VA-SOM：ベクトル近似法を用いた自己組織化マップの高速化手法

出木原　裕　順

（受付　2017 年 10 月 30 日）

1. は　じ　め　に

近年，機械学習の技術革新に伴い人工知能の分野が注目されている［1, 2］。特に，囲碁のような従来コンピュータには困難であったゲームや，画像認識，音声認識などの一部の処理においては，人工知能が人間と同等レベルやそれ以上の結果を示すようになってきている［3, 4］。ビッグデータやセンサネットワークの普及に伴い，今後，人工知能の需要はますます増加するものと言える。

本稿では，人工知能を実現する主要なアプローチの一つであるニューラルネットワークに着目し，ニューラルネットワークの一種である自己組織化マップ（Self- Organizing Map: SOM）［5］を高速化する方式を提案する。提案法では，VA-File［6］で提案されているようなベクトル近似法の概念を用いて，SOM を改良した新しい手法 VA-SOM を提案する。VA-SOM によって，高次元のデータを対象とした機械学習の高速化が実現できることにより，経済活動や社会現象などを表した複雑なデータの機械学習や，パターン認識，判別分析，行動解析などを高速に実施できるようになると期待できる。

これまでに，様々な分野に SOM［7–13］を応用したシステムが数多く提案されている。その中でも，索引層や学習初期段階の省略などを用いて SOM の学習高速化を図った改良法もいくつか提案されている［14–17］。これらのシステムは，オリジナルの SOM の学習プロセスを簡素化・省略化して学習の高速化を実現している。しかし，その結果として，オリジナルの SOM とは学習プロセスが異なるものもある。本稿では，オリジナルの SOM と全く同じ学習プロセスを実現しながら，高次元データの学習を高速化する新しい手法である VA-SOM を提案する。すなわち，VA-SOM では，アルゴリズムにおける冗長な処理を削減することで，学習処理の高速化を実現しており，オリジナルの SOM と同様の学習プロセスを実現することから，他の SOM の改良方式との併用が可能な，極めて汎用性の高い高速化手法の実現が期待できる。VA-SOM では，画像などの多次元ベクトルから特徴抽出する際に用いられるベクトル近似法の概念［6］を SOM の学習に適用し，多次元ベクトルの情報をベクトル近似値に圧縮して処理することで学習プロセスの高速化を図っている。

以下，第 2 章において自己組織化マップ SOM について述べる。第 3 章では，ベクトル近似法を用いた提案法について解説する。また，第 4 章で数値実験結果を示して提案法の有用性を検討する。最後に，第 5 章において本稿をまとめる。

2.　自己組織化マップ（SOM）

コホネンの自己組織化マップ（Self-Organizing Map: SOM）について，その概要および学習アルゴリズムについて述べる。学習アルゴリズムの詳細については，文献［5］に詳しい。

2.1　自己組織化マップ（SOM）

自己組織化マップは，コホネンが，脳内で行われている（と予想される）自己連想記憶モデルの研究に取り組んでいる中で，その自己連想のアルゴリズムを簡略化し，利便性の高いものとして提案した教師なしニューラルネットワークである。また，SOM は，データごとの特徴や関連性などの予備知識がない中で，その分類や特徴抽出が可能な階層型ニューラルネットワークの一種である。SOM の構造は，入力層と出力層の 2 層で構成され（図 1 参照），出力層は競合層とも呼ばれる。出力層では，入力層の入力データ群を SOM のアルゴリズムで分類した結果を視覚的に判別するために，1 次元配列や 2 次元配列の形状でマッピングされる。それぞれ 1 次元 SOM，2 次元 SOM と呼ばれ，図 1 は，M×N の 2 次元 SOM の例である。出力層は，ニューロン u を格子状に配置して構成されており，それぞれの u は，参照ベクトル W を持つ。SOM の大まかなアルゴリズムの流れは以下の通りである。

