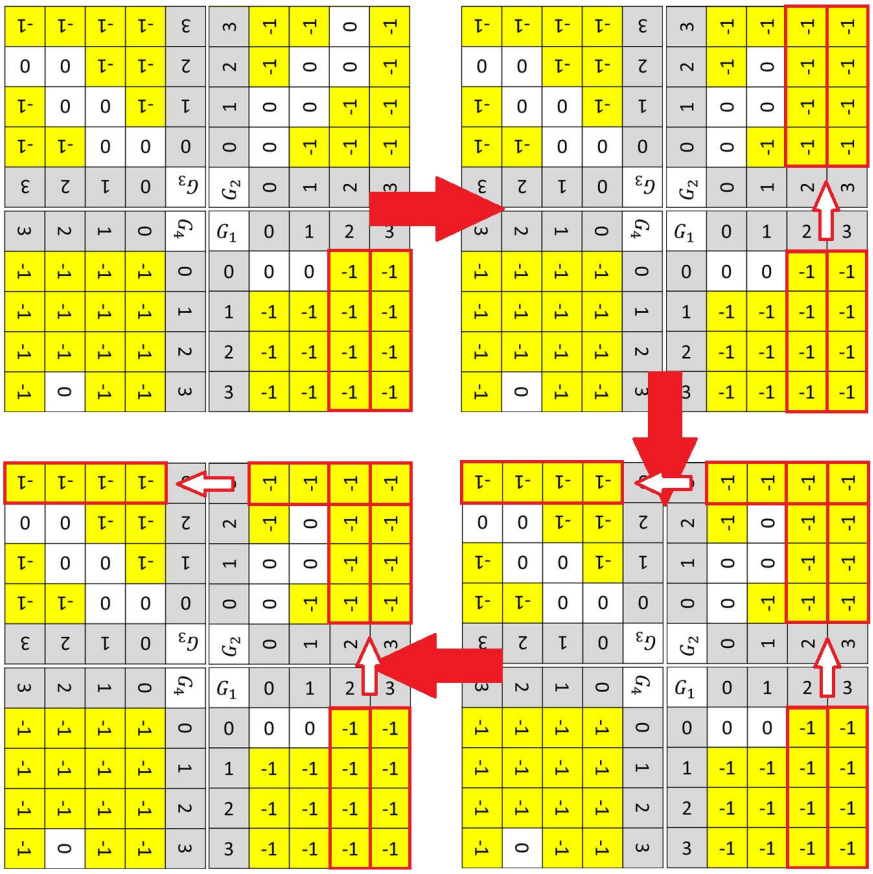


図 20: 前向きによる行列集合 $\mathbb{G}$ の更新

■ STEP　2　：後ろ向きによる更新

前向きによる更新後，今後は逆向きに更新を行っていく．遷移回数が最終の $t = l$ から始め行列 G_t における i 行のすべてが -1 の場合は，行列 G_{t-1} における i 列全てを -1 に設定する．この操作を $t = 2$ になるまで繰り返し，行列集合 $\mathbb{G}$ を再び更新する（図 21）．

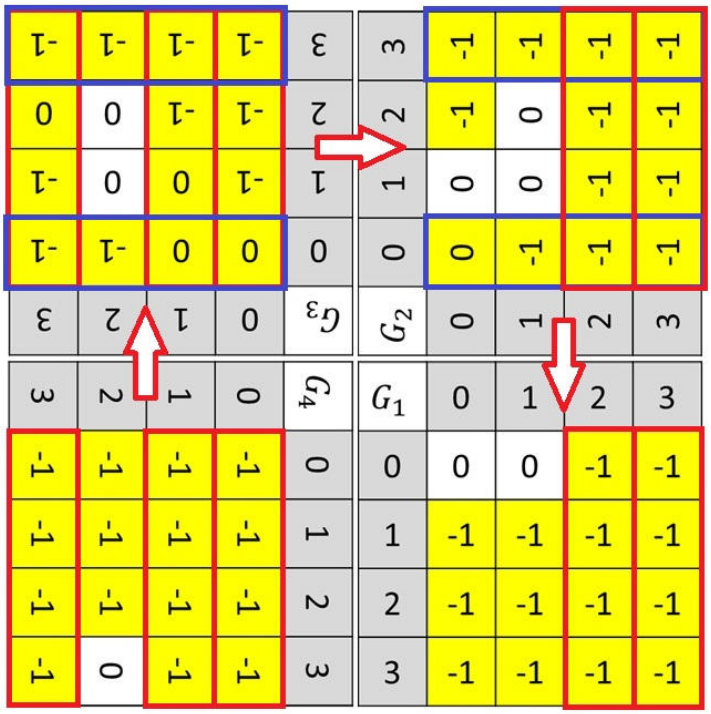


図 21: 後ろ向きによる行列集合 $\mathbb{G}$ の更新

こうしてSTEP 1～STEP 2までの前向きと後ろ向きによる更新を行うことで，遷移回数 t 回目において状態 i から状態 j に遷移できるかできないかは行列 G_t における $\Gamma_t(i,j)$ が 0 であるか -1 であるかによって判断できる．0 であれば遷移可能で -1 であれば遷移不可能となる．このことは，シンボル系列 $AABC$ を出力できるすべての経路が明らかになったことであり，実際に図 22 に示す経路が表 1 に示す経路と等しくなる．

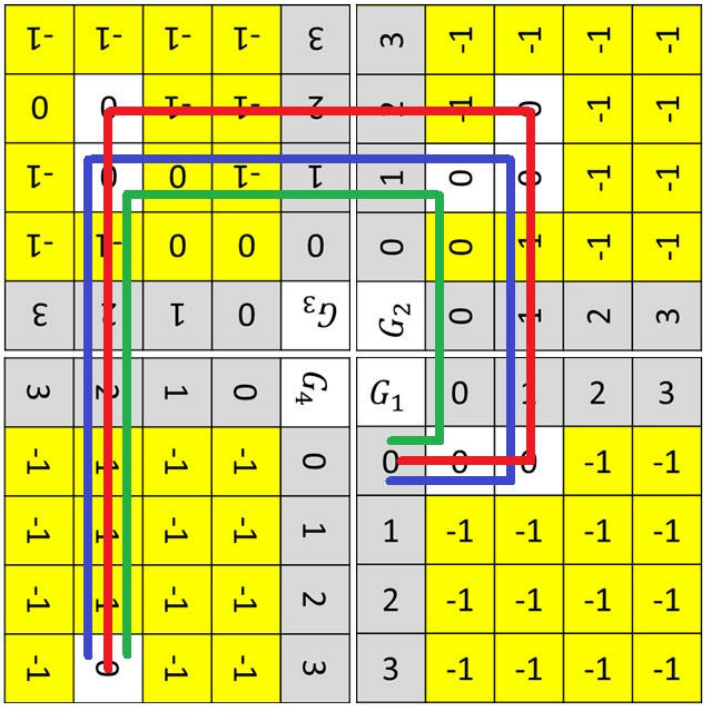


図 22: 行列集合 $\mathbb{G}$ から明らかになる経路

4　隠れマルコフ球面自己組織化マップ (HMM-S-SOM)

隠れマルコフ球面自己組織化マップとは,S-SOM のノードに隠れマルコフモデルを用いたものであり（図 23）, 入力データには, ベクトルではなく文字列集合を用いる. ノード上に内包された隠れマルコフモデルは, すべて同じ構造を持つ. この自己組織化マップは, データを直接分類するのではなくデータの背景にある確率モデルにもとづいて, 確率モデルを球面上にクラスタリングし, その分類結果を視覚的にも容易に提示することのできる学習モデルである. S-SOM と HMM-S-SOM の違いについては表 2 に示す. ノードや入力データの形式の違い以外に, 勝者ノードの決定方法や, ノードの更新の方法が異なる. その詳細については, 次の HMM-S-SOM の学習アルゴリズムの説明に記す.

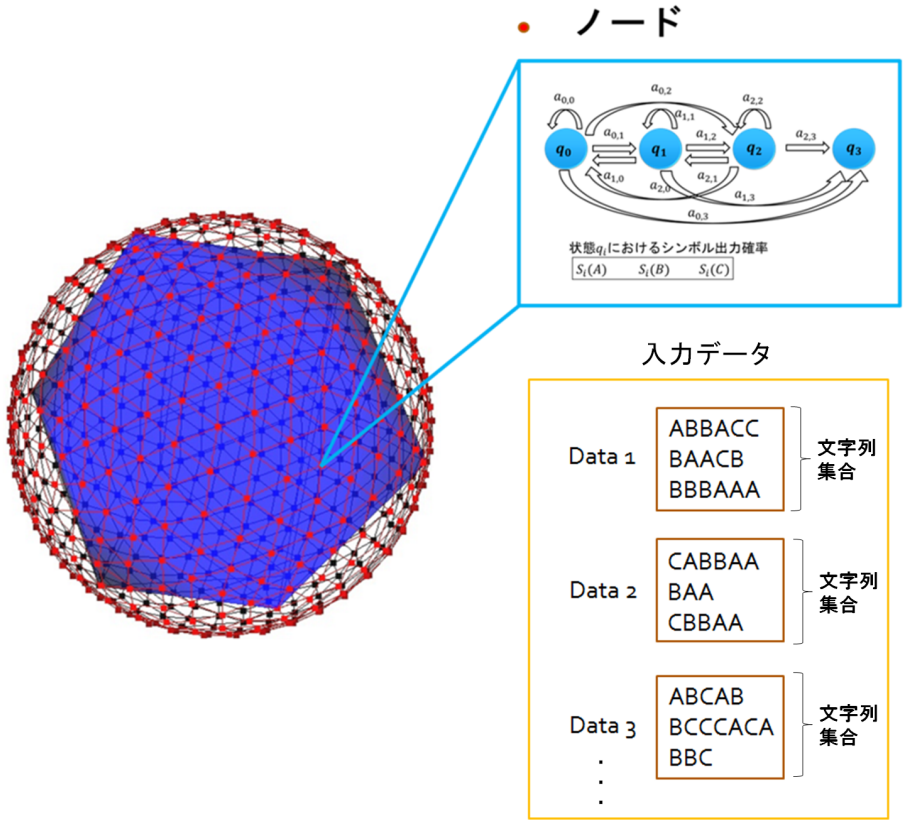


図 23: 隠れマルコフ球面自己組織化マップ

表 2: S-SOM と HMM-S-SOM の比較

	S-SOM	HMM-SOM
SOMのノード	ベクトル	HMM
入力データ	ベクトル	文字列集合
勝者決定方法	ユークリッド距離が最小	尤度が最大
勝者ノードの更新	重みの調節を用いたベクトルの更新	入力データを用いたBaum-Welch algorithmによるHMMパラメータの更新
近傍ノードの更新	重みの調節を用いたベクトルの更新	重みの調節を用いたHMMパラメータの更新

4.1 HMM-S-SOM の学習アルゴリズム

STEP 0: ノードの初期化

球面上における各ノードが内包する HMM のパラメータ Θ_k を乱数を用いて初期化する

STEP 1: 入力データの読み込み

入力データである文字列集合を一つ読み込む

STEP 2: 勝者ノードの決定

文字列集合に対し各ノードのパラメータを Baum-Welch algorithm を用いて十分に更新したと仮定した際にその文字列集合を出力する確率（尤度）の極値が STEP 3 の更新によって最も高くなるノードをそのデータに対する勝者ノードとする．十分に更新という意味は, 更新しても尤度に変化が見られなくなり, 値が極大値に近づくまで更新するという意味である

STEP 3: ノードの更新　勝者ノードにおける HMM モデルを Baum-Welch algorithm を用いて更新し, 近傍のノードにおける HMM のパラメータを SOM のアルゴリズムを用いて更新する

SOM のアルゴリズムと同様, STEP1〜STEP3 を全てのデータについて行い, これを学習回数分くり返す．STEP2 において, 更新後の尤度を用いるのは, 更新前に尤度が最も高いモデルが更新後に最も尤度の高いモデルになるとは限らないからであり, 実際に, 図 24 に示す 2 つのモデルの尤度の変化のように更新前に尤度が高かったモデルが, 更新後に他のモデルより尤度が低くなる場合がある．また, 更新後の尤度を知るために,Baum-Welch algorithm によるモデルの更新をノード上の HMM に対して行うと, モデルのパラメータが変わってしまうため, プログラム上では, 他のメモリ上にモデルのパラメータを移して,Baum-Welch algorithm によるモデルの更新後の値を計算する．実際のマップ上における HMM モデルのパラメータの更新は,STEP3 で行う．

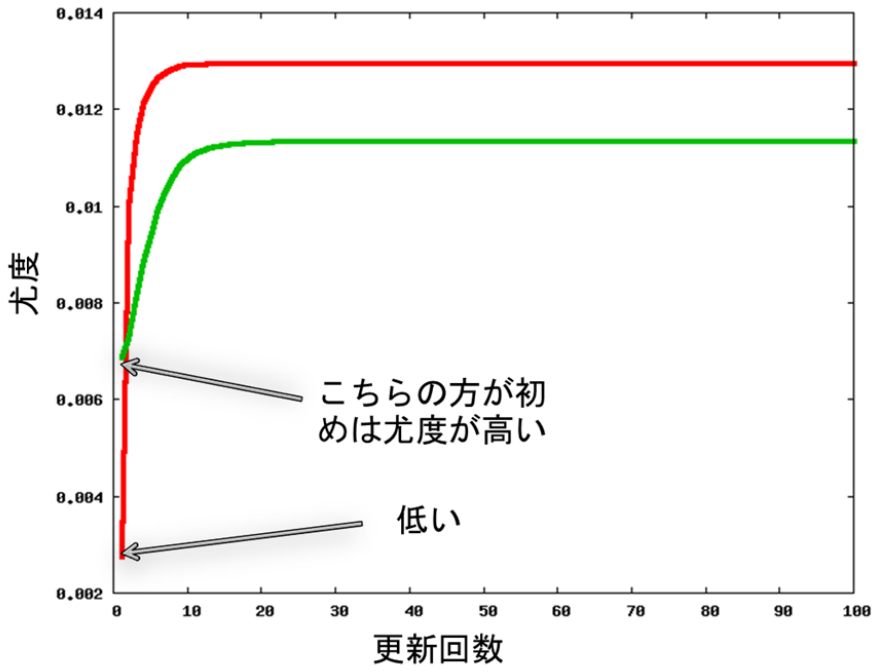


図 24: 尤度の更新前と更新後の変化

4.2 人工データを用いた実験

分類の対象である隠れマルコフモデルは, 状態遷移が前の状態に戻らない left to right 型と, 前の状態にも戻れる Ergodic 型の混合したサンプルデータを用いて分類の評価を行った．この際、マップ球面の表示において, ノード間のパラメータの差をより視覚的に表現できる U-matrix と呼ばれる手法を用いた．この手法を用いて本研究ではノード間のパラメータ同士の差を, ノードとノードの間にある U-matrix に保存しておきその U-matrix に基づいてノードの位置を高くすることでノード間の差を山で表現している．また, この山の高さによって色を変え, より分類結果が容易に判断できるようにした．ノード間におけるモデルの差の計算方法は, 人工データについては状態数４, シンボルは A,B,C の 3 つを内包する HMM を用いた．このモデルのパラメータが異なる HMM モデル $\boldsymbol{M_1}, M_3, ..., M_{10}$ の１０個用意し, それぞれのモデルから人工的に初期状態から, 最終状態までこれらのパラメータに基づいて確率的に遷移させながらシンボル列を出力させ, これをそれぞれのモデルから５０個用意し, それぞれのモデルから観測されるこれらのシンボル列集合を入力データとして用いた（図 25）.

