一般化自己組織化マップを実現するモジュラーネットワーク

Modular Network for Generalized Self-Organizing Map

徳永憲洋 Kazuhiro TOKUNAGA

アブストラクト データの視覚化はデータの関係性や隠れた情報を解釈するうえで重要なテクニックである。データの視覚化の方法としては主成分分析が広く知られているが、人工ニューラルネットワークの一つである Kohonen が提案した自己組織化マップ (SOM: Self-Organizing Map) もデータの視覚化ツールとして広く用いられている。SOM は多次元空間上のデータを二次元程度の低次元空間に位相を保存したまま写像することによりデータの視覚化が可能である。しかし課題によってはデータを生成するもとのモデルを対象とし、モデル間の類似性に基づいた分類やマップ生成をしたほうがよい場合もある。このような課題を解決する方法として、筆者は SOM を発展させたモジュラーネットワーク型自己組織化マップ (mnSOM: Modular Network SOM) を提案した。mnSOM は様々なモデルを対象としてマップを生成できるため一般化された SOM としての枠組みをもつものといえる。本稿では SOM の解説も交えながら mnSOM の理論や学習アルゴリズムについて解説をする。

キーワード 自己組織化マップ, モジュラーネットワーク型自己組織化マップ

Abstract Data visualization is an important technique for interpreting complex data and finding relationships between data. Although principal component analysis is widely known as a method of data visualization, the Self-Organizing Map (SOM) proposed by Kohonen, one of the artificial neural networks, is also widely used as a data visualization tool. The SOM performs a topology-preserving transformation from a higher-dimensional vector space to a lower one, and generates a map that represents the relationships between data vectors. In some cases, however, it is necessary to generate maps based on the similarity between models that generate the data vectors. The author proposed a modular network self-organizing map (mnSOM: Modular Network SOM) as a method to solve such problems. The mnSOM has an architecture as a generalized SOM since the mnSOM can generate maps for a variety of models such as input-output functions, dynamical models, manifolds and so on. In this paper, the theory and learning algorithms of mnSOM are explained with an explanation of SOM.

Key words self-organizing map, modular network self-organizing map

1. はじめに

データマイニングにおいてデータの構造や関係性を視覚化することは隠れた情報を解釈するうえで重要である。データの視覚化としては主成分分析や多次元尺度構成法といった多変量解析の手法がよく利用されている。どちらの方法も多変量のデータに対して、データを低次元空間に配置することでその関係性を視覚的に把握することが可能である。また主成分分析は多変量データのデータ次元の削減にも利用される。

他方、データを視覚化するニューラルネットワークとして Kohonen が提案した自己組織化マップ (SOM: Self-Organizing Map) がある⁽¹⁾。SOM は代表的な教師なし学習法である。SOM

は多次元空間上のデータを二次元程度の低次元空間に位相を保存したまま写像することによりデータの視覚化が可能となる。主成分分析と SOM の違いは大雑把にいうとデータの射影先が線形部分空間か非線形部分空間かの違いといえる。主成分分析はデータのばらつきを表現する共分散行列の固有値問題を解き、固有値の大きい固有ベクトルで作られた線形部分空間上にデータを射影する。一方で SOM は位相を保存したまま低次元空間に写像するマップを生成するので、多次元データにフィットするようなネットワークを形成する。生成されたネットワークは非線形多様体のモデルとして表現され、データは非線形部分空間上に射影される。以上のことから、データの分布が複雑でない場合は主成分分析でも十分であるが、空間で曲がりくねったようにデータが分布するような課題においては SOM が適していると考えられる。上記のような特徴から SOM はデータマイニングツールとして分野を問わず多くの実課題で利用されており研究も盛んに行われている。

これまでに SOM のネットワーク構造や学習アルゴリズムの拡張をした手法が多く提案されている。例として適応部分空間 SOM⁽²⁾、階層型 SOM⁽³⁾などが挙げられる。その中の一つとし

徳永憲洋 正員 水産大学校

E-mail tokunaga@fish-u.ac.jp

Kazuhiro TOKUNAGA, Member (National Fisheries University, 2-7-1 Nagata Honmachi, Shimonoseki-shi, Yamaguchi, 759-6595 Japan).

電子情報通信学会 基礎・境界サイエンス

Fundamentals Review Vol.14 No.2 pp.97-106 2020 年 10 月

©電子情報通信学会 2020

て、著者が提案したモジュラーネットワーク型自己組織化マップ (mnSOM: Modular Network SOM) がある⁽⁴⁾。主成分分析も SOM も扱っている対象はデータベクトルであるが、mnSOM の対象は“モデル”である。すなわち、mnSOM はモデルの類似性をマップとして示すことが可能となる。モデルとは入出力関係を表現する関数やダイナミカルモデル、更には基底ベクトルからなる空間モデルなどであり、mnSOM は様々なモデルを対象として取り扱うことが可能である。mnSOM は SOM を拡張させたものであるが、mnSOM は一般化自己組織化マップとしての枠組みをもつともいえる⁽⁴⁾。また mnSOM はモデル間の類似性をマップに表現するとともにモデル間を内挿するモデルの自己組織的発現も可能となる。mnSOM の応用として、一つは人間や動物などをシステムとして考えた場合の振る舞いや行動モデルの類似性の解析や分類をする場合に利用できると考える。また振る舞いなどを mnSOM であらかじめ分類することでパターン認識や適応制御などへの応用も可能である⁽⁵⁾。ほかにも、ロボットの行動制御などにおいて、センサから入力される時系列データのダイナミクスをモデル化及びマップ化させることで、ダイナミクスのクラス分けと各クラスに対する適切な行動割り当てを自動的に学習するメカニズムも実現可能となる (Ishikawa らが mnSOM と強化学習を階層化させたネットワークでロボットの行動学習に応用させている⁽⁶⁾)。