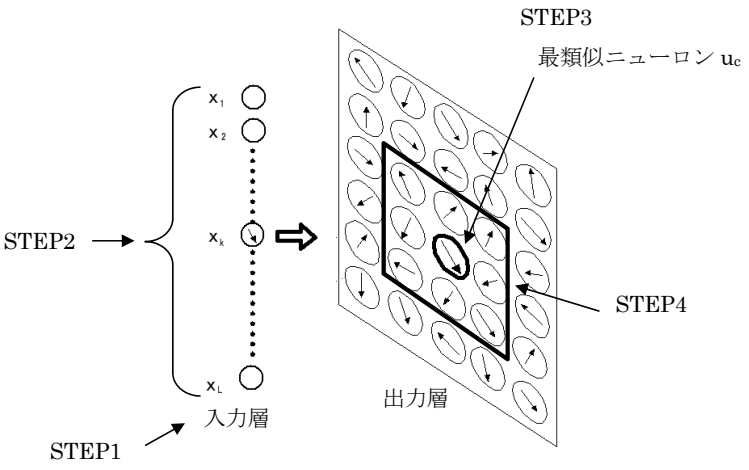


図 1　自己組織化マップの学習の流れ

STEP 1：出力層の初期化
STEP 2：入力データのランダム選択
STEP 3：入力データに最も似ているニューロンを選択
STEP 4：最類似ニューロンを中心に入力ベクトルを学習
STEP 5：STEP 2 から再処理

SOM の学習アルゴリズムは，大脳皮質の神経機能，主に脳の情報処理の仕組みをモデル化しており，その基本的なモデルは次の式（1）で表現されている。

$$W_{ij}(t+1) = W_{ij} + h_{ij}(t)\{x(t) - W_{ij}(t)\} \quad \cdots式（1）$$

なお，i と j はそれぞれ 1 以上の自然数とし，t は 0 以上の自然数とする。h は，近傍関数と呼ばれ，出力層における学習対象となるニューロンの範囲を設定する関数であり，t に関する単調減少関数で t が無限大に近づくと h は 0 に収束していく。学習の大きさの初期値を学習率係数 $\alpha(t)$ で，学習の範囲の初期値を学習範囲係数 $\beta(t)$ で任意に設定する。学習の例を図 2 に示す。最類似ニューロンを中心に学習範囲を学習範囲係数 $\beta(t)$ で決定し，学習範囲に含まれるニューロンに対して，学習率係数 $\alpha(t)$ を基に入力データのニューロン値を反映していく。以上により，学習範囲内にあるニューロンが入力データの値に近づいていく。なお，近傍関数や各係数などの詳細については，文献［5］に詳しい。次節に，SOM のアルゴリズムを示す。

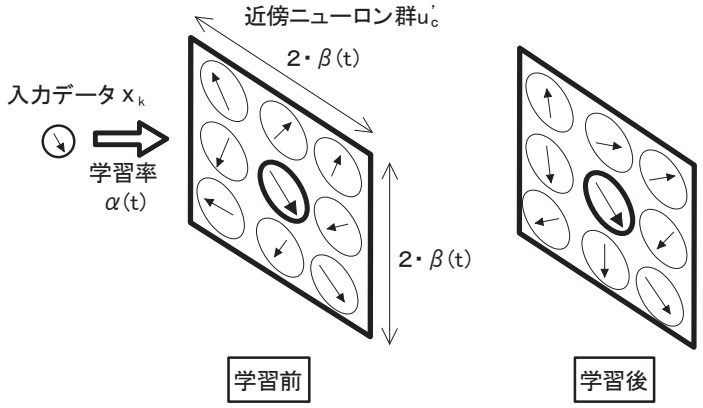


図 2　ニューロンの学習前後の変化

2.2　SOM の学習アルゴリズム

入力層では，n 次元の特徴ベクトルを持つ L 個の実数ベクトル $x = \{x^{(1)}, x^{(2)}, \cdots, x^{(n)}\}$ を入

力データとして与えられ，出力層を M×N 個の格子点にニューロン u を配置した 2 次元 SOM とする。また，u_{ij} は，参照ベクトル $W_{ij}^{(t)} = \{w_{ij}^{(1)}, w_{ij}^{(2)}, \cdots, w_{ij}^{(n)}\}$ を持ち，学習を繰り返す回数は，T とする。なお，t は，任意の学習回数番号を示し，n，M，N，L および T は，任意の自然数とする。

＜オリジナル SOM の学習アルゴリズム＞

STEP 1：繰り返し回数 t = 0 において，出力層の各ニューロンの参照ベクトルを初期化する。初期化では，ランダム初期化と線形初期化のどちらか一方を任意に選択する。

