博 士 論 文

大規模ニューラルネットワークの並列学習法に関する研究

指導教官　堀口 進 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

山森 一人

1997 年 3 月 31 日

要旨

入出力間の詳しい関係を記述することなく情報の処理が可能なニューラルネットワークは，新しい情報処理手法として近年注目されている．しかし，ニューラルネットワークの学習には膨大な時間が必要であり，実用規模への問題に対する応用が困難である．そのため，学習速度の向上が強く望まれている．本論文では，ニューラルネットワークの代表的な形態である階層型ニューラルネットワークと Kohonen による自己組織化マップ（SOM）を取りあげ，その並列学習法について議論する．

階層型ニューラルネットワークについては，従来より多くの並列学習法が提案されてきたが，多くは特定の計算機に特化した実装法であり，誤差逆伝搬学習が持つ基本的な並列性についての議論は不十分であった．本論文では，誤差逆伝搬学習が内包する並列性を利用した 3 種類の並列学習法をとりあげ，それらの並列学習法の特性について議論する．同時に超並列計算機上に各並列学習法を実装し，その並列処理性能について評価検討する．一方，階層型ニューラルネットワークのハードウェア実装による高速学習手法では，ハードウェア故障が避けられず，故障を補償する学習法が必要とされている．しかし，従来の故障補償法では学習に極めて長い時間が必要であったり，補償を行う際に条件を加えるなどの問題があった．そこで，階層型ニューラルネットワークのハードウェア故障を考慮した部分再学習法を提案し，その有効性について議論する．提案する部分再学習法は，故障の影響を受けるユニットのみをに注目して再学習することにより高速に故障補償した学習が可能である．

SOM の並列学習には競合層を分割する手法が提案されているが，競合層のユニット数が多くなると通信時間の増加のため十分な高速化が達成できず，PE 間の負荷のばらつきが大きいという問題があった．そこで，入力層分割による並列学習法を提案し，その処理性能について議論する．また，SOM の学習では学習セットを反映しないデッドノードが発生する場合がある．これを解決するために最小結合木を用いた手法が提案されているが，ユニットの空間的隣接関係が保障されないという問題がある．そこで，空間的隣接関係を保障しつつデッドノードを削除する忘却学習法を提案し，デッドノードが効果的に削除できることを示す．さらに，提案した忘却学習法がハードウェア上に SOM を実装する場合に必要な故障補償メカニズムとして動作可能かを詳しく検討する．

目 次

図 目 次

第 1 章

緒言

1.1 本研究の背景と目的

1.1.1 並列計算機

近年の計算機技術の発展に伴い，膨大な計算機資源を必要とする大規模な問題を計算機で扱うことが可能となってきた．微細加工，クロック周波数を高めるなどの VLSI 技術，インストラクションキャッシュやベクトルプロセッサといった計算機アーキテクチャ上の改良により，チップ単体での処理速度は上昇の一途をたどっている．これらの技術を結集してスーパーコンピュータと呼ばれる計算機が実用化され，流体計算や物性解析などの分野で用られるようになってきており，数値解析などを行う強力な手法となっている．

しかしながら，チップ単体での高速化は現行の技術では限界に近づき，現在これに代わる高速計算機のあり方が模索されるようになってきている．次世代の高速計算機として最も有望視されているものの一つに超並列計算機がある．超並列計算機は多数のプロセッサを相互結合網で結び，問題を細かく分割して並列に処理を行うことにより高速な処理を実現する．

現在実用化されている超並列計算機は大きく 2 つに分類され，それぞれ SIMD 型，MIMD 型と呼ばれる．SIMD 型超並列計算機は，異なるデータに対し複数のプロセッサが同一の命令を実行するものであり，MIMD 型超並列計算機は異なるデータに対し各プロセッサが独立に命令を実行する．

SIMD 型超並列計算機は，巨大データに対して同一の処理を行う画像処理などの分野へ

の応用が期待されており，数万規模のプロセッサを備える CM-2 などがある．MIMD 型超並列計算機は各プロセッサが異なる命令を実行できるため，SIMD 型超並列計算機よりも複雑な処理を行うことができる．代表的な MIMD 型超並列計算機としては，ICOT の PIM や nCUBE 社の nCUBE/2，Thinking Machines 社の CM-5 などが挙げられる．

高速な計算機による数値計算が盛んに行われる一方で，人間の脳が行う情報処理手法に着目した研究が行われるようになってきた．人間が言語を用いて理論的に思考する過程は逐次的であり，従来のノイマン型計算機のモデルとなってきた．こうした見地からの研究はチューリング以来数多く行われており，現在のコンピュータ技術の発展の基礎となっている．また，脳が持つ個別の情報処理システムを単純化し，計算機を用いてさまざまなルールに基づき意思を決定する人工知能の分野は Winston[1] がまとめたように様々な領域に広がり，エキスパートシステムなどの技術が開発された．これらの新しい理論をもとに，データフローマシンやリダクションマシン，LISP や prolog などの記号処理言語を高速に実行する計算機などが，超並列計算機とともに発展しつつある．

1.1.2 ニューラルネットワーク

従来の計算機は単純にモデル化された空間内で情報を単純処理するという形で発展してきた．ところが近年，人間の脳の機能的な解明から考案された新しい情報処理手法が注目されている．人間の脳はニューロンと呼ばれるしきい素子が相互に多数結合しており，結合部分はシナプス結合と呼ばれる．ニューロンの発火頻度によりカルシウムイオンやカリウムイオンの濃度を変化させてシナプス結合の強度を変化させることで脳は学習を行い，情報を処理していることが次第に明らかになっている．

こうした脳の機能は，1943 年に McCulloch と Pitts[2] により工学的にモデル化され，1949 年には Hebb[3] によりシナプス結合強度の変化規則が提案された．これらの研究を元に，Rosenblatt[4] により Perceptron が提案され，ニューラルネットワーク研究の主流となる．しかし， Perceptron では線型分離不可能な問題を扱えないことなどを Minsky[5] らが指摘し，Rumelhart ら [6] による誤差逆伝搬学習がこれに代わる．誤差逆伝搬学習は極めて強力な手法であり，パターン認識や制御分野などで現在では一部実用に用いられるようになってきている．

しかしながら，大規模なネットワークを必要とする実用規模の問題に対しては，学習を行ってネットワークを収束させるのに膨大な時間が必要になる．そこで近年，誤差逆伝搬

学習自体を改良して高速化を行ったり，並列計算機を用いたニューラルネットワークの高速学習に関する研究が数多く行われるようになってきている．

学習アルゴリズムの改良としては，たとえばFahlman[7]は誤差逆伝搬学習のパラメータ変更による挙動変化を詳しく調べ，quick-propagationと呼ばれる変形誤差逆伝搬学習法を提案している．また，Schiffmanら[8]は，学習率などの値を学習状態に応じて適切に変更することで，学習の高速化ができることを示した．また，塚本ら[9]は，1回の学習パターン提示で学習を行う瞬時学習法を提案している．

しかし，アルゴリズムの改良では学習パターンに条件を設ける場合も多く，また増大する大規模化への要求を満たすことはできない．そこで，誤差逆伝搬学習を行う階層型ニューラルネットワークの並列化に関する研究が盛んに行われている．例えば曽根原ら[10]はSIMD型，MIMD型並列計算機上への階層型ニューラルネットワークの実装法について報告している．Singer[11]はThinking Machines社製の超並列計算機CM-2上で，並列誤差逆伝搬学習を行う複数の実装法について述べ，誤差逆伝搬学習が持つ並列性について言及した．Zhangら[12]はSingerと同様にCM-2を使用し，格子網をベースにして並列誤差逆伝搬学習を行う手法を提案した．Zhangらによる報告では約6万5千個のプロセッサを装備したCM-2により，おおよそ150MWUPS（Mega Weight Update Per Second）の学習速度が達成されている．Mullerら[13]は，45個のDSPを用いた分散メモリ形式の並列計算機を製作し，階層型ニューラルネットワークの各層を複数のプロセッサに割り当てる並列化手法を提案している．Witbrockら[14]は，超並列計算機GF-11上に階層型ニューラルネットワークを実装した結果から，ニューラルネットワークを高速に学習させる並列計算機のアーキテクチャについて考察している．

一方，ニューラルネットワークの基礎となった脳では，教師信号を使わず自律的に自己組織化を行っているらしいことが報告されている．近年の脳に関する生理学的な研究により，脳は各種の機能を別々の領域で処理していることが明らかになってきた．特に視覚野における研究では線分の傾きに対して選択的に反応する細胞が発見され，同じような傾きに反応する細胞が皮質上で隣り合うよう配置していることが分かっている[15]．こうしたトポグラフィックマッピングを自律的に行う手法として，自己組織化ニューラルネットワークが注目されている．特に，Kohonen[16]が提案した自己組織化マップ（SOM）は，簡単な規則でトポグラフィクマッピングを行う手法として注目されている．たとえば，Obermayerら[17]はSOMを用いて指先の感覚器に対する受容野のマッピングを行い，SOM上に各指

に対応した領域が生成されることを示した．

SOM においても，数多くのニューロンをネットワークとして接続して情報処理を行うということから，ネットワーク規模が増大すると学習時間が爆発的に増加するという問題がある．このため，誤差逆伝搬学習と同じく並列学習による高速化が試みられている．Myklebust[18] らは DSP を用いた並列計算機を製作し，学習セットとネットワーク双方を分割して並列に処理する手法について報告している．Chwan ら [19] は，2 次元格子網と直鎖網上に競合層のニューロンを分割して配置する手法について述べている．

しかし，競合層を分割するこれらの並列化手法では，学習の進展にともなってプロセッサ間の負荷が不均衡になるという問題がある．SOM では学習の進展にしたがって，重み更新を行うユニット数が減少する．このため，競合層を分割して並列化する手法では，学習後半にアイドル状態となるプロセッサが増えてしまい，並列化の効率が悪くなる．また，SOM には初期状態や学習セットにより，学習にまったく関与しないデッドノードの発生問題が指摘されており [20]，こうしたデッドノードもプロセッサ間の負荷の不均一を生じる原因となる．また，画像処理などを行う SOM のように巨大な入力層を必要とする SOM では，従来行われている競合層分割法では十分な高速化が行えない可能性が考えられる．

さらなる高速化の手法として，ニューラルネットワーク自体を VLSI や WSI といったハードウェア上に実装する手法が提案されている [21]．しかし，ハードウェアには製作時に生じる欠陥が避けられず，こうした欠陥があっても動作するようにニューラルネットワークを学習させねばならない．丹ら [22] は階層型ニューラルネットワーク自体が持つ耐故障性に着目し，故障の影響を受けないように学習を行う手法について述べている．Khunasaraphan ら [23] は，故障した結合が持つ重みを他の健全な結合に分担させることにより故障を回避する手法を提案している．當麻 [24] はニューラルネットワークの一部が故障しても，ネットワークが内包する冗長性を用いて他の最適解を探索できることを指摘している．