本稿は mnSOM の基本である理論や学習アルゴリズムについて解説し紹介することを主旨とする。しかし、mnSOM は SOM のネットワーク構造や学習アルゴリズムをベースとしているため、SOM の理解が必要不可欠となる。そのため、まず SOM について解説をした後に mnSOM について解説をする。

2. 自己組織化マップ (SOM: Self-Organizing Map)

SOM は Kohonen が提案したニューラルネットワークの一つであり、多次元ベクトルデータを位相保存したまま低次元空間へ写像する学習アルゴリズムを備えている。これによりデータ間の関係を視覚的に捉えやすい“マップ”として表現することができる。この特徴からデータマイニングツールとして多くの研究課題で利用されている。更に単なるデータマイニングとしてのツールだけでなく画像・音声の解析や認識^{(7)・(8)}、遺伝子解析⁽⁹⁾、医療⁽¹⁰⁾などの幅広い分野で工学的応用がなされている^(注1)。他方で、歴史的には Willshaw と von der Malsburg⁽¹¹⁾ や Amari⁽¹²⁾などに代表される大脳視覚野における自己組織的機能分化の数理モデル化に関する研究の一つのモデルとして SOM が提案されている⁽¹³⁾。このように SOM は工学的側面と脳科学的側面の二つの顔をもっており研究対象として興味深い手法である。

(注1)：ここに挙げたのはごく一部で参考文献 (1) に数多くの文献が紹介されている。

2.1 SOM のネットワーク

SOM はベクトルを表現するニューロン (本稿ではノードと呼ぶ) が一次元あるいは二次元程度の低次元空間に配置された構造をもち、各ノードの位置 (座標) は固定されている (図 1)。本稿ではノードが配置された空間をマップ空間と呼ぶ。マップ空間の形は課題に合わせて適宜設計する必要があるが、一般的には二次元格子状のマップが利用されることが多い。ほかにも円状に形成されたサークル SOM や球状に形成された球状 SOM などが利用されている^{(14)・(15)}。また各ノードは観測空間上の 1 点を表現するベクトルを保持しており、参照ベクトルなどと呼ばれる。ニューラルネットワークの観点から述べると、参照ベクトルがシナプス荷重に相当し、学習によって更新されるものである。

2.2 SOM の学習アルゴリズム

SOM の目的は、与えられた入力ベクトルを多次元ベクトル空間 (観測空間と呼ぶ) からマップ空間に位相を保存したまま写像することである (図 1)。これを実現するための学習アルゴリズムは、大きく分けると、初期化、競合過程、協調過程、適応過程の 4 過程で構成されている。

今、 d 個の特徴からなる I 個の d 次元ベクトル $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id})$ $i = 1, \dots, I$ が与えられているとする。また SOM は K 個のノードから構成されており、 k 番目のノードは参照ベクトル $\mathbf{w}_k$ を保持する。更に各ノードのマップ空間上の座標を $\mathbf{g}_k$ とする。ここで、本説明ではノードは二次元格子状に配置されていることを前提として説明する。

process 0: 初期化

参照ベクトル $\mathbf{w}_k$ に初期値を与える。初期値の与え方は様々に提案されている。通常は乱数での初期化でよいが、実応用では主成分分析を用いた初期化法を取り入れたほうがよい場合もある。また、学習ステップ数 t を $t = 0$ として学習を開始させる。

process 1: 競合過程

各入力データ $\mathbf{x}_i$ に対して、最も距離の近い参照ベクトルをもつノードを勝者ノードとする^(注2)。すなわち与えられている全ての入力データに対して以下の式で勝者ノードを求める。

$$k_i^* = \arg \min_k \|\mathbf{x}_i - \mathbf{w}_k\|^2 \quad \forall i \quad (1)$$

ここで k_i^* は、 i 番目の入力データに対する勝者ノードのインデックスを表す。

(注2)：本稿では距離としてユークリッド距離を用いるが、課題によっては距離の定義を変えたほうがよい場合もある。ユークリッド距離が一般的には用いられるが、課題によってはマンハッタン距離やマハラノビス距離なども用いられている。

図1 観測空間とマップ空間上の SOM ネットワーク SOM ネットワークのノードは観測空間上のベクトル（参照ベクトルという）を保持するとともに、マップ空間上におけるノードの位置（座標）ももつ。ただしマップ空間上のノードの座標は固定されており、学習によって更新されることはない。更新されるのは観測空間上の参照ベクトルだけである。学習によって観測空間からマップ空間への写像が得られる。図内の SOM のネットワークのグラフはノード間の隣接関係を表している。

process 2: 協調過程

各ノードの各データに対する学習量を以下の式で計算する。

$$\phi_{ik} = \exp \left(-\frac{\|\xi_{k_i^*} - \xi_k\|^2}{2\sigma^2} \right) \quad (2)$$

ここで、 $\xi_{k_i^*}$ は i 番目の入力ベクトルに対する勝者ノードのマップ空間上の座標を表す。 ϕ_{ik} の式は近傍関数と呼ばれており、勝者ノードに近いノードほど学習量の配分が多くなる。また式中の σ は近傍関数の範囲を決めるパラメータである。 σ は学習ステップ数とともに単調減少する。 σ の定義は課題に合わせて定義する。本稿では以下の式で定義される単調減少関数を用いる。

$$\sigma = \sigma_\infty + (\sigma_0 - \sigma_\infty) \exp \left(-\frac{t}{\tau} \right) \quad (3)$$

