

## オンライン教師なし分類のための追加学習手法

小倉 和貴<sup>†</sup>      申 富饒<sup>††</sup>      長谷川 修<sup>†,††</sup>

An Incremental Learning Mechanism for Online Unsupervised Classification

Tomotaka OGURA<sup>†</sup>, Furao SHEN<sup>††</sup>, and Osamu HASEGAWA<sup>†,††</sup>

あらまし オンライン教師なし学習手法である Self-Organizing Incremental Neural Network (SOINN) (Shen & Hasegawa, 2006) を改良した新しい手法として, Enhanced-SOINN (ESOINN) を提案する. ESOINN は SOINN のもつオンライン教師なし学習の機能を失うことなく追加学習を可能とした. 改良点は以下のとおりである. (1) 二層構造で学習を行う SOINN に対し, 一層構造による学習が可能である. (2) 分布に高密度の重なりをもつクラスを分離できる. (3) SOINN と比べて少ないパラメータで動作する. (4) SOINN と比べて安定性が高い. 人工データ及び実データを用いた実験により, ESOINN は SOINN より優れていることを示す.

キーワード 自己増殖型ニューラルネットワーク (SOINN), 一層構造, 分布の高密度の重なり, 安定性

### 1. ま え が き

教師なし学習の目的の一つはクラスタリングである. 代表的なクラスタリング手法である  $k$ -means 法は初期値依存性を有し, 大域的最適解を保証しない. この点を改良した Global  $k$ -means 法 [1] や Adaptive Incremental LBG [2] は計算負荷が大きいという問題点がある. 一般に  $k$ -means 法に基づく手法では利用者がクラス数  $k$  の値を決定する必要があるが, 事前に適切な値を選択することは難しい. また, クラスの形状が超球形であることを仮定しており, 複雑な形状のクラスに対して適切な結果を得ることは難しい. EM アルゴリズム [3] も初期値依存性を有し, 大域的最適解を保証しない. その他, Single-Link [4], Complete-Link [5], CURE [6] などは記憶容量と計算負荷の面で問題がある.

教師なし学習の別の目的は高次元データの分布を

位相構造として抽出することである. 自己組織化マップ (SOM) [7] はネットワークの構造を事前に決定する必要があり, 競合ヘブブ学習 (Competitive Hebbian Learning) と組み合わせられた Neural Gas (NG) [8] はネットワークのノード数を事前に決定する必要がある. そのため, これらの手法は追加学習に適さない. Growing Neural Gas (GNG) [9] は定期的にノードを挿入することで追加学習に対応しているが, ノード数が恒久的に増加するため永続的な学習に適さない.

追加学習は非定期的な入力を学習する場合に必要となる. つまり, 既にある学習結果に対して新しいクラスのデータが入力された場合, それに対応するクラスを追加的に形成するために必要である. このとき, 以前の学習結果を改悪あるいは忘却せずに, 新しい入力データに適応することが求められる. GNG-U [11] はデータが低密度に分布する領域に位置するノードを削除することで非定期的な入力への対応を実現しているが, 追加学習には対応しておらず, 以前の学習結果は破壊されてしまう. Hamker (2001) [12] は GNG を拡張して教師ありでの追加学習を実現したが, この手法は教師なし学習には適さない.

Shen と Hasegawa は Self-Organizing Incremental Neural Network (SOINN) [13], [14] と呼ばれるオンライン教師なし学習の手法を提案した. SOINN は二層ネットワーク構造をしており, 非定期的な入力を学習でき, 分布に複雑な形状をもつクラスに対しても適

<sup>†</sup> 東京工業大学大学院総合理工学研究科知能システム科学専攻, 横浜市

Department of Computational Intelligence and System Science, Tokyo Institute of Technology, 4259 Nagatsuta-cho, Midori-ku, Yokohama-shi, 226-8503 Japan

<sup>††</sup> 中国南京大学計算機科学技術系, 中国

The Department of Computer Science of Nanjing University, China

<sup>†††</sup> 東京工業大学像情報工学研究施設, 横浜市

Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, 4259 Nagatsuta-cho, Midori-ku, Yokohama-shi, 226-8503 Japan

切なクラス数及び位相構造を出力できるなど多くの利点をもつ。また、追加学習も部分的に実現している。一方で以下の問題点がある。(1) 1層目は追加学習が可能であるが、2層目は追加学習ができない。(2) 2層目の学習を開始するタイミングを適切に決定することが難しい。(3) 分布に低密度の重なりがあるクラスは分離できるが、重なりが高密度の場合には分離できず、複数のクラスが連結して一つのクラスタを形成してしまう。

本論文では以上のような SOINN の問題点を解決した新しい手法として Enhanced-SOINN (ESOINN) を提案する。ESOINN はオンライン教師なし学習が可能な SOINN のもつすべての機能を失うことなく、追加学習を可能とした手法であり、SOINN より少ないパラメータで動作する。

## 2. SOINN の概要

ここでは、提案手法のもととなった SOINN の概要について述べる。

### 2.1 学習アルゴリズム

SOINN は二層ネットワーク構造により学習を行うが、1層目と2層目は同じ学習アルゴリズムで動作する。図1に SOINN のフローチャートを示す。SOINN は入力ベクトルが与えられると、入力ベクトルに最も近いノード(以降、第1勝者ノードと呼ぶ)及び2番

目に近いノード(以降、第2勝者ノードと呼ぶ)を探索し、3.2で述べる類似度しきい値という基準を用いて、入力ベクトルが同一のクラスタに属すかどうかを判定し、第1勝者ノードと異なるクラスタに属す場合は入力ベクトルと同じ位置にノードを挿入し、次の入力を処理する。このときの挿入はクラス間挿入と呼ばれる。また、同一のクラスタに属す場合は、第1勝者ノード及び第2勝者ノードを辺によって直接的に接続しているノード(隣接ノード)の重みベクトルを更新する。また、「辺の年齢」という基準を用いて第1勝者ノードに接続している辺のうち不要な辺を削除する。更に、入力が入回与えられるごとにノイズと判定されたノードを削除し、入力データとノードの位置との誤差が大きい領域にノードを挿入する。ただし、挿入によって誤差が減少しない場合は、挿入は取り消される。このときの挿入はクラス内挿入と呼ばれ、Growing Neural Gas (GNG) や Growing Cell Structure (GCS) などの伝統的な追加的ネットワーク構造学習と同様の仕組みである。

SOINN の1層目は入力回数あらかじめ設定された回数  $LT$  に達すると学習結果としてノードの重みベクトル及び位相構造を出力する。学習結果として得られた重みベクトルは、学習データとして2層目に入力される。ただし、追加学習を行う場合は、2層目に残っている以前の学習結果を消去し、その上で2層目の学習を行う。2層目の学習が終了すると、SOINN はクラス数及び各クラスの代表的なプロトタイプベクトルを出力する。ここで、プロトタイプベクトルとは代表的な入力データを集約的に表現したベクトルのことである。SOINN 及び ESOINN においてはノードの重みベクトル(アルゴリズム 4.3における  $W$ )がプロトタイプベクトルに相当する。