これらの故障補償法は階層型ニューラルネットワークに対するものであり，SOM の耐故障性についてはいまだ十分議論されていない．また，階層型ニューラルネットワークの故障補償法についても，学習時に故障パターンに対応した重みをあらかじめ準備しておく必要があったり，故障補償を実現する学習に膨大な時間が必要であるなどの問題がある．また，ニューラルネットワークが大規模化したときに，実用的な時間で故障補償が可能であるか十分検討されているとは言えない．

1.1.3 本研究の目的

ニューラルネットワークは入出力間の関係を記述することなく，適切な学習セットを与えれば適切な動作が期待できる．このため，非常に複雑な処理になりがちな人工知能分野などでも応用が試みられている．また，脳自身の動作を解明する一つの手段としてのアプローチも行われつつある．こうした応用での最大の問題は，ネットワーク規模の増大に伴う学習時間爆発である．

並列計算機についての 1.1.1節と，ニューラルネットワークについての 1.1.2節で述べた背景を踏まえ，本研究は,「大規模ニューラルネットワークの高速な並列学習法」を明らかにすることを目的とする．

階層型ニューラルネットワークについては，誤差逆伝搬学習が持つ 3 つの並列性に着目した並列学習モデルをとりあげ，その学習時間について数学的な解析および並列計算機上への実装を通じて詳しく検討する．SOM については入力層に着目した並列学習法を提案する．従来から行われている競合層分割法では，画像などの大きな入力層を必要とする SOM の学習に対して十分な高速化が行えないことを示し，入力層のユニット数が競合層のユニット数より多い SOM において入力層分割法が有効であることを示す．また，入力層分割法を超並列計算機 nCUBE/2 上に実装し，プロセッサ間の負荷不均衡が発生しないことを示す．さらに，SOM の学習において学習パターンを反映しないデッドノードを削除し，より学習パターンに即したマップを生成できる忘却学習法を提案し，その性能について議論する．

超並列計算機で実現できる以上の学習速度を得ようとする場合，ニューラルネットワークを直接ハードウェア上に実装する手法が考えられる．ハードウェアには製造上の欠陥が避けられないため，こうした故障の回避を行うメカニズムを学習法は備えていなければならない．本論文では，ニューラルネットワークをハードウェア上に実装したときに起こりうる故障について検討し，階層型ニューラルネットワークにおいてリンク故障を高速に補償する部分学習法を提案し，その性能を評価する．故障ノードを含んだ SOM に対しては，忘却学習法を拡張することにより故障のない場合と同様に学習が可能であることを示す．

1.2 本論文の構成

本論文は全 7 章より構成されている．はじめにニューラルネットワークについて概説し，階層型ニューラルネットワークの並列学習法と故障補償法，次いで SOM の並列学習法と

故障補償法について議論する．図 1.1に本論文の構成を示し，以下に各章の概要を述べる．

第 2 章 本研究の対象とする階層型ニューラルネットワーク及び SOM について説明する．それぞれについて学習アルゴリズムを解説し，逐次処理を用いた場合の学習時間を概算する．また，従来の学習法での問題点について述べる．

第 3 章 3 種類の階層型ニューラルネットワークの並列学習法について説明し，超並列計算機 nCUBE/2 上に実装した場合の性能を評価する．階層型ニューラルネットワークは従来多くの研究が行われ，並列学習についても様々な提案が行われている．しかし，多くは実装する計算機のアーキテクチャに依存した並列化が行われており，誤差逆伝搬学習が持つ基本的な並列性について十分に比較・検討されていない．本章では誤差逆伝搬学習が持つユニット並列性，学習セット並列性，パス並列性について，モデルによる解析と超並列計算機への実装結果から比較検討を行う．

第 4 章 階層型ニューラルネットワークの故障補償法について述べる．並列誤差逆伝搬学習をさらに高速に実行するプラットフォームとして，VLSI や WSI などのハードウェアが有望視されている．しかし，これらのハードウェアには製造時に生じる欠陥が避けられない．このため，実装されるニューラルネットワークに耐故障性を持たせるか，故障補償を行う再学習メカニズムが必要である．従来の研究では，耐故障性を備えるように重みを決定するためには非常に長い学習時間を要するなどの問題があった．本章では実用的な時間で故障補償を行う部分学習法を提案する．はじめに，製造時に生じる故障がニューラルネットワークに及ぼす影響を定義する．この定義に基づき，実用的な時間でリンク故障の補償を行う部分再学習法を提案し，その性能について XOR 問題を用いた小規模ネットワーク，顔画像認識を行う大規模ネットワークの双方で検討する．

第 5 章 Kohonen の SOM を並列に学習する手法について議論する．SOM では入力層，競合層のユニットが 2 次元平面上に配置されて完全結合しているため，ユニットの増加は大幅な学習時間の増大をもたらす．そこで様々な並列学習モデルによる高速化が試みられている．本章では，はじめに従来行われている SOM の並列化手法で問題となっている負荷の不均一について検討する．次に負荷の均一化を実現し，大規模な入力層を必要とする画像処理などで有効な入力層分割並列学習法を提案し，その効果を検討する．

第 6 章 SOM の学習で問題となるデッドノードの発生に対し，デッドノードを学習するうちに忘却により取り除く，忘却学習法を提案する．デッドノードとは，学習パターンを表現しないユニットであり，学習パターンの分布が偏っている場合などに多く発生する．本章では，デッドノードが発生しやすい構造的分布を持つ学習セットを用い，その性能を評価する．同時に，忘却学習法を故障を持つ SOM に適用して故障補償を実現できることを示す．WSI などのハードウェア実装時に発生する故障ユニットは，学習パターンに反応することがないという観点からデッドノードと見なすことができる．そこで，近傍拡張を伴う忘却学習法を用いて SOM の故障補償を行う手法について議論する．

第 7 章 結言であり，本研究についてまとめる．

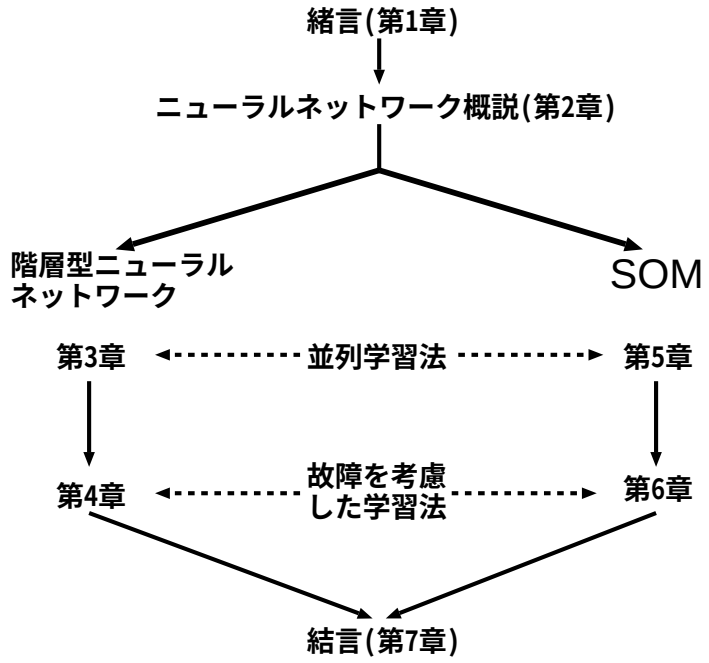

図 1.1: 本論文の構成

第 2 章

ニューラルネットワークと学習法

2.1 はじめに

ニューラルネットワークは基本的なしきい動作を行うユニットが多数結合し，学習を通じて結合の強度を調節することにより複雑な情報処理を可能にしている.「学習」を通じて自律的に情報処理の手法を獲得していくため入出力間の関係を明確に記述する必要がなく，文字や画像の認識，ロボット制御などの分野で一部実用的に使用されるようになってきている.

ニューラルネットワークがどのような動作を行うかは，与えられる情報とその学習アルゴリズムにより決定される．従来より，高速かつ効率的な学習を目指してさまざまな学習アルゴリズムが提案されてきた．本章では代表的なニューラルネットワークの形態として階層型ニューラルネットワークと Kohonen による自己組織化マップ（SOM）をとりあげ，階層型ニューラルネットワークでは誤差逆伝搬学習法，SOM では Kohonen の学習アルゴリズムについて説明する．さらに，学習に必要な時間について検討すると同時に，これらの学習アルゴリズムに存在する問題点について考察する.

2.2 階層型ニューラルネットワークと誤差逆伝搬学習

誤差逆伝搬学習は Rumelhart ら [6] によって提唱された，階層型ニューラルネットワークを学習させる手法である．パーセプトロンでは実現できなかった線型分離不可能な問題を扱うことができ，パターン認識などの分野で広く応用されている．ここでは階層型ニュー

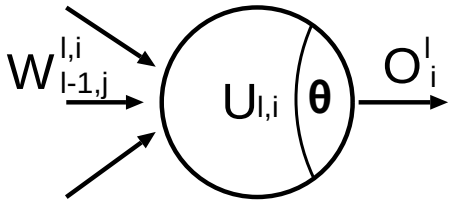

図 2.1: ニューラルネットワークのユニット

ラルネットワークを学習させる最も一般的な学習アルゴリズムである誤差逆伝搬学習について述べ，その問題点を考察する．

2.2.1 階層型ニューラルネットワーク

ニューラルネットワークの説明に先立ち，必要な用語と記号の定義を行う．

定義 2.1 (ニューラルネットワーク) ニューラルネットワークは，しきい素子であるユニット，ユニット間を結合するリンク，各リンクが持つ重みの 3 要素から構成される．

定義 2.2 (ユニット) ネットワークのノードにあたる処理要素であり，人間の脳内ではニューロンにあたる．多数の入力リンクを通じて内部状態を変化させる．図 2.1にその概要を示す．以降，第 l層のユニット i を $u_{l,i}$と表す．

定義 2.3 (リンク) ネットワークのアークにあたる信号伝達経路で，層間のユニットを結合している．人間の脳内では軸索とシナプス結合と呼ばれる．

定義 2.4 (重み) 個々のリンクに付随しており，そのリンクが結合するユニット間の相互作用の強さを表す．ニューラルネットワークの学習は，重みの値を変化させることによ行われる．以降，第 $l-1$ 層のユニット jより，第 l層のユニット i へのリンクに付随する重みを $w^{l,i}_{l-1,j}$と表す．

階層型ニューラルネットワークの例を図 2.2に示す．ニューラルネットワーク内のユニットは階層的に配置され，層間はリンクで結合されている．下位層（入力層）に入力された情報は上位層（出力層）へ向かう方向へ結合しており，各ユニットが完全結合したフィードフォワード型モデルとなっている．同一層内のユニット間にはリンクは存在しない．

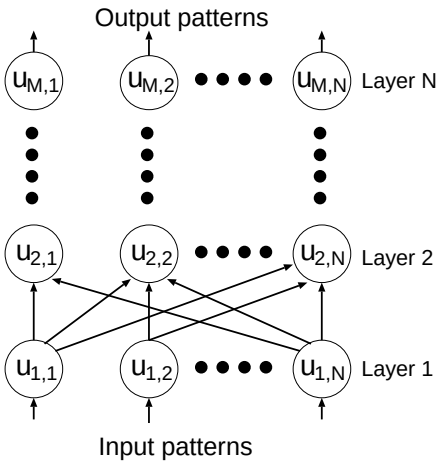

図 2.2: 階層型ニューラルネットワーク

階層型ニューラルネットワークの基本形は、1958 年に Rosenblatt[4] が提案した Perceptron である．ネットワークは S（Sensory），A（Association），R（Response）の 3 層からなる階層的なもので、各層は適当な数のユニットからなり、層内の結合はなく、層間の結合は入力層（S 層）から出力層（R 層）へ向けての一方向のみであった．その後、Minsky, Rumelhart らといった研究者によって階層型ニューラルネットワークについて詳細な解析が行なわれ、現在では次のような能力を持つことが証明されている [25].