STEP 2：L 個の入力データ群からデータ x_k をランダムに選び出す。

STEP 3：x_k の特徴ベクトルに最も類似した参照ベクトルを持つニューロン u_c を探し出す。u_c には，x_k とのユークリッド距離 $|x_k - W_{ij}^{(t)}|$ を最小にするニューロン u_{ij} が選ばれる。

STEP 4：まず，近傍関数 h より u_c の近傍ニューロン群 u_c’ を求め，次に式（1）を用いて，u_c および u_c’ の参照ベクトルに x_k の特徴ベクトルを学習させる。

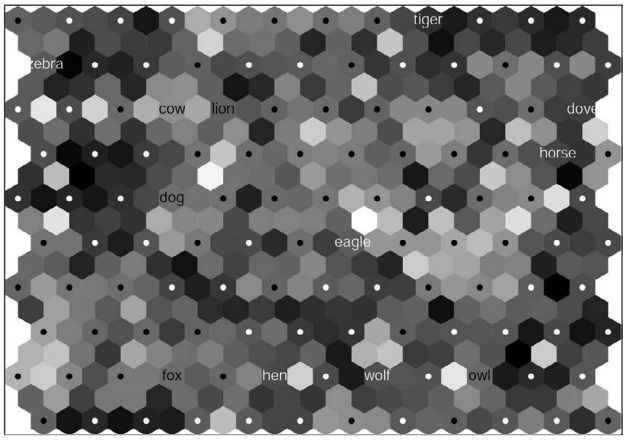
STEP 5：t をカウントしながら，STEP 2 から STEP 4 までを T 回繰り返す

図 1 より，オリジナルの SOM の学習アルゴリズムについて述べる。まず，STEP 1 において，出力層にマッピングされた各ニューロンの参照ベクトルの値を初期化する。初期化では，ランダム初期化と線形初期化の 2 通りある。次に STEP 2 では，入力データ群からランダムに 1 つ入力データを選び出す。そして STEP 3 で，選ばれた入力データの特徴ベクトルと出力層の各ニューロンの参照ベクトルをユークリッド距離に基づいて比較し，最も距離が近いニューロンを探し出す。STEP 4 において，出力層のニューロンを順番に調べ，探索されたニューロンを中心として，任意の学習範囲係数 β で与えられた範囲内のニューロン群を式（1）によって学習させる。また，学習の際には，学習率係数 α を基準に学習する大きさを決定する。α と β は，任意の初期値として与えられ，学習が繰り返されるたびに，$\alpha(t)$ と $\beta(t)$ は，0 に収束していく。

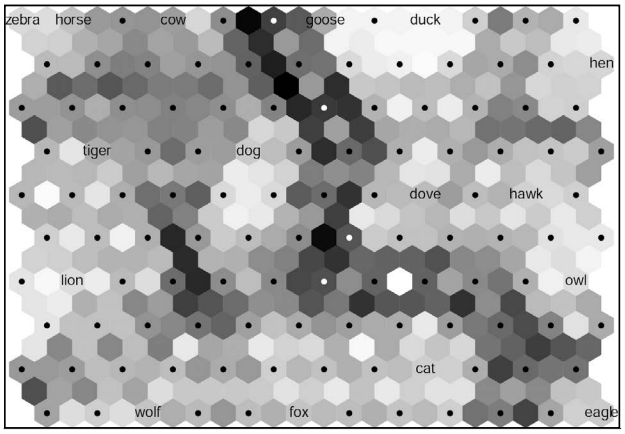
入力データの例を図 3 に示す。また，SOM の初期マップと出力マップの例を図 4 に示す。図 3 のような入力データを SOM に与え，STEP 1 にて，作成したランダム初期化のマップを図 4（a）に，学習終了後の学習結果のマップを図 4（b）に示す。図 4 の例では，マップの配置を六角格子型としている。図 4（b）の SOM の出力マップでは，似ているデータは近くに，異なるデータは遠くに配置され，色が薄くなるほど距離が近く，色が濃くなるほど距離が遠くなることを示している。学習前と学習後のマップを比較すると，草食動物や肉食動物，鳥類といったように比較的種族として似ている動物が近くに集まっている。このように，