各モデルの状態遷移確率のパラメータは表 3〜12 に示す値に設定し, 各モデルのシンボル出力確率のパラメータは表 13〜22 に示す値に設定した.

表 3: M_1 における状態遷移確率

i \ j	0	1	2	3
0	0.2	0.8	0.0	0.0
1	0.0	0.2	0.8	0.0
2	0.0	0.0	0.2	0.8

表 4: M_2 における状態遷移確率

i \ j	0	1	2	3
0	0.8	0.2	0.0	0.0
1	0.0	0.2	0.8	0.0
2	0.0	0.0	0.2	0.8

表 5: M_3 における状態遷移確率

i \ j	0	1	2	3
0	0.2	0.8	0.0	0.0
1	0.0	0.2	0.8	0.0
2	0.0	0.0	0.8	0.2

表 6: M_4 における状態遷移確率

i \ j	0	1	2	3
0	0.8	0.2	0.0	0.0
1	0.0	0.8	0.2	0.0
2	0.0	0.0	0.8	0.2

表 7: M_5 における状態遷移確率

i \ j	0	1	2	3
0	0.2	0.8	0.0	0.0
1	0.0	0.8	0.2	0.0
2	0.0	0.0	0.8	0.2

表 8: M_6 における状態遷移確率

i \ j	0	1	2	3
0	0.8	0.2	0.0	0.0
1	0.0	0.8	0.2	0.0
2	0.0	0.0	0.2	0.8

表 9: M_7 における状態遷移確率

i \ j	0	1	2	3
0	0.5	0.3	0.0	0.2
1	0.2	0.5	0.3	0.0
2	0.3	0.1	0.2	0.4

表 10: M_8 における状態遷移確率

i \ j	0	1	2	3
0	0.5	0.3	0.0	0.2
1	0.3	0.1	0.2	0.4
2	0.2	0.5	0.3	0.0

表 11: M_9 における状態遷移確率

i \ j	0	1	2	3
0	0.3	0.1	0.2	0.4
1	0.5	0.3	0.0	0.2
2	0.2	0.5	0.3	0.0

表 12: M_{10} における状態遷移確率

i \ j	0	1	2	3
0	0.2	0.5	0.3	0.0
1	0.3	0.1	0.2	0.4
2	0.5	0.3	0.0	0.2

表 13: M_1 におけるシンボル出力確率

i \ x	A	B	C
0	0.3	0.2	0.5
1	0.3	0.5	0.2
2	0.2	0.3	0.5

表 14: M_2 におけるシンボル出力確率

i \ x	A	B	C
0	0.3	0.5	0.2
1	0.3	0.2	0.5
2	0.4	0.2	0.2

表 15: M_3 におけるシンボル出力確率

i \ x	A	B	C
0	0.2	0.6	0.2
1	0.3	0.5	0.2
2	0.3	0.2	0.5

表 16: M_4 におけるシンボル出力確率

i \ x	A	B	C
0	0.3	0.2	0.5
1	0.3	0.1	0.6
2	0.3	0.5	0.2

表 17: M_5 におけるシンボル出力確率

i \ x	A	B	C
0	0.3	0.5	0.2
1	0.2	0.7	0.1
2	0.3	0.2	0.5

表 18: M_6 におけるシンボル出力確率

i \ x	A	B	C
0	0.2	0.6	0.2
1	0.3	0.2	0.5
2	0.3	0.5	0.2

表 19: M_7 におけるシンボル出力確率

i \ x	A	B	C
0	0.5	0.3	0.2
1	0.3	0.2	0.5
2	0.3	0.5	0.2

表 20: M_8 におけるシンボル出力確率

i \ x	A	B	C
0	0.3	0.5	0.2
1	0.5	0.3	0.2
2	0.3	0.2	0.5

表 21: M_9 におけるシンボル出力確率

i \ x	A	B	C
0	0.3	0.2	0.5
1	0.5	0.3	0.2
2	0.3	0.5	0.2

表 22: M_{10} におけるシンボル出力確率

i \ x	A	B	C
0	0.5	0.3	0.2
1	0.3	0.5	0.2
2	0.3	0.2	0.5

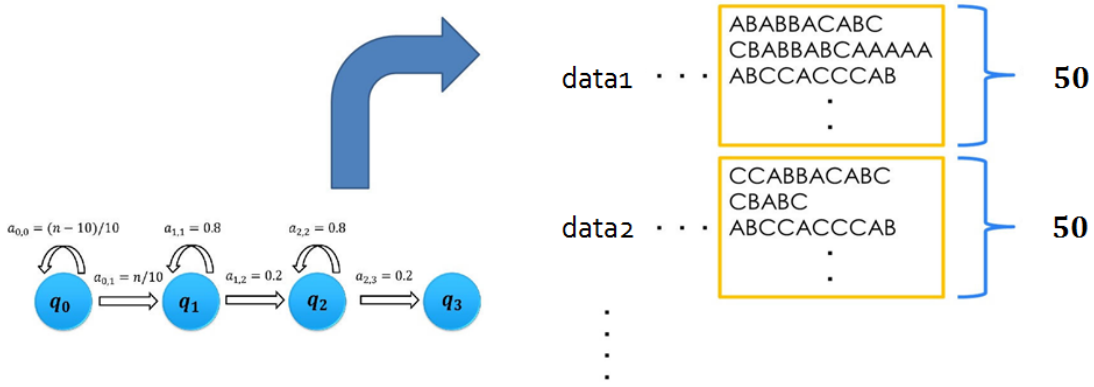


図 25: 実験に用いた人工モデル

4.2.1 HMM-S-SOM における人工データでの実験結果

このように各モデルから作成したデータを HMM-S-SOM に学習させた結果を下図 26 に示す.

この図 26 におけるグラフの縦軸はモデル M_1 に対応する文字列集合 $Data1$ を M_1〜M_{10} に対して尤度を求めたものである．このグラフからわかることは，元のモデルにおいて M_1 に対し M_{10} が類似度の高いモデルといえ，$M_3, M_8, M_9, M_7, M_2, M_5, M_6, M_4$ の順でモデル M_1 との類似度は低くなっている．また,U-matrix の計算は HMM のパラメータの差を用いて算出しており, この U-matrix の表示は HMM におけるモデルの構造の類似度を示している．勝者ノード間の境界は山となってところどこ

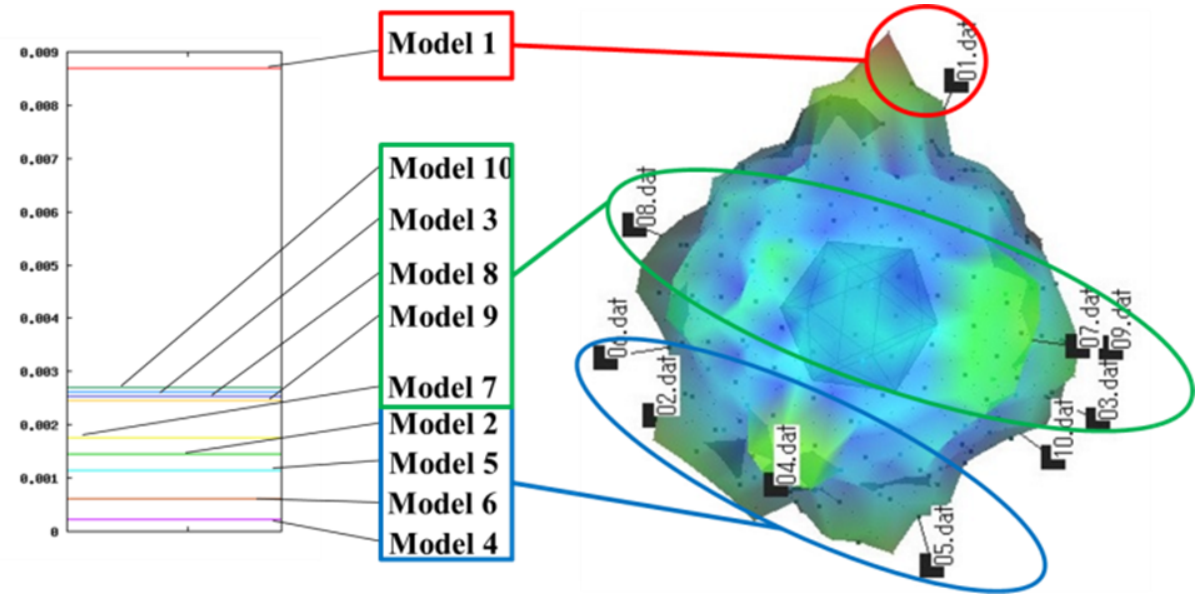


図 26: モデルの構造にもとづいた表示結果

ろ現れているが, 山に連続性はなく HMM のパラメータでの学習は大まかにしかできていないことが分かる.

また, モデル間の類似度を表す尤度の表示においては, 図 27 のようになった.

マップ上に不規則に山ができていることから、その場所では学習過程においてあるモデルの強い学習がおこり, あるモデルの局所解を学習しているノードがあることを示すもので, マッピングが適切に行えていないことを示している. 実際に, 同じ実験を数回行ったが,model 1 と model 4 が同じ位置にマッピングされるなどマップの再現性に乏しい結果となった.

以上の結果から,HMM のパラメータおよび尤度に関する U-matrix の表示のどちらに関してもノード間の差が明確ではなく, モデルの分類を視覚的に理解することは困難である.

これらの問題を解決するためにベクトル統合型隠れマルコフ球面自己組織化マップを開発した. 次章に, 詳細を示す.

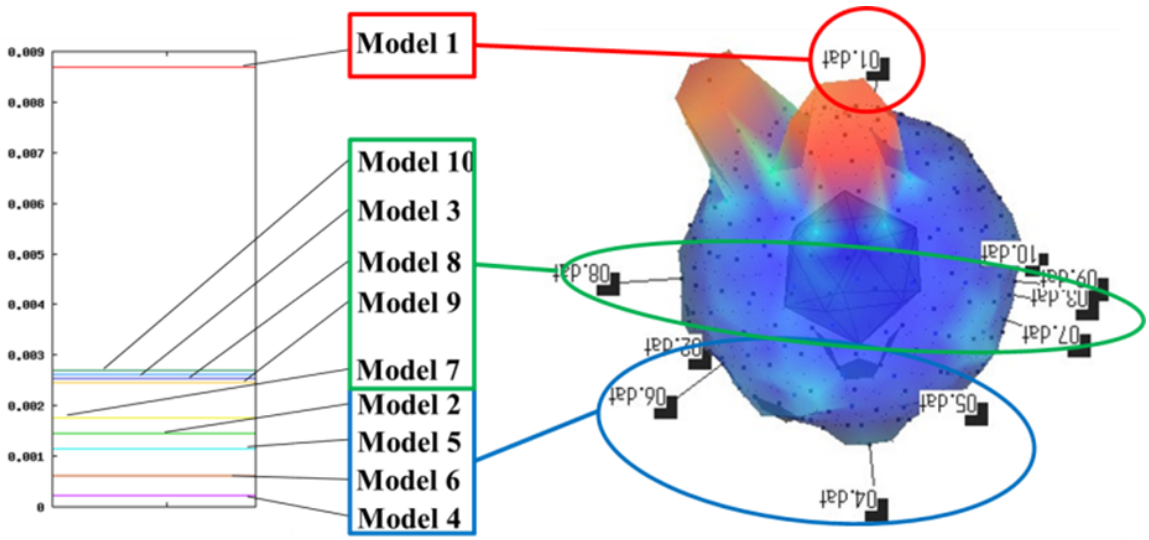


図 27: 尤度にもとづいた表示結果

5　ベクトル統合型隠れマルコフ球面自己組織化マップ (F-HMM-S-SOM)

前章で紹介した隠れマルコフ球面自己組織化マップの学習アルゴリズムにおいて, 勝者ノードを更新する際に Baum-Welch algorithm を用いたが, この手法によるノードの更新では,SOM による学習率を無視する形で過学習が起こることが分かった．この問題を解決するためには,HMM モデルの状態遷移とシンボル出力の傾向を考慮しつつ, 更新の際の学習率も考慮できるような学習の手法が必要である．そこで, 従来の SOM の学習に用いられるベクトルの形式で HMM モデルの状態遷移とシンボル出力の傾向を考慮できる頻度ベクトルを提案し, このベクトルを隠れマルコフ球面自己組織化マップのノードに統合したベクトル統合型隠れマルコフ球面自己組織化マップ (F-HMM-S-SOM) を開発した (図 28).

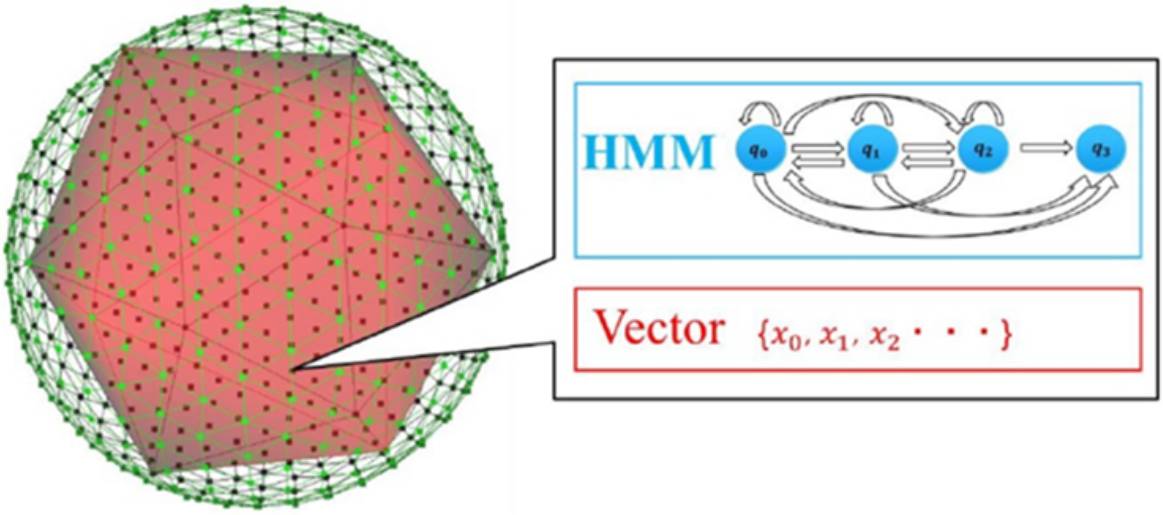


図 28: F-HMM-S-SOM におけるノード

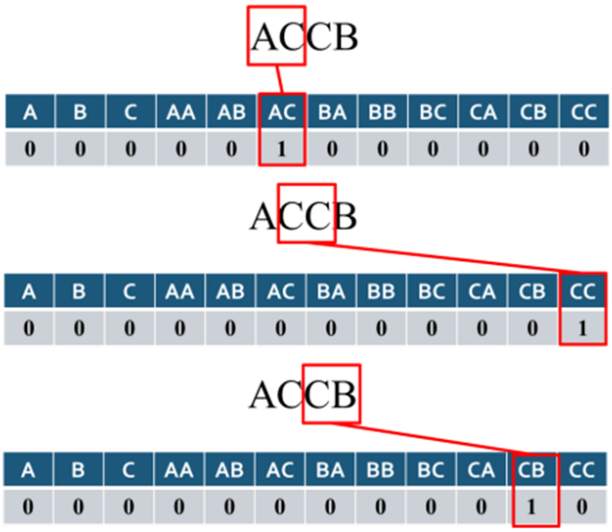
なお, マップ上の全てのノードは同じ構造をもっており, 各ノード上の HMM が内包する状態数とシンボル数は同じで, 頻度ベクトルの次元についても各ノードは同じ次元をもつ．この自己組織化マップはシンボル列集合を入力データとし, 各入力データにおけるシンボル系列集合を出力するような確率モデルをそれぞれ推測

し, これらのモデルをマップ上に学習していくことでモデルの分類を行っていく. F-S-HMM-SOM の学習アルゴリズムにおいて扱われるこの頻度ベクトルは, 入力データのシンボル列集合におけるシンボルの出力頻度をベクトル化したデータである. 頻度ベクトルは次のように構成する.