- それぞれのユニットがしきい素子で構成される 3 層の階層型ニューラルネットワークで中間層のユニットを必要なだけ使用すれば、任意の 2 値論理関数を実現できる．
- それぞれのユニットがシグモイド状の関数である 4 層のニューラルネットワークによって、中間層のユニットを必要なだけ使用すれば、任意の連続関数 f を任意の精度で近似できる．

以上のように、ニューラルネットワークが持つ能力についての研究が進められる一方で、そうした能力を達成するための学習アルゴリズムの研究も盛んに行われている．次節では、ニューラルネットワークを学習させるアルゴリズムの１つである誤差逆伝搬学習法ついて述べる．

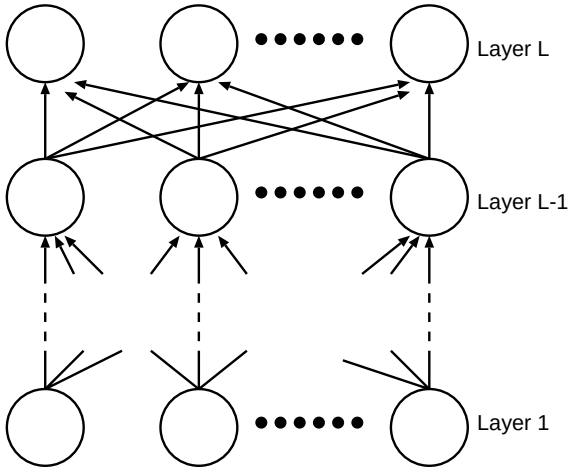

図 2.3: 誤差逆伝搬学習の対象ネットワーク

2.2.2 誤差逆伝搬学習法

Rumelhart ら [6] が発表した誤差逆伝搬学習法は，学習パターンにより提供される教師信号とネットワークの出力との二乗誤差を最小にするよう，すべてのリンクの重みを調節することにより学習を行う．以下では誤差逆伝搬学習のアルゴリズムについて述べる．

図 2.3で示すような，L 層のネットワークを考え，第 1 層を入力層，第 L 層を出力層とする．第 k層の第 m ユニットへの入力の総和を i_m^k，出力を o_m^kとする．各層に n 個のユニットがあると仮定し，各ユニットの入出力関係を fとすると，i_m^kと o_m^kは次式で表される．

$$o_m^k = f(i_m^k) \tag{2.1}$$

$$i_m^k = \sum_n w_{k-1,n}^{k,m} o_n^{k-1} \tag{2.2}$$

ある時刻t でのネットワークの重みを$\boldsymbol{w}_t$，情報源からの入力を$\boldsymbol{x}_t$，$\boldsymbol{x}_t$に対するネットワークの出力を$\boldsymbol{z}(\boldsymbol{w}_t, \boldsymbol{x}_t)$，教師信号を$\boldsymbol{y}_t$，学習パターン $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ による誤差修正量を$\Delta\boldsymbol{w}_t$とすると，時刻 $t+1$ の重み$\boldsymbol{w}_{t+1}$は次式により求められる．

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \Delta\boldsymbol{w}_t \tag{2.3}$$

重みが（2.3）式により更新されて学習が行われるとしたとき，次の定理が証明されている [25]．

定理 2.1 εを正の定数，Cを正定値の行列として，$\boldsymbol{w}$の修正量$\Delta\boldsymbol{w}_t$を

$$\Delta\boldsymbol{w}_t = -\varepsilon C\nabla r(\boldsymbol{z}(w_t, x_t), y_t) \tag{2.4}$$

とする．このとき，εが十分小さい値ならば，学習により，評価 rに対して最適な対応を与える$\boldsymbol{w}$がいくらでも良い精度で得られる．

さて，損失関数 rを誤差の 2 乗とすると，ある学習パターン $(\boldsymbol{x}, \boldsymbol{y})$ が与えられたときの rは（2.5）式で表される．

$$r = \sum_n (o_n^L(\boldsymbol{w}, \boldsymbol{x}) - y_n)^2 \tag{2.5}$$

ここで，$\boldsymbol{w}$はネットワークが持つ重みをすべてまとめたものである．定理 2.1より，$\boldsymbol{w}$の修正量を求めるには rの$\boldsymbol{w}$についての勾配を計算すればよい．

定理 2.1で Cを単位成分ごとに書くと,（2.6）式となる．

$$\Delta w_{k-1,m}^{k,n} = -\varepsilon\frac{\partial r}{\partial w_{k-1,m}^{k,n}} \tag{2.6}$$

（2.6）式を変形すると,（2.7）式を得る．

$$\frac{\partial r}{\partial w_{k-1,m}^{k,n}} = \frac{\partial r}{\partial i_n^k}\frac{\partial i_n^k}{\partial w_{k-1,m}^{k,n}} = \frac{\partial r}{\partial i_n^k} \tag{2.7}$$

$k \neq L$ のとき，fの導関数を f'とすると（2.7）式は（2.8）式のように変形することができる．

$$\begin{aligned}\frac{\partial r}{\partial i_n^k} &= \sum_l \frac{\partial r}{\partial i_l^{k+1}}\frac{\partial i_l^{k+1}}{o_n^k}\frac{\partial o_n^k}{\partial i_l^k} \\ &= \sum_l \frac{\partial r}{\partial i_l^{k+1}} w_{k,n}^{k+1,l} f'(i_n^k)\end{aligned} \tag{2.8}$$

ここで，$\partial r/\partial i_n^k = d_n^k$とおくと，重みの修正量$\Delta w_{k-1,m}^{k,n}$は次式により計算できる．

$$\Delta w_{k-1,m}^{k,n} = -\varepsilon d_n^k o_m^{k-1} \tag{2.9}$$

$$d_n^L = 2(o_n^L - y_n) f'(i_n^L) \tag{2.10}$$

$$d_m^k = \left(\sum_l w_{k,m}^{k+1,l} d_l^{k+1}\right) f'(i_m^k) \tag{2.11}$$

fがすべてのユニットで共通であり,(2.12）式で示すシグモイド関数であるとすると，その導関数 f'は（2.13）式で表される．

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.12}$$

$$f'(x) = f(x)(1-f(x)) \tag{2.13}$$

(2.13）式をニューラルネットワークの入出力 i_m^k, o_m^kで表すと（2.14）式となる．

$$f'(i_m^k) = o_m^k(1-o_m^k) \tag{2.14}$$

重みの修正量を求めるにあたり，振動を減らして学習の収束を早めるために（2.15）式を用いる．

$$\Delta w_{k-1,m}^{k,n}(t+1) = -\varepsilon d_n^k o_m^{k-1} + \alpha \Delta w_{k-1,m}^{k,n}(t) \tag{2.15}$$

ここで，αはモメンタム項と呼ばれる小さな正の定数である．これを，全ての学習パターンについて，誤差が規定以下になるまで繰り返すことにより学習を行う．

2.2.3 誤差逆伝搬学習の学習時間

本節では，計算機により誤差逆伝搬学習を行う時に必要な処理時間について検討する．入力層 L 個，隠れ層 M個，出力層 N個のユニットを持つ $L-M-N$ネットワークを用いて誤差逆伝搬学習を行なうとする．入力層-隠れ層間のリンクが持つ重みは隠れユニット，隠れ層-出力層間のリンクが持つ重みは出力ユニットが持つものとする．学習時間の検討に必要な各処理の単位時間を以下に定義する．

t_{add}: 2 項間の加減算 1 回に必要な時間.

t_{multi}: 2 項間の乗除算 1 回に必要な時間.

t_{act}: 活性値を計算するのに必要な時間.

フォワード パス

ある学習パターンが入力され，各入力ユニットにおいて活性値を計算する．このとき必要な時間 T_{FI}は（2.16）式で表される．

$$T_{FI} = Lt_{act} \tag{2.16}$$

各入力ユニットの活性値は，リンクを通じて隠れ層に送られる．隠れユニットでは,(2.1) 式,(2.2) 式により活性値を計算するため，ここで必要となる時間T_{FH}は (2.17) 式となる．

$$\begin{aligned} T_{FH} &= L(t_{multi} + t_{add}) \times M + t_{act} \times M \\ &= M\left\{L(t_{multi} + t_{add}) + t_{act}\right\} \end{aligned} \tag{2.17}$$

出力層においても隠れ層と同様な計算が行われ，これに要する時間 T_{FO}は (2.18) 式で表される．

$$T_{FO} = N\left\{M(t_{multi} + t_{add}) + t_{act}\right\} \tag{2.18}$$

以上より，誤差逆伝搬学習のフォワードパスで必要な時間 T_{FP}は,(2.16) 式,(2.17) 式,(2.18) 式の和となり,(2.19) 式となる．

$$\begin{aligned} T_{FP} &= T_{FI} + T_{FH} + T_{FO} \\ &= (L + M + N)t_{act} + (LM + MN)(t_{multi} + t_{add}) \end{aligned} \tag{2.19}$$

バックワードパス

出力層では，自身の活性値と教師信号との 2 乗誤差を (2.5) 式，各リンク毎の勾配を (2.10) 式により計算し，重みの修正量を決定する.(2.5) 式の計算に必要な時間は (2.20) 式,(2.10) 式の計算に必要な時間は，活性化関数が全てシグモイド関数であるとして (2.21) 式で表される．

$$N(t_{multi} + 2t_{add}) \tag{2.20}$$

$$N(2t_{add} + 3t_{multi}) \tag{2.21}$$

(2.10) 式により計算された d に従い，各リンク毎の誤差修正量は (2.15) 式により更新される．ユニット一つ当たり M本のリンクを持つので，全出力ユニットが持つ重みの更新量計算と重みの更新に必要な時間は，

$$MN(3t_{multi} + 2t_{add}) \tag{2.22}$$

となり，出力層における重み更新に必要な時間 T_{BO} は，(2.20）式，(2.21）式，(2.22）式の和である（2.23）式で表される．

$$T_{BO} = N\{2(M+1)t_{add} + (3M+4)t_{multi}\} \tag{2.23}$$

隠れ層での重み更新操作は，(2.11）式と（2.15）式に基づいて行われる．隠れ層の 1 ユニットにおいて，(2.11）式の計算に要する時間は $(N+2)t_{multi} + (N+1)t_{add}$ なので，隠れ層全体では（2.24）式となる．

$$M\{(N+2)t_{multi} + (N+1)t_{add}\} \tag{2.24}$$

（2.11）式で計算した d に基づき誤差の修正量を計算して重みを修正するためには，全リンクでは（2.25）式で示される時間が必要である．

$$LM(3t_{multi} + 2t_{add}) \tag{2.25}$$

以上より，隠れ層でのバックワードパスの所要時間 T_{BH} は（2.24）式，(2.25）式の和となり，(2.26）式で表される．

$$T_{BH} = M(3L+N+2)t_{multi} + M(2L+N+1)t_{add} \tag{2.26}$$

以上より，バックワードパスにおける所要時間 T_{BP} は（2.27）式となる．

$$\begin{aligned} T_{BP} &= T_{BO} + T_{BH} \\ &= (3LM + 4MN + 2M + 4N)t_{multi} + (2LM + 3MN + M + 4N)t_{add} \end{aligned} \tag{2.27}$$