### 2.2 1層目と2層目の違い

1層目では入力データの分布が未知であるので、類似度しきい値を適応的に変化させる。しかし、2層目では1層目の結果から入力データの分布を知ることができる。このため、2層目では類似度しきい値として適切な値を算出し、その値(固定値)を用いる。これにより2層目は1層目比べて学習結果が安定する。一方で、2層目では学習領域全体で同じ類似度しきい値を用いるため、局所的に類似度しきい値の値が不適切となる領域が存在する。そのような領域では入力ベクトルとノードの位置との誤差が増加するため、クラス内挿入が多く行われる。これに対して、1層目では

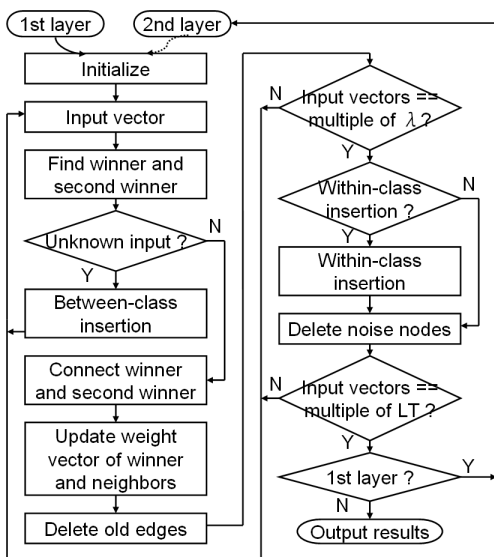


図1 SOINN のフローチャート  
Fig. 1 Flowchart of SOINN.

類似度しきい値が適切に変化するため、未知の入力が適切に検出され、クラス間挿入が行われる。そのため、誤差があまり増加せず、クラス内挿入はほとんど行われない。

通常、1層目と2層目ではノイズ削除に関するパラメータに対して異なる値を設定して削除の振舞いを変化させる。すなわち、1層目では隣接ノード数が一つであるノードをノイズとして無条件に削除する。一方、2層目では位相構造を適切に抽出するために、隣接ノード数が一つであるノードのうち第1勝者ノードになった回数が他と比べて極端に少ないノードのみ削除する。追加学習において2層目の学習結果を削除する理由はここにある。追加学習において新規に挿入されるノードは、長期間学習を行ったノード（以前の学習結果）に比べれば第1勝者ノードになった回数が少ない。そのため、追加学習のために必要なノードのほとんどがノイズとして削除されてしまう。これを防ぐため、以前の学習結果を削除し、再学習を行う。

### 3. Enhanced-SOINN (ESOINN)

#### 3.1 SOINN と ESOINN の違い

SOINN は二層ネットワーク構造により学習を行うが、そのために次の問題点をもつ。

- 1層目の学習を停止して2層目の学習を始める適切なタイミングを利用者が決定する必要がある。
- 1層目はオンラインでの追加学習が可能であるが、2層目はそれができない。

ESOINN は SOINN と同じ機能を一層ネットワーク構造で実現する。そのため、ESOINN では2層目の学習を始めるタイミングを決定する必要がない。また、ESOINN は SOINN の1層目をもとに改良を加えた手法であり、SOINN の1層目もつ追加学習などの機能を受け継いでいる。つまり、オンラインでの追加学習を実現できる。

SOINN では追加学習のために1層目及び2層目においてネットワークに新しいノードを挿入する。このとき、クラス内挿入とクラス間挿入の2種類を用いるが、クラス内挿入には多くのパラメータが必要である。そのため、SOINN の動作には合計八つのパラメータが必要となる。一方、ESOINN ではクラス間挿入のみを用いるため、SOINN より少ない合計四つのパラメータで動作する。また、クラス内挿入の削除による学習結果への影響はほとんどない。これらの詳細は3.2. で述べる。

SOINN では3.3 で述べる競合ヘップ学習則によりノード間を接続する。ESOINN ではこの接続規則に以下の改良を加える。(1) 接続が必要かどうか事前に判定する。(2) 一定期間ごとにクラス間における分布の重なり領域を検出し、そこに存在する接続を消去する。これにより、ESOINN は分布に高密度の重なりをもつクラスを分離できる。これらの詳細は4. で述べる。

#### 3.2 ノードの挿入

追加学習を実現し、適切な位相構造を獲得するには、ネットワークを成長させることが重要である。これはノードの挿入によって実現できるが、ノード数の恒久的な増加による悪影響を防ぐには、十分な数のノードが得られた時点で挿入を停止する必要がある。

SOINN ではノードの挿入にあたってクラス間挿入とクラス内挿入の2種類を用いているが、2.2 で述べたように、クラス内挿入は1層目の学習にはほとんど寄与しない。そのため、ESOINN ではクラス間挿入のみを用いてノードの挿入を行う。クラス間挿入では、入力ベクトルが未学習領域への入力であるかどうかを判定し、そのような入力の場合にノードを挿入する。すなわち、入力ベクトルとその最近傍ノード  $k$  との距離がノード  $k$  の類似度しきい値  $T_k$  より大きい場合には、未学習領域への入力であると判断し、入力ベクトルの位置に新しいノードを挿入する。類似度しきい値はポロノイ領域の考えに基づいて計算されるが、ノードの位置は学習を進めることで次第に変化し、それに伴ってポロノイ領域も変化する。つまり、類似度しきい値はノードの位置変化に応じて適応的に変化する必要がある。

ノード  $i$  の重みベクトルを  $W_i$  とするとき、ポロノイ領域  $V_i$  は以下の式で定義される。

$$V_i = \{W \mid \|W - W_i\| \leq \|W - W_j\|, i \neq j\} \quad (1)$$

ポロノイ領域  $V_i$  の内部に入力ベクトルが与えられた場合、その入力はノード  $i$  によって学習済みの領域に入力されたと判断する。ここで、ポロノイ領域の定義から、ポロノイ領域  $V_i$  の内部にはノード  $i$  以外のノードは存在しないため、ノード  $i$  から最も遠い隣接ノードまでの距離より離れた領域は明らかにノード  $i$  のポロノイ領域  $V_i$  の外側であると考えられる。つまり、類似度しきい値を隣接ノードまでの最大距離とすることで、未学習領域を適切に判定できる。ただし、本論文における隣接ノードは位相構造上の隣接ノード

を指す。すなわち、辺  $(i, j)$  が存在するときのみ、ノード  $i$  とノード  $j$  は隣接ノードである。ESOINN ではアルゴリズム 3.1 に従って類似度しきい値を計算する。このアルゴリズムは SOINN における類似度しきい値の計算とほぼ同じである。

#### アルゴリズム 3.1: 類似度しきい値 $T$ の計算

Step 1. 新しく挿入されたノード  $i$  の類似度しきい値  $T_i$  を  $+\infty$  とする。

Step 2. ノード  $i$  が入力ベクトルから最も近いノードまたは 2 番目に近いノードになったとき、類似度しきい値  $T_i$  を以下の手順で更新する。