	dove	fox	hen	lion	goose	eagle	dog	wolf	zebra	duck	cat	owl	tiger	horse	hawk	cow
小さい	1	0	1	0	1	0	0	0	0	1	1	1	0	0	1	0
中ぐらい	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0
大きい	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	1
夜行性	0	0.5	0	0	0	0	0	1	0	0	0.5	1	0.5	0	0	0
2本足	1	0	1	0	1	1	0	0	0	1	0	1	0	0	1	0
4本足	0	1	0	1	0	0	1	1	1	0	1	0	1	1	0	1
髪を持つ	0	1	0	1	0	0	1	1	1	0	1	0	1	1	0	1
有蹄類	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
たてがみ	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0
羽あり	1	0	1	0	1	1	0	0	0	1	0	1	0	0	1	0
縞あり	0	0	0	0	0	0	0	0	1	0.3	0	0	1	0	0	0
狩猟性	0	1	0	1	0	1	0	1	0	0	1	1	1	0	1	0
走る	0	0	0	1	0	0	1	1	1	0	0	0	1	1	0	0
飛ぶ	1	0	0	0	1	0	0	0	0	1	0	1	0	0	1	0
泳ぐ	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
草食性	0.5	0	0.5	0	0.5	0	0	0	1	0.5	0	0	0	1	0	1

図 3　入力データの例



(a)　初期マップ



(b)　出力マップ

図 4　**SOM** のマップの例

大きさ・模様などの姿形や食べ物・習性などの特性といった様々な特徴を入力データとして与えるだけで分類され，更に可視化できるのが SOM の特徴である。

3. 提　案　法

本稿で提案する SOM の高速化手法は，SOM で取り扱う入力データやニューロンごとのベクトルを 0 と 1 のビット列に近似表現し，ビット列からデータ量を圧縮したベクトル近似値を算出する。データ圧縮されたベクトル近似値を用いることで，最類似ニューロンの疎な抽出を行うことにより，SOM の学習アルゴリズムの高速化を図っている。また，各ニューロンに対応する行番号を保持させ，最類似ニューロンの近傍学習において，学習範囲の選定に行番号をキーとしたハッシュテーブルを利用することで，効率的な学習を実行する。

3.1 特徴ベクトルの近似

提案法では，特徴ベクトル空間の各次元軸を軸ごとに中間点で 2 分割して，各軸のベクトルを 0 と 1 の 1 ビットに変換し，特徴ベクトル全体を表現したビット列により，ベクトルデータを近似表現する。n 次元の特徴ベクトルを持つ実数ベクトル $x = \{x^{(1)}, x^{(2)}, \cdots, x^{(n)}\}$ の次元軸 i の成分 $x^{(i)}$ をビット $b^{(i)}$ で表現するとき，$b^{(i)}$ は次の式（2）により変換される。ただし，i は $1 \le i \le n$ の自然数であり，p は各次元軸の中間点とする。

$$b^{(i)} = \begin{cases} 0 & \text{if} \quad x^{(i)} < p \\ 1 & \text{otherwise} \end{cases} \qquad \cdots\text{式（2）}$$

式（2）を用いて全次元軸についてビット表現を行い，さらにそれらのビット群 b の総和が実数ベクトル x のベクトル近似値 v であり，次の式（3）により計算する。

$$v = \sum_{i=1}^{n} b^{(i)} \qquad \cdots\text{式（3）}$$

学習時において，最類似ニューロンを決定するとき，すなわち，2.2節のオリジナル SOM の学習アルゴリズム STEP 3 において，まず入力データの特徴ベクトル x のベクトル近似値 v_x と，出力層に配置されたニューロンの参照ベクトル W の近似値 v_w を比較して疎な検索を行う。具体的には，$|v_w - v_x| \le \theta$（t）の範囲内の近似値を持つものを疎な検索で抽出し，抽出結果に対してユークリッド距離を最小にする最類似ニューロンを決定する。なお，$\theta(t)$ は，疎な検索の範囲係数であり，初期値 θ_s から収束値 θ_e に収束するものとする。θ_s と θ_e の値については，第 4 章の数値実験にて検討する。