シンボル出力の種類を A,B,C の三種類とした場合, 状態遷移を行うごとにシンボルを出力する隠れマルコフモデルから n 回目の遷移で出力されるシンボルと, 次の遷移 $n+1$ 回目で出力されるシンボルの組み合わせは,A,B,C の二つの組み合わせを考えると 9 通りできる. また, 一回の遷移で遷移を終えるような場合は, 一文字しか出力しないので, 一文字を出力する A,B,C の三通りを含めて, 状態遷移前と状態遷移後に出力する文字組み合わせは１２通りの可能性がある. 入力ベクトルにおける文字列集合において最大のシンボル列の長さを l とおくと, 最大でも $l-1$ 回目の遷移で出力されるシンボルと、次の遷移 l 回目で出力されるシンボルの組み合わせを考えることになるため, このシンボル列を頻度ベクトルに変換した際のベクトルの最大の次元数は $12*(l-1)$ となる. したがって文字の種類を m, 最大のシンボル列の長さを l_{max} とおくと, 入力頻度ベクトルの次元数は $(m^2+m)(l_{max}-1)$ になる. 入力データの文字列集合を ACCB, BCC とした場合具体的に入力データから入力頻度ベクトルへの変換は次のようになる.

まず, 2 つの文字列の長さをもったウィンドウを用意し, それぞれの文字列についてその文字列の先頭からウィンドウを一文字ずつずらしながら, ウィンドウ内の 2 文字の組み合わせが１２通りのうちどれにあてはまるかを調べる. たとえば,ACCB の文字列であれば, 初めの 2 文字の組み合わせは AC であり, 図 29 のように AC の要素を 1 にセットし, 他の要素は 0 にセットすることで, 初めのウィンドウ内の文字列は”000001000000”のベクトルに変換される. 同じようにして, その次のウィンドウ内の文字列 CC はベクトル”000000000001”に変換され, さらにその次のウィンド内の文字列 CB はベクトル”000000000010”に変換される. このようにして ACCB は

ベクトル”000001000000000000000001000000000010” として定義される．このような方法で,BCC の文字列はベクトル”000000001000000000000001”に変換される．



ACCB→000001000000000000000001000000000010 (36 elements)
BCC →000000001000000000000001 (24 elements)

図 29: シンボルのベクトル変換の手法

次に, このようにしてそれぞれの文字列から求めた、すべてのベクトルについての和をもとめ, それぞれの要素を変換されたベクトルの個数で割ることで, 入力データの文字列集合に対する頻度ベクトルをそれぞれの要素の平均値で定義する図 30．ただし, ベクトルの長さは最長のもに合わせ, 残りの成分を 0(図 30:青色の部分) にする．

この学習の手法（学習アルゴリズム）については次に詳細を記す．

■ Θ_k : HMM－S-SOM での各ノードにおける HMM のパラメータで $\Theta_k = \{A_k, S_k\}$ であるとする．ここで,

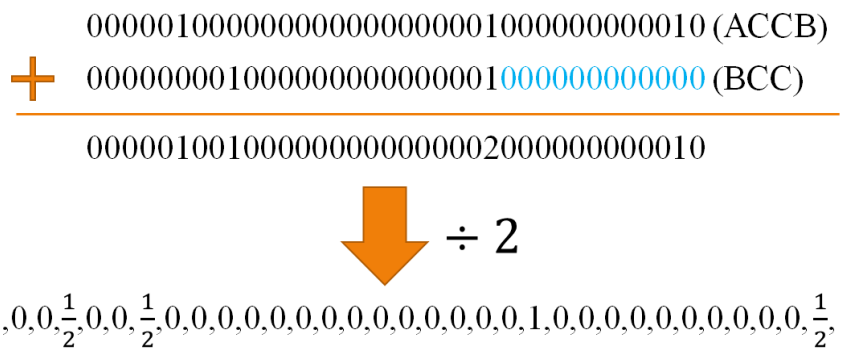


図 30: 頻度ベクトル

$$A_k = \begin{bmatrix} a_{01}^k & a_{02}^k & \cdots & a_{0j}^k \\ a_{11}^k & a_{12}^k & \cdots & a_{1j}^k \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1}^k & a_{i2}^k & \cdots & a_{ij}^k \end{bmatrix} \qquad S_k = \begin{bmatrix} s_{01}^k & s_{02}^k & \cdots & s_{0h}^k \\ s_{11}^k & a_{12}^k & \cdots & s_{1h}^k \\ \vdots & \vdots & \ddots & \vdots \\ s_{i1}^k & s_{i2}^k & \cdots & s_{ih}^k \end{bmatrix}$$

$a_{i,j}$ は状態 i から状態 j に遷移する確率．$s_{i,h}$ は状態 i においてシンボル x_h を出力するシンボル出力確率

■ X : 学習させるシンボル列集合

$X = \{x_1, x_2, x_3, ..., x_h\}$

$V = \{v_1, v_2, v_3, ..., v_h\}$

STEP 1: ノードの初期化　球面上における各ノードが内包する HMM のパラメータ Θ_k とベクトルをランダムな値で初期化しておく．

STEP 2: 類似度の計算　各入力データ Wn に対して、Baum-Welch algorithm を用いて球面上ノード N_i における HMM のパラメータを更新した際の尤度 L_i と, 各ノード N_i におけるベクトルと入力ベクトルとのユークリッド距離 E_i を計算する．

STEP 3: 勝者ノードの決定　各入力データに対して,

$(1 - \frac{E_i}{E_{max}}) * L_i$　が最小となるノードを勝者ノードとする．ここで,E_{max} は入力ベクトルと各ノードのベクトルとのユークリッド距離 E_i のうち最大の値を表す.

STEP 4: 勝者ノードの更新 勝者ノードのパラメータを Baum-Welch algorithm を用いて十分に更新する．また, 勝者ノードが内包するベクトルを入力データにおけるベクトルに近づける.

STEP 5: 近傍ノードの決定 近傍ノードのパラメータをを勝者ノードのパラメータに近づける.

ベクトルについては,SOM の学習パラメータに従って更新をおこなう．隠れマルコフモデルのパラメータについては次のように更新を行う．勝者ノード $N^n_{k^*}$ とその近傍のノードの行列パラメータ Θ_k を以下の式に従って更新する．ただし，STEP 4 の実行後における勝者ノード $N^n_{k^*}$ が内包する HMM の行列パラメータを $\Theta^n_{k^*} = \{A^n_{k^*}, S^n_{k^*}\}$ とする.

$$\begin{cases} \Delta A = \beta \cdot h(P_{nx^*}, P_{nx})(A^n_{k^*} - A_k) \\ A^{new}_k = A_k + \Delta A \end{cases}$$

$$\begin{cases} \Delta S = \beta \cdot h(P_{nx^*}, P_{nx})(S^n_{k^*} - S_k) \\ S^{new}_k = S_k + \Delta S \end{cases}$$

Θ^{new}_k を Θ_k 更新後のパラメータとすれば $\Theta^{new}_k = \{A^{new}_k, S^{new}_k\}$ となる.

ただし,

$$h(P_{nx^*}, P_{nx}) = exp(\frac{-|P_{nx^*} - P_{nx}|}{\theta^2})$$

P_{jx^*} は勝者ノード $N^j_{k^*}$ のマップ上における位置，P_{jx} はその近傍ノードの位

置を表し $|P_{jx^*} - P_{jx}|$ は球の中心に対する勝者ノードと近傍ノードのマップ上での立体角である．なお，この立体角は学習回数とともに小さくしていく．学習係数 β は学習回数とともに減衰する関数をとる．

この後,Baum-Welch algorithm で５回更新をおこなう．

STEP 6: SOM におけるパラメータの更新 SOM におけるパラメータを更新し、STEP2〜STEP5 を学習回数分繰り返す．

なお, 本研究においては, 学習アルゴリズムの開発だけでなく, 頻度ベクトルでのノード間の類似関係を U-matrix の表示に反映できるようなシステムや, データを操作するためのユーザインターフェースついても開発をおこなった．

5.1 F-HMM-S-SOM における人工データを用いた実験

この実験においては, ベクトル統合型隠れマルコフ球面自己組織化マップでのモデルの推定および分類について, HMM-S-SOM の実験に用いたものと同じ人工データを用いて検証することで, 従来と比較してより適切にマッピングできるかどうかの評価をおこなった.

5.1.1 F-HMM-S-SOM における人工データでの実験結果

HMM のパラメータの類似度での U-matrix 表示における分類結果を図 31 に示す. 球面上に現れた境界は, 隣接したノードにおけるモデルのパラメータの差が大きいことを示すもので, モデルの違いを学習できていることが分かる.

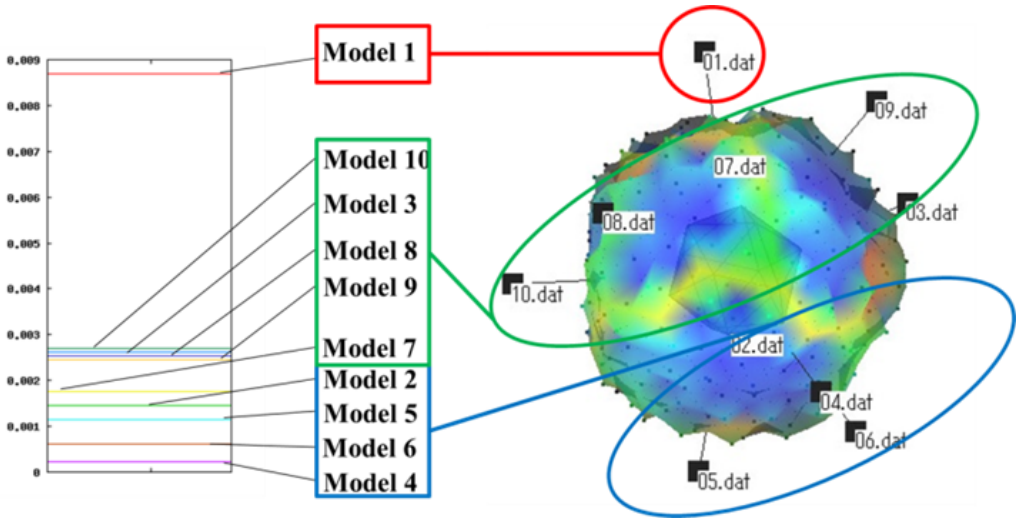


図 31: HMM パラメータもとづく U-matrix 表示での分類結果

また, モデルとしての類似度を表す尤度の表示 (図 32) においては, モデル間の類似度を表すグラフとおなじ相対関係を表す境界がマップ上に表示され, モデルの類似度を適切に学習できたといえる.

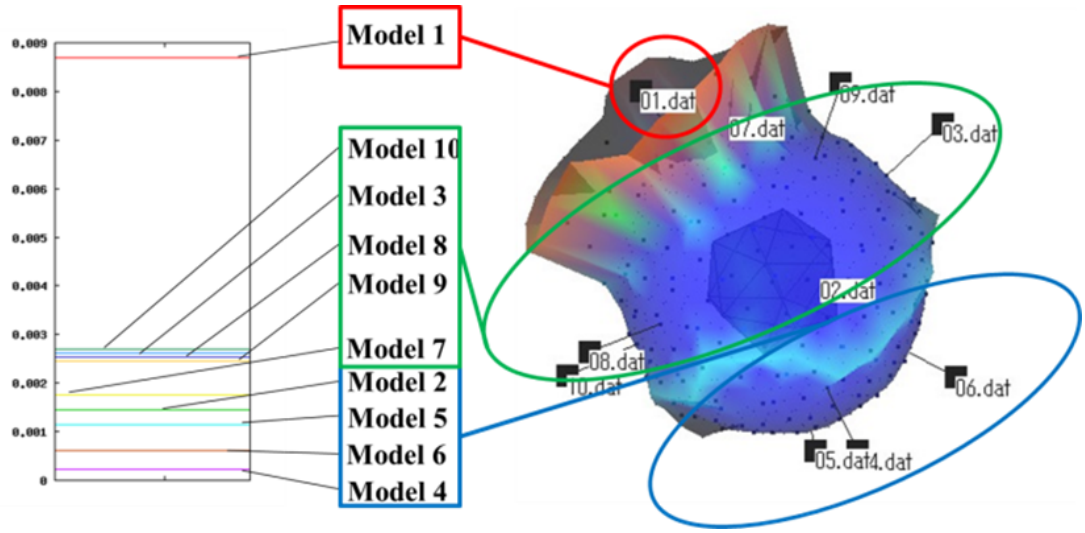


図 32: 尤度にもとづく U-matrix 表示での分類結果

5.2 F-HMM-S-SOM における実データを用いた実験

この実験においては, ベクトル統合型隠れマルコフ球面自己組織化マップでのモデルの推定および分類について, 実データとして,DNA データと株価データを用いて従来のアルゴリズム (HMM-S-SOM) と比較してより適切にマッピングできるかどうかの評価をおこなった.

5.2.1 DNA データを用いた実験とその結果

この実験においては, 生物の DNA の塩基配列である ATGC の配列を入力データとして学習を行った. この生物データは,ATGC の塩基配列を文献 [14] の方法で処理し, 長さ 6 シーケンスをそれぞれの生物に対し 1024 個用意した. つまり, 生物一つに対して 6 文字列が 1024 個集まった文字列集合が一つの生物におけるデータとなる. 生物の種類としては, ヒト (Hsall), ネズミ（Mmuall）, 犬 (Cfaall), 大腸菌（Ecoall）, ショウジョウバエ (Dmaall), イネ (Osaall) を用意した. これらを,HMM-S-SOM に学習させた結果を, 尤度における類似度にもとづいた U-matrix 表示で表したものを図 33 に示す.