- ノード  $i$  が隣接ノードをもつ場合:  $T_i$  を隣接ノードへの最大距離とする ( $T_i = \max_{c \in N_i} \|W_i - W_c\|$ )。ここで、 $N_i$  はノード  $i$  の隣接ノードの集合とする。
- ノード  $i$  が隣接ノードをもたない場合: ノード  $i$  からそれ以外の各ノードへの距離を計算し、その最小距離を  $T_i$  とする ( $T_i = \min_{c \in A \setminus \{i\}} \|W_i - W_c\|$ )。ここで、 $A$  はノードの集合とする。

ESOINN においてクラス内挿入を削除したことは学習結果に影響しない。既に述べたように、SOINN のクラス内挿入は 1 層目の学習にはほとんど寄与しない。つまり、一層構造で学習を行う ESOINN ではクラス間挿入のみで位相構造を表現するにあたって十分な数のノードを得ることができる。クラス間挿入では未学習領域への入力を与えられた場合にノードを挿入する。そして、その後の処理により、挿入されたノードと既にクラスタを形成しているノードが接続されることで、それらが一つのクラスタとしてまとまる。このとき、類似度しきい値が適応的に変化することで、ネットワークの形成に十分な数のノードが挿入される。また、ノードが各クラスの境界位置まで挿入されると、未学習領域が存在しなくなり、ノードの挿入は自動的に停止される。つまり、ノードの永続的な挿入を防ぐことができる。5. で述べる実験結果も、クラス間挿入のみで適切な位相構造を獲得できることを示している。

### 3.3 ノードの接続

SOINN では、ノード間を接続するために Martinez が Topology Representing Network (TRN) [16] において提案した競合ヘップ学習則を用いる。すなわち、第 1 勝者ノード及び第 2 勝者ノードを探索し、それらの間に辺を作ることによってそれらを接続する（既に辺が存在する場合はその辺を更新する）。この規則により生成されるネットワークは位相構造を適切に表現することが証明されている。オンライン学習において、ノ

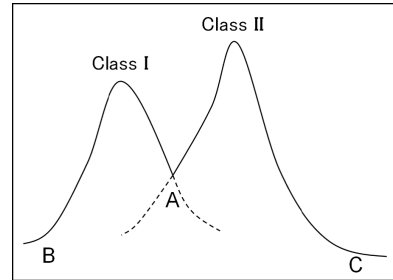


図 2 分布に重なりのあるクラス  
Fig. 2 Overlapped classes.

ドはその位置を恒久的に少しずつ変化させる。それゆえ、はじめは隣接していたノードが一定期間を経た後には隣接しないことが起こり得る。このような場合にも、一定期間を経ても更新されない辺を削除することで対処できる。

ESOINN では競合ヘップ学習則に変更を加える。すなわち、ESOINN では第 1 勝者ノード及び第 2 勝者ノードを接続する前にその必要性を判定し、接続が必要な場合のみ辺を生成する。例えば、図 2 において、二つのクラス間には分布の重なり（図 2 における A の部分）が存在する。このため SOINN の仕組みを用いると、二つのクラスは接続されて一つのクラスタを形成してしまう。これを防ぐためには、第 1 勝者ノードと第 2 勝者ノードが分布の重なり領域に位置する場合には、それらを接続しない仕組みが必要である。

#### アルゴリズム 3.2: ESOINN における辺の生成

Step 1. 入力ベクトルに対し、第 1 勝者ノードと第 2 勝者ノードを探索する。

Step 2. 第 1 勝者ノードに接続されているすべての辺の年齢を 1 増加させる。

Step 3. 第 1 勝者ノードと第 2 勝者ノードの間に辺が必要かどうか判定する。

- 必要な場合: 第 1 勝者ノードと第 2 勝者ノードの間に辺が存在しなければ、辺を生成し、第 1 勝者ノードと第 2 勝者ノードの間に存在する辺の年齢を 0 に更新する。

- 不要な場合: 第 1 勝者ノードと第 2 勝者ノードの間に辺が存在すれば、それを削除する。

アルゴリズム 3.2 において、第 1 勝者ノードと第 2 勝者ノードが分布の重なり領域に位置するかどうかを基準として接続の必要性を判定することで、分布に重なりのあるクラスを分離できる。

ESOINN は分布に高密度の重なりのあるクラスを

分離できる。これは、辺の生成にあたってその必要性を判定すること、及びクラス間の分布の重なり領域を検出する仕組みを導入したことで実現される。これらの詳細は 4. で述べる。

### 3.4 ノードの属すクラスを決定する

[13] では、いくつかの辺を通して二つのノードが接続していることを、それら二つのノード間には“パス”が存在すると表現している。つまり、 $a, b, x_i \in A, i = 1, 2, \dots, n$  として  $(a, x_1), (x_1, x_2), \dots, (x_{n-1}, x_n), (x_n, b) \in C$ , という辺の連続が存在するとき、ノード  $a$  とノード  $b$  の間にはパスが存在する。ここで、 $A$  はノードの集合、 $C$  は辺の集合である。

Martinetz と Schulten は [16] において、競合ヘブリング学習則によって、与えられた多様体を適切に表現するパスが形成されることを証明した。パスが適切に形成されているとき、与えられた多様体はどの部分が分離しているか、すなわち、異なるクラスを形成しているかを決定できる。つまり、二つのノードがパスによって接続していれば、それらのノードは同一のクラスに属すと判断できる。ESOINN では、ノードの属すクラスを決定する際に SOINN と同じアルゴリズムを用いる。

#### アルゴリズム 3.3: ノードの属すクラスを決定する

Step 1. すべてのノードをどのクラスにも属していない状態にする。

Step 2. どのクラスにも属していないノード  $i$  をランダムに選択し、新しいクラスのラベルを貼る。

Step 3. ノード  $i$  とパスによって接続しているノードをすべて探索し、ノード  $i$  と同じラベルを貼る。

Step 4. どのクラスにも属していないノードが存在するなら、Step 2 に戻る。

### 3.5 ノイズに相当するノードを削除する

SOINN では分布が低密度の領域に位置するノードをノイズとみなして削除する。すなわち、入力されたベクトルの総数がパラメータ  $\lambda$  の倍数になったとき、隣接ノード数が一つ以下であるノードを低密度の領域に位置するとみなして削除する。ただし、一次元のデータ若しくはノイズの少ないデータに対しては、削除対象のノードが第 1 勝者ノードになった回数を利用して削除の振舞いを調節する。これは次の理由による。一次元のデータの場合、分布の端に生成されるノードは隣接ノード数が必ず 1 となる。そのため、隣接ノード数が一つ以下という条件でノードを削除すると分布

の端の領域で学習結果が不安定になってしまう。また、ノイズの少ないデータについても同様のことが起き、学習結果が不安定になってしまう。これらを防ぐために SOINN では削除の振舞いを調節している。また、SOINN が二層構造であることもノイズの削除において有効な役割を果たす。つまり、1 層目と 2 層目のそれぞれでノイズ削除を行うため、より効果的にノイズを削減できる。

ESOINN では、SOINN とよく似た仕組みを用いてノイズに相当するノードを削除する。すなわち、入力されたベクトルの総数がパラメータ  $\lambda$  の倍数になったとき、隣接ノード数が二つ以下であるノードを削除する。ESOINN では隣接ノード数が二つであるノードを削除する点が SOINN と異なる。また、ESOINN ではノードの密度という概念、及び二つのパラメータ  $c_1$  (隣接ノード数が二つの場合)、 $c_2$  (隣接ノード数が一つの場合) を用いて、削除の振舞いを調節する。ノードの密度の詳細は 4. で述べる。