3.2　データ構造の拡張

近似値を格納するために，入力データに以下のデータ構造を追加する。

＜bit［m］, v＞

なお，bit［m］は n 次元ベクトルの各次元軸の 0 と 1 の情報を格納するビット配列であり，m は任意の定数である。また，v は式（3）の値を格納する整数型変数である。

また，出力層ニューロンに以下のデータ構造を追加する。

＜bit［m］, v, col＞

なお，bit［m］と v の定義は，前述の入力データのものと同様である。また，col は M×N の出力層において，各ニューロンが所属する行番号である。

3.3　提案法による学習アルゴリズム

近似値を用いて改良した SOM の学習アルゴリズムを以下に示す。各変数の定義は，2.2節および3.2節を参照すること。

＜近似値を用いた改良版学習アルゴリズム＞

STEP 1：繰り返し回数 t = 0 において，出力層の各ニューロンの参照ベクトルを初期化する。初期化では，ランダム初期化と線形初期化のどちらか一方を任意に選択する。ただし，参照ベクトルの初期化時に，以下の処理を行う。

STEP 1-1：出力層の各ニューロンが所属する行番を各ニューロンの col に設定する。

STEP 1-2：各次元の値を調べ，もし次元の中間点 p 未満ならば 0 を，そうでなければ 1 を bit［m］に設定する。

STEP 1-3：式（3）より，v の値を設定する。

STEP 1-4：v の数値順に並べたニューロンの双方向リスト List を作成する。

STEP 2：L 個の入力データ群からデータ x_k をランダムに選び出す。

STEP 3：x_k の特徴ベクトルに最も類似した参照ベクトルを持つニューロン u_c を探し出す。

STEP 3-1：List を先頭から順に探索していく。もしニューロン u_{ij} が $|v_w - v_x| \leq \theta(t)$ となる v_w を持つならば，次の処理を行う。そうでないならば，次の候補を調べる。ただし，v_w が $v_x + \theta(t)$ を超えた場合は，探索を終了する。

STEP 3-2：x_k の特徴ベクトルと u_{ij} の参照ベクトルとのユークリッド距離 $|x_k - W_{ij}^{(t)}|$ を計算する

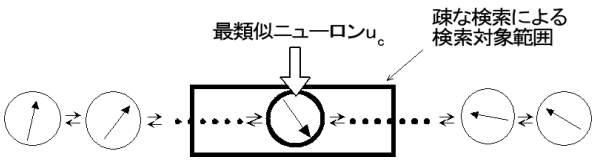
STEP 3-3：u_{ij} のユークリッド距離が最小ならば，u_c に u_{ij} を設定する。

STEP 4：まず，近傍関数 h より u_c の近傍ニューロン群 u_c' を求め，次に式（1）を用いて，u_c および u_c' の参照ベクトルに x_k の特徴ベクトルを学習させる。なお，近

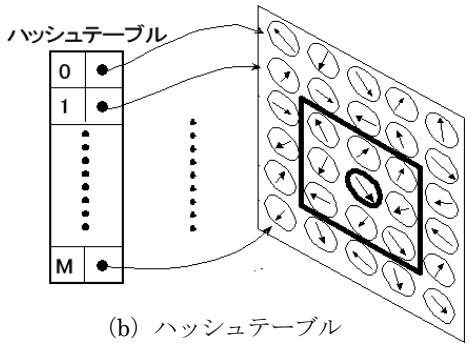
傍ニューロン群へのアクセスには，u_c の col を引数としたハッシュテーブルを用いる。特徴ベクトルの学習時に，各次元で中間点 p を超えた場合には，その結果を bit［m］に反映させる。

STEP 5：t をカウントしながら，STEP 2 から STEP 4 までを T 回繰り返す。

図 5 に，提案法におけるベクトル近似値を用いたリストとハッシュテーブルによる行頭検索の概要を示す。ベクトル近似値による最類似ニューロンの疎な検索を効率的に行うために，STEP 1 において，出力層のニューロン群をベクトル近似値順にソートした双方向リンク List を作成しておく（図 5 （a）参照)。これにより，ベクトル近似値による疎な線形探索が可能となる。ただし，線形探索時には，検索対象範囲のみのユークリッド距離を精査し，その範囲の調査後は，探索を終了する。また，STEP 4 において，近傍関数 h により求めた近傍ニューロン群 u_c' へのアクセスを効率的に行うために，図 5 （b）のようなハッシュテーブルを用いて更新範囲の先頭行の頭出しを行い，それ以降は，オリジナルの SOM と同様に，出力層の各ニューロンの近傍を調べていき，必要があれば参照ベクトルの学習を行う。