フォワードパス，バックワードパスを合わせた誤差逆伝搬学習全体の学習時間 T_{total} は，(2.19）式と（2.27）式の和となり（2.28）式で表される．

$$\begin{aligned} T_{total} &= T_{FP} + T_{BP} \\ &= (L+M+N)t_{act} + (4LM + 5MN + 2M + 4N)t_{multi} \\ &\quad + (3LM + 4MN + M + 4N)t_{add} \end{aligned} \tag{2.28}$$

（2.28）式で示した学習時間は，学習パターン 1 つを処理する時間なので，学習セットに含まれる学習パターン数を N_p，学習に要する Epoch 数を N_{epoch} とすると，最終的に必要な時間 T_{BPL} は，これらを乗じた（2.29）式となる．

$$T_{BPL} = N_p N_{epoch} T_{total} \tag{2.29}$$

2.2.4 誤差逆伝搬学習の問題点

2.2.2節で説明した誤差逆伝搬学習は，階層型ニューラルネットワークにおける強力な学習アルゴリズムではあるが，以下のような問題点も指摘されている．

- 中間層のユニット数の決定を解析的に行うことができない．
- 学習パターンや重みの初期値により学習が不安定になる場合がある．
- 一度学習を行った後，学習パターンの追加などによる再学習が困難である．
- ネットワークの規模が大きくなると，学習に必要な時間が大幅に増加する．

中間層のユニット数決定の問題は，問題を扱うのに十分でかつ最小の中間層ユニット数がヒューリスティックにしか決定できないというものである．甘利ら [26] は，n 次元の入力ベクトル m 個に対し，h 個の隠れユニットを持つ 3 層階層型ニューラルネットワークが処理できる容量は，

$$nh < m < nh \log h$$

という上下限を与えた．しかし，h を制限したときにどのような制約が現れるのかは分かっていない．

学習が不安定という問題については，誤差逆伝搬学習が原理的に持つ問題である．誤差逆伝搬学習は，各学習パターンに対して 2 乗誤差を小さくするよう重みの調整を行うが，これは必ずしも誤差最小となる重みに収束することを保障するものではない．これは局所最小問題とも呼ばれ，扱う問題によっては大きな障害となる．また，重みの初期値によっては学習が収束せず，振動してしまう場合もある．こうした問題は重みの初期値や学習率，モメンタム項を適切に与えることで回避できる場合もあるが，これらのパラメータをあらかじめ最適に決定することは困難である．

誤差逆伝搬学習は，与えられた学習パターンに対する 2 乗誤差を最小にするように動作するアルゴリズムなので，新に学習パターンが追加された場合などは，再度最初から学習をしなおす必要がある．このため，考えられるすべての場合の学習パターンをあらかじめ用意しておかねばらなず，学習環境のダイナミックな変化には対応できない．また，ニューラルネットワークを VLSI などのハードウェア上に実装して高速化を図る場合，ハードウェアの製造上の欠陥のため正常な動作が行えないことが考えられる．この問題に対しては，あ

らかじめあらゆる故障に対応できるようにネットワークを学習させておくか，故障発生時にあらためて再学習を行うしかない．耐故障性を持つようネットワークを学習させる方法では，考えうるすべての組合せに対応できるよう学習をするのは困難である．再学習による重みの修正でも，学習パターンの追加の場合と同様に，ダイナミックな学習変化への対応は困難である．

学習の高速化については，現在最も解決が望まれる問題である．階層型ニューラルネットワークを誤差逆伝搬法で学習させる場合，ユニット数 Nのニューラルネットワークには $\mathcal{O}(N^2)$ のリンクがあることから同数の重みを修正しなければならない．このため，実用規模の問題でネットワークの規模が大きくなる場合には膨大な学習時間が問題となる．

本論文では，3章において並列処理を用いた誤差逆伝搬学習の高速化について検討する．さらに 4章において，ニューラルネットワークを VLSI，あるいは WSI に実装したハードウェアによる高速化において問題となる故障の回避法について部分再学習法を提案し，その性能について評価検討を行う．

2.3 自己組織化ニューラルネットワーク

2.2節で述べた階層型ニューラルネットワークの誤差逆伝搬学習は最急降下法による最適化であり，学習の結果得られる重みとユニットの空間的位置とは特に関係がない．これに対し，Malsburg ら [27] は人間の脳内に見られる視覚野の複雑な構造がどのように得られているかについて考察し，神経細胞の活動に依存した自己組織化によりトポグラフィックマッピングが得られるモデルを提案した．視覚におけるトポグラフィックマッピングとは，2 つの神経場の間の連続的な結合関係のことであり，網膜と視覚野の関係がこれにあたる．Malsburg らのモデルをもとに多くの研究者が様々なモデルを提案しているが，倉田 [28] はこれらのモデルが以下の 3 つの条件を基本にしているという点で共通していることを指摘している．

- 入力層に与えられるパターン群（信号空間）は，パターン間の内積の大きさにより定義されるトポロジーを持つ．
- 出力層で近くの細胞同士は興奮性の結合で結ばれ，離れた細胞間は抑制性の結合で結ばれる．これにより，ある細胞が発火すると周りの細胞も同時に発火する．出力層は発火パターンにより定義されるトポロジーを持つ．

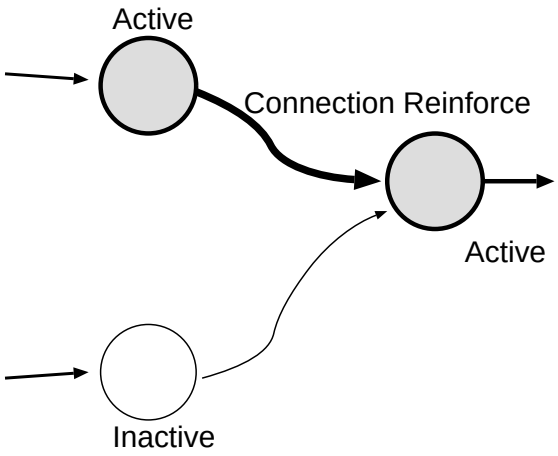

図 2.4: Hebb の学習則

- 入力層と出力層は重み付きリンクで結ばれ，重みの値は Hebb の学習則で更新される．

1 番目の条件は，隣合った細胞は同時に発火することが多いという網膜上で見られる現象を理想化して，入力層で隣合った一群の細胞を同時に発火させるような入力を考えるということである．したがって，学習パターン間の内積（類似度）が大きいものは，出力層でも隣接したニューロンを発火させることになり，トポグラフィクマッピングが行われることになる．

2 番目の条件は，生成されるトポグラフィックマップをより明確にするためのものであり，ある学習パターン群に反応するニューロンを結集させ，反応しないニューロン群を離反させる作用をもつ．

3 番目の条件にある Hebb の学習則 [3] の概念を図 2.4に示す．Hebb の学習則は，2 つの細胞が同時に発火したとき，これらの細胞間のシナプス結合が強化されるという仮説である．これは条件反射を説明するために導入されたものであるが，多くのニューラルネットワークの学習理論の基礎となっている．実際に，Kandel ら [29] は海馬などの脳の多くの場所で Hebb の学習則に従ったシナプス変化が観測できることを報告している．

以下の節では，代表的な自己組織化ニューラルネットワークである Kohonen の自己組織化マップ（SOM）について説明する．同時に，SOM が持つ問題点を検討する．

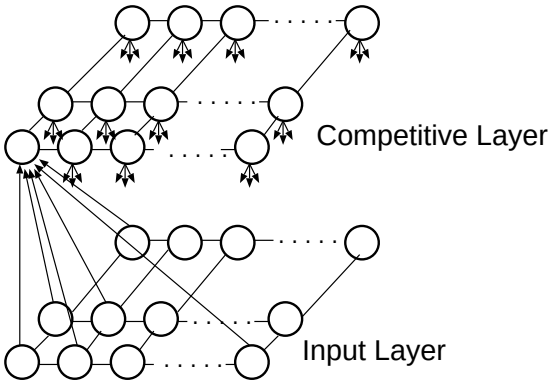

図 2.5: 自己組織化マップ

2.3.1 Kohonen の自己組織化マップ（SOM）

T.Kohonen[16, 30] により提案された自己組織化マップは，脳の機能部位で見られる情報のトポグラフィックマッピングを自律的に行うアルゴリズムである．Kohonen により提案された SOM は図 2.5に示すように 2 層の構造を持ち，それぞれ入力層，競合層と名付けられている．

入力層-競合層間は完全結合しており，競合層の各ユニットは同一の入力を受ける．競合層の各ユニットは結合重み$\boldsymbol{w}_i$を持っており，この重みを変更していくことでトポグラフィックマッピングを実現する．Kohonen による SOM の学習法について，以下に説明する．

$\boldsymbol{x} \in \mathcal{R}^n$を確率的な入力ベクトルとする．このとき，SOM は高次元入力データベクトル$\boldsymbol{x}$の確率密度関数 $p(\boldsymbol{x})$ の 2 次元表示上への非線型写像であるということができる．ベクトル$\boldsymbol{x}$は全ての$\boldsymbol{w}_i$と比較され，多くの場合，ユークリッド距離 $||\boldsymbol{x} - \boldsymbol{w}_i||$ を最小にするユニットが最整合ユニット (ウィナー）c として（2.30）式により定義される．

$$\begin{aligned} c &= argmin_i\{||\boldsymbol{x} - \boldsymbol{w}_i||\} \\ ||\boldsymbol{x} - \boldsymbol{w}_c|| &= min_i\{||\boldsymbol{x} - \boldsymbol{w}_i||\} \end{aligned} \tag{2.30}$$

競合層上で c の近傍にあるユニットは，c からの距離に反比例した強度で発火し，結合重みを（2.31）式により変更する．

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + h_{ci}(t)(\boldsymbol{x}(t) - \boldsymbol{w}_i(t)) \tag{2.31}$$