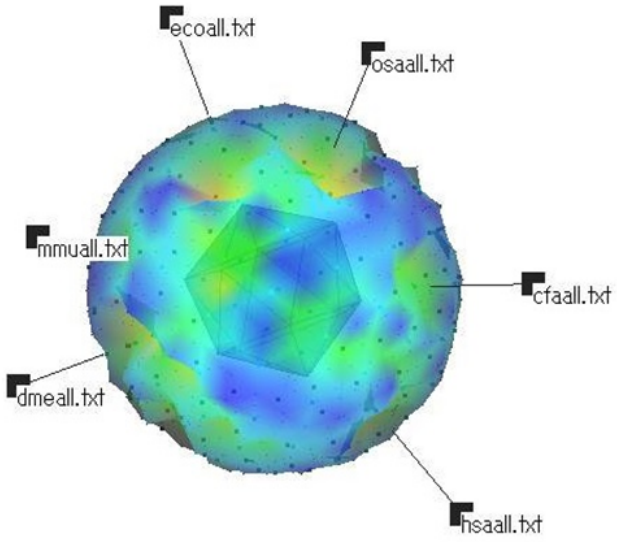


図 33: HMM-S-SOM における生物データの分類結果

この図におけるくぼみのあるところは, 局所的な学習を表している場所で, それぞれのデータの位置がこのくぼみにマッピングされているのがわかる. これはモデルが学習の過程で孤立してしまい適切な分類ができなかったことを示している. 一方 F-HMM-S-SOM については,HMM パラメータに関する U-matrix の表示から, モデルもパラメータの違いを学習できており (図 34), モデルの類似度に関しても局所的に学習が行われているノードはなく均一的に学習が行えたといえる (図 35). 分類結果のマップから大きく分けて, 犬, イネ, ショウジョウバエのグループと大腸菌, ヒト, ネズミに分類がされ, 犬はイネとショウジョウバエに対し境界ができており, モデルとしては離れていることが確認できた. また, 大腸菌についても, ヒトとネズミに対して境界線をもち, ヒトとネズミのモデルから少し離れた関係にあることが示され適切にモデルの違いを学習できているといえる.

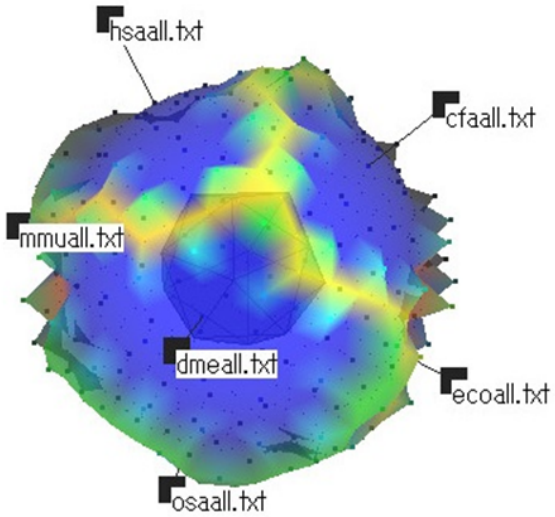


図 34: F-HMM-S-SOM における生物データの分類結果:尤度にもとづいた U-matrix 表示

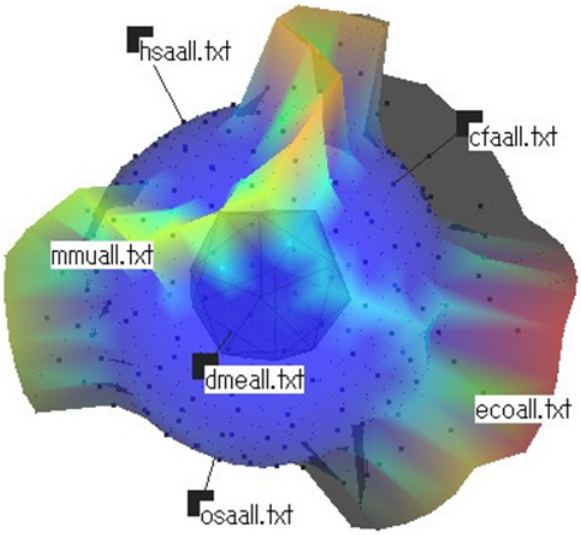


図 35: F-HMM-S-SOM における生物データの分類結果:モデルの構造にもとづいた U-matrix 表示

5.2.2　株価データを用いた実験とその結果

この実験においては, 時系列の実データとして 2010 年の九州電力会社の日足株価データを用いた．図 36 は６１日分の１０月から１２月までの株価データである．しかし, 株価のグラフは数値のデータであり, このままでは今回作成した, 離散的な隠れマルコフモデルでの学習は行えない．そこで, これらの数値としてのデータを文字列集合のデータに変換する必要がある．文字列集合への具体的な変換方法は次のように行った.

まず初めに, ５日間の幅を持ったウィンドウを用意し, このウィンドウを一日ずつずらしながら１０日分のデータを文字列集合に変換することを考える.

例えば, 図 36 のようにはじめは赤の場所にウィンドウがある場合, このウィンドウ内における５日間のグラフの上がり下がりについてみると初めは下がり, 次は上がり, そして少し停滞して最後に下がる．このとき, 上がるときは U, 下がるときは D, 停滞するときは S の文字に変換し, 上がりが急な時はいくつかの U に変換する．下がるときも、停滞するときも同様に変化の大きいときは連続した D に変換をおこなう．このようにして一つの文字列ができる．このウィンドウを緑の位置にくるまで１日ずつずらしながら同じようにこの５日間の区間を文字列に変換していくことで１０個の文字列が含まれる文字列集合を１組生成することができる．このようにして, ６１日の株価データからは文字列集合が４７組得られ, これらを F-HMM-S-SOM の入力データとして用いた.

このようにして変換された文字列集合の背景にあるモデルの学習を行うために,F-HMM-S-SOM に用いるノード上における HMM は全て同じ構造をもち, 図 37 のように状態数は最終状態を含め 4 つ, 出力シンボルの種類は U,S,D の 3 つとした.

実験結果

実験結果については,Umatrix によるモデル間の表示に加え, データの時系列にも

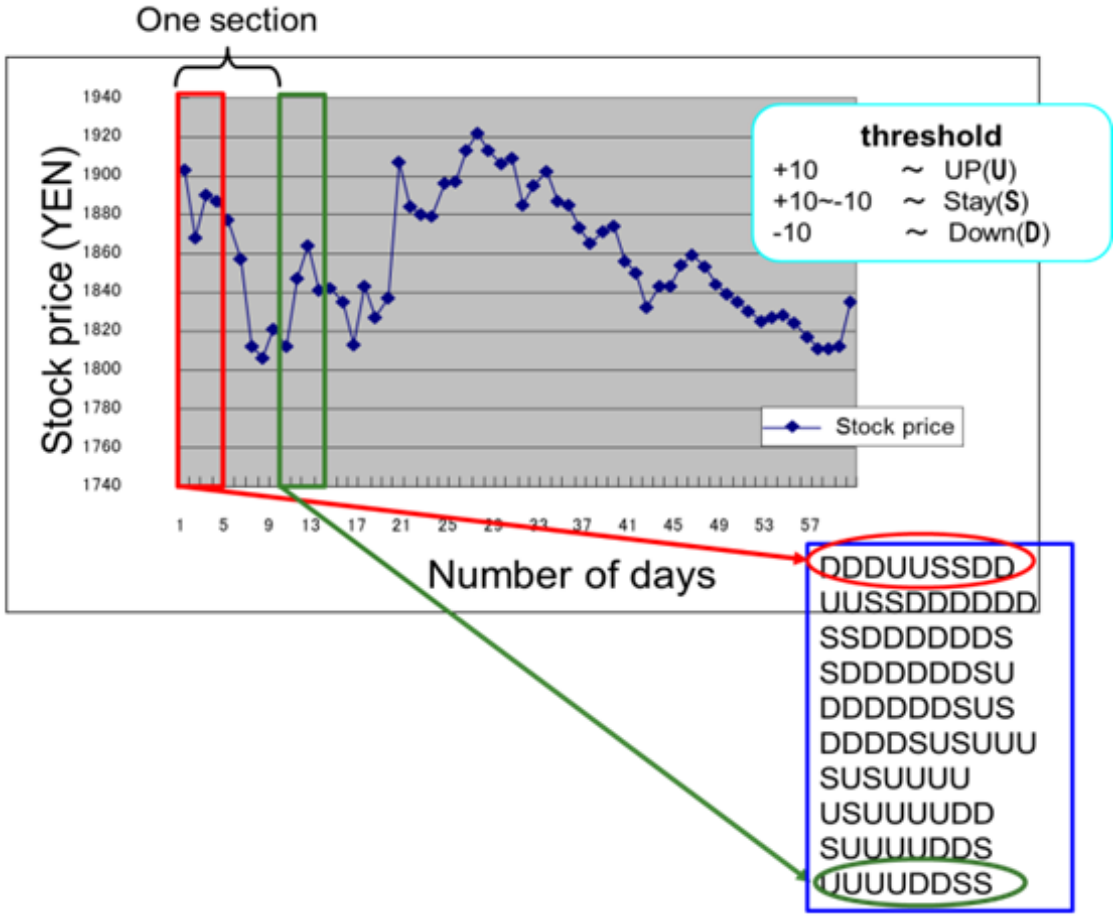


図 36: 九州電力の日足株価

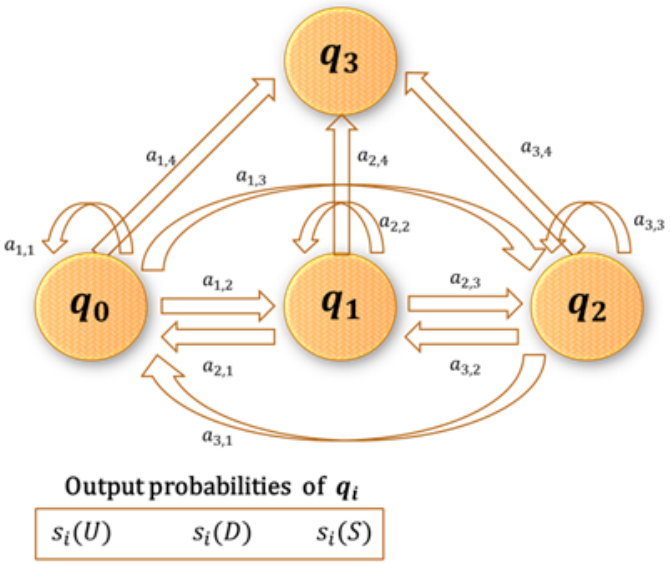


図 37: 株価分類の実験に用いる隠れマルコフモデルの構造

とづいて順番に線で繋いでいくことで, 単にモデルの分類結果を示すのではなく, 時系列の流れがわかるような表示で分類結果を示している. 具体的には時刻 t 番目のデータと t+1 番目のデータの勝者ノードの 2 点間を線で繋なぎ, 時系列でのデータのつながりを視覚化できるようにした. HMM-S-SOM と F-HMM-S-SOM のマッピング結果の比較を次の 2 つの図 38 と図 39 に分けて示す. データ 0 からデータ 2 2 までのモデルの分類結果を図 38 に, データ 22 からデータ 46 までの分類結果を図 39 に示す.

それぞれの図において, 上が頻度を考慮しない S-HMM-SOM の場合と, 下が頻度を考慮した F-S-HMM-SOM 場合である. マップ上の赤の線は, 前の日のデータと次の日のデータを線で結んだもので, これがデータの時系列における連続性を表示したものである. S-HMM-SOM のマップは赤い線が球面を貫いて互いに交わり、時系列情報を保持できていないことが分かる. また,U-matrix 表示についてはマップに孤立した山が見られる. これは学習過程において過学習がおこりモデルを適切に分類できていないことを示すもので, モデルの類似性についても適切にマッピングできていないことがわかる. 一方で,F-S-HMM-SOM のマップは, 線が球面上