ここで、 $t = 0$ 時の近傍関数の広さを σ_0 、 $t = \infty$ としたときの近傍関数が収束する広さを σ_∞ 、 τ を単調減少関数の時定数とする。これらのパラメータは学習初期に、ユーザが課題に合わせて適切に設定する。

process 3: 適応過程

全参照ベクトルを入力ベクトルの重みつき平均となるように以下の式で更新を行う。

$$\mathbf{w}_k(t+1) = \frac{\sum_{i=1}^I \phi_{ik} \mathbf{x}_i}{\sum_{i'=1}^{I'} \phi_{i'k}} \quad (4)$$

更新を行った後に、 t を 1 増加させ process 1 に戻る。以上の過程を十分に繰り返す。目安としては t が τ の 5 倍以上となるまで繰り返すとよい。

2.3 一括学習と逐次学習

上記で説明した学習アルゴリズムは全入力データを一括して学習に用いているため、一括学習法（バッチ学習法）と呼ばれ

ている。既に十分にデータが観測されている状況であれば一括学習法を用いたほうが生成されるマップ（学習結果）に一貫性が保たれる。しかし課題によっては、逐次的にデータが観測される場合もある。その場合には逐次学習法を用いる。逐次学習法では、一つのデータに対して、勝者ノード、学習量がそれぞれ式 (1), (2) によって求められる。そして適応過程の式 (4) が以下のように置き換わる。

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \lambda(t) \phi_{ik} \|\mathbf{x}_i - \mathbf{w}_k(t)\| \quad (5)$$

上記の式において $\lambda(t)$ はマップの生成を固定化させるためのスケジューリング関数である。単調減少関数が用いられる。ただ、設計段階や検証段階の際は、 $\lambda(t) = 1$ としても差し支えない。

さらに一括学習と逐次学習を折衷したセミバッチ学習法もある。セミバッチ学習法は入力データが非常に多い場合に計算コストを削減するテクニックでもある。入力データの中から $I > I'$ となる I' 個のデータをランダムに抽出し、抽出したデータに対して一括学習と同様に競合過程、協調過程を行う。そして適応過程において、以下の式で参照ベクトルを更新する。

$$\mathbf{w}_{kd}(t+1) = (1 - \epsilon) \mathbf{w}_{kd}(t) + \epsilon \sum_{i'=1}^{I'} \phi_{i'k} \mathbf{x}_{i'd} \quad (6)$$

ϵ は $0 < \epsilon < 1$ で与えられるパラメータである。

以降はデータをランダムに選びなおし学習を繰り返す。

2.4 SOM を用いたマップ生成

シミュレーションとして 2 種類のデータを学習データとして SOM に学習させた。一つは三次元空間上に S 字状に分布するデータ、二つめは MNIST の手書き文字データ⁽¹⁶⁾である。以下、それぞれで生成されたマップについて説明をする。

2.4.1 S 字状に分布する人工データ

本シミュレーションでは、図 2 内の S 字状に分布したデータ

(青点)を学習データとして用いた。データは500点の三次元ベクトルである。

SOMは 15×15 の二次元格子状に参照ベクトルユニットを配置したネットワークを用いた。 $\sigma_0 = 15$, $\sigma_\infty = 0.5$, $\tau = 300$ とした。学習法はバッチ学習法を用い、 $t = 1600$ となるまで学習を行った。

図2に、 $t = 0$, $t = 200$, $t = 600$, $t = 1600$ 時における観測空間上でのSOMの学習過程を示す。赤色の点がSOMのノードを示し、ノード間を結ぶエッジはノード間の隣接関係を表す。 $t = 0$ 時には参照ベクトル $\mathbf{w}_k$ を一様乱数 $[-1, 1]$ で初期化しているためネットワークが観測空間上の原点付近に集まっている。 $t = 200$ 時にはSOMの格子状のネットワークがきれいに広がっているのが分かる。 $t = 600$ 時にはマップがS字状に生成され、最終的($t = 1600$)にはデータにフィットする形でネットワークが生成された。

この課題のようにデータが観測空間上で曲がって分布している場合は主成分分析ではデータの本質を見出すことが難しくなる。SOMは競合・協調作用によってデータの位相を保存するような写像を実現するため、曲がったデータにフィットするマップが生成される。この性質からSOMは多様体学習でも利用されている。

2.4.2 MNISTの手書き文字データ

MNISTの手書き文字データを学習データとして用いた。本データは様々な人が書いた0~9までの数字を画像化したデータである。1枚の画像は $28 \times 28 = 784$ (pix)であり、1ピクセルごとに0~255の輝度値が格納されている。MNISTの学習データは60000個の手書き数字の画像が用意されているので、本シミュレーションでは784次元のベクトルデータ60000個を入力データとして利用した。

SOMは 20×20 の二次元格子状に参照ベクトルユニットを配置したネットワークを用いた。 $\sigma_0 = 21$, $\sigma_\infty = 0.5$, $\tau = 2000$ とした。また学習データが多いうえに次元が大きいことからセミバッチ学習法を用い、 $t = 20000$ となるまで学習を行った。

図3に $t = 20000$ 時の結果を示す。図はマップ空間を示し、各格子が参照ベクトルユニットに対応する。また各格子に描かれる画像は対応する参照ベクトルユニットが学習の結果、獲得した参照ベクトルを画像化したものである。

結果から、各参照ベクトルが様々な手書き文字を表現していることが分かる。また近い参照ベクトルユニットは似た形の数字を表現しており、マップ全体でみると数字の形が連続的に変化している。このようにマップ空間上で参照ベクトルユニットを確認することでデータ間の関係性を視覚的に捉えることが可能となる。