（a）リスト



（b）ハッシュテーブル

図 5　リストとハッシュテーブル

4. 数 値 実 験

提案法の有用性を示すために，オリジナルの SOM と提案法を用いて拡張した SOM との比較実験を行う。

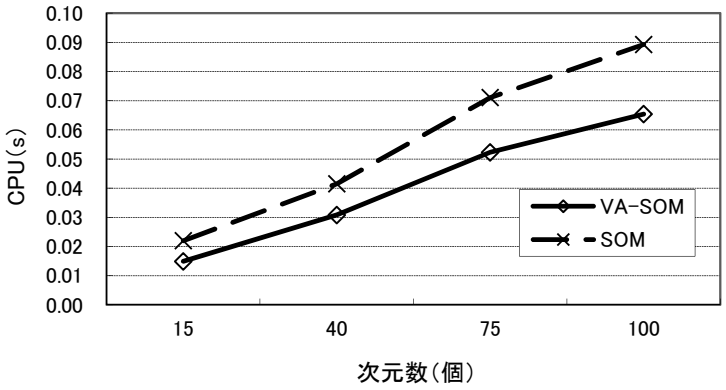
4.1 実 験 条 件

数値実験では，オリジナルの SOM と提案法を用いて拡張した SOM（以下，VA-SOM と呼ぶ）の 2 つを用意した。入力データは，［0, 1］の範囲の実数ベクトルを n 次元持つ L 個のサンプルデータとする。なお，数値実験では，n＝{15・40・75・100}，L＝{10・100・500・1,000} に変化させて調査した。また，学習回数 T＝{1,000・4,000・7,000・10,000} で変化させ，出力マップを正方形として一辺を {10・20・30・40} で変化させた。学習率係数（α）は0.01，近傍初期半径（β）は 6，近傍関数（h）は bubble 型とした。また，実験環境として，実験コンピュータの OS は Red Hat Enterprise Linux，CPU は2.00GHz，メモリは4G である。

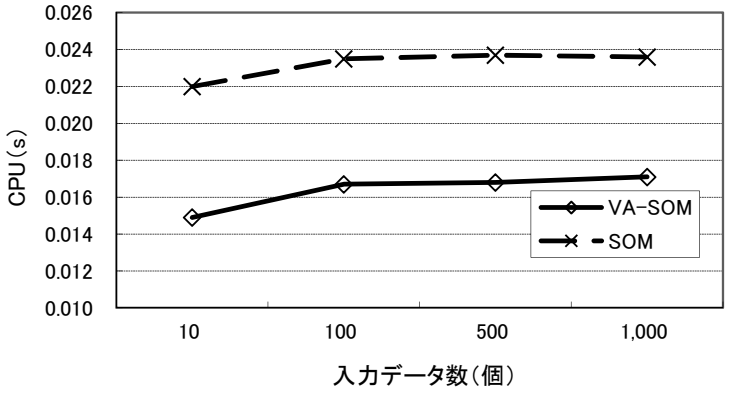
4.2 実 験 結 果

次元 n＝15，入力データ数 L＝10，学習回数 T＝1,000および出力マップの一辺10を基本条件とし，次元，入力データ数，学習回数および出力マップの大きさを変えた実験結果を図 6 と図 7 に示す。なお，図 6 と図 7 の結果は，実験を10回行った平均値を示している。図 6 と図 7 より，特徴ベクトルの次元数，入力データ数，学習回数および出力マップサイズの大きさを変更したとしても，提案法を適用した VA-SOM は従来法であるオリジナルの SOM の1.35−1.47倍程度高速化されていることが確認できた。