## 4. クラス間における分布の重なり

ここでは、ノードの密度の定義、クラス間における分布の重なりを検出する方法、及び第 1 勝者ノードと第 2 勝者ノードの接続における判定基準について述べる。重なり領域の検出にあたって、重なり領域の密度はクラス中心部分の密度より低いという仮定を用いる。

### 4.1 ノードの密度

ノードの密度は局所的に与えられる入力の数によって定義できる。すなわち、あるノードの近くに多くの入力を与えられるなら、そのノードの密度は高く、逆にその近くに入力ほとんど与えられないなら、そのノードの密度は低いと考えられる。つまり、ノードが第 1 勝者ノードになった回数 (勝者の回数) をノードの密度として定義できる。この定義は GCS や SOINN などの手法に導入されている。これは自然な定義であるが以下の問題点をもつ。

(1) 一般に分布が高密度の領域には多くのノードが生成されるため、そのような領域では第 1 勝者ノードになる機会が少ない。つまり、より高密度の領域に位置するノードほど勝者の回数が多いとは限らない。

(2) 追加学習を行う場合、以前の学習において生成されたノードは第 1 勝者ノードにならないことが多い。つまり、追加学習によってそれらのノードの勝者の回数が相対的に少なくなり、以前の学習で得られた結果に悪影響を与えてしまう。

ESOINN では、上記の問題を解決するために密度に対する新しい定義を用いる。基本的な考えは“勝者の回数”と同じであるが、“回数”の代わりに“ポイント”を用い、ノードの密度を累積ポイントの平均として定義する。“勝者の回数”による定義では特定のノードのみを用いて密度の計算を行うが、新しい定義ではノード間の関係を考慮する。ポイントの計算において、はじめに隣接ノードへの平均距離  $\bar{d}_i$  を計算する。

$$\bar{d}_i = \frac{1}{m} \sum_{j=1}^m \|W_i - W_j\|_2^2 \quad (2)$$

ここで、 $m$  はノード  $i$  における隣接ノードの数、 $W_i$  はノード  $i$  の重みベクトルとする。そして、ノード  $i$  のポイント  $p_i$  を以下の式で算出する。

$$p_i = \begin{cases} \frac{1}{(1+\bar{d}_i)^2} & \text{第 1 勝者ノードの場合} \\ 0 & \text{それ以外の場合} \end{cases} \quad (3)$$

隣接ノードへの平均距離が大きい場合は、その領域にはノードが少ないと考えられ、逆に平均距離が小さい場合は、その領域にはノードが多いと考えられる。式 (3) を用いることで、ノードの多い領域で第 1 勝者ノードになった場合には高いポイントが与えられ、逆にノードの少ない領域で第 1 勝者ノードになった場合には低いポイントが与えられることが分かる。つまり、“勝者の回数”による密度の定義における最初の問題点を解決できる。式 (3) において、ポイントの値を 1 未満にするために分母に 1 を加えている。

連続して与えられる入力を一定回数  $\lambda$  ごとの区間に分け、各区間においてノード  $i$  に与えられたポイントの合計を累積ポイント  $s_i$  と呼ぶことにする。

$$s_i = \sum_{j=1}^n \left( \sum_{k=1}^{\lambda} p_i^{(j,k)} \right) \quad (4)$$

ここで、 $n$  は区間の総数を表す ( $n$  は入力総数を  $LT$  とするとき、 $LT/\lambda$  によって計算される)。また、 $p_i^{(j,k)}$  は  $j$  番目の区間における  $k$  番目の入力によってノード  $i$  に与えられたポイントを意味する。

そして、ノード  $i$  の密度  $density_i$  を累積ポイントの平均として以下で定義する。

$$density_i = \bar{s}_i = \frac{1}{N} s_i = \frac{1}{N} \sum_{j=1}^n \left( \sum_{k=1}^{\lambda} p_i^{(j,k)} \right) \quad (5)$$

ここで、 $N$  は与えられたポイントの合計が 0 以上の

区間の数である ( $N$  と  $n$  は必ずしも同じでない点に注意する)。追加学習に対応するために  $n$  の代わりに  $N$  を用いている。追加学習において、以前の学習で生成されたノードにはポイントが与えられないことが多く、 $n$  を用いて密度を算出すると、以前学習したノードの密度は次第に低くなってしまふ。しかし、 $N$  を用いて密度を算出することで、追加学習を長時間行っても、追加されるデータが以前学習したノードの近くに入力されない限り、そのノードの密度は変化せずに保持される。つまり、“勝者の回数”による密度の定義における 2 番目の問題点を解決できる。

#### 4.2 分布の重なり領域の検出

密度の定義を用いて分布の重なり領域を検出する簡単な方法は、密度の低い領域を重なり領域として検出する方法である。GCS や SOINN などの手法はこの考えを用いている。しかし、この考えが正しいとは限らない。例えば、図 2 のようないくつかのガウス分布で与えられるクラスにおいて、分布の重なり領域である領域 A の密度は重なり領域でない領域 B や領域 C の密度より高い。これは、重なり領域が複数のクラスから構成されるため、密度がある程度高くなってしまふために起きる。つまり、単純に密度の高低を用いるだけでは重なり領域の検出は難しい。この問題を解決するため、ESOINN では新しい方法を導入する。

SOINN において、いくつかのクラス間の分布に重なりがある場合、ある程度学習が進むと、それらのクラスに属すノードが互いに接続して一つのクラスタを形成することがある。ここでは、そのような混合クラスタ (複数のクラスからなるクラスタ) において分布の重なり領域を検出する方法、及び異なるクラスから生成されたクラスタが接続されることを防止し、クラスタを適切に分離する方法について述べる。

はじめに、分布の重なり領域を検出するために、混合クラスタを以下の規則に基づき、いくつかのサブクラスタに分割する。

アルゴリズム 4.1: 混合クラスタのサブクラスタへの分割

Step 1. 密度が局所的に最大となっているノードをサブクラスタの頂点とする。そのような頂点をすべて探索し、それぞれに異なるラベルを貼る。

Step 2. それ以外の全ノードに対して、密度が最大の隣接ノードと同じラベルを貼る。

Step 3. 異なるラベルを貼られたノード間に接続がある場合、それらのノードをサブクラスタの境界、つま

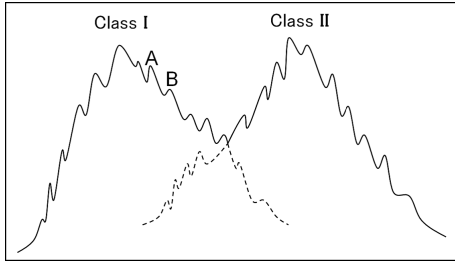


図3 分布に重なりのあるクラス (細かい凹凸がある場合)  
Fig. 3 Overlapped classes with small irregularities.