以上、二つのシミュレーションのように、SOMは多様体学習としての特徴ももち、なおかつ視覚化のツールとしての機能ももっており、使い次第で様々な応用できる万能なツールだと考える。

3. mnSOMについて

SOMは観測空間上のデータベクトルを対象として扱う手法である。一方で、課題によってはベクトル単体の類似性ではなく、データベクトルを生成しているモデルの類似性を調べたほうがよいこともある。例えば機械の振る舞いや動物の行動などを分類あるいは類似性を見たいと考えたとき、それらは単一のデータベクトルではなく、一連の入力に対する振る舞い(出力)や時系列データをモデルとして現したうえで取り扱ったほうがよい場合もある。このような場合にmnSOMが利用できる。

mnSOMはSOMにおける参照ベクトルユニットをニューラルネットワークなどの“モジュール”に置き換えたモジュラーネットワークの構造をもち、各モジュールの学習にはSOMの競合・協調学習が取り入れられている。これにより、mnSOMはベクトル空間ではなくモジュールが表現するモデルの空間におけるマップを生成する。このことからmnSOMは単にSOMを拡張したものではなく、様々なデータタイプに適用できるような一般化したSOMを実現するといえる。

本節ではmnSOMの枠組みと学習アルゴリズムについて以下に説明する。なお、mnSOMのアーキテクチャや学習アルゴリズムなどの詳細については参考文献⁽⁴⁾, ⁽¹⁷⁾, ⁽¹⁸⁾を参照していただきたい。

3.1 mnSOMの枠組み

まず具体的にmnSOMの枠組みについて説明する。なお、本稿ではモジュールを多層パーセプトロン(MLP: Multi-Layer Perceptron)で構成したMLP-mnSOMを基本として説明する。

mnSOMは図4に示すように、 K 個のモジュールで構成され、モジュールは一、二次元の格子状に配置される。一方、学習に用いるデータは I 個のシステム(関数)からそれぞれ複数に観測された入出力データペア群の集合 $\mathcal{X} = \{X_i\}$, $X_i = \{(\mathbf{x}_{ij}, \mathbf{y}_{ij})\}$ である($\mathbf{x}$ を入力、 $\mathbf{y}$ を出力とする)。本稿では観測された一連のデータ集合 $\mathcal{X}$ をエピソードと呼ぶ。mnSOMにはサンプリングされたエピソード $\mathcal{X}$ だけが与えられ、入出力モデル(関数形)は未知とする。この状況でmnSOMは、以下を目的とし動作する。

- (1) I 個のデータ集合 X_i に対してそれぞれ勝者モジュールを求めるとともに元となるモデルを学習により獲得・再現する。モデルの獲得は教師あり学習で行う。
- (2) 勝者以外のモジュールは学習によって勝者モジュール間を補間する中間的なモデルを獲得し、全体のモジュールではモデルの自己組織化マップが生成される。マップの生成は教師なし学習によって生成される。またモデル同士の類似・相違度を表す指標として関数空間における距離を用いる。

ここで関数空間上のモデル間の距離は以下の式で定義される。

$$L^2(f, g) = \int \|\mathbf{f}(\mathbf{x}) - \mathbf{g}(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x} \quad (7)$$

図 2 SOM の学習過程 青点は学習データで、観測空間上で S 字カーブ上に分布する．赤色で描かれたグラフは SOM のネットワークを示す．また学習ステップ数が $t = 0$, $t = 200$, $t = 600$, $t = 1600$ 時の SOM ネットワークを示す．

図 3 MNIST の手書き文字を SOM に学習させた結果 各格子が参照ベクトルユニットに対応する．格子内の画像は対応するユニットが学習の結果で獲得した参照ベクトルを画像として表示したものである．マップ上の左上のユニットは数字の“1”，左下は“3”のように各ユニットは数字を表現し，マップ空間上では似た形の数字が近くにマップされるように各ユニットは様々な数字の形を表現している．

図4 mnSOMのネットワーク構造 mnSOMはSOMの参照ベクトルユニットをニューラルネットワークなどのモジュールに置き換えたモジュラーネットワークの構造をもつ。学習アルゴリズムではSOMの協調・競合過程を導入する。

$g(\mathbf{x})$ 及び $f(\mathbf{x})$ は関数空間上のモデル（入出力関数）を表し、 $p(\mathbf{x})$ は $\mathbf{x}$ の確率密度関数である。ここで、 $p(\mathbf{x})$ は与えられる全てのシステムで同じである必要があり、異なる場合は上式を定義できない。ただし異なる場合は、 $\mathbf{x}$ の平均が0、分散が1になるように事前にデータを標準化するなど前処理を施すことで対応が可能となる。

4. mnSOMの学習アルゴリズム

mnSOMの学習アルゴリズムはSOMと同じように、初期化、競合過程、協調過程、適応過程の四つの過程で構成されている。尚、学習ステップ数を t とし、学習の開始時は $t=0$ であり、一通り処理を終えた後に t を1増加させる。