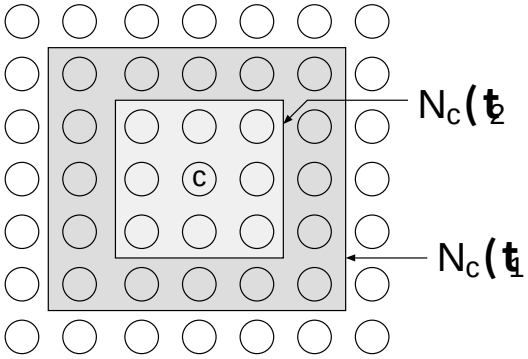

図 2.6: 近傍集合

ここで，t は離散時間座標であり，$h_{ci}(t)$ は近傍関数と呼ばれて Kohonen の SOM では中心的な役割をする．(2.31）式が収束するためには，$t \to \infty$ のときに $h_{ci}(t) \to 0$ であることが必要である．多くの場合，$h_{ci}(t) = h(||\boldsymbol{r}_c - \boldsymbol{r}_i||, t)$ とされる．ここで，$\boldsymbol{r}_c \in \mathcal{R}^2, \boldsymbol{r}_i \in \mathcal{R}^2$ は それぞれ競合層の中でのユニット c と i の位置ベクトルであり，$||\boldsymbol{r}_c - \boldsymbol{r}_i||$ が増加するにつれて $h_{ci} \to 0$ とする．

h_{ci}としては主に以下の 2 種類が用いられる．一つは，c の周りのユニットを近傍集合 N_c とするものである．図 2.6 にその概念を示す．N_cは時間の関数であり，$N_c(t)$ と表され，時間が進むにつれて単調減少する．このとき，近傍関数 h_{ci}は（2.32）式で定義される．

$$h_{ci} = \begin{cases} \alpha(t) & i \in N_c \\ 0 & \text{otherwise} \end{cases} \tag{2.32}$$

（2.32）式において，$\alpha(t)$ は学習率関数と呼ばれて $0 < \alpha(t) < 1$ の範囲の値をとり，$N_c(t)$ と同じく時間について単調減少する．

（2.32）式よりも滑らかな近傍を得るために,（2.33）式が用いられる場合がある．

$$h_{ci} = \alpha(t) \exp\left(-\frac{||\boldsymbol{r}_c - \boldsymbol{r}_i||^2}{2\sigma^2(t)}\right) \tag{2.33}$$

ここで$\sigma(t)$ は近傍の幅を定義し，時間に関して単調減少する．

十分な数の$\boldsymbol{x}$を用いて競合学習を行うことにより，競合層の各ユニットが持つ重みを参照ベクトルとするボロノイ・モザイク領域が形成される．図 2.7にボロノイ・モザイク領域の例を示す．ボロノイ・モザイク領域は線（一般には超平面）で境界がつけられており，図 2.7で示すように複数の領域に分割されている．各領域内には，同一領域 A_n内のどのベク

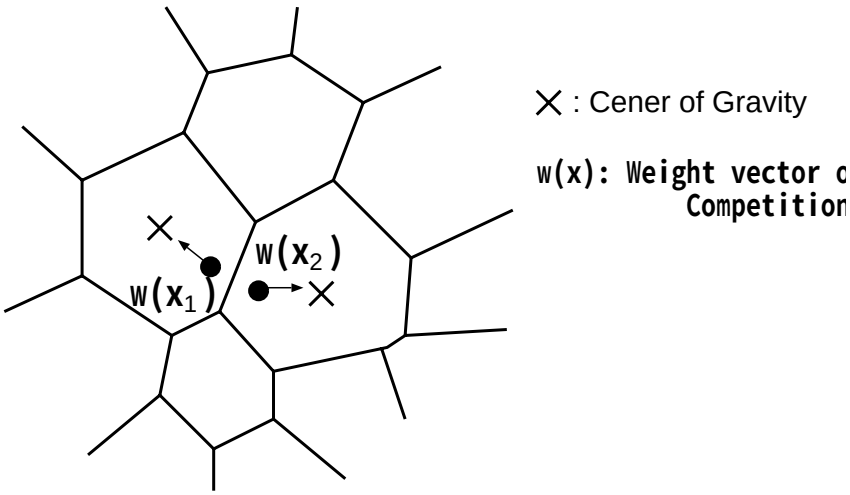

図 2.7: ボロノイ・モザイク領域

トル$\boldsymbol{x} = (\xi_1, \xi_2)$ に対しても「最近接近傍」となる参照ベクトル$\boldsymbol{w}_n$が存在する．すなわち，競合層のユニットを発火させる入力ベクトルの集合毎に入力信号空間が分割されていく．

2 つのユニットが持つ重みベクトル$\boldsymbol{w}_1$，$\boldsymbol{w}_2$が同一領域内で並んでいるような場合では，これらのユニットの受け持ち領域の境界は$\boldsymbol{w}_1$，$\boldsymbol{w}_2$を結ぶ線分の垂直 2 等分線の一部となる．別々の領域にある$\boldsymbol{w}_1$，$\boldsymbol{w}_2$が極めて接近しているとすると，これらのベクトルはそのユニットの受け持ち領域の境界近くに位置することになる．したがって，受け持ち領域の重心は重みベクトルから見て境界の反対側に位置し，2 つの重みベクトルはその重心方向へ移動するため$\boldsymbol{w}_1$，$\boldsymbol{w}_2$間にはみかけ上斥力が働き，重みベクトルは結果として信号空間一様に広がることになる．同時に近くに位置するユニットは近傍関数によって類似した入力に反応するため，隣接したユニットの重みベクトル間には引力が働く．この斥力と引力により，重みベクトルは発散せず，同時に集中することなくトポグラフィックマッピングを行うことができる．

2.3.2 SOM の学習時間

2.3.1節で述べた Kohonen の自己組織化マップを逐次処理により実行するときの学習時間について考察する．入力層，競合層はそれぞれ $M \times M$，$N \times N$の 2 次元配列を構成しており，それぞれのユニットの活性値は考えないものとする.. また，近傍関数は簡単化のた

め（2.32）式を用い，重み更新の対象となる近傍ユニット $N_c(t)$ は（2.34）式で表されるとした．

$$N_c(t) = N_c(0)(1.0 - \frac{t}{T}) \tag{2.34}$$

ここで，Tは最大学習回数とし，$N_c(0) = \frac{N}{2}$とする．

入力層に提示された学習パターンは，そのまま競合層の各ユニットへの結合を通じて送られる．送られた入力パターンに対し，競合層の各ユニットでは結合重みとの距離 d_iを計算する．d_iをユークリッド距離とすると，これに必要な時間 T_dは次式で与えられる．

$$T_d = \left\{ M^2(t_{add} + t_{multi}) + t_{sq} \right\} N^2 \tag{2.35}$$

ここで，t_{sq}は平方根を計算するのに必要な時間である．次に，得られた d_iが最も小さいユニット（ウィナー）の決定を行う．これに必要な時間を $T_s = t_s$とする．ウィナーを決定した後に近傍の範囲を（2.34）式により決定し，同時に学習率関数による学習率の決定を行う．近傍範囲，および学習率を決定するのに必要な時間 T_nは（2.36）式で表される．

$$T_n = 2(t_{add} + 2t_{multi}) \tag{2.36}$$

さて,（2.34）式で求められる値は近傍範囲の直径であり，この範囲内にあるユニット数は $N_c^2(t)$ である．しがたって，$N_c^2(t)$ 個の競合層のユニットの重み変更に必要な時間 T_{uw}は次式で表される．

$$T_{uw} = M^2(2t_{add} + t_{multi})N_c^2(t) \tag{2.37}$$

したがって，N_p個の学習パターンを用いて N_{epoch}回の学習を行う場合の学習時間 T_{total}^{som}は次式で表される．

$$\begin{aligned} T_{total}^{som} &= \sum_{t=1}^{N_{epoch}} N_p\{(M^2N^2 + 2M^2N_c^2(t) + 2)t_{add} \\ &\quad +(M^2N^2 + M^2N_c^2(t) + 4)t_{multi} + N^2t_{sq} + t_s\} \end{aligned}$$

2.3.3 SOM の問題点

トポグラフィックマッピングを実現する最も簡単なモデルとして，Kohonen の SOM について述べた．SOM は画像符号化分野でのベクトル量子化（VQ）などに用いられ，一定

の評価を得た．しかしながら，Kohonen 自身や他の研究者により，以下の問題点が指摘されている．

- 学習率関数や近傍関数の決定が困難である．
- 入力の構造により，マッピングの品質が劣る場合がある．
- 学習に全く関与しないデッドノードの発生がある．
- 2 次元の入力層，競合層を持ち，それぞれが完全結合しているため，ネットワークの規模が大きくなると膨大な計算時間が必要となる．

学習率や近傍半径の初期値は生成されるマップの品質を決める重要な値であるが，慣習的に決定されているのが現状である．Kohonen は学習率や近傍半径の解析的な決定法についても検討しているが [16]，現在でも解決されているとは言えない．

入力がある構造を持つとき，生成されるマップの品質が悪くなる場合がある． たとえば，入力ベクトルの分布が突出した形状をしているとき，競合層上のウィナーが一部のユニットに集中する傾向がある．また，ゼロ密度領域に存在する重みベクトルを持つユニットは，非ゼロ密度領域全域からの入力ベクトルにより影響を受けやすい．この問題に対して Kangas ら [20] は 2 次元のトポグラフィックマッピングをやめ，ユニットの隣接関係を最小結合木構造（MST）で表現する手法を提案し，学習パターン分布の形状によっては効率的にトポグラフィックマップが生成できることを示した．しかし，この手法では競合層ユニットの空間的隣接関係に基づく位相関係の定義を放棄することになる．

また，学習ベクトルが分布する領域数が競合層のユニット数より少ないとき，まったく学習に関与しないデッドノードが発生する場合がある．Choi ら [31] は，学習の初期には競合層のユニットは 1 つだけとし，学習に進展に伴ない，必要に応じてユニットを木状に追加していく手法を提案した．その結果，より少ないユニット数で学習セットを表現することが可能であることを示した．しかし，ユニットの追加・削除が随時行われるために処理が複雑になり，並列化などの高速化が難しいという問題点がある．

SOM では入力層，競合層が共に平面をなしているため，マップの大規模化を行うと計算量が膨大となってしまう．(2.38) 式で示したように，各層のユニット数を $n \times n$ とすると，SOM の学習には $\mathcal{O}(n^4)$ という膨大な計算量が必要となり実用的ではない．この問題に対しては 5.2節で述べるように並列処理による高速化が多く提案されている [17, 18, 19]．しか

し，任意のユニット数に対応することが難しく，学習の進展により重み更新を行うユニット数が減少するため負荷のばらつきが大きくなるなどの問題がある．

本論文では，5章において並列処理を用いたSOMの高速化手法について述べる．従来の並列化と異なって入力層に着目した分割を行うことにより，各プロセッサの負荷の均等化を実現し，より大きな入力層を持つ大規模SOMを高速に学習できることを示す．さらに6章において，学習に関与しないユニットに対して忘却を導入することによりデッドノードを削除する手法について検討し，同時に忘却による学習が故障ユニットを含むSOMの学習についても有効であることを示す．