り、分布の重なり領域とする。

アルゴリズム 4.1 を用いることで、図 2 には二つのサブクラスタがあり、領域 A が重なり領域であることが検出できる。

この方法にはいくつかの問題が存在する。例えば、図 3 には二つのクラスが存在するが、密度の分布には細かい凹凸がある (これはノイズや学習サンプルが少ないことが原因となって起きる)。このような状況でアルゴリズム 4.1 を用いると、非常に多くのサブクラスタ及び重なり領域が検出されてしまう。この問題を解決するには、サブクラスタへの分離を行う前に、図 3 を図 2 のように平滑化が必要がある。実際には、ある基準を満たした二つのサブクラスタを一つに統合することで平滑化を実現する。

例えば、図 3 における二つのサブクラスタ A と B において、サブクラスタ A の頂点の密度が  $A_{max}$ 、サブクラスタ B の頂点の密度  $B_{max}$  であるとする。ここで、第 1 勝者ノード及び第 2 勝者ノードがサブクラスタ A 及び B の間にある分布の重なり領域に位置し、以下の条件のいずれかを満足するとき、サブクラスタ A と B を統合する。

$$\min(density_{win}, density_{sec-win}) > \alpha_A A_{max} \quad (6)$$

$$\min(density_{win}, density_{sec-win}) > \alpha_B B_{max} \quad (7)$$

ここで、 $density_{win}$  及び  $density_{sec-win}$  はそれぞれ第 1 勝者ノード及び第 2 勝者ノードの密度を表す。

式 (6) 及び式 (7) によって密度の凹凸の大きさを判定している。つまり、サブクラスタ A と B の間にある谷間の密度 (左辺) がしきい値 (右辺) より大きい場合、小さな凹凸であると判定する。しきい値はサブクラスタの頂点の密度を  $\alpha$  倍した値とし、 $\alpha \in [0, 1]$  は式 (8) により算出する。

$$\alpha_A = \begin{cases} 0.0 & 1 \geq A_{max}/mean_A - 1 \\ 0.5 & 2 \geq A_{max}/mean_A - 1 > 1 \\ 1.0 & A_{max}/mean_A - 1 > 2 \end{cases} \quad (8)$$

ここで、 $mean_A$  はサブクラスタ A に属すノードの密度の平均値である。

$$mean_A = \frac{1}{N_A} \sum_{i \in A} density_i \quad (9)$$

ここで、 $N_A$  はサブクラスタ A に属すノードの数を表す。

式 (8) において、サブクラスタの密度の凹凸の程度を判定し  $\alpha$  の値を決定している。 $A_{max}/mean_A$  の値によって  $A_{max}$  と  $mean_A$  の大きさの比を算出できるが、 $A_{max} \geq mean_A$  であるため  $A_{max}/mean_A$  の最小値は 1 となる。式 (8) では、最小値を 0 にするために  $A_{max}/mean_A - 1$  を用いている。ここで、その値が 1 以下の場合には、 $A_{max}$  と  $mean_A$  の値は同程度であり、密度の凹凸はノイズの影響によるものと判断する。そして、 $\alpha$  の値を 0.0 とすることで、サブクラスタが統合されるようにしている。また、値が 2 を超える場合には  $A_{max}$  が  $mean_A$  に比べて十分大きく、明らかな密度の凹凸が存在すると判断する。そして、 $\alpha$  の値を 1.0 とすることで、サブクラスタが分離されるようにしている。それ以外の場合は、 $\alpha$  の値を 0.5 とすることで、密度の凹凸の大きさに応じてサブクラスタが統合または分離されるようにしている。

以上をまとめると、混合クラスタを異なるサブクラスタに分割し、それらを適切に統合することで、分布の重なり領域を検出できる。そして、異なるサブクラスタに属するノード間に存在する接続を削除することで、分布に重なりのあるクラスを分離できる。

アルゴリズム 4.2: ノード間を接続する条件

Step 1. 第 1 勝者ノードまたは第 2 勝者ノードが、どのサブクラスタにも属していないノードであるなら、ノード間に辺を生成し、それらを接続する。

Step 2. 第 1 勝者ノード及び第 2 勝者ノードが同じサブクラスタに属す場合、ノードの間に辺を生成し、それらを接続する。

Step 3. 第 1 勝者ノードと第 2 勝者ノードが異なるサブクラスタ A と B に属すなら、以下の方法により、二つのノード間の接続が必要かどうかを判定する。

- $m = \min(density_{win}, density_{sec-win})$  を計算する。ここで、 $density_{win}$  及び  $density_{sec-win}$  はそれぞれ第 1 勝者ノードの密度及び第 2 勝者ノードの密度

を表す .

- $\alpha_A A_{max} > m$  かつ  $\alpha_B B_{max} > m$  を満たすなら, 二つのノードを接続しない . また, 既にノード間に接続がある場合には接続を削除する .

- $m \geq \alpha_A A_{max}$  または  $m \geq \alpha_B B_{max}$  を満たすなら, 二つのノードを接続し, サブクラスタ A と B を統合する .

サブクラスタへの分割処理において, ノイズなどの影響によりサブクラスタが誤って分割される場合がある . このような場合でも正しい再分割を行うために, アルゴリズム 4.2 では第 1 勝者ノードと第 2 勝者ノードが異なるサブクラスタに属す場合も, それらのノードを接続する機会を与えている .

ESOINN の結果は SOINN の結果に比べて安定するが, これは以下の理由による . クラス間に低密度の重なりがある場合, SOINN はある程度それらを正しく分割できるが, 分割に失敗して一つのクラスタを形成することもあり, 結果が安定しない . 一方, ESOINN では低密度の分布の重なりを安定的に分離できる . また, 平滑化の手法を導入することで, 分けすぎによる不安定化も防止している .

#### 4.3 ESOINN 全体のアルゴリズム

2. 及び 3. のまとめとして, ESOINN 全体のアルゴリズムを述べる .

アルゴリズム 4.3: Enhanced-SOINN (ESOINN)

Step 1. 入力ベクトル群からランダムに二つのベクトルを選択し, ノード集合  $A$  をそれらに対応する二つのノードのみを含む集合として初期化する . また, 辺集合  $C \subset A \times A$  を空集合として初期化する .

Step 2. 新しい入力ベクトル  $\xi \in R^n$  を入力する .

Step 3.  $\xi$  に最も近いノード  $a_1$  (第 1 勝者ノード) 及び 2 番目に近いノード  $a_2$  (第 2 勝者ノード) を探索する .

$$a_1 = \arg \min_{a \in A} \|\xi - W_a\|$$

$$a_2 = \arg \min_{a \in A \setminus \{a_1\}} \|\xi - W_a\|$$

ここで,  $\xi$  と  $a_1$  または  $\xi$  と  $a_2$  の距離が類似度しきい値  $T_{a_1}$  または  $T_{a_2}$  より大きければ, 入力ベクトルを新しいノードとしてノード集合  $A$  に加え, Step 2 に進み, 次のベクトルを処理する . 類似度しきい値  $T$  はアルゴリズム 3.1 によって算出される .

Step 4. アルゴリズム 4.2 を用いて,  $a_1$  と  $a_2$  の間に接続が必要かどうかを判定し, アルゴリズム 4.2 に従って接続を生成する .