また数式の表記において、変数の添え字が煩雑になることを避けるため、便宜上本稿ではモジュールのインデックス k は上付きにする。例えば、 ξ^k はマップ空間上の k 番めのモジュールの座標を表す。

process 0: 初期化

$t=0$ 時に、各モジュールの結合荷重 $\mathbf{w}$ は一様乱数などを利用してランダムに初期化しておく。

process 1: 競合過程

まず各モジュールの表現するモデル $g^k(\cdot)$ と全システムが本来もつモデル $f_i(\cdot)$ との距離を式(7)で評価する。ただし、実際には各システムのモデル $f_i(\cdot)$ は未知のため、各システムから得られているエピソード X_i を利用し、近似的に距離 $L^2(f_i, g^k)$ を評価する。具体的には、 X_i における一つの入出力データベクトル $\mathbf{x}_{ij}$ を全モジュールに与え、各モジュールの出力 $\hat{\mathbf{y}}_{ij}^k$ を求める。そして教師信号 $\mathbf{y}_{ij}$ との誤差 e_{ij}^k を計算する。この過程を全システム、全ベクトルに対して行い、 e_{ij}^k のアンサンブル平均 E_i^k で式(7)を近似的に評価する。式で定義すると以下のようになる。

$$L^2(f_i, g^k) \simeq E_i^k = \frac{1}{J} \sum_{j=1}^J e_{ij}^k = \frac{1}{J} \sum_{j=1}^J \|\mathbf{y}_{ij} - \hat{\mathbf{y}}_{ij}^k\|^2 \quad (8)$$

評価後に全システムのモデルに対してそれぞれ勝者モジュールを決める。つまり次式に示すように i 番めのモデルに対して最も評価誤差 E_i^k が小さかったモジュールを勝者モジュール k_i^* とする。

$$k_i^* = \arg \min_k E_i^k \quad (9)$$

process 2: 協調過程

SOMと同じように式(2)の近傍関数で i 番めのシステムのエピソードに対する k 番めのモジュールの学習量 $\phi_i^k(t)$ を求める。

process 3: 適応過程

全てのモジュールを次式のように誤差逆伝搬法によって学習させる。

$$\Delta \mathbf{w}^k = -\eta \sum_{i=1}^I \Psi_i^k \frac{\partial E_i^k}{\partial \mathbf{w}^k} = -\eta \frac{\partial E^k}{\partial \mathbf{w}^k} \quad (10)$$

$$\Psi_i^k = \frac{\phi_i^k(t)}{\sum_{i'} \phi_{i'}^k(t)} \quad (11)$$

ここで、 $\mathbf{w}^k$ は k 番めのモジュールの結合荷重を表す。 η は誤差逆伝搬法の学習率である。また $E^k = \sum_i \Psi_i^k E_i^k$ である。これは、各システムのモデル $\{f_i\}$ の関数空間上の重みつき内分点となるモデル、すなわち $g^k = \sum_{i=1}^I \Psi_i^k f_i(\mathbf{x})$ を近似的に誤差 E^k で表現していることに相当する。

式(10)による $\mathbf{w}^k$ の更新は Ψ_i^k を固定したまま十分な回数を繰り返す。ただし、繰り返し数が多くなると学習時間に影響するため、課題の難しさに合わせて、適切に設定する必要がある。

以後は process 1 に戻り、学習を繰り返す。上記のように mnSOM は基本的な学習メカニズムは SOM と同じである。また上記のアルゴリズムそのままでは計算コストの問題などがあり実課題には適さない。そのため mnSOM を実装する際の幾つかのテクニックを次の節で説明する。

5. mnSOMを実装する際のテクニック

これまでに書かれた mnSOM の学習アルゴリズムをプログラ

ミングし、実際に動かすと幾つかの問題が生じる。ここでは問題を解決するテクニックについて説明する。

5.1 隣接モジュールコピー法によるローカルミニマム回避

mnSOM が最急降下法などを学習で用いるモジュールで構成されている場合はローカルミニマムに陥るモジュールが高い確率で出現する。ローカルミニマム問題が生じれば、マップの連続性、トポロジー保存性が失われるため、mnSOM を実課題で運用するには重大な問題となる。これを回避する方法が、下記に説明する“隣接モジュールコピー法”である。

mnSOM における各モジュールのタスクは誤差関数 $E^k = \sum_i \Psi_i^k E_i^k$ を最小化することであり、この式を利用することで、モジュールがローカルミニマムに陥っているかどうかを判断することが可能になる。

今、 k 番めのモジュールの各システムに対する誤差関数を E_i^k 、 k 番めのモジュールの隣接モジュールの誤差関数を $E_i^{k'}$ とし、 E^k と $E_i^{k'}$ とを一時的に置き換えた場合の総誤差関数 $\tilde{E}^{k'}$ を以下の式で計算する。

$$E^k = \sum_i \Psi_i^k E_i^k \quad (12)$$

$$\tilde{E}^{k'} = \sum_i \Psi_i^k E_i^{k'} \quad (13)$$

もし置き換えて総誤差が改善するならば、そのモジュールはローカルミニマムに陥っている可能性が高いため、隣接モジュールの結合荷重をコピーすることで、ローカルミニマムを回避することができる。これを定式化すると次式になる。

$$\mathbf{w}^k(t+1) = \mathbf{w}^{k'}(t) \quad \text{if } \tilde{E}^{k'} < \beta E^k \quad (14)$$

β は安全項（通常 0.8~0.9 周辺の値をとる）である。上記のコピー法を、適応過程の前に隣接モジュール間で行うことで、ローカルミニマム問題を防ぐことが可能である。また本コピー法は、毎学習ステップで行う必要はない。 σ の時定数 τ を考慮しながら、コピー法を行うステップ数を設定すればよい。

5.2 簡易評価適応過程による計算速度向上

純粋な学習アルゴリズムで計算をさせると、並列計算のプログラム開発を行わない限り、課題によっては計算に時間がかかる。理由は競合過程における距離評価及び適応過程において全システム、全データを式 (8) で評価、学習をさせるためである。