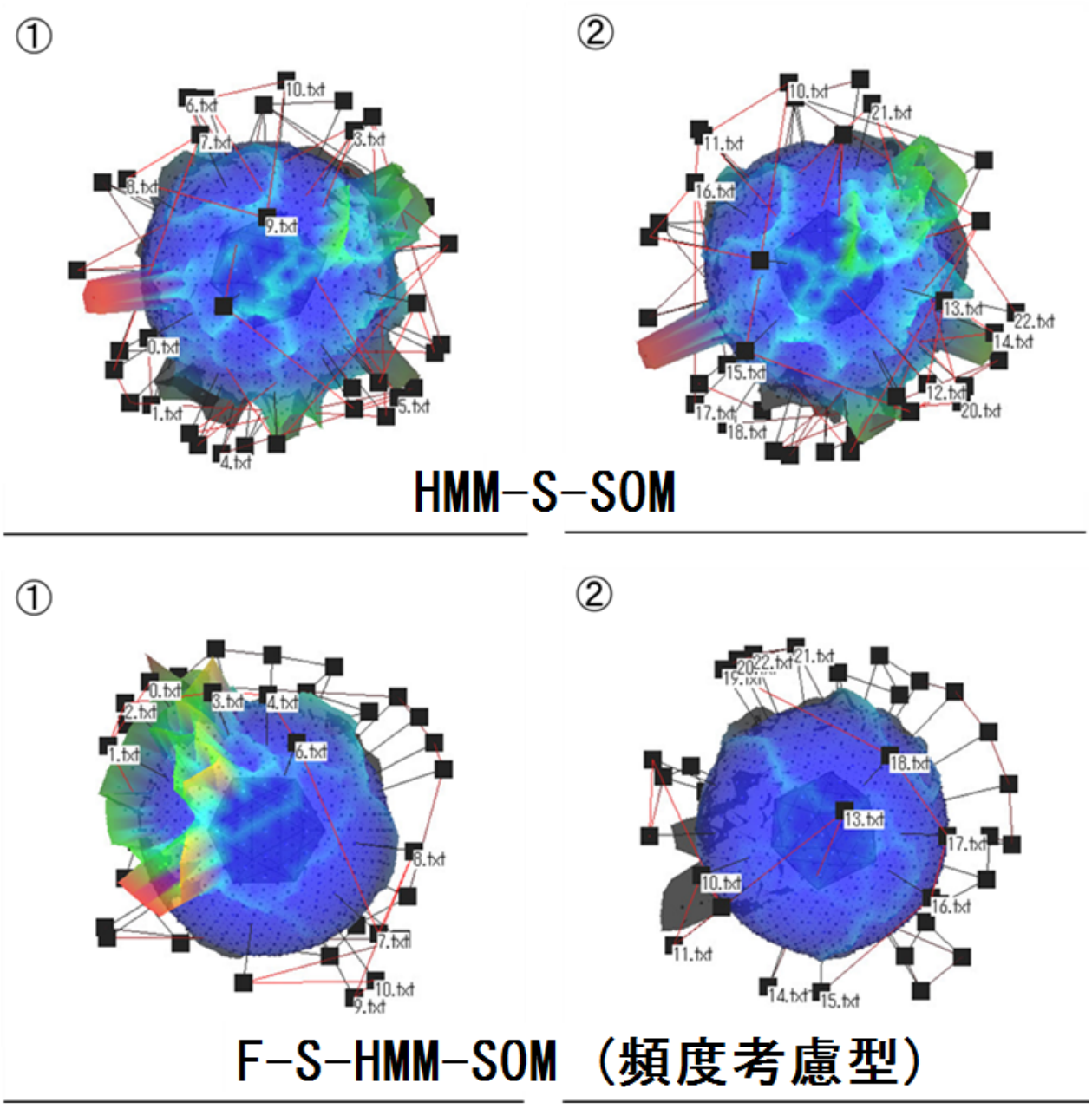


図 38: 株価データの分類結果の比較

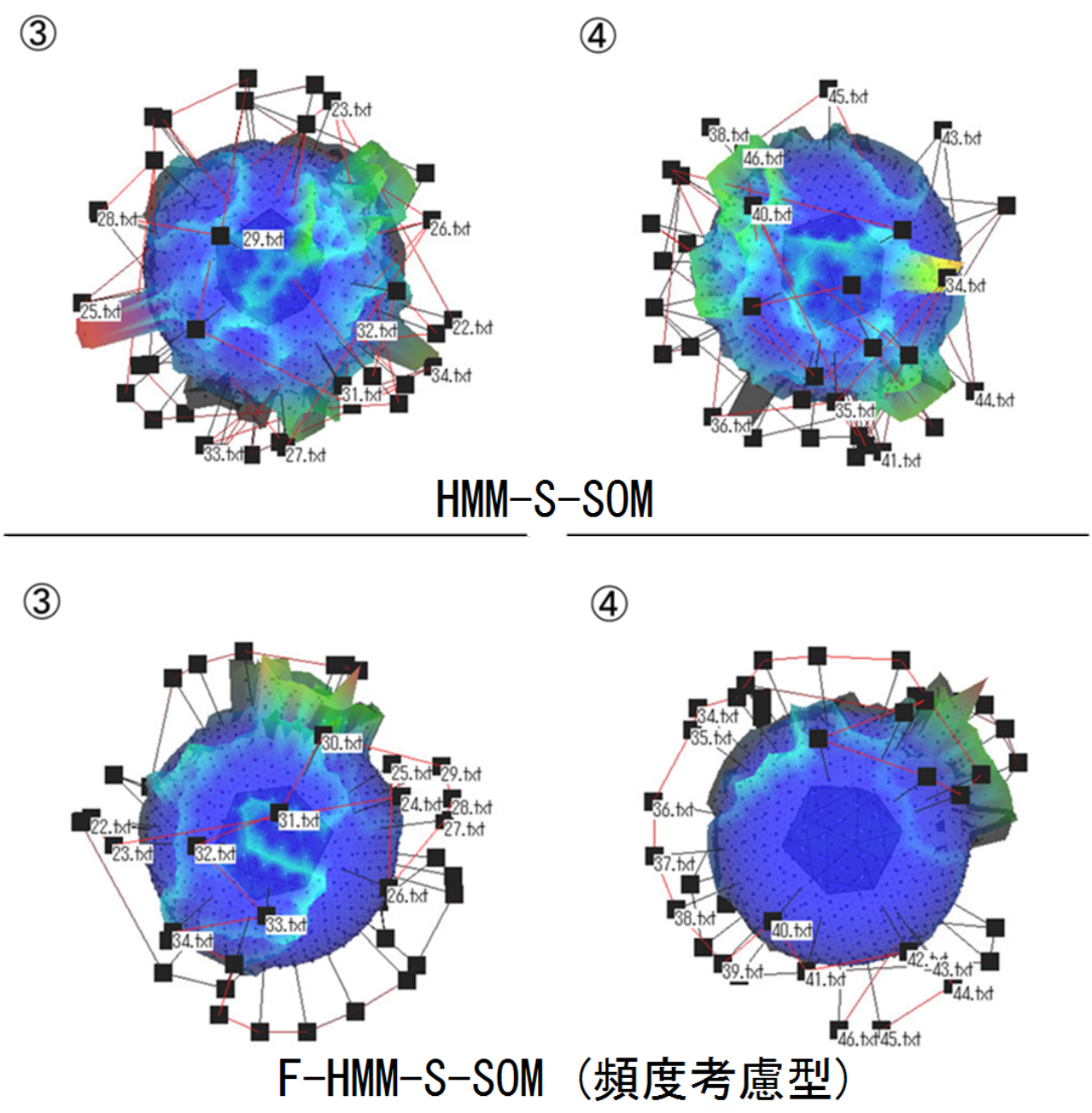


図 39: 株価データの分類結果の比較

で連続的につながり, 時系列情報が順に写像されていることが分かる．U-matrix 表示においても, モデル間に境界線みられ, モデルを分類できていると考えられる．時系列の並びに関しては, 前の日のデータと次の日のデータの球面に沿った距離の平均をとると, F-S-HMM-SOM では 1.377,S-HMM-SOM では 2.623 であった．この値は球面上にマッピングされた時系列データがどれほど連続的にマッピングされているかを示しており, 値が小さいほど, 連続性が良いといえる. 一方で, 2 つの連続したデータにおけるモデルの尤度の差を算出した結果を図 40 に示す．この図におけるグラフは, 縦軸がモデルの違いの度合いを示し, 横軸がデータの番号を表している．グラフの頻度を考慮しない S-HMM-SOM（図 40 の上）では, それぞれのモデルで過学習がおこりグラフに激しい上下が見られ, モデルの数に対しクラスタ数が多いことを示しており, モデルを適切に分類できていないことが分かる．一方, 頻度を考慮した F-S-HMM-SOM（図 40 の下）では, 色で示した位置でモデルの変化が起きていることを示しており, モデルを適切にクラスタリングできていることが分かる.

しかし, マップ上にモデルが分類できたことと, 株価の変動を推測するために適切にクラスタリングされているかについては, 別の問題であり, これを評価する必要がある.

そこで, 株価の変動を評価するために「移動平均線」と呼ばれるグラフを用いることにした．この移動平均線は株価変動を知るための指標として, 区間ごとの平均値をグラフにしたものであり, 実際に, 多くのトレーダーにとって, 区間ごとの平均値は企業の株価変動を見極める重要な指標となっている．この移動平均線から企業の状態や, 株価の変動を読み解くことはトレーダーにとって, まず初めに習得すべきスキルであり, このような背景から, 移動平均線の指標は株価変動の傾向を推測するための指標として機能している.

したがって, 移動平均線の傾向がマップに反映されていなくては, 適切なマップとは言えない．本研究でもちいた電力会社の株価の移動平均線とマップを比較す

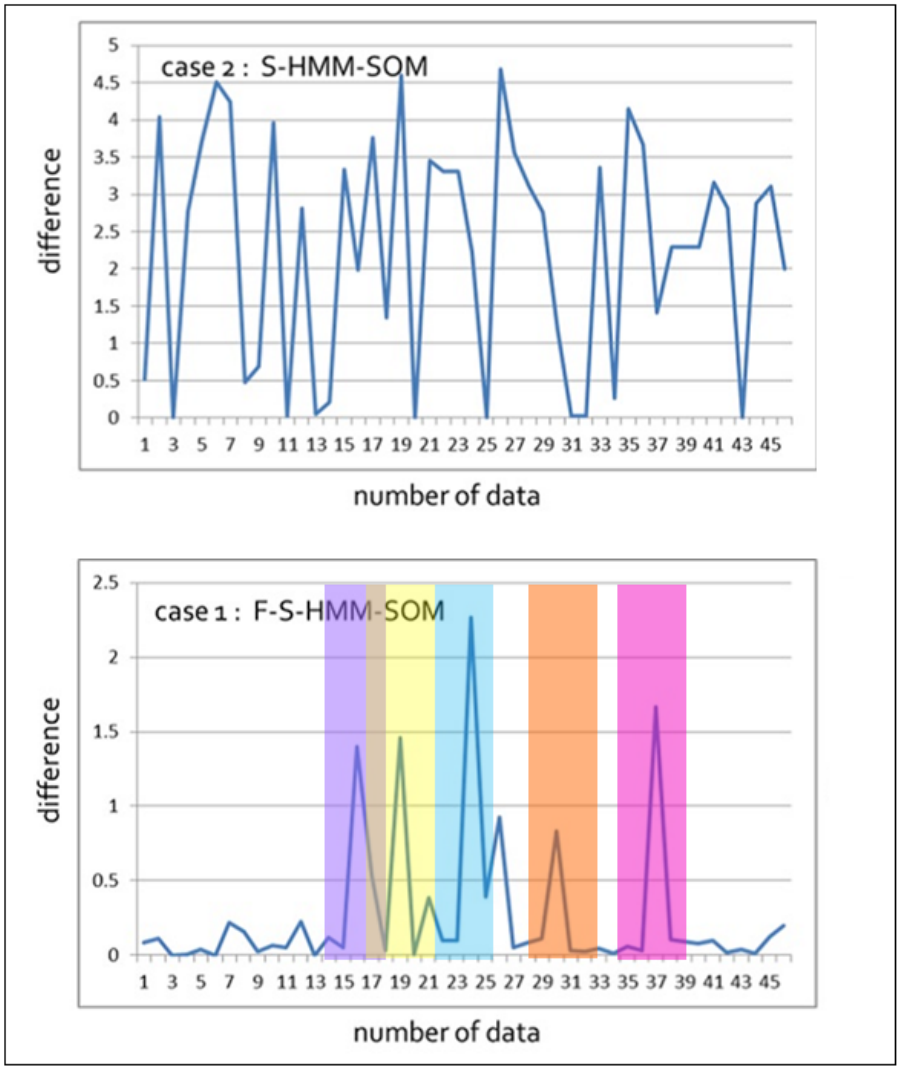


図 40: マップ上におけるモデルの差

るために, まずは, 企業のそれぞれの区間での平均値をグラフにしたものを図 41 の下に示す.

このグラフからから, 読み取れることは, 初めの７つ目までのデータは株価平均は下がり, その後上昇を続けたあと停滞を経て, 平均株価は下がりはじめ緩やかに下がり幅が小さくなっていく様子が分かる.

この移動平行線のグラフとデータのマッピング結果との比較を次の２つの図に分けて評価した. データ 0 からデータ 23 までのモデルの分類結果を図 42 に, データ 24 からデータ 46 までの分類結果を図 43 に示す.

企業業績が良くない初めの約 5 つまでのデータはマップの赤い領域にマッピングされているのが分かる. その後業績が良い方向に転じ, マップ上においてデータ５からデータ７にかけてマッピングされる位置に大きな変化が見られる. その後, 企業の業績は良い状態が続き, マッピングされる位置は緑の領域になるが, その後の業績は停滞し, マッピングされる位置もやや上の方へ移動していくのがわかる.

データ２４以降は企業の業績は緩やかに下がり始め, 分類結果においてはオレンジの領域にマッピングされている.

企業の業績の下がり方は徐々に緩やかになり, マッピングされる位置においても停滞の要素を含んだ緑色の領域にこのようにやや寄っていくのが分かる. これは, この区間のデータは停滞の要素を含んでいるためマッピングの位置も停滞の集まる領域に引っ張られる形でモデルが分類されたためと考えられる.

このように, モデルの分類を行いながら, 時系列情報と株価の移動平行線の傾向を再現することができ, 適切な分類結果を得ることができたといえる.

しかし，本研究の目的は，テストデータを適切に表現できるモデルを得ることではない. 我々の興味は，このモデルが未知のデータを再現することができるかということである.