2.4 まとめ

本章では，ニューラルネットワークの代表的形態として階層型ニューラルネットワークとSOMを取りあげ，それぞれを学習させるアルゴリズムである誤差逆伝搬学習法とKohonenによる学習法について説明した．同時に，各学習に必要な学習時間について検討し，その問題点について考察した．

2.2.1節では，パターン認識などに広く応用されつつある階層型ニューラルネットワークについて，また2.2.2節では，階層型ニューラルネットワークを学習させる誤差逆伝搬学習について説明した．さらに，2.2.3節において，誤差逆伝搬学習を逐次処理で行う際に必要な計算時間について解析した．また，2.2.4節において誤差逆伝搬学習が持つ問題点について検討した．

2.3節では，近年の大脳生理学の発達を受けて提案されてきた自己組織化ニューラルネットワークについて概説した．2.3.1節では代表的な自己組織化ニューラルネットワークであるKohonenのSOMについて説明し，トポグラフィックマップが生成されるメカニズムについて述べた．2.3.2節では，誤差逆伝搬学習のときと同様にSOMの学習に必要な時間について解析を行い，逐次処理では大規模なSOMの学習が不可能であることを示した．また，2.3.3節ではSOMが持つ問題点について説明した．

ニューラルネットワークの研究では，その動作の解析的な検討とともに実際の応用面での振舞いから新しい情報処理のスキームを得ようという研究も多い．特に視覚系のシミューレションなど大規模ネットワークによる複雑な情報処理が数多く行われており，処理の高速化が強く要求されている．次章では，並列処理による誤差逆伝搬学習の高速化について検討を行う．

第 3 章

階層型ニューラルネットワークと並列学習

3.1 はじめに

2.2.4節で述べたように，階層型ニューラルネットワークで誤差逆伝搬学習を用いる場合の最大の問題点は，学習に必要な時間が膨大なものになるということである．特に実用規模の問題を扱う場合にはネットワークの規模も大きくなり，スーパーコンピュータのような高速な計算機を用いたとしても実用的な時間で解を得ることは困難である．

近年，ニューラルネットワークが持つ並列性に着目し，多数のプロセッサを相互に結合した超並列計算機上での並列学習法が注目されるようになってきた．Zhang ら [12] は，ハイパーキューブ網を持つ超並列計算機 Connection Machine 2(CM-2) 上にニューラルネットワークを実装する手法を提案した． Muller ら [13] は，学習パターン空間および重み空間双方を分割・並列化し，Witbrock ら [14] は学習パターン空間を分割して超並列計算機 GF11 上に実装する手法について報告している．

しかしながら，多くの報告は対象とする計算機に特化した実装法の提案を目的としたものであり，誤差逆伝搬学習そのものが持つ並列性の基本的な評価は不十分である．本章では，誤差逆伝搬学習が持つ複数の並列性に注目し，3 種類の並列学習法について評価検討を行う．3.2節では，対象とする並列誤差逆伝搬学習法をモデル化し，学習に必要な時間を解析的に導出する．3.3では，導出した各並列学習法の学習時間について比較検討を行う．3.4 節では，超並列計算機 nCUBE/2 上に 3 種類の並列学習アルゴリズムを実装して，学習速度性能の評価検討を行う．

3.2 並列誤差逆伝搬学習法

誤差逆伝搬学習法には，学習時間短縮のため様々な並列化が考えられているが，これらの基礎となっている並列モデルとして以下の 3 つが挙げられる．

ユニット並列モデル: （2.1）式で示される各ユニットの出力計算において，k, m の双方で並列化を行う手法.

学習セット並列モデル: （2.3）式で示される重みの更新の線型性を利用し，学習セットを分割して並列化を行う手法.

パス並列モデル: 誤差逆伝搬学習におけるフォワードパス，バックワードパスの 2 つのフェーズを分割して並列化を行う手法.

本節では，上の 3 つの各並列学習法について説明し，2.2.3節と同様に学習時間の検討を行う．問題を並列化して処理する場合，PE 間の通信が数多く発生する．学習時間の検討を行うにあたり，PE 間で行われる通信時間を以下のように定義する．

t_{comm}: 2PE 間の通信で，一方が情報を送信，他方が情報を受信するために必要な時間．通信路確保などのセットアップ時間も含める．

3.2.1 ユニット並列モデル

ニューラルネットワーク内の各ユニットをそれぞれ並列に動作させて学習を行う方法をユニット並列モデルと呼ぶ．誤差逆伝搬学習では，各ユニットは自身への入力が揃い次第，(2.1）式により出力を計算する．すなわち，出力計算に関して，層番号 k，ユニット番号 m 双方に関して並列化を行なう．誤差の計算終了後,(2.15）式に従って重みの更新を行う．このときも出力計算時と同様に，層番号 kとユニット番号 m に関して並列化を行うモデルである．

フォワードパス

入力ユニットでの活性値は（2.16）式で計算される．ユニット並列モデルでは，L 個の PE で並列に処理するため，処理時間 T_{FI}^{up}は次式で表される．

$$T_{FI}^{up} = T_{FI}/L = t_{act} \tag{3.1}$$

入力層から隠れ層への通信では，M個の隠れユニットが並列に，L 個の入力ユニットの活性値を受けとる．したがって，入力層-隠れ層間の通信時間 T^{up}_{FIH}は（3.2）式で表される．

$$T^{up}_{FIH} = Lt_{comm} \tag{3.2}$$

隠れ層，出力層においても,（2.17）式,（2.18）式をそれぞれ M個，N 個の PE で計算する．したがって，ユニット並列モデルにおける隠れ層，出力層の出力値計算に必要な時間 T^{up}_{FH}，T^{up}_{FO}は次式のようになる．

$$T^{up}_{FH} = L(t_{multi} + t_{add}) + t_{act} \tag{3.3}$$

$$T^{up}_{FO} = M(t_{multi} + t_{add}) + t_{act} \tag{3.4}$$

隠れ層から出力層への通信時間 T^{up}_{FHO}は，N個の PE が並列に M回の通信を行うために,（3.5）式となる．

$$T^{up}_{FHO} = Mt_{comm} \tag{3.5}$$

以上から，ユニット並列モデルのフォワードパスにおける所要時間 T^{up}_{FP}は（3.6）式となる．

$$\begin{aligned} T^{up}_{FP} &= T^{up}_{FI} + T^{up}_{FH} + T^{up}_{FO} + T^{up}_{FIH} + T^{up}_{FHO} \\ &= 3t_{act} + (L + M)(t_{multi} + t_{add} + t_{comm}) \end{aligned} \tag{3.6}$$

バックワードパス

ユニット並列モデルでは，複数の PE に各出力ユニットの活性値と教師信号が割り当てられる．したがって，ある学習パターンによる誤差を計算するには，各 PE が持つ 2 乗誤差を 1ヶ所に集める必要がある．ここでは図 3.1に示すように出力ユニットは一つのホストノードと接続しており，各出力ユニットはホストノードへ自分が担当する要素の 2 乗誤差を送り，ホストノードからその加算結果を受けとる，というモデルを仮定する．

ユニット並列モデルの出力ユニット N個が並列に 2 乗誤差を計算するのに必要な時間は $(t_{add} + t_{multi})$，ホストノードが N個の PE からデータを受け取るのに Nt_{comm}，ホストノードが送られたデータ N 個を加算するのに Nt_{add}，ホストノードから出力ユニットへのデー

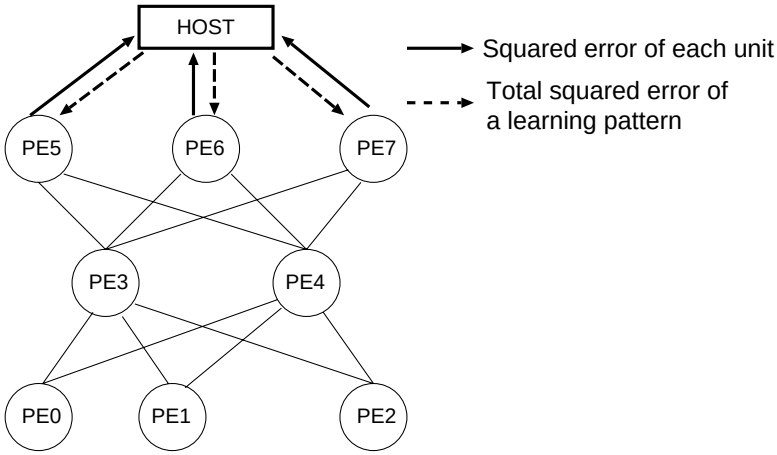

図 3.1: 出力ユニットとホストノード

タ送信には Nt_{comm} の時間が必要である．したがって，出力層での 2 乗誤差の計算に必要な時間 T_{Br}^{up} は（3.7）式となる．

$$\begin{aligned} T_{Br}^{up} &= (t_{add} + t_{multi}) + Nt_{comm} + Nt_{add} + Nt_{comm} \\ &= (N+1)t_{add} + t_{multi} + 2Nt_{comm} \end{aligned} \tag{3.7}$$

続いて各出力ユニットでの d を（2.10）式に従って並列に計算するのに必要な時間 T_{BOd}^{up} は（3.8）式，重みの修正量 $\boldsymbol{\Delta w}$ を（2.15）式により計算するのに必要な時間 T_{BOdw}^{up} は（3.9）式で表される．

$$T_{BOd}^{up} = 2t_{add} + 3t_{multi} \tag{3.8}$$

$$T_{BOdw}^{up} = t_{add} + 3t_{multi} \tag{3.9}$$

計算した重みの修正量により重みを修正する加算の時間を考え，これを M 個の重みについて行なうのに必要な時間 T_{BOuw}^{up} は（3.10）式となる．

$$T_{BOuw}^{up} = M(2t_{add} + 3t_{multi}) \tag{3.10}$$

出力層-隠れ層間では，出力ユニットが計算した d を受けとるための通信が必要である．これに必要な時間 T_{BOHd}^{up} は，M 個の PE が並列に N 個のデータを受信する時間に等しいの

で（3.11）式で表すことができる．

$$T^{up}_{BOHd} = Nt_{comm} \tag{3.11}$$

隠れユニットでは，出力層より受け取った d をもとに（2.11）式を用いて自身の d を計算する．これに必要な時間 T^{up}_{BHd}は（3.12）式となる．

$$T^{up}_{BHd} = (N+1)T_{add} + (N+2)T_{multi} \tag{3.12}$$

次に，重みの修正量を計算して重みを更新する操作が L 回行われるので，これに必要な時間 T^{up}_{BHuw}は（3.13）式となる．

$$T^{up}_{BHuw} = (2L+N+1)T_{add} + (3L+N+2)t_{multi} \tag{3.13}$$

以上より，ユニット並列モデルにおけるバックワードパスに要する時間 T^{up}_{BP}は（3.14）式で表される．

$$\begin{aligned} T^{up}_{BP} &= T^{up}_{Br} + T^{up}_{BOd} + T^{up}_{BOuw} + T^{up}_{BOHd} + T^{up}_{BHd} + T^{up}_{BHuw} \\ &= (2L+2M+3N+5)t_{add} + (3L+3M+2N+8)t_{multi} + 3Nt_{comm} \end{aligned} \tag{3.14}$$