この問題を解決する一つの方法として、本稿では E_i^k を逐次平均法で近似的に求めるテクニックを紹介する。

まず競合過程における距離評価において、ランダムにシステムのインデックス i 及びデータのインデックス j を選択する。選択されたデータベクトルのペア $(\mathbf{x}_{ij}, \mathbf{y}_{ij})$ について、以下を計算し E^k を近似的に求める。

$$E_i^k(t) = (1 - \gamma)E_i^k(t-1) + \gamma e_{ij}^k \quad (15)$$

γ は $0 < \gamma < 1.0$ で設定されるパラメータである。その後、選ばれた $(\mathbf{x}_{ij}, \mathbf{y}_{ij})$ について全モジュールを学習させる。例えばモジュールが多層パーセプトロンの場合は誤差逆伝搬法を用いて学習を行う。以上のデータのランダム選択及び式 (15) で表される評価及び学習を数十から数百回繰り返す。

上記の評価、適応過程を本稿では便宜上、簡易評価適応過程と呼ぶ。簡易評価適応過程は $t > 0$ について行い、 $t = 0$ についてはネットワークや E_i^k の初期化の意味も含めて通常の学習過程を利用する。 E_i^k は $t = 0$ 時の初期化以外は全学習を通して値を更新し続ける。

この方法の利点は、計算速度が学習データの数に依存せず、学習スピードを向上させられることである。簡易評価適応過程内のループ回数はシステムの数、データの数 considering して設定する。

5.3 オンライン学習

これまで説明してきた mnSOM の学習アルゴリズムは、既存の複数のシステムからそれぞれデータ集合 $\mathcal{X}$ が得られていることを前提とし、それを一括で学習する一括学習形のアルゴリズムである。

しかし課題によっては逐次的に複数のシステム（数もモデルも未知）から入出力データのペアの集合 $X(t) = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ が与えられることも考えられる。この場合は mnSOM の学習アルゴリズムをオンライン学習形に置き換えることで対応が可能である。すなわち適応過程における式 (11) を以下の式に置き換える。

$$\Psi_i^k = \frac{\phi_i^k(t)}{S^k(t)} \quad (16)$$

$$S^k(t) = (1 - \epsilon)S^k(t-1) + \epsilon \phi_i^k(t) \quad (17)$$

ϵ は $0 < \epsilon < 1.0$ で設定されるパラメータである。 $S^k(t) = 1$ としてもマップの生成は可能である。ただし、SOM と同じように一般的なオンライン学習を行うと、データの学習順序やパラメータによって、生成されたマップに一貫性が保たれなくなる。ゆえに、たとえ近似的な方法であっても近傍関数を正規化する $S^k(t)$ を入れたほうが安定性は向上する。

5.4 パラメータの設定について

mnSOM はモジュールのパラメータだけでなく mnSOM の学習パラメータもあり、多くのパラメータを設定しなければならない。パラメータの設定の指針としては、まず 1 モジュールでのモデル生成をしっかりと検証し、モジュールに関わる学習パラメータを先に固定することである。MLP などの場合には、中間層ユニットの数についても過学習にならないよう最適なネットワーク構成を調査しておくといよい。その後、mnSOM の学習パラメータを決めていく。1 モジュールでの検証をもとに、どのぐらいの学習ステップでモデルを生成できるのか、誤差曲線の減少の様子、などをもとに、mnSOM の学習パラメータである τ や簡易評価適応過程のループ回数を決めるといよい。近傍関数の広さについては σ_0 はマップサイズと同程度（例えば 10×10 ）

のマップサイズであれば $\sigma_0 = 10$ 程度) が経験上よいと考える。 σ_∞ は、マップサイズと学習するシステムの数 I から決める。例えば 10×10 のマップサイズで 6 個のシステムを学習するのであれば $\sigma_\infty = 2.0 \sim 2.5$ 程度が経験上よいと考える。すなわち互いの勝者モジュールが影響を及ぼさないかの境界になるよう設定しておくといよい。 σ_∞ が小さすぎると、連続したマップが崩れるので注意が必要である。

5.5 過学習モジュールについて

一つのモジュールが複数のシステムの間関数を包括するようなモデルを表現することがしばしばある。例えば多層パーセプトロンの中間層ユニットを多くすると、過学習が生じ、複雑なモデルを表現できるようになる。このようなモジュールを mnSOM のモジュールに組み込むと、多数のシステムに対して一つのモジュールが勝者モジュールになり (一人勝ちの状態)、マップの生成に支障が出る。そのため、1 モジュールでの検証で、できるだけコンパクトなモジュールになるようにしたほうがよい。

また一つのモジュールが複数のシステムに対して勝者にならないよう、勝者になったモジュールはほかのエピソードで勝者にならないようプログラムコーディングの際に回避策を設けるとよい (ただしシステム数がモジュール数より多い場合は除く)。

6. mnSOM のシミュレーション

本節では多項式関数を用いて mnSOM のマップ生成について解説をする。

6.1 シミュレーションの枠組み

シミュレーションでは多項式関数として次式で示すルジャンドル多項式で構成された三次関数を用いる。

$$y_{ij} = a_{i1}p_1(x_{ij}) + a_{i2}p_2(x_{ij}) + a_{i3}p_3(x_{ij}) \quad (18)$$

$$p_1(x_{ij}) = x_{ij}$$

$$p_2(x_{ij}) = \frac{1}{2}(3x_{ij}^2 - 1)$$

$$p_3(x_{ij}) = \frac{1}{2}(5x_{ij}^3 - 3x_{ij})$$

上式の a_{i1} , a_{i2} , a_{i3} を様々に設定することで異なる入出力モデルを実現する。本シミュレーションでは以下の二つのケースについて行う。