Step 5. 式 (5) を用いて第 1 勝者ノードの密度を更新する .

Step 6. 第 1 勝者ノードの累積入力数  $M_{a_1}$  を 1 増やす .

$$M_{a_1} = M_{a_1} + 1$$

Step 7. 第 1 勝者ノード及びその隣接ノードの重みベクトルを微量  $\epsilon_1(t)$ ,  $\epsilon_2(t)$  及び入力ベクトルまでの距離を用いて更新する .

$$\Delta W_{a_1} = \epsilon_1(M_{a_1})(\xi - W_{a_1})$$

$$\Delta W_i = \epsilon_2(M_{a_1})(\xi - W_i)$$

ここで, ノード  $i$  はノード  $a_1$  の隣接ノードとする . また, 重みベクトルの更新は SOINN と同じ仕組みを用いて行う . すなわち, 次の式を用いて, 入力数に応じて学習率を変化させる .

$$\epsilon_1(t) = 1/t$$

$$\epsilon_2(t) = 1/100t$$

ここで,  $t$  は入力回数であるが, システムへの入力回数ではなく, 第 1 勝者ノードへの累積入力数  $M_{a_1}$  を用いる . これは, 追加学習に対応するためである .

Step 8. あらかじめ設定されたしきい値  $age_t$  を超えた年齢をもつ辺を削除する .

Step 9. これまでに入力されたベクトルの総数がパラメータ  $\lambda$  の倍数となった場合, 以下を行う .

(a) 各ノードのサブクラスタのラベルをアルゴリズム 4.1 を用いて更新する .

(b)  $A$  に属す全ノードに対して, 以下の手順でノイズに相当するノードを削除する .

i. 隣接ノードを二つもつ場合: ノードの密度が  $c_1 \sum_{j=1}^{N_A} density_j / N_A$  未満なら, ノードを削除する .

ここで,  $N_A$  はノード集合  $A$  の要素数である .

ii. 隣接ノードを一つもつ場合: ノードの密度が  $c_2 \sum_{j=1}^{N_A} density_j / N_A$  未満なら, ノードを削除する .

iii. ノードが隣接ノードをもたない場合: ノードを削除する .

Step 10. オンライン学習を終了するなら, アルゴリズム 3.3 を用いてノードの分類を行い, クラス数及び各クラスのプロトタイプベクトルを出力し, 学習を停止する .

Step 11. 学習を終了しない場合, Step 2 に進み, 学習を続ける .



## 5. 実 験

### 5.1 人工データ

最初に、図 4 に示した人工データセットを用いて SOINN と ESOINN の比較実験を行った。データセットは分布に重なりのある二つのガウス分布、二つの同心円、及びサインカーブの合計五つのクラスによって構成されている。また、実世界の環境を想定して、10%のノイズが加えられている。SOINN では [13] において同じデータセットを用いた実験が行われ、定常的な環境及び非定常的な環境における実験とともに、五つのクラス及び各クラスの位相構造を適切に出力することが示された。ここで、定常的な環境では、データセット全体からランダムに入力ベクトルを与え、非定常的な環境では、データセットを図 4 に示す七つの領域に分け、一定期間ごとに入力環境を表 1 に従って切り換えながら入力ベクトルを与えた。非定常的な環境における実験はオンラインでの追加学習を想定して行われた。

同じデータセットを用いて、ESOINN (パラメータは  $\lambda = 100$ ,  $age_t = 100$ ,  $c_1 = 0.001$ ,  $c_2 = 1.0$  とした。これらは実験により定めた) による実験を行った。図 5 は定常的な環境及び非定常的な環境における実験結果である。どちらの環境においても、ESOINN は五つのクラス及び各クラスタの位相構造を適切に出力

した。これにより、ESOINN が SOINN と同程度の機能を有していることが確認された。

次に、図 6 に示した人工データセットを用いて比較実験を行った。データセットは分布に重なりのある三つのガウス分布から構成され、10%のノイズが加えられている。このデータセットは図 4 に示したデータセットに比べて分布に高密度の重なりをもつ。定常的な環境では、データセット全体からランダムに入力ベクトルを選択し、非定常的な環境では、各クラスから順にそれぞれ 10,000 回ずつ入力ベクトルを選択して学習を行った。

図 7 は SOINN (パラメータは  $\lambda = 200$ ,  $age_t = 50$ ,  $c = 1.0$ ,  $\alpha_1 = 1/6$ ,  $\alpha_2 = 1/4$ ,  $\alpha_3 = 1/4$ ,  $\beta = 2/3$ ,  $\gamma = 3/4$  とした。 $\lambda$ ,  $age_t$ ,  $c$  は実験により定め、 $\alpha_1$ ,

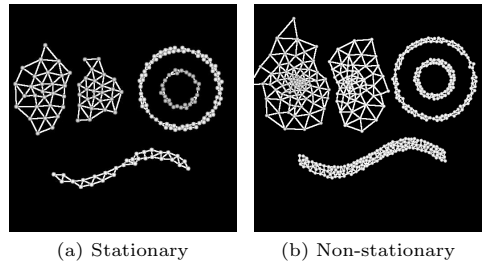


図 5 データセット I における ESOINN の結果  
Fig. 5 ESOINN results for data set I.

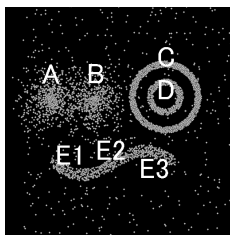


図 4 人工データセット I  
Fig. 4 Artificial data set I.

表 1 入力環境  
Table 1 Input environments.

| Area | Environment |    |     |    |   |    |     |
|------|-------------|----|-----|----|---|----|-----|
|      | I           | II | III | IV | V | VI | VII |
| A    |             | x  |     | x  | x | x  | x   |
| B    | x           |    | x   |    | x | x  | x   |
| C    | x           | x  |     | x  | x |    | x   |
| D    | x           | x  | x   |    |   | x  | x   |
| E1   | x           | x  | x   | x  |   | x  | x   |
| E2   | x           | x  | x   | x  | x |    | x   |
| E3   | x           | x  | x   | x  | x | x  |     |

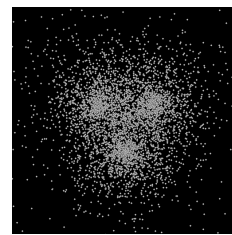


図 6 人工データセット II  
Fig. 6 Artificial data set II.

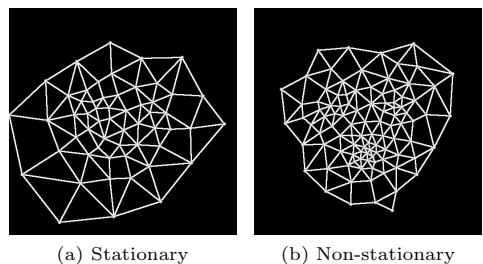
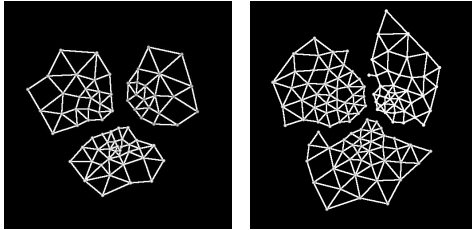


図 7 データセット II における SOINN の結果  
Fig. 7 SOINN results for data set II.