したがって，ユニット並列モデルを用いて学習パターン一つを並列に学習するのに要する時間 T^{up}_{total}は（3.15）式となる．

$$\begin{aligned} T^{up}_{total} &= T^{up}_{FP} + T^{up}_{BP} \\ &= 3t_{act} + (3L+3M+3N+5)t_{add} \\ &\quad + (4L+4M+2N+8)t_{multi} + (L+M+3N)t_{comm} \end{aligned} \tag{3.15}$$

以上より，ユニット並列モデルを用いて，N_p個の学習パターンを N_{epoch}回学習するのに必要な時間 T^{up}は次式で求められる．

$$T^{up} = N_{epoch}N_pT^{up}_{total} \tag{3.16}$$

3.2.2 学習セット並列モデル

学習セット並列モデル

誤差逆伝搬学習では，重みの更新は（2.3）式によって行なわれる．これは，ある学習パターンによる重みの修正量はその重みに線型に加算されることを意味している．この線型性より，学習セットを複数に分割し，同一のネットワーク構成を持つ複数の PE により誤差修正量を計算した後，重みの更新を行う．したがって，重みの更新は学習パターン毎に行われるのではなく，学習セット毎に行われる．こうした重みの更新方法をバッチ更新と呼ぶ．

学習セット並列モデルでは，複数の PE で$\Delta \boldsymbol{w}_t$を並列に計算した後，まとめて$\boldsymbol{w}_t$に加算するため，(2.3）式は以下のように変形される．

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \sum_j \Delta \boldsymbol{w}_t^j \tag{3.17}$$

ここで，w_t^jは j番目の PE で計算された$\Delta \boldsymbol{w}_t$であり，PEjが担当する学習パターン数をN_{lp}^j，学習パターン p による重みの修正量を$\Delta \boldsymbol{w}_t^{j,p}$とすると（3.18）式で表される．

$$\Delta \boldsymbol{w}_t^j = \sum_{p=1}^{N_{lp}^j} \Delta \boldsymbol{w}_t^{j,p} \tag{3.18}$$

学習セット並列モデルの計算時間の見積りを以下に示す．ここで用いる記号の定義は2.2.3節と同じである．

フォワードパス

各 PE に異なる学習パターンが提示され学習を行うとき，各 PE で行われるフォワードパスは通常の誤差逆伝搬学習と全く同じである．したがって，学習セット並列モデルでのフォワードパスに要する時間 T_{FP}^{lp} は（3.19）式である．

$$\begin{aligned} T_{FP}^{lp} &= T_{FP} \\ &= (L+M+N)t_{act} + (LM+MN)t_{add} + (LM+MN)t_{multi} \end{aligned} \tag{3.19}$$

バックワード パス

学習セット並列モデルでは，各 PE で担当する学習パターンによる重みの修正量を加算しておき，1Epoch 分の学習が終了したあと各 PE が持つ修正量を加算し，その結果をもとに全 PE が持つ重みが一斉に変更される．

各 PE で加算されてきた重みの修正量はホストノードに送られ，そこで加算されて全学習パターンに対する誤差の修正量を決定後，各 PE にブロードキャストすると仮定する．すると，各 PE では重みの修正量の計算までを行うことになり，学習セット並列モデルでのバックワードパスに要する時間 T_{BP}^{lp}は（3.20）式となる．

$$T_{BP}^{lp} = (LM + 2MN + M + 4N)t_{add} + (3LM + 4MN + 2M + 4N)t_{multi} \tag{3.20}$$

学習セットを分割して別々の PE で処理するため，N_p個の学習パターンを一通り処理するのに要する時間 T_{FPBP}^{lp}は（3.21）式となる．

$$\begin{aligned} T_{FPBP}^{lp} &= \frac{N_P}{N_{PE}}(T_{FP}^{lp} + T_{BP}^{lp}) \\ &= \frac{N_p}{N_{PE}}\{(L + M + N)t_{act} + (2LM + 3MN + M + 4N)t_{add} \\ &\quad +(4LM + 5MN + 2M + 4N)t_{multi}\} \end{aligned} \tag{3.21}$$

各 PE での学習パターンの処理が修了した後，各 PE は計算しておいた重みの修正量をホストに送り，ホストでこれらを加算して新しい重みを決定し，その結果を各 PE に送信する．各 PE が持つ重みの修正量は $(LM + MN)$ 個であるので，重みの更新処理に必要な時間 T_{UW}^{lp}は（3.22）式で表される．

$$T_{UW}^{lp} = (LM + MN)t_{add} + 2(LM + MN)t_{comm} \tag{3.22}$$

以上より，学習セット並列モデルを用いて N_p個の学習パターンを持つ学習セットを N_{epoch} 回学習するのに必要な時間 T_{total}^{lp}は（3.23）式となる．

$$\begin{aligned} T_{total}^{lp} &= N_{epoch}(T_{FPBP}^{lp} + T_{BP}^{lp}) \\ &= N_{epoch}\frac{N_p}{N_{PE}}\{(L + M + N)t_{act} + (2LM + 3MN + M + 4N)t_{add} \end{aligned}$$

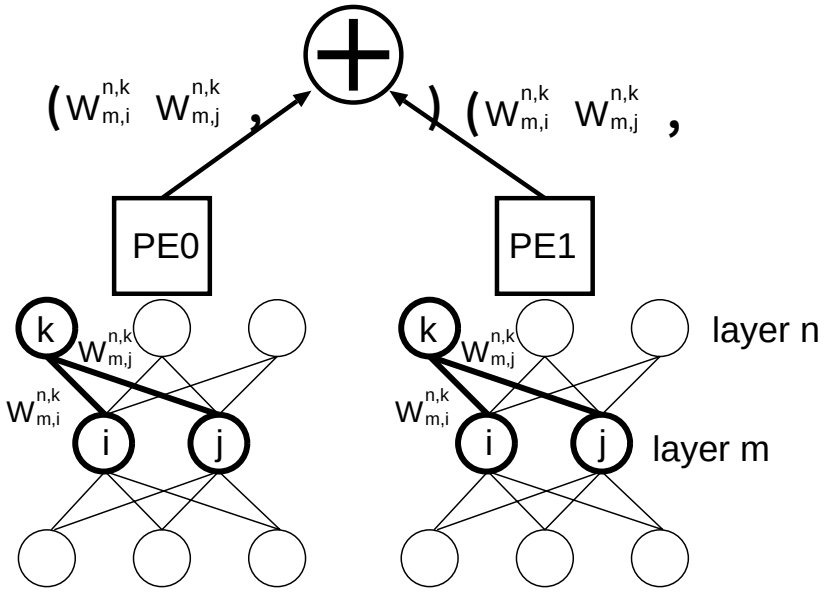

図 3.2: 複数の重みの同時通信と加算

$$+(4LM + 5MN + 2M + 4N)t_{multi}\}$$
$$+N_{epoch}\{(LM + MN)t_{add} + 2(LM + MN)t_{comm})\} \quad (3.23)$$

（3.23）式を見ると，通信時間が PE 数やメッセージ長に依存しない理想並列計算機では，PE 数 N_{PE} を増やせば線型に処理時間が減少するはずである．実際には同期通信のよる処理のブロックや通信網形態の違いから，PE 数に比例した速度向上率が得られない．さらなる高速化を図るためには，大規模ネットワークでは特に問題となりうる通信回数 $(LM+MN)$ を少なくする必要がある．3.2.3節では，通信回数を減らす手法として改良学習セット並列モデルを提案し，その性能を評価する．

3.2.3 改良学習セット並列モデル

3.2.2節で述べたように，学習セット並列モデルを用いた並列誤差逆伝搬学習を効率良く行うには通信回数を大幅に減らさなければならない．そこで，複数の重みをまとめて一度に通信し，加算を行なう改良学習セットモデルを提案し，その性能について議論する．

複数の重みをまとめて通信するときの概念（配列加算通信）を図 3.2に示す．本手法では，上位層が持つ入力リンクをまとめて通信して加算を行なう．これにより，(3.22）式の通信時間と重み更新項の係数 $(LM + MN)$ は $(M + N)$ となり，PE 間通信の影響を減らすこと

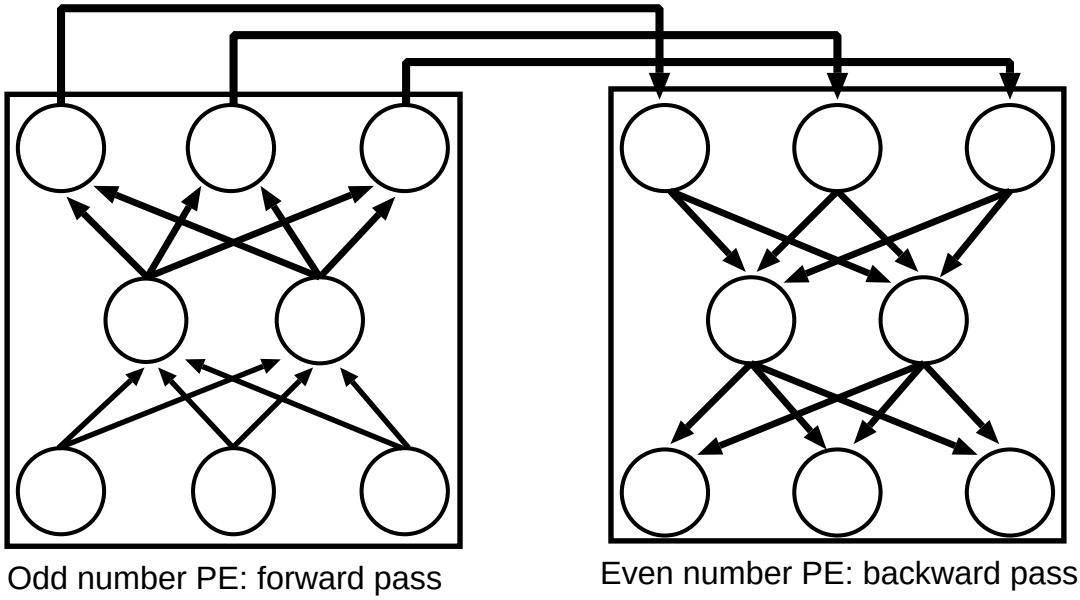

図 3.3: パス並列モデル

ができる．

3.2.4 パス並列モデル

パス並列モデルは，誤差逆伝搬学習におけるフォワードパス，バックワードパスをそれぞれ別の PE で処理する手法である．ユニット並列モデルは（2.1）式の m，学習セット並列モデルは（2.3）式における$\Delta \boldsymbol{w}_t$の計算を並列に行うモデルであるが，パス並列モデルは（2.1）式の kについて並列化を行うモデルである．