ケース 1: システムの数が少なく、各システムからの得られている入出力データが多い場合

ケース 2: システムの数は多いがシステムから得られる入出力データが少ない場合

6.1.1 ケース 1 のシミュレーション

本シミュレーションでは式 (18) の三次関数から得られた 6 個のエピソードを用いる。各エピソードの分布を図 5 に示す。また各エピソードのパラメータを表 1 に示す。エピソード内のデー

図 5 ケース 1 のシミュレーションで用いた 6 エピソード

表 1 各エピソードのパラメータ

	a_{i1}	a_{i2}	a_{i3}
X_1	0.4	0	0
X_2	0.4	0.4	0.4
X_3	0	0	0.4
X_4	-0.4	0	0
X_5	-0.4	-0.4	-0.4
X_6	0	0	-0.4

タ点数は 100 個とし、 x_{ij} は一様乱数 $[-1, 1]$ とした。更に y_{ij} には正規分布 $\mathcal{N}(0, 0.04)$ に従ったノイズを加えた。

一方、mnSOM は 10×10 の二次元格子状に MLP モジュールを配置した。MLP は入力、中間、出力層の 3 層からなりそれぞれ 1, 8, 1 個のユニットで構成した。MLP の誤差逆伝搬法の学習率 η は 0.1 とした。また mnSOM は隣接モジュールコピー法と簡易評価適応過程を実装した学習アルゴリズムを用いた。学習パラメータは $\sigma_0 = 10$, $\sigma_\infty = 2.5$, $\tau = 2000$, $\beta = 0.85$, $\gamma = 0.005$ とした。簡易評価適応過程内のループ数は 100 回とした。mnSOM の学習は 10000 ステップまで行った。

図 6 に結果を示す。各格子はモジュールに対応し、格子内の曲線は対応するモジュールが獲得したモデルが描かれている。モデルはモジュールの $[-1, 1]$ の入力をモジュールに与えた際の出力をもとに描かれている。また赤い格子は mnSOM に与えた 6 個のエピソードそれぞれに対する勝者モジュールを示す。結果から、各モジュールが様々なモデルを獲得した。また、隣り合うモジュールが類似したモデルを生成し、マップ全体ではモデルが連続的に変化している。mnSOM には 6 個のモデルのエピソードしか与えていないが、その 6 個のモデルを内挿するように各モジュールがモデルを自己組織的に獲得した。

6.1.2 ケース 2 のシミュレーション

本シミュレーションでは式 (18) の三次関数から作られた 100 個のモデルを利用する。各モデルの関数形を図 7 に示す。モデルの数は多いが、各モデルからは 4 点だけのデータしか観測されていないとする。図 8 にエピソードの例を示す。図のように 4 点だけでは元のモデルを推定することは困難である。この状況下で mnSOM にエピソードを学習させる。すなわち、多数のエピソードがあるが各エピソード内のデータは少数しか観測されていない、という場合に mnSOM が各エピソードの本来のモデルを正しく推定し、更にマップも生成できるかを検証するこ

図 6 ケース 1 のシミュレーション結果 各格子はモジュールに対応している。格子内の曲線は対応する MLP モジュールが獲得したモデルを示す。また赤色の格子は枠内に書かれているエピソードの勝者モジュールを示す。

とが本シミュレーションの目的である。

mnSOM の構成とパラメータについてはほとんどケース 1 と同じである。ケース 1 と異なるパラメータは σ_∞ と簡易評価適応過程のループ数であり、 $\sigma_\infty = 0.5$ 、簡易評価適応過程内のループ数は 1000 回とした。 σ_∞ をケース 1 とケース 2 で異なる値に設定した理由は、マップサイズに対するエピソード数が異なるためである。ケース 1 ではマップのサイズに比べてエピソード数が少ないので、多くのモジュールがエピソードの影響を受けるように σ_∞ を大きく設定した。一方でケース 2 ではマップのサイズに対してエピソード数が大きいため $\sigma_\infty = 0.5$ のように小さい値に設定した。またループ数については、ケース 2 はエピソード数は多いかわりにデータ数が少ないのであるべく多くのエピソードのデータに学習の機会を与えるためにループ数を大きく設定した。

結果を図 9 に示す。図の見方はケース 1 と同じである。結果から、各モジュールはもととなるモデル (図 7) を少ないデータから推定でき、なおかつマップの生成もできている。図 8 のデータのように一つ一つのエピソードを個別にみるとデータの生成元であるモデルを正しく把握することは困難であるが、mnSOM ではモジュール間の協調作用によって本来のモデルを推定することが可能になると考えられる。

7. おわりに

本稿では、SOM の解説も交えながら mnSOM の基本を解説した。mnSOM は SOM の機能を一般化したものであり、多様な課題へ応用することも可能だと考える。

mnSOM の発展や応用としては、例えば Minatohara らは制御器と予測器のペアで構成されるモジュールをベースとする mnSOM (Minatohara らは SOAC: Self-Organizing Adaptive

図 7 ケース 2 のシミュレーションで用いた 100 個のモデル

図 8 ケース 2 のシミュレーションで用いたエピソードの例 1 エピソード内のデータ数は 4 点でありモデルの推定が困難である。

図 9 ケース 2 のシミュレーション結果 各格子はモジュールに対応している。格子内の曲線は対応する MLP モジュールが獲得したモデルを示す。

Controller と呼んでいる) を提案し適応制御に応用させている⁽⁵⁾。SOAC は制御対象のパラメータが変わっても瞬時に適応的に制御し、なおかつ、学習をしていないパラメータの制御対象 (学習