(a) Stationary (b) Non-stationary

図 8 データセット II における ESOINN の結果  
Fig. 8 ESOINN results for data set II.

$\alpha_2, \alpha_3, \beta, \gamma$  は [13] と同様とした) における結果である。これらの結果から、定常的な環境、非定常的な環境ともに、SOINN では高密度の重なりのあるクラスを分離できないことが分かる。図 8 は ESOINN (パラメータは  $\lambda = 200, age_t = 50, c_1 = 0.001, c_2 = 1.0$  とした。これらは実験により定めた) における結果である。これらの結果から、ESOINN では高密度の重なりをもつクラスを分離でき、三つのクラス及び各クラスの位相構造を適切に出力することが分かる。

### 5.2 実データ

ここでは 2 種類の実データセットを用い、SOINN と ESOINN の性能を比較する。

最初に、AT&T\_FACE データベース (<http://www.uk.research.att.com>) を用いた比較実験を行った。データセットは、AT&T\_FACE データベースから選択された 10 クラス (各クラスは 10 個のサンプルを含む) を用いた。オリジナル画像は  $92 \times 112$  ピクセル、グレースケール 256 階調である ([13] の図 11 を参照)。実験にあたって、オリジナルの画像を  $23 \times 28$  ピクセルに縮小し (最近傍法により補間)、ガウス分布 (サイズ 4, 分散 2) による平滑化を行った画像を用いた。これらの処理によって得られた特徴ベクトル ([13] の図 12 を参照) を入力ベクトルとした。

定常的な環境では、入力ベクトルをデータセット全体からランダムに選択し、非定常的な環境では、各クラスから順にそれぞれ 1,000 回ずつ入力ベクトルを選択して学習を行った。定常的な環境、非定常的な環境ともに ESOINN (パラメータは  $\lambda = 25, age_t = 25, c_1 = 0.0, c_2 = 1.0$  とした。これらは実験により定めた) はクラス数として 10 クラスを出力する機会が多かった。また、そのとき得られたプロトタイプベクトルの一つを用い、[13] における SOINN の実験と同様に、オリジナルのデータセットの識別を行った。そ

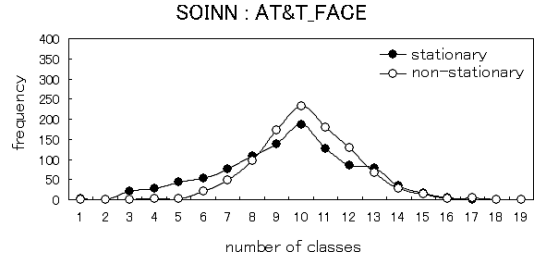


図 9 SOINN の出力クラス数  
Fig. 9 SOINN distribution of number of classes.

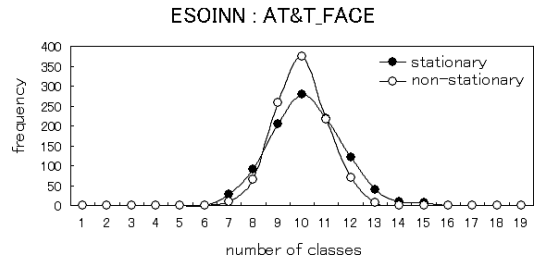


図 10 ESOINN の出力クラス数  
Fig. 10 ESOINN distribution of number of classes.

の結果、SOINN と同程度の認識率 (定常的な環境で 90%、非定常的な環境で 86%) が得られた。SOINN と ESOINN で同程度の結果が得られたが、これはデータセットのサンプル数が少なく、クラス間の分布の重なりが低密度であるためと考えられる。これにより、ESOINN は SOINN と同程度の機能を有していることが確かめられた。

SOINN 及び ESOINN の安定性を比較するため、それぞれ 1,000 回ずつ実験を行い、出力クラス数の頻度を調べた。図 9 は SOINN における結果であり、図 10 は ESOINN における結果である。これらの図は、定常的な環境及び非定常的な環境ともに ESOINN は SOINN に比べて 10 クラス前後を出力する回数が多いことを示している。この結果から、ESOINN は SOINN に比べて安定していることが分かる。

更に、別の実データを用いて比較実験を行った。データとして、Optical Recognition of Handwritten Digits database (Optdigits) (<http://www.ics.uci.edu/~mllearn/MLRepository.html>) を選択した。このデータセットは 10 クラスの手書き数字からなり、学習データとして 3,823 個、テストデータとして 1,797 個のサンプルがある。サンプルの次元数は 64 である。

まず、SOINN (パラメータは  $\lambda = 200, age_t = 50, c = 1.0, \alpha_1 = 1/6, \alpha_2 = 1/4, \alpha_3 = 1/4, \beta = 2/3,$



図 11 SOINN におけるプロトタイプベクトル  
Fig. 11 SOINN result of Optdigits data set.



図 12 ESOINN におけるプロトタイプベクトル  
Fig. 12 ESOINN result of Optdigits data set.

$\gamma = 3/4$  とした。  $\lambda$ ,  $age_t$ ,  $c$  は実験により定め、  $\alpha_1$ ,  $\alpha_2$ ,  $\alpha_3$ ,  $\beta$ ,  $\gamma$  は [13] と同様とした) による実験を行った。その結果、定常的な環境、非定常的な環境ともに、クラス数として 10 クラスを出力する場合が最多であった。図 11 は SOINN の結果におけるプロトタイプベクトルの一例である。ここで、SOINN で得られたプロトタイプベクトルを用いてテストデータの分類を行った。その結果、定常的な環境で 92.2%、非定常的な環境で 90.4% の認識率を得た。更に、SOINN について 100 回の実験を行い、クラス数の変動を調べたところ、定常的な環境、非定常的な環境ともに 6~13 クラスが出力された。

次に、ESOINN (パラメータは  $\lambda = 200$ ,  $age_t = 50$ ,  $c_1 = 0.001$ ,  $c_2 = 1.0$  とした。これらは実験により定めた) による実験を行った。その結果、定常的な環境、非定常的な環境ともに、クラス数として 12 クラスを出力する場合が最多であった。図 12 は ESOINN の結果におけるプロトタイプベクトルの一例である。図 11 と図 12 を比べると、数字の「1」を、SOINN では一つのクラスとして出力し、ESOINN では 2 クラスに分けていることが分かる (同様に、ESOINN では数字の「9」も 2 クラスに分けている)。オリジナルのデータセットにおいて、図 12 の 1 と 1' のようなサンプルの間には大きな違いがある (同様に 9 と 9' の間にも大きな違いがある) ため、ESOINN ではこれらを分離したと考えられる。また、SOINN では図 12 の 1' や 9' のようなサンプルは抽出されていない。これらの結果から、ESOINN では分布に重なりのあるクラス (図 12 の 1 と 1' や 9 と 9') を分離でき、オリジナルデータの情報を SOINN より適切に保存できることが示された。

更に、ESOINN で得られたプロトタイプベクトルを用いてテストデータの分類を行った結果、定常的な環境で 94.3%、非定常的な環境で 95.8% という SOINN