また，図 6 と図 7 の実験結果に対応した疎な検索の範囲係数の初期値 θ_s および収束値 θ_e を表 1 に示す。表 1 の値は，各実験条件において，VA-SOM の学習プロセスがオリジナルの SOM の学習プロセスと全く同様になる最小のものである。実験の基本条件（次元 n＝15，入力データ数 L＝10，学習回数 T＝1,000および出力マップの一辺10）に対して，表 1 に記された条件のみを変更することで，各実験の条件となおる。表 1 より，最悪時の場合，すなわち，入力データ数の多い場合は，次元数 n に対して，θ_s は約 0.73×n，θ_e は約 0.47×n となる。最良時の場合，すなわち，入力データ数に対して次元数 n が多い場合は，θ_s は約 0.20×n，θ_e は約 0.10×n となる。その他の場合は，おおよそ次元数 n に対して，θ_s は約 0.30×n，θ_e は約 0.20×n となっている。



(a) CPU v.s. 次元数



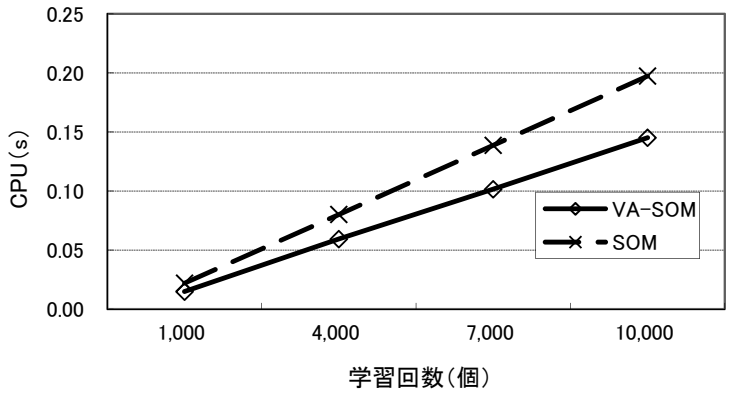
(b) CPU v.s. 入力データ数

図 6　次元数と入力データ数に対する学習時間の実験結果

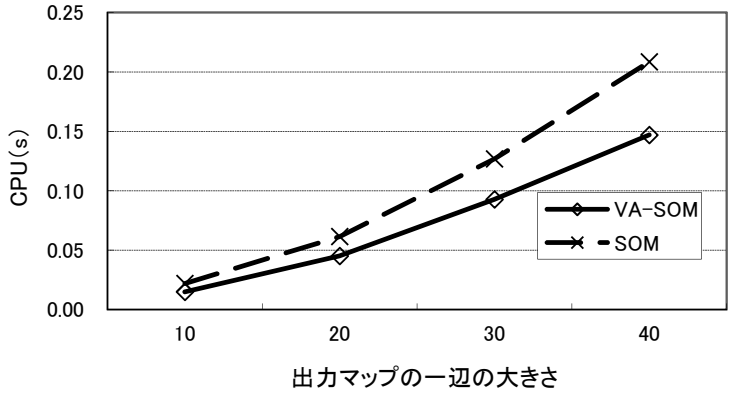
表 1　実験における初期値 θ_s と収束値 θ_e の対応表

	図 6 (a)：次元数				図 6 (b)：入力データ数			
	15	40	75	100	10	100	500	1,000
θ_s	5	10	16	20	5	9	10	11
θ_e	3	5	8	10	3	6	7	7

	図 7 (a)：学習回数				図 7 (b) 出力マップ			
	1,000	4,000	7,000	10,000	10	20	30	40
θ_s	5	5	6	6	5	5	5	5
θ_e	3	3	4	4	3	3	3	3



（a）CPU v.s. 学習回数



（b）CPU v.s. 出力マップサイズ

図 7　学習回数と入力マップサイズに対する学習時間の実験結果

5. お　わ　り　に

本稿では，高速で効率的な人工知能の実現のために，ニューラルネットワークの 1 つである自己組織化マップ（SOM）の高速化を目的とする一方式として VA-SOM を提案した。提案法では，オリジナルの SOM と全く同じ学習プロセスを保持したまま，オリジナルの SOM の1.35–1.47倍程度の高速化が確認できた。提案法は，オリジナルの SOM と学習プロセスが同じであるため，SOM の他の改良法との併用が理論的に可能である。今後の予定としては，SOM の他の改良法との併用化の検証や他のニューラルネットワークシステムへの適用，経済データを対象とした解析システムの構築などが考えられる。