した制御対象の中間的パラメータの制御対象)に対しても適応的に制御することが可能である。また、古川はモジュールにSOMをのものを利用した高階形自己組織化マップの提案と理論の導出をするとともに形状認識などへの応用もしている⁽¹⁹⁾、⁽²⁰⁾。また動的モデルのマップ生成と応用もなされている⁽²¹⁾。

mnSOMはケース2の実験のように協調作用によって各モジュールがお互い助け合うことで少ない学習データで多くのモデルを表現できる。この特徴を活かせるようなパターン認識や適応制御の課題に今後取り組む予定である。本稿によって読者の皆様がSOMやmnSOMへ興味をもって頂くとともに、学習メカニズムの理解への手助けになれば幸いである。

謝辞 mnSOMの体系化については多くの助言と協力を頂いた九州工業大学大学院生命体工学研究科教授の古川先生に深く御礼申し上げる。

文 献

- (1) T. Kohonen, Self-organizing Maps, Springer-Verlag, 2001.
- (2) T. Kohonen, S. Kaski, and H. Lappalainen, "Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM," Neural Computation, vol. 9, no. 6, pp. 1321–1344, 1997.
- (3) J. Lampinen and E. Oja, "Clustering properties of hierarchical self-organizing maps," Journal of Mathematical Imaging and Vision, vol. 2, pp. 261–272, 1992.
- (4) K. Tokunaga and T. Furukawa, "Modular network SOM," Neural Networks, vol. 22, no. 1, pp. 82–90, 2009.
- (5) T. Minatohara and T. Furukawa, "The self-organizing adaptive controller," International Journal of Innovative Computing, Information and Control, vol. 7, no. 4, pp. 1933–1947, 2011.
- (6) M. Ishikawa and K. Ueno, "Hierarchical architecture with modular network SOM and modular reinforcement learning," ICANN '09: Proceedings of the 19th International Conference on Artificial Neural Networks: Part I, pp. 546–556, Sept. 2009.
- (7) M. Haritopoulos, H. Yin, and N.M. Allinson, "Image denoising using self-organizing map-based nonlinear independent component analysis," Neural Networks, vol. 15, no. 8, pp. 1085–1098, 2002.
- (8) M. Kurimo, "Thematic indexing of spoken documents by using self-organizing maps," Speech Communication, vol. 38, no. 1, pp. 29–45, 2002.
- (9) T. Abe, S. Kanaya, M. Kinouchi, Y. Ichiba, T. Kozuki, and T. Ikemura, "Informatics for unvailing hidden genome signatures," Genome Research, vol. 13, pp. 693–702, 2003.
- (10) M.K. Markey, J.Y. Lo, G.D. Tourassi, and C.E. Floyd, "Self-organizing map for cluster analysis of a breast cancer database," Artificial Intelligence in Medicine, vol. 27, no. 2, pp. 113–127, 2003.
- (11) D.J. Willshaw and C.V.D. Malsburg, "How patterned neural connections can be set up by self-organization," Proceedings of the Royal Society of London B, vol. 194, pp. 431–445, 1976.
- (12) S. Amari, "Topographic organization of nerve fields," Bulletin of Mathematical Biology, vol. 42, no. 3, pp. 339–364, 1980.
- (13) T. Kohonen, "Self-organized formation of topologically correct feature maps," Biological Cybernetics, vol. 43, pp. 59–69, 1982.
- (14) M. Kiang, U. Kulkarni, K. Goul, A. Philippakis, R. Chi, and E. Turban, "Improving the effectiveness of self-organizing map networks using a circular Kohonen

- layer," Proceedings of the Hawaii International Conference on System Sciences, pp. 521–529, Dec. 1997.
- (15) Y. Wu and M. Takatsuka, "Spherical self-organizing map using efficient indexed geodesic data structure," Neural Networks, vol. 19, no. 6, pp. 900–910, 2006.
- (16) Y.L. Cun, C. Cortes, and C.J.C. Burges, "The MNIST database of handwritten digits," <http://yann.lecun.com/exdb/mnist/>
- (17) 徳永憲洋, 肝付謙二, 安井湘三, 古川徹生, "関数空間型 SOM," 日本神経回路学会誌, vol. 12, no. 1, pp. 39–51, 2005.
- (18) 徳永憲洋, 古川徹生, "一般化自己組織化マップ:教師あり学習と教師なし学習の融合," 電子情報通信学会技術研究報告 NC, ニューロコンピューティング, vol. 106, no. 102, pp. 75–80, 2006.
- (19) T. Furukawa, "SOM of SOMs," Neural Networks, vol. 22, no. 4, pp. 463–478, 2009.
- (20) 薬師寺翔, 古川徹生, "高階化 SOM による形状表現マップ," 知能と情報, vol. 24, no. 2, pp. 648–659, 2012.
- (21) 大久保貴之, 辻純一, 徳永憲洋, 古川徹生, "力学系群の自己組織化アルゴリズム: Parametric bias 法と mnSOM によるアプローチ," 電子情報通信学会技術研究報告 NC, ニューロコンピューティング, vol. 108, no. 480, pp. 381–386, 2009.

(SIS研究会提案, 2020年6月5日受付,
2020年7月20日再受付)

徳永憲洋 (正員)

2001 九州工業大学情報工学部制御システム工学科卒業。2003 同大学大学院生命体工学研究科博士前期課程修了。2006 同大学大学院生命体工学研究科博士後期課程修了。博士(工学)。同年, 同大学研究員。2011 一般財団法人ファジィシステム研究所主任研究員。2013 水産大学校海洋機械工学科助教。2016 より同校同学科講師。脳型情報処理モデルの開発やソフトコンピューティングの応用に関する研究に従事。日本機械学会, 水産工学会各会員。