を超える認識率を得た。また、ESOINN について 100 回の実験を行い、クラス数の変動を調べた結果、定常的な環境、非定常的な環境ともに 10~13 クラスが出力され、SOINN より安定することが示された。

実データ Optdigits を用いた実験の結果から、ESOINN は SOINN と比べ、分布に重なりのあるクラスを分離できること、高い認識率を得ること、及び安定性が高いことが示された。

### 5.3 パラメータの決定

ここでは、各パラメータの役割とパラメータ決定に関する指針を述べる。

#### 5.3.1 SOINN におけるパラメータ

$\lambda$  はノイズとみなされるノードを削除する周期である。小さな値を設定すると頻繁にノイズ処理が行われるが、極端に小さくすると実際はノイズではないノードを誤って削除してしまう。逆に極端に大きな値を設定するとノイズの影響で生成されたノードを適切に取り除くことができない。

$age_t$  はノイズなどの影響で誤って生成された辺を削除するために用いられる。小さな値を設定すると辺が削除されやすくなりノイズによる影響を防ぐことができるが、極端に小さくすると頻繁に辺が削除されるようになり学習結果が不安定になる。逆に極端に大きな値を設定するとノイズの影響で生成された辺を適切に取り除くことができない。

$c \in [0, 1]$  は隣接ノード数が一つであるノードをノイズとみなすかどうかの判定に用いられる。0 に近い値を設定するとすべてがノイズとみなされる。また、1 に近い値を設定するとほとんどがノイズでないとみなされる。通常は 1 層目ではノイズを効果的に削除するために 1 に近い値を用い、2 層目では位相構造を適切に抽出するために 0 に近い値を用いる。

$\alpha_1, \alpha_2, \alpha_3, \beta, \gamma$  はクラス内挿入で用いられる。入力データの分布が局所的には一様であるとの仮定を用いると、 $\alpha_1 = 1/6$ ,  $\alpha_2 = 1/4$ ,  $\alpha_3 = 1/4$ ,  $\beta = 2/3$ ,  $\gamma = 3/4$  が数学的に導かれ、通常はこれらの値を用いる。

#### 5.3.2 ESOINN におけるパラメータ

$\lambda$  及び  $age_t$  は SOINN と同じパラメータである。 $c_1$ ,  $c_2$  は SOINN におけるパラメータ  $c$  に相当し、 $c_1$  は隣接ノード数が二つであるノードを、 $c_2$  は隣接ノード数が 1 であるノードをそれぞれノイズとみなすかどうかの判定に用いられる。通常、隣接ノード数が一つであるノードはノイズであることが多いので  $c_2$  は 1 に

近い値を用いる。また、隣接ノード数が2であるノードはノイズでないことが多いので  $c_1$  は0に近い値を用いる。ESOINN ではクラス内挿入を行わないため、パラメータ  $\alpha_1, \alpha_2, \alpha_3, \beta, \gamma$  は存在しない。

## 6. む す び

本論文では、SOINN を改良した新しい手法として Enhanced-SOINN (ESOINN) を提案した。提案手法は SOINN のもつすべての機能を実現できるため、オンライン教師なし学習が可能である。また、二層構造で学習を行う SOINN に対して、ESOINN は一層構造での学習を実現し、完全なオンラインでの追加学習が可能である。ESOINN は、ノード間を接続する条件を与えることで、高密度の重なりのあるクラスを分離できる。また、ノードの挿入にあたってクラス間挿入のみを採用したことで、SOINN より四つ少ないパラメータで動作する。更に、平滑化の手法を導入したことで、SOINN より安定的に動作する。

残された課題として以下が挙げられる。ESOINN は SOINN より少ないパラメータで動作するが、まだ四つのパラメータの値を決定する必要があり、これらのパラメータは学習結果に影響を与える。ESOINN は SOINN より安定して動作するが、ノイズが多く、学習データが少ない場合には結果が安定しない。このため、いかなるタスクに対しても十分な安定性をもつとはいえない。将来的にはこれらの課題を解決する必要がある。

## 文 献

- [1] A. Likas, N. Vlassis, and J.J. Verbeek, "The global  $k$ -means clustering algorithm," *Pattern Recognit.*, vol.36, no.2, pp.451–461, Feb. 2003.
- [2] F. Shen and O. Hasegawa, "An adaptive incremental LBG for vector quantization," *Neural Netw.*, vol.19, no.5, pp.694–704, June 2006.
- [3] A.K. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol.22, no.1, pp.4–37, Jan. 2000.
- [4] H.A. Sneath and R.R. Sokal, *Numerical taxonomy*, Freeman, London, UK, 1988.
- [5] B. King, "Step-wise clustering procedures," *J. American Statistical Association*, vol.69, no.3, pp.86–101, 1967.
- [6] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," *Proc. ACM SIGMOD Conference on Management of Data*, pp.73–84, June 1998.
- [7] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol.43,

no.1, pp.59–69, Jan. 1982.

- [8] T.M. Martinetz, S.G. Berkovich, and K.J. Schulten, "“Neural-gas” network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Netw.*, vol.4, no.4, pp.558–569, July 1993.
- [9] B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems (NIPS)*, vol.7, pp.625–632, 1995.
- [10] G.A. Carpenter and S. Grossberg, "The art of adaptive pattern recognition by a self-organizing neural network," *Computer*, vol.21, no.3, pp.77–88, March 1988.
- [11] B. Fritzke, "A self-organizing network that can follow non-stationary distributions," *Proc. ICANN'97*, pp.613–618, 1997.
- [12] F.H. Hamker, "Life-long learning cell structures — Continuously learning without catastrophic interference," *Neural Netw.*, vol.14, no.4-5, pp.551–573, May 2001.
- [13] F. Shen and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Netw.*, vol.19, no.1, pp.90–106, Jan. 2006.
- [14] F. Shen, *An algorithm for incremental unsupervised learning and topology representation*, Ph.D thesis, 2006.
- [15] B. Fritzke, "Growing cell structures — A self-organizing network for unsupervised and supervised learning," *Neural Netw.*, vol.7, no.9, pp.1441–1460, 1994.
- [16] T. Martinetz and K. Schulten, "Topology representing networks," *Neural Netw.*, vol.7, no.3, pp.507–522, 1994.

(平成 18 年 6 月 15 日受付, 9 月 28 日再受付)



小倉 和貴

2005 明大・理工・情報科学卒。現在、東工大大学院知能システム科学専攻修士課程在学中。



申 富饒

1995 中国南京大・数学卒。1998 同数学専攻卒。2006 東工大大学院知能システム科学専攻博士課程了。現在、中国南京大学計算機科学技術系助教。



長谷川 修 (正員)

1993 東京大学大学院電子工学専攻博士課程了。博士(工学)。同年電子技術総合研究所。1999 から 1 年間米国カーネギーメロン大学客員研究員。2001 産業技術総合研究所主任研究員。2002 東京工業大学像情報工学研究施設助教授。2003 ~ 2006

JST さきがけ研究 21 研究者(兼任)。情報処理学会, 日本認知科学会, 人工知能学会, IEEE-CS 等各会員。