実際に，先によって得られたマップ上の株価の変動を学習したモデルを用いて株価の予測実験を行い，未知のデータに対しても汎用性の高いモデルが得られて

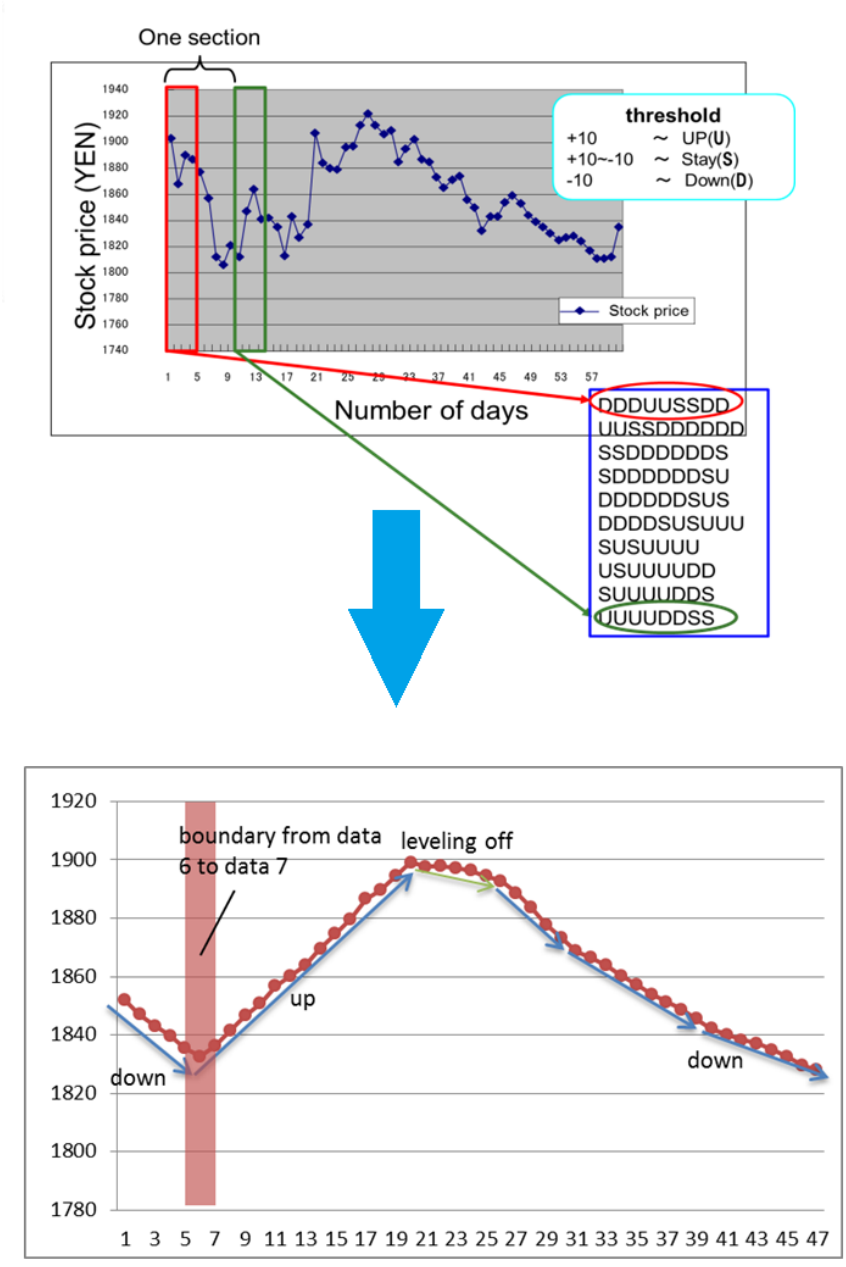


図 41: 株価のデータの移動平均線

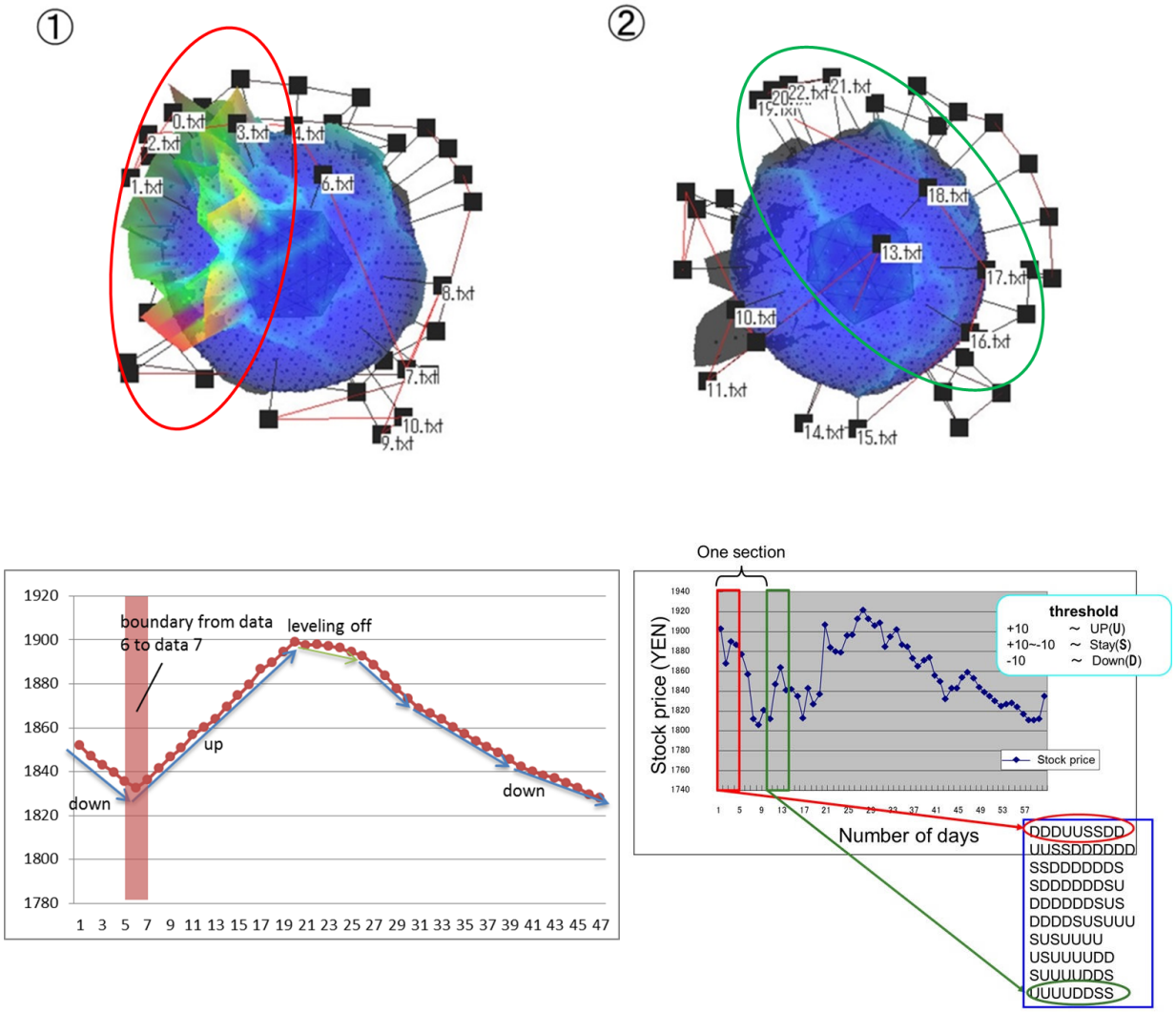


図 42: 移動平均線との比較

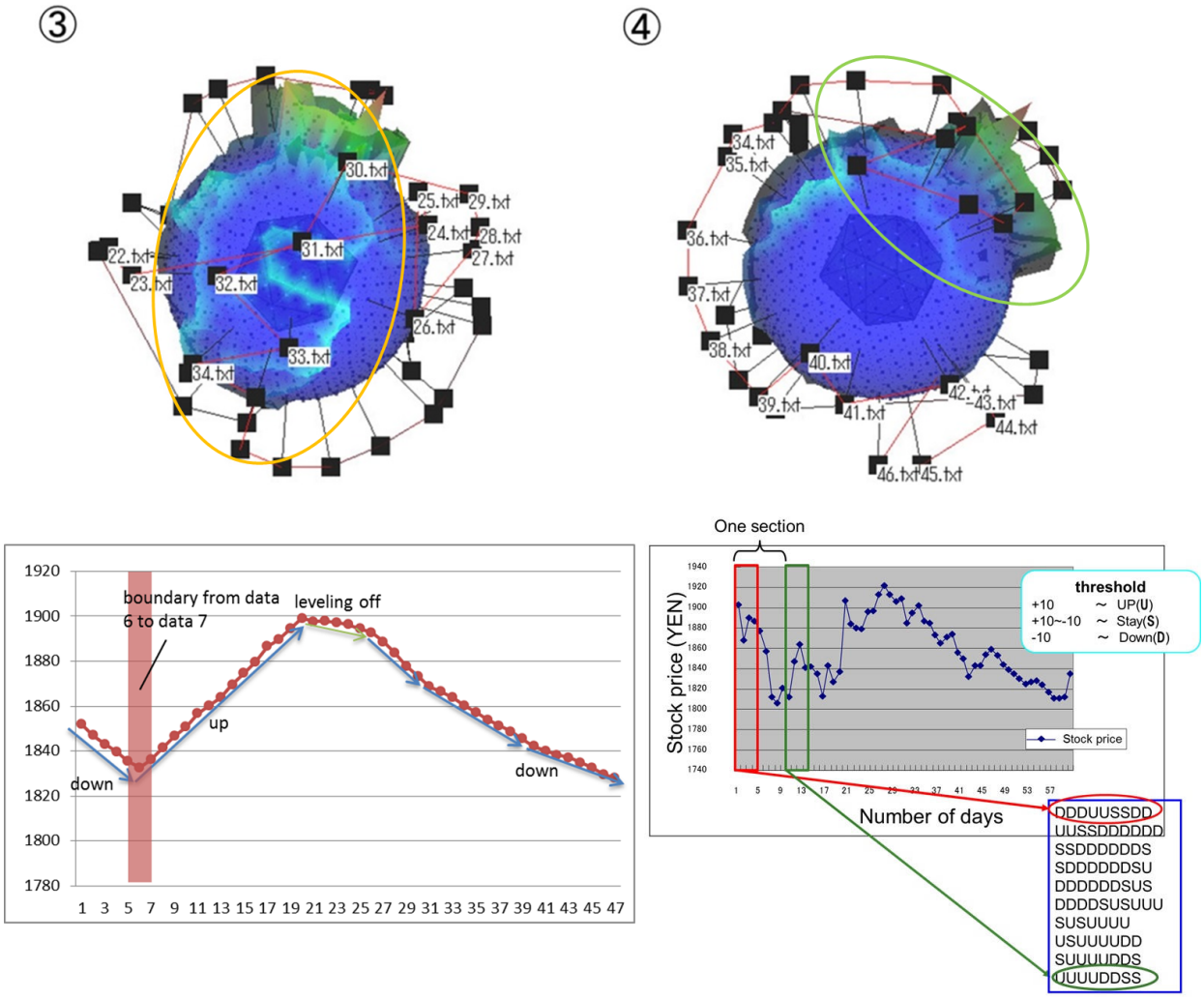


図 43: 移動平均線との比較

いるかの定量的評価をおこなった.

予測に用いる学習データは，翌年２０１１年度における九州電力の株価データを 230 日分のデータに対して予測の評価を行った.

予測の方法は，今後の株価の上がり下がりについて予測をしたいデータに対し，過去１４日分のデータを，図 36 に示す変換方法で文字列集合に変換し，この文字列集合を出力する確率が最も高いモデルをマップ上から探す．これによって，現在の株価の状態が，２０１０年の，どの日の状態に近いかを知ることができる．そして，その日から５日後の株価の変動を調べることで，株価の変動を予測する.

予測の評価方法は，現在の株価の状態に最も類似している過去の状態の５日後の株価が上がっていれば，実際の５日後の株価も上がると予測し，下がっていれば下がると予測する．このようにして，２３０日分のデータ中何日分当たっているかを評価する.

この実験結果を表 23 に示す.

表 23: 予測結果

	予測精度
頻度考慮なし(HMM-S-SOM)	49.17%
頻度考慮あり (F-HMM-S-SOM)	64.95%

実験の結果，頻度を考慮しない場合の予測精度が約４９％であるのに対し，頻度を考慮した場合は約１６％予測精度が向上し，新たなデータに対しても有用な結果を得ることができた．これは，頻度を考慮した場合は，考慮しない場合に比べて汎化性能が高いことを示しており，既知データだけでなく，新たなデータに対してもモデルの当てはまりが良いことを示している.

6　因果関係考慮型二層球面自己組織化マップ

データ分析においては, まず初めに因果関係がある要素を適切に抽出する必要がある．分析対象となるモデルが決まっているときは, 分析対象でないモデルと分析対象のモデルとの間の要素の違いを解析すればよい．しかし, 対象となるモデルが決定しておらず, 異なるモデルが混在している可能性のある場合は要素間の特徴を解析することは極めて困難になる．実際にこれまでの研究における隠れマルコフモデルの分類は, 一つのデータの背景にあるモデルの数が 1 種類ということを前提にしており, モデルが複数考えられる場合には応用することができない．そこでまずは, 3 次元のデータに対して分類が可能なアルゴリズムを考え, そのアルゴリズムを発展させていくことにした．異なるモデルから出力されるデータが混合した場合, それらを識別するためには各要素の分布のわずかな差を学習する必要がある．そこで注目したのが自己組織化マップにおける, 類似したデータを学習の際に引き寄せる性質である．逆に言えば引き寄せられない要素は類似していないということでもある．自己組織化マップのアルゴリズムにおいて, 勝者ノードの近傍範囲を更新する前に, 近傍範囲に存在する他の勝者ノードが内包する各要素の分布の広がりをチェビシェフの不等式を用いることで評価し, 要素間の関係性を学習できる自己組織化マップ（因果関係考慮型二層球面自己組織化マップ）の開発をおこなった．この自己組織化マップの表面は仮想的に球面を 2 層（図 44）にしたもので, 上部から 1 層目,2 層目となる構造である．本研究においては, 2 層目で止めているがより細かな分類のために将来的には多層になることを想定してアルゴリズムが組まれている．

因果関係考慮型 SOM は入力データ $\boldsymbol{v_j}$ を学習するためのベクトル $\boldsymbol{x}_i$ を 1 層目のノードに, 2 層目のノードには因果関係のある要素を記憶するためのマスクベクトル (mask-vector)$\boldsymbol{w}_i$ をそれぞれ内包している．マスクベクトル $\boldsymbol{w}_i$ の要素数は, 入力ベクトルの要素数をもち, 各要素は 0 から 1 までの値をとる．この 1 と 0 の意味については,1 に近い要素同士は因果関係があり,0 に近い要素は因果関係のない要素

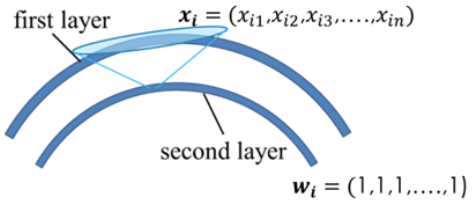


図 44: マップの構造と受容野

であることを示す．初め, このマスクベクトルの要素は全て 1 に設定してあり, 学習によって 0 か 1 に近づくように更新されていく．また,1 層目には 2 層目の受容野があり, その広さは立体角で 30 °とする．これは最終的な近傍範囲の約 5 倍に設定してあり, あまり大きすぎるとノイズが学習に悪影響を与えてしまい, また, 小さすぎると要素間の因果関係の学習が不十分になり, 実験により決定した．

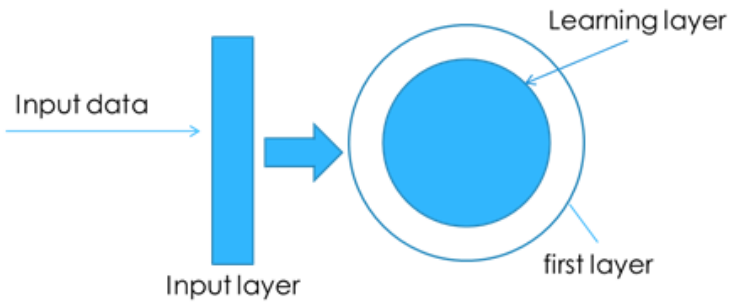


図 45: 因果関係考慮型 SOM の概念図

また, 入力データは直接学習させるのではなく, 入力層を介して学習させる（図 45）．入力層のノードは入力データの要素にフィルタをかけて, 必要でない要素が入力されないように抑制をかける働きをもつフィルタベクトル $\boldsymbol{h}_a$ がノードに内包されている．入力層におけるノードの個数は入力データ数に等しく, そのノードに内包されているフィルタベクトルの要素数についても入力データと同じ要素数をもっている．

以下が学習アルゴリズムである．

STEP 1: 初期化

1 層目のベクトル $\boldsymbol{x}_i = \{x_{i0}, x_{i1}, ..., x_{in}\}$ は乱数によって初期化し, 入力層のフィルタベクトル $\boldsymbol{h}_a = \{h_{a0}, h_{a1}, ..., h_{an}\}$ と 2 層目のマスクベクトル $\boldsymbol{w}_i = \{w_{i0}, w_{i1}, ..., w_{in}\}$ の要素を全て 1 に設定する.

STEP 2: 勝者ノードの決定

次の式を最小値にするようなノード i を勝者ノードとする.

$$[argmin]_i \frac{1}{L} \sqrt{\sum_j \{w_{ij} * (h_{aj} * v_{aj} - w_{ij} * x_{ij})\}^2} \tag{34}$$

ただし,L は次の式に従う.

$L = \sqrt{\sum_j w_{ij}^2}$

このときの勝者ノードにおける $\boldsymbol{w}_i$ を $\boldsymbol{w}_i^*$ と表記する.