参　考　文　献

［1］松尾豊：「人工知能は人間を超えるかディープラーニングの先にあるもの」，KADOKAWA/中経出版（2015）
［2］AI ビジネス研究会，福林一平：「60分でわかる !AI ビジネス最前線」，技術評論社（2017）
［3］D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis: “Mastering the Game of Go with Deep Neural Networks and Tree Search”, Nature 529（7587），pp.484–489（2016）
［4］W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, and D. Yu, G. Zweig: “Achieving Human Parity in Conversational Speech Recognition”, CoRP, abs/1610.05256（2017）
［5］Kohonen, T.: “Self-Organizing Maps”, Springer Series in Information Sciences, Volume 30（1995）
［6］R. Weber, H.-J. Schek, and S. Blott: “Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces”, Proc. of 24th Int. Conf. on Very Large Databases（VLDB’98），pp.194–205（1998）
［7］コホネン , T., 徳高平蔵，岸田　悟，藤村喜次郎訳：「自己組織化マップ」，シュプリンガー・フェアラーク東京（1996）
［8］徳高平蔵，岸田　悟，藤村喜次郎：「自己組織化マップの応用」，海文堂（1999）
［9］Dittenbach, M., Merkl, D. and Rauber, A.: “The Growing hierarchical self-organizing map”, Proceedings of the IEEE-INNS-ENNS International Joint Conference, Vol.6, pp.15–19（2000）
［10］Y. Maniwa, Y. Ikeda, H. Kurosawa, H. Tokutaka, K. Fujimura, M. Usami, and A. Tsutou: “Construction of Checkup System which Uses Self-Organizing Maps（SOM）”, Journal of Biomedical Fuzzy Systems Association, Vol.5, No.1, pp.9–16（2003）
［11］H. Wakuya, Y. Horinouchi, and H. Itoh: “Development of an application for mobile devices to analyze data set by a Self-Organizing Map : A case study on Saga Prefecture sightseeing information”, International Journal of Contents, Vol.9, No. 3, pp.15–18（2013）
［12］亀岡　瑶，宗像昌平，八木圭太，山本義郎：「自己組織化マップによる顧客の分類とその可視化」，計算機統計学，第29巻，第 2 号，pp.181–188（2016）
［13］P. Dewan, R. Ganti and M. Srivatsa: “SOM-TC: Self-Organizing Map for Hierarchical Trajectory Clustering”, 2017 IEEE 37th International Conference on Distributed Computing Systems（ICDCS），pp.1042–1052（2017）
［14］Mu-Chun Su and Hsiao-Te Chang: “Fast Self-Organizing Feature Map Algorithm”, IEEE Transactions on Neural Networks, Vol.11, No.3, pp.721–733（2000）
［15］木ノ内誠，高田直志，工藤喜弘：「一括学習型自己組織化マップの高速学習」，情報化学討論会講演要旨集，JP37（2001）
［16］佐々木英史，高橋由雅：「索引層を用いた SOM の学習高速化アルゴリズム」，2004年度人工知能学会全国大会（第18回）論文集，1F2–04（2004）
［17］後藤康路，廣田雅春，横山昌平，福田直樹，石川　博：「MapReduce を用いた並列 SOM の高速化手法の提案」，DEIM Forum 2012，C2–3（2012）

Abstract

VA-SOM: A Speed-up Technique of Self-Organizing Map Using Vector Approximation

Hiroyuki Dekihara

In the fields of Artificial intelligence based on neural network systems for Intelligent Transfer Systems, Disaster Recovery Systems, Robotics Systems, and so on, these systems must react quickly, responding to various situations. In this paper, a speed-up technique has been developed for neural network systems using vector approximation like VA-File. The technique presented here is to be applied to Self-Organizing Map (SOM) that is one of the neural network systems. The proposed method is called Vector Approximation SOM (VA-SOM). The VA-SOM manages input data by the list and the hash table based on the vector approximation. In a series of experimental tests, the results have been shown to be more efficient than current techniques. The performance of the VA-SOM is in proportion to the high dimensional.

Keywords: Neural network, Self-Organizing Map, Speed-up, Vector Approximation, VA-File