STEP 3: パラメータの更新

勝者ノードの真下にある二層目のノード（$\boldsymbol{w}_i^*$）の受容野範囲に,

他の勝者ノードがある場合, 勝者ノードのベクトルと他の勝者ノードのベクトルとのユークリッド距離の平均値をベクトル $\boldsymbol{\mu}_a$ に保存する.

ベクトル $\boldsymbol{\mu}_a$ の各要素 $\boldsymbol{\mu}_{ai}$ は小さい順に並び替えてベクトル $\boldsymbol{\omega}$ に写像する（図 46）.

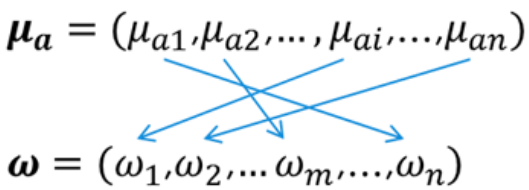


図 46: ベクトルのソート

チェビシェフの不等式を用いた次式 (35) を満たすような最小の m を検索する.

$$[argmin]_m\{\frac{1}{m}\sum_{k=2}^{m}(\frac{\omega_k}{\omega_m})^2\} \leq \phi \tag{35}$$

ここで,ϕ は学習の閾値で 0.5 以上の数値で調節される. μ の要素 t から ω の要素 k に変換する関数を $f(t) = k$ とすると, $f(t) \leq m$ のとき

$$w_{it}^{*new} = w_{it}^{*old} + \gamma(1 - w_{it}^{*old}) \tag{36}$$

$m < f(t)$ のとき

$$w_{it}^{*new} = w_{it}^{*old} + \gamma(0 - w_{it}^{*old}) \tag{37}$$

となるように更新する. ただし,γ は $0 < \gamma < 1$ となるような学習係数である.

また, 二層目におけるベクトル $\boldsymbol{w}_i^*$ の全要素が閾値を超えたときベクトル $\boldsymbol{w}_i^*$ のベクトルはそれらの閾値に従い 1 または 0 の値に設定され勝者ノードに対応するフィルタベクトル $\boldsymbol{h}_a$ に転写される.

STEP 4: 勝者ノード, 近傍ノードの更新

勝者ノードと近傍ノードの各パラメータは次の式に従い更新される.

$$\boldsymbol{x}_j^{new} = \boldsymbol{x}_j^{old} + \alpha(\boldsymbol{v}_a - \boldsymbol{x}_j^{old}) \tag{38}$$

$$\boldsymbol{w_j}^{new} = \boldsymbol{w}_j^{old} + \beta(\boldsymbol{w}_i^* - \boldsymbol{w}_j^{old}) \tag{39}$$

α は学習係数で, 学習回数と共に収束し,$0 < \alpha < 1$ の値をとり, β は学習回数に寄らず一定の値を取る定数で $0 < \beta < 1$ の間で調節される.

STEP2〜STEP4 の操作を, 各入力データに対して行い, 学習回数分これらの操作を繰り返すことで因果関係の学習を行う.

このようにして, 最終的に導出された各入力データに対する入力層におけるフィルタベクトル $\boldsymbol{h}_a$ の各成分が, そのデータがどの要素に因果関係をもつかを示すことになる.

6.1 人工データを用いた実験とその結果

実験結果の表示においては, フィルタベクトル $\boldsymbol{h}_a$ の値を因果関係の組み合わせごとに自動識別して分類し, 3 次元空間上にプロットした.

本研究において用いた人工データを生成するモデルはそれぞれ以下のモデル 1〜モデル 3 の三種類用意し, 次のようなデータになっている, 以下に示す, 数式内の $noise$ は擬似乱数列生成器の一つである Mersenne Twister[15] を用いて 0〜1 の一様乱数を生成している.

【モデル 1】x,y に正の相関があり,z はランダムなデータを生成するモデル.

$$\begin{cases} y = 3 * x + noise \\ z = noise \end{cases}$$

【モデル 2】x の値が 0.5 以下のとき, x,y に正の相関があり,z はランダムなデータを出力するモデルと、x,y に負の相関があり,z はランダムなデータを出力するモデルの混合モデル

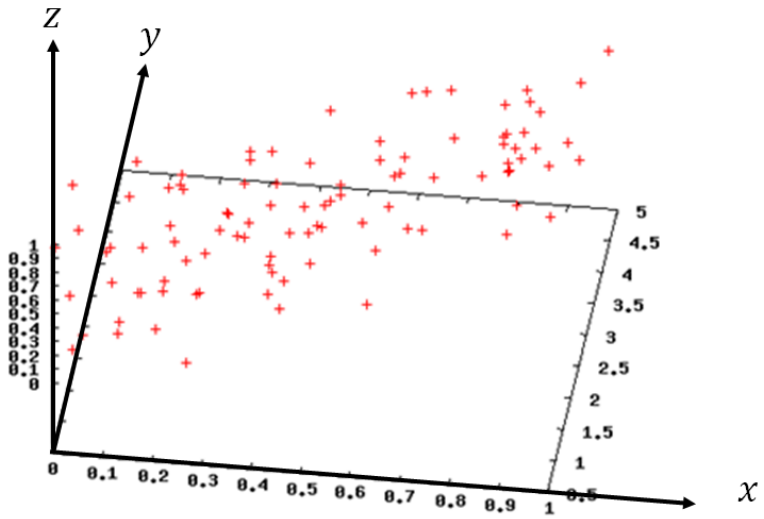


図 47: モデル 1

$$\begin{cases} y = 3 * x + noise \\ z = noise \end{cases} \quad (x \leq 0.5)$$

$$\begin{cases} y = -2 * x + noise \\ z = noise \end{cases} \quad (x > 0.5)$$

【モデル 3】x の値が 0.5 以下のとき,x と y に相関があり,z にはランダムノイズがのったデータを出力するモデルと, x の値が 0.5 以上のとき,y と z に相関があるデータを出力するモデルの混合モデル.

$$\begin{cases} y = 3 * x + 2 * noise \\ z = noise \end{cases} \quad (x \leq 0.5)$$

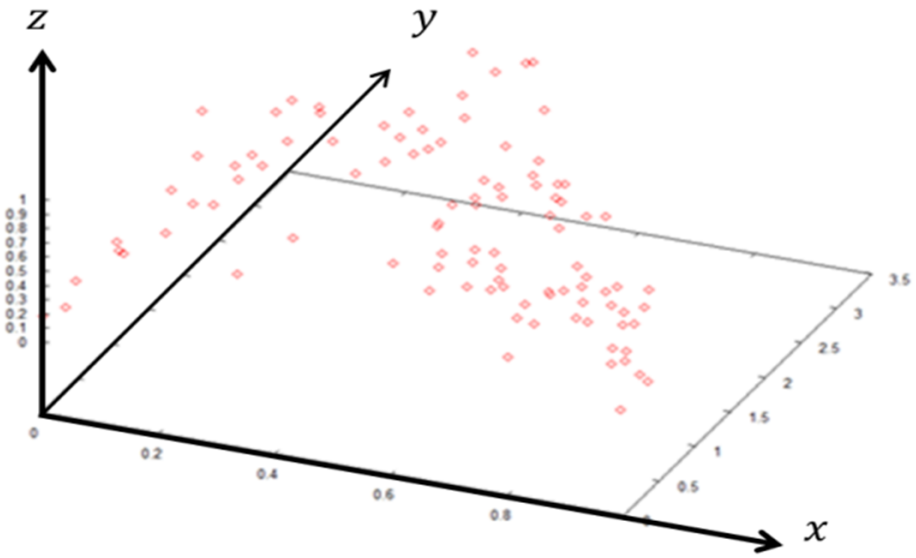


図 48: モデル 2

$$\begin{cases} y = -2 * z + noise + 2 \\ x = noise \end{cases} (x > 0.5)$$

以下にそれぞれの実験結果を示す.

6.1.1 モデル１に関する実験結果

モデル 1 に対する実験結果を図 50 に示す．この, 実験結果の緑の点は因果関係考慮型のアルゴリズムによって, x と y に因果関係があると判定されたデータである.

model 1 においては因果関係を適切に判別できたデータの割合は 100 %であった．相関のない z の要素は学習時にマスクベクトルの要素から外され, 適切な学習が行えたといえる．実際の x と y の相関係数は 0.819 であり, 強い相関係数において因果関係を適切に学習できた.

この学習アルゴリズムの勝者ノードの決定について一般的な SOM の手法で行った場合の実験結果を図 51 に示す.

従来の SOM の手法においては, 他の約 8 割のデータは, 全ての要素に因果関

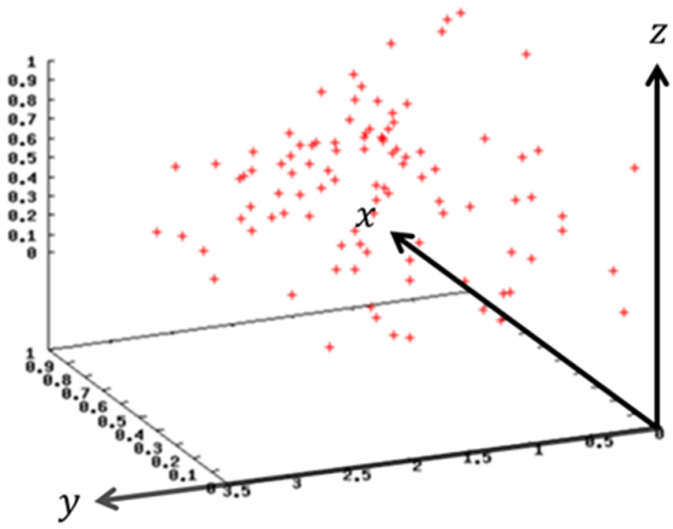


図 49: モデル 3

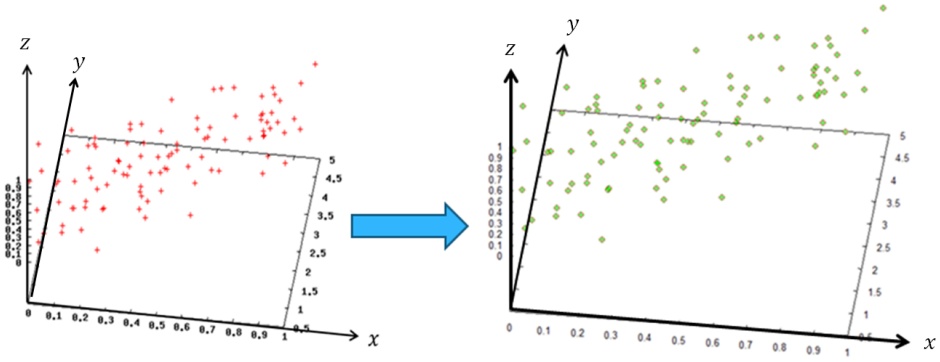


図 50: モデル 1 の実験結果

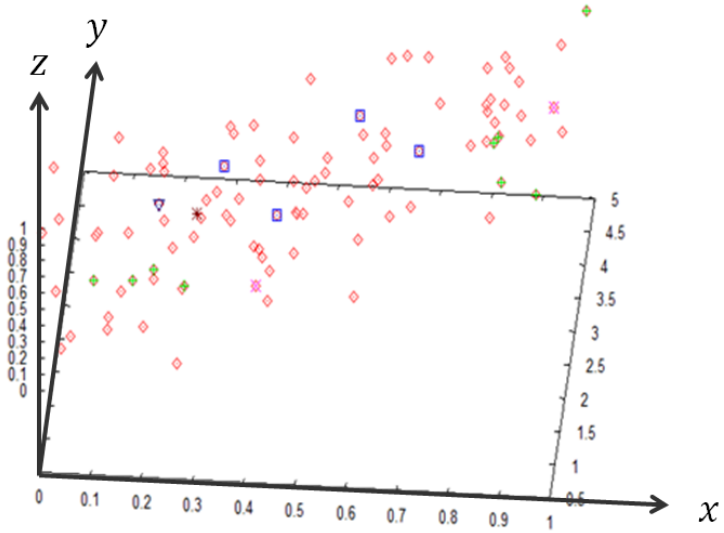


図 51: SOM の学習アルゴリズムにおけるモデル 1 の実験結果

係があると判定されており, x と y に因果関係があると判定されたデータは 9 点だけで, 従来のアルゴリズムでは因果関係を発見することはできないという結果を得た.

6.1.2 モデル 2 に関する実験結果

モデル 2 に対する実験結果を図 52 に示す. この, 実験結果の緑の点は x と y に因果関係があると判定されたデータでありまた, 青の点は x と y と z に因果関係のあるデータと判定されたデータである. モデル 2 においては, 因果関係を適切に判別できたデータの割合は 95 %であった. 相関係数が 0.335 と弱い相関であるにも関わらず, 有用な結果を得ることができた.

従来の SOM のアルゴリズムにおいては図 53 のような結果を得た,

この分類結果では, 約 8 割のデータが x と y と z に因果関係をもち, x と y に因果関係をもつと判定されたデータは 2 点のみであった. 従来のアルゴリズムでは,model 2 の因果関係についても因果関係の識別できなかった.

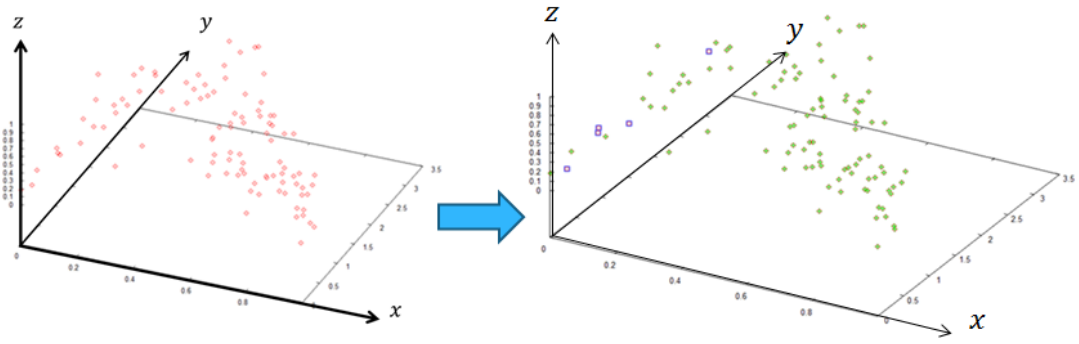


図 52: モデル 2 の実験結果

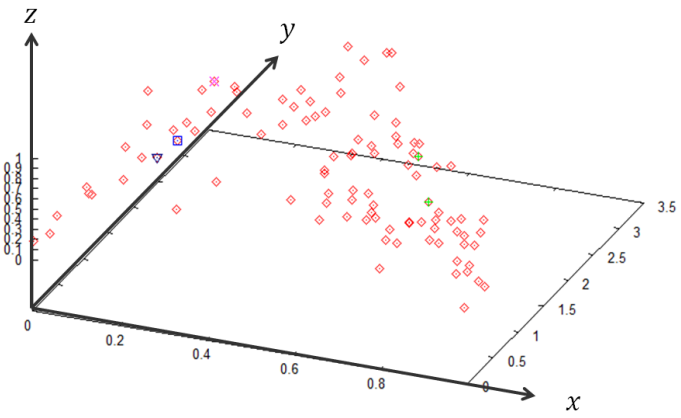


図 53: SOM の学習アルゴリズムにおけるモデル 2 の実験結果

6.1.3　モデル 3 に関する実験結果

モデル 3 における, 因果関係考慮型自己組織化マップの学習結果を図 54 に示す. 図 55 は図 54 を xy 軸平面についてみた場合を示す．青い点は x と y に因果関係があることを示し, 緑は y と z に因果関係があることを示すデータである．この結果を見ると,x,y のみに因果関係があるはずのデータの中に,y と z 間に因果関係があることを示す点が含まれていた．x の値が 0.5 以上のグループと 0.5 以下のグループにきれいに分かれるのが理想的な結果であるが, 実験結果はそうならなかった．しかし, は因果関係を適切に判別できたデータの割合は約 78 %であり, 有用な結果を得ることができたといえる.

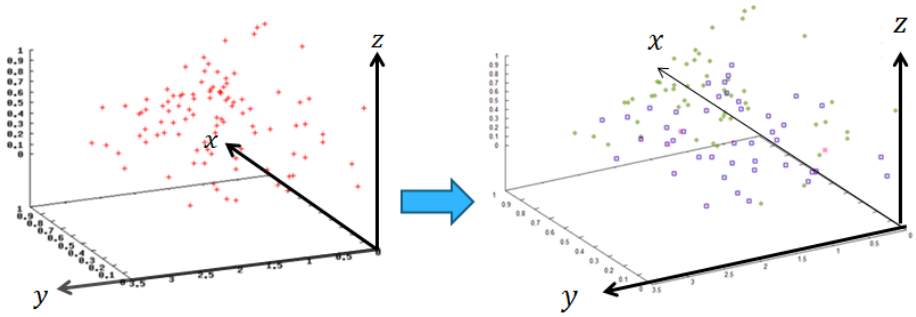


図 54: モデル 3 の実験結果

yz の因果関係のデータが,xy の因果関係のデータに混入してしまう結果になったのは, 図 56 をみると理解できる.

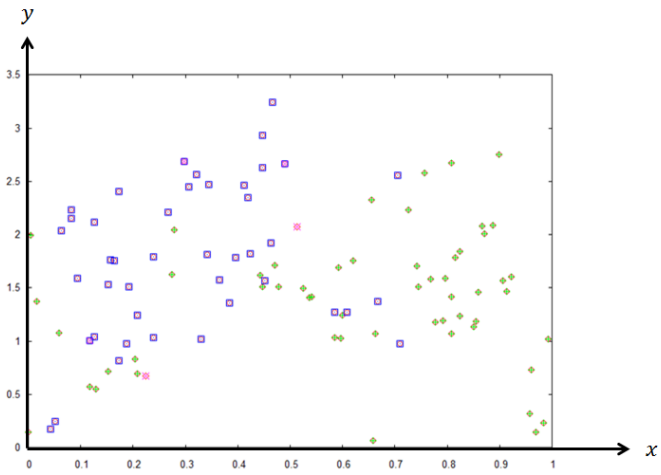


図 55: モデル３の実験結果：ｘｙ平面

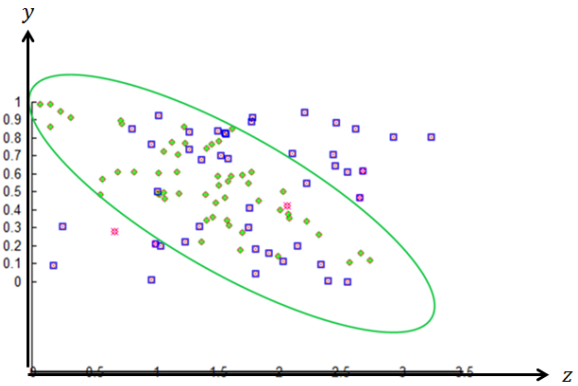


図 56: モデル３の実験結果：ｙｚ平面

図 56 は, 図 54 を yz 平面についてみた図である．この緑の円の領域における緑の点がが,yz に因果関係があると判定されたデータであり, yz 平面でみると非常にデータが密集し,xy に因果関係のあるデータも当然含まれている．実際には,x 方向の奥行があるが, 学習の過程において yz での因果関係をもつと判断され始めた緑の円の領域では,x 方向の奥行について学習を徐々に弱めるようになるため, このときに, まだ学習しきれていない xy に因果関係をもつデータがいくつかこの領域のデータに引きずられデータが誤認識されたと考えられる．したがって, このような引きずりを抑えることで, 解析精度は向上すると考えられる．

一方で, 従来の SOM による学習においては, 図 57 に示すような結果となった．

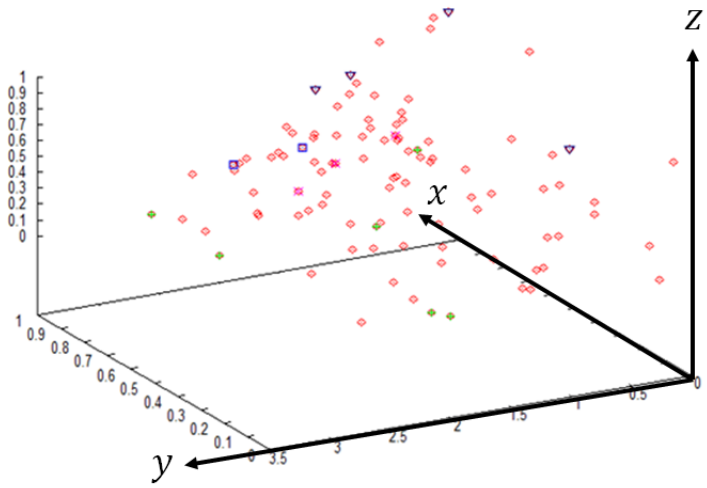


図 57: SOM の学習アルゴリズムにおけるモデル 3 の実験結果

従来のアルゴリズムにおいては, これもほとんどのデータが全ての要素間に因果関係があるとされ, その他のデータでは元の因果関係には含まれない x と z の間に因果関係のあるデータとして分類されるものまであった．SOM のアルゴリズムの因果関係を学習することができず, 実験結果にノイズが現れただけだった．

7　結論

本研究では、隠れマルコフモデルを用いて, 確率モデルなどの分類を可能にするために隠れマルコフ自己組織化マップを改良し，学習データの背景にある元のモデルにもとづいて, 隠れマルコフモデルを最適にクラスタリングすることが可能なアルゴリズムを開発し, 人工データと実データの両方を用いて実験と評価をおこなった．従来の学習アルゴリズムに, 入力データにおけるシンボルの出力頻度を考慮することで, 人工データ, 実データともに, モデルの分類を適切に行うことができた, 実データでの DNA の分類に関しては, 学習時の過学習を抑えたことで生物の類似性に基づいてモデルが分かれ, 尤度の表示においては生物のモデル間に境界線が現れ, 適切なマップを得ることができた．また, 株価を用いた企業動向の分類においては, 時系列情報を保持しながらモデルの学習を行うことができ, また, マップ上に移動平均線の傾向が再現でき, 有用な結果を得ることができた．このことは, もとから文字列であるようなデータを分類できるだけではなく, グラフのように徐々に変化する時間的に連続なデータについても分類でき, グラフで表されるデータの多くを隠れマルコフモデルで分類できる可能性があることを意味しており, これは今回提案した手法が幅広い分野に応用できる可能性を示唆するものである.

次に, 因果関係を学習する自己組織化マップにおいては, 本研究で開発したアルゴリズムによって, 人工データでの実験では適切に因果関係を学習することができ, 有用な結果を得ることができた．また, 因果関係を学習することのできるアルゴリズムについても開発を行った．開発した学習アルゴリズムの実験では, 一つのモデルを含んだモデル 1 のデータの因果関係を判別する実験においては, 全てのデータを適切に学習することができた．またモデル 2 においては同じ要素間に因果関係をもつ異なる 2 つのモデルの混合で, その学習結果においては同じ要素間の因果関係をもつデータとして三次元空間にプロットされた. このことは, 因果関係が同じ要素を含む異なるモデル間のモデル同士を分類できないことを示しているが, 要素間の因果関係にもとづいて, まずはデータをこの学習アルゴリズムを用いて分類し,

その後はそれぞれカテゴライズされたデータに対して, k-means 法などを適用することで分類が可能なため, この問題に対しては議論する必要はない. モデル 3 におけるデータの学習結果においては, データ同士がある軸の方向から見ると重なっているような場合には適切に学習することのできるデータの割合がモデル 1 とモデルに 2 の学習結果に比べ少なく, 学習アルゴリズムを改善する必要があることが分かった. しかし, この課題は学習過程で他の要素に更新値が引きずられないように, 更新式に禁止項を考慮することなどによって改善できると考えられる. 今後, これらの課題を解決し, 因果関係考慮型 SOM と F-HMM-S-SOM を組み合わせたアルゴリズムを開発することで, モデル数が未知数であるデータの分類や, 混合モデルの分類に応用できることが期待できる.

謝辞

本研究の遂行ならびに本論文の作成にあたり, 日頃から丁寧で親切な御指導、御鞭撻をいただいた, 村松和弘 教授には, 終始ご親切なご教示と多大なるご指導を賜りました．心からの感謝の意を表し, 厚くお礼申し上げます.

また, 本論文をまとめるにあたり, 本学大学院工学系研究科, 後藤聡 教授, 高橋英嗣 教授及び堂園浩 准教授には多くの御指導, 御助言を頂きました. ここに深く感謝の意を評します．本研究を進めるにあたり, 御協力をいただいた高炎輝 助教，築地浩 技術職員には深く感謝致します．また, 本研究の過程でご助力をいただきました村松研究室の学部生, 大学院生の皆様に深くお礼申し上げます.

参考文献

[1] 片山 雄介, 後藤滋樹：隠れマルコフモデルによる トラフィックの分類,2012.

[2] 木村 広希, 天笠 俊之, 川島 英之, 北川 博之：サポートベクターマシンと隠れマルコフモデルを用いた 気圧配置分類手法, DEIM Forum 2010 F4-2,2010.

[3]Teuvo Kohonen, 大北 正昭, 徳高 平蔵: 自己組織化マップ, シュプリンガーフェアラーク東京.

[4]Chie Morita and Hiroshi Tsukimoto:Knowledge discovery from numerical data, Knowledge-based Systems,Vol.10,No. 7,pp.413-419,1998.

[5] 藤巻遼平:線形時間異種混合モデル選択のための期待情報量基準最小化法,2009.

[6] ソニー株式会社/青山 一美, 南野 活樹, 下村 秀樹：HMM-SOM に基づく認知行動の獲得とその学習.

[7] 田中 耕平, 堂園　浩：HMM-SOM を用いた MIDI 音楽小節のクラスタリング.

[8] 中塚 大輔, 藤村 喜久郎, 大北 正昭, 大藪 又茂, 徳高 平蔵: A spherical SOM and its examples/Thecnical Report of IEICE.

[9] 徳高平蔵, 藤村喜久郎, 大北正昭, 中塚大輔: S-SOM を用いたクラスタ分析バイオメディカル・ファジィ・システム学会講演論文集, 0 巻 0 号（頁 153 〜 156）,2004.

[10] 吉 見 真 聡 , 西 本 要 , 王 路 易 , 廣 安 知 之 , 三 木 光 範:球面 SOM によるパレート解集合の可視化の検討 , 情報処理学会研究報告, Vol.2010-MPS-77 No.29,2010.

[11] 高塚 正浩，Ying Xin WU：球面 SOM のデータ構造と量子化誤差の考察およびイン タラクティビティの向上，日本知能情報ファジィ学会誌：知能と情報，Vol.19, No.6, pp.611-617,2007.

[12]N2Factory DirectX シェーダプログラミング仕組みからわかるゲームエフェクトテクニック.

[13]Jeffrey Richter, Christophe Nasarre:Advanced Windows 第 5 版 上, 日経 BP ソフトプレス,2008.

[14]Hiroshi Dozono. et.al:A Design Method of DNA Chips for sequence Analysis Using Self Organizing Maps, Proceeding of WSOM 2003,pp.116-121,2003.

[15]M. Matsumoto and T. Nishimura:A 623-dimensionally equidistributed uniform pseudorandom number generator, ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, January pp.3-30,1998.

研究業績

査読付学術論文

1. Gen Niina, Kazuhiro Muramatsu, Hiroshi Dozono: "Basic Study on the Classification of Time Series Data Using a Frequency Integrated Spherical Hidden Markov Self Organizing Map", Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol.19 No.2, pp.212-216, 2015.

査読付国際会議論文

1. Gen Niina, Kazuhiro Muramatsu, Hiroshi Dozono: "Causal analysis of data using 2-layerd Spherical Self-Organizing Map", The 2015 International Conference on Computational Science and Computational Intelligence (CSCI'15), Las Vegas, USA, pp.198-202, 2015.

2. Gen Niina, Kazuhiro Muramatsu, Hiroshi Dozono, Tatsuya Chuuto: "The data analysis of stock market using a Frequency Integrated Spherical Hidden Markov Self Organizing Map", ICSIIT 2015, 4th International Conference on Soft Computing, Intelligent System and Information Technology, March 11-14, 2015 / Bari, Indonesia, pp.195-204, 2015.

3. Gen Niina, Kazuhiro Muramatsu, Hiroshi Dozono: "The Frequency Integrated Spherical Hidden Markov Self Organizing Map for Learning Time Series Data", ISIS2013, The International Symposium on Advanced Intelligent Systems, Daejeon, Korea, pp.362-370, 2013.

4. Hiroshi Dozono, Gen Niina and Kazuhiro Muramatsu,: ”Mapping of DNA sequences using Hidden Markov Model Self-Organizing Maps”, Proceedings of The 10th annual IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, IEEE Press, Singapore, 2013.

5. Hiroshi Dozono, Gen Niina and Yutaro Kaneko: ”Mapping the Groups of DNA Sequences using Hidden Markov Model Self Organizing Maps”, The First BMIRC International Symposium on Frontiers in Computational Systems Biology and Bioengineering, Fukuoka, Japan, 2013.

6. Gen Niina, Hiroshi Dozono: ”The Spherical Hidden Markov Self Organizing Map for Learning Time Series Data”, 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, pp.563-570, Part I, 2012.

その他

1. 新名玄, 村松　和弘, 堂園　浩: 「頻度考慮型隠れマルコフ球面自己組織化マップを用いた時系列データの分類に関する研究」, 第 15 回自己組織化マップ研究会, 福岡, 2014.

2. 新名玄, 堂薗浩: 「DirectX を用いた球面 SOM の実装」第 19 回電子情報通信学会九州支部学生会講演会, 2011.

3. 甲斐大翔, 新名玄, 堂薗浩: 「隠れマルコフモデルと自己組織化マップを用いた時系列情報の解析」第 64 回電気関係学会九州支部連合大会, 2011.