図 3.3にパス並列モデルの概念を示す．3 層の階層型ニューラルネットワークをパス並列モデルで並列化した場合，入力層から隠れ層へ活性値を送る間に，出力層から隠れ層へ重みの修正量を送ることになる．したがって，パス並列モデル単独では大幅な高速化は達成できないため，ここでは改良学習セット並列モデルと組み合わせて使用した．

次に，パス並列モデルの計算時間について検討する．2.2.3節と同様の定義により，N_p個の学習セットを持つ問題を N_{PE}個の PE を用いて，パス並列モデルに改良学習セット並列モデルを組み合わせて学習する場合を考える．

パス並列モデルでは，2 つの PE を組にしてフォワードパス，バックワードパスを分担

して行う．このため，より長い処理を行うPEの処理時間が各PEの組での処理に必要な時間である．学習セット並列モデルによる誤差逆伝搬学習のフォワードパスの所要時間は（3.19）式，バックワードパスでの所要時間は（3.20）式である.（3.19）式より(3.20）式を引いて式を整理すると,（3.24）式となる．

$$\begin{aligned} T_{FP}^{lp} - T_{BP}^{lp} &= (L+M+N)t_{act} + (LM+MN)t_{add} + (LM+MN)t_{multi} \\ &\quad -(LM+2MN+M+4N)t_{add} - (3LM+4MN+M+4N)t_{multi} \\ &= (L+M+N)t_{act} - (MN+M+4N)t_{add} \\ &\quad -(2LM+3MN+M+4N)t_{multi} \end{aligned} \tag{3.24}$$

ここで，$L=M=N=n$とおき，活性値の計算には乗除算，加減算の20倍の時間が必要であるとし，$t=\frac{t_{act}}{20}=t_{add}=t_{multi}$とおく[1]．この仮定の元で（3.24）式を簡単化すると次式が得られる．

$$T_{FP} - T_{BP} = -nt(6n-49) \tag{3.25}$$

$n>0, t>0$なので（3.25）式が正となる条件は$0<n<\frac{49}{6}\simeq 8$となり，ネットワークのユニット数24以上ではバックワードパスの時間の方が長いことが分かる．3.4節での実験に使用するネットワークで，ノード数24以上のものはencode/decode問題を学習する場合に該当する．ここでは，バックワードパスに必要な時間が支配的となるencode/decode問題のような場合について考える．

パス並列モデルでの通信時間は，組となったPE間で出力層の活性値を送信・受信し，$N_{PE}/2$個のPEが重み修正量を通信して加算を行い，その結果を全PEにブロードキャストするだけの時間が必要である．学習セット並列モデルでの解析と同じく，バックワードパスを担当するPEはそれぞれと結合しているホストノードに重みの修正量を送り，ホストノードによりそれらを加算，その結果を全PEにブロードキャストするモデルを考える．

フォワードパスを担当したPEがネットワークの出力をを組となるPEに送る通信は全て並列に行われ，出力値をまとめて一度に通信するときの所要時間はt_{comm}である．バックワードパスにより重み修正量を計算したPEは，その結果をホストノードに並列に送信する．改良学習セット並列モデルと同様に上位層ユニットが持つ重みをまとめてホストノー

[1] nCUBE/2での実測値では，$t_{act}=20\mu$，$t_{add}=t_{multi}=1\mu$（単位:秒）であった．

ドに送信するとすると，$(M+N)t_{comm}$の時間が必要になる．ホストが受信した重み修正量を基に重みを修正するのに必要な時間は$(M+N)t_{add}$，新しい重みを全PEにブロードキャストする時間は$(M+N)t_{comm}$である．以上より，ある学習パターンによる重み更新で必要な時間T_{UW}^{pp}は次式で表される．

$$T_{UW}^{pp} = 2(M+N)t_{comm} + (M+N)t_{add} \tag{3.26}$$

以上より，パス並列モデルで各PEが割り当てられた学習パターンの学習に要する時間T_{total}^{pp}は（3.27）式となる．

$$\begin{aligned} T_{total}^{pp} &= T_{BP}^{lp} + T_{UW}^{pp} \\ &= (LM + 2MN + 2M + 5N)t_{add} \\ &\quad +(3LM + 4MN + 2M + 4N)t_{multi} + (M+N)t_{comm} \end{aligned} \tag{3.27}$$

したがって，N_p個の学習パターンをN_{epoch}回学習するときの学習時間T^{pp}は（3.28）式となる．

$$T^{pp} = N_{epoch}\frac{2N_p}{N_{PE}}T_{total}^{pp} + N_{epoch}T_{UW}^{pp} \tag{3.28}$$

3.3 各並列学習モデルの比較

3.3.1 理想並列計算機上での比較

本節ではユニット並列モデル，学習セット並列モデル，パス並列モデルの各並列誤差逆伝搬学習法について比較検討を行う．PE数や学習パターン数の相違による通信オーバーヘッドや速度向上率の違いについて，3.2節で導出した学習時間をもとに検討する．はじめに，PE間通信の条件として以下を仮定し，理想的な並列計算機上での各並列学習モデルの理論学習時間を比較する．

- 通信時間はメッセージの大きさに依存しない．
- 通信時間はプロセッサ数に依存しない．

なお，モデルを用いた学習時間の計算には$L = M = N = n$として各層にn個のユニットがあると仮定し，各計算要素の処理時間はnCUBE/2での実測値に基づき以下の比率とした．

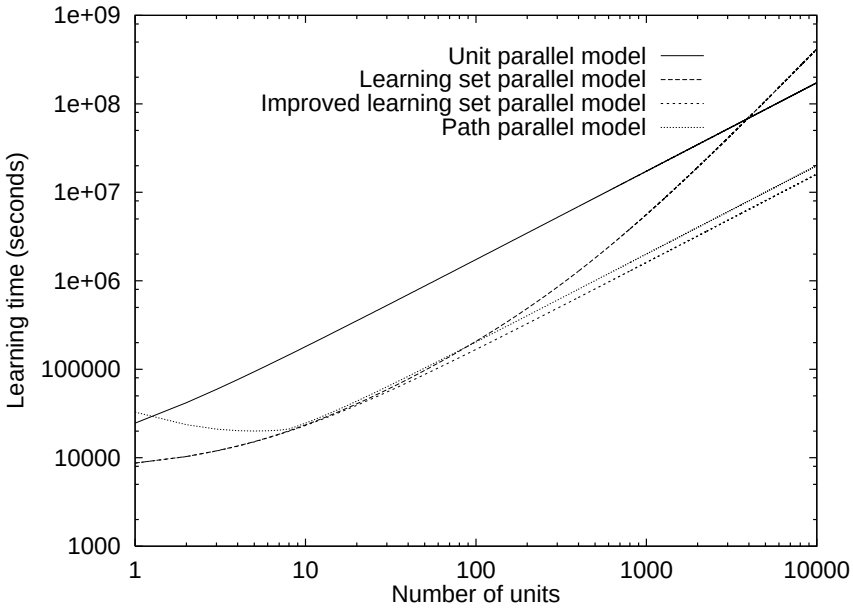

図 3.4: 理想並列計算機上での並列誤差学習法の学習時間

$$
\begin{aligned}
t_{add} &= 1\mu sec. \\
t_{multi} &= 1\mu sec. \\
t_{act} &= 20\mu sec. \\
t_{comm} &= 100\mu sec.
\end{aligned}
$$

図 3.4に，理想並列計算機上での各並列学習モデルの学習時間を示す．このときの学習パターン 数は $N_p = 10000$，学習回数 $N_{epoch} = 10000$ とした．各モデルを比較する都合上，ユニット並列モデルでは，各層におけるユニット数 n は $N_{PE}/3$ とし，それ以外のモデルでは $n = N_{PE}$とした．

図 3.4より，ユニット並列モデルが最も長い学習時間が必要であり，次いで学習セット並列モデル，パス並列モデル，改良学習セット並列モデルの順となっている．これはユニット並列モデルでは N_{PE}は学習対象となるニューラルネットワークの規模で一意に定まり，(3.16) 式より分かるように速度向上に繋がらないためである．学習セット並列モデルでは通信回数がユニット数 n の 2 乗で増加するため，ネットワークの規模が大きくなると通信時間が大幅に増加する．各層のユニット数がおよそ 4000 個を越えると，ユニット並列モデ

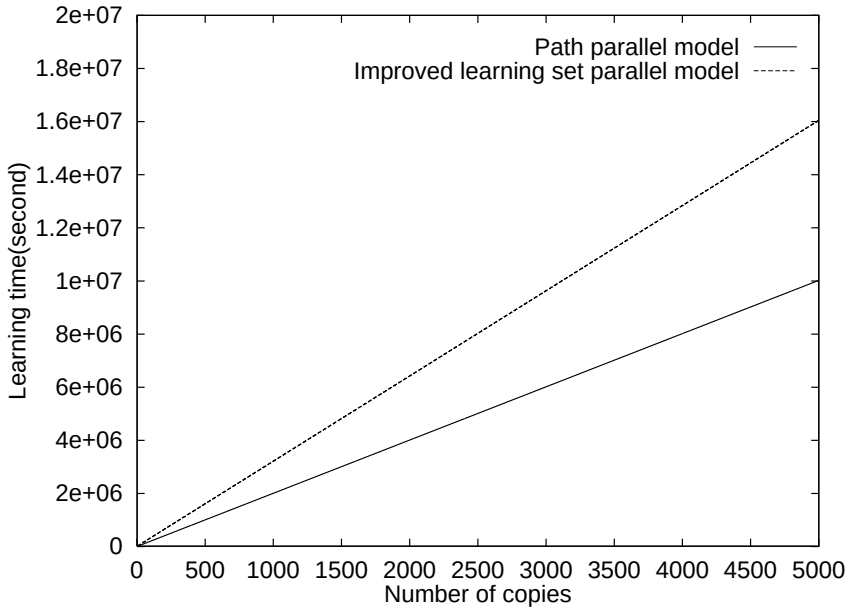

図 3.5: パス並列モデルと改良学習セット並列モデルのコピー数による比較

ルよりも長い学習時間が必要となる．通信回数を削減した改良学習セット並列モデルでは，学習セット並列モデルよりもはるかに通信オーバーヘッドの増加率が小さく，最も高速なモデルであることが分かる．

パス並列モデルは，ユニット数が少ない場合ではユニット数が増加（PE 数も同時に増加）すると学習時間が減少する．これは，PE 数が少ない領域では通信処理よりも学習処理での時間が長く，学習フェーズを分割することによる高速化が大きくあらわれるためである．さらにユニット数（PE 数）が増加すると，通信時間の影響が大きくなり全体として学習時間が増大していくことが分かる．このときの増加の傾向は，同時に用いた改良学習セット並列モデルと同様である．

改良学習セット並列モデルとパス並列モデルを比較すると，パス並列モデルでは作成されるネットワークコピー数が少なくなるため，同じ PE 数で比較すると改良学習セット並列モデルの方が高速である．用いるネットワークのコピー数で比較すると，図 3.5 で示すようにパス並列モデルの方がやや高速である．以上のことから，パス並列モデルでパスを分割して高速化する以上に，学習セットを分割する効果が大きいことが分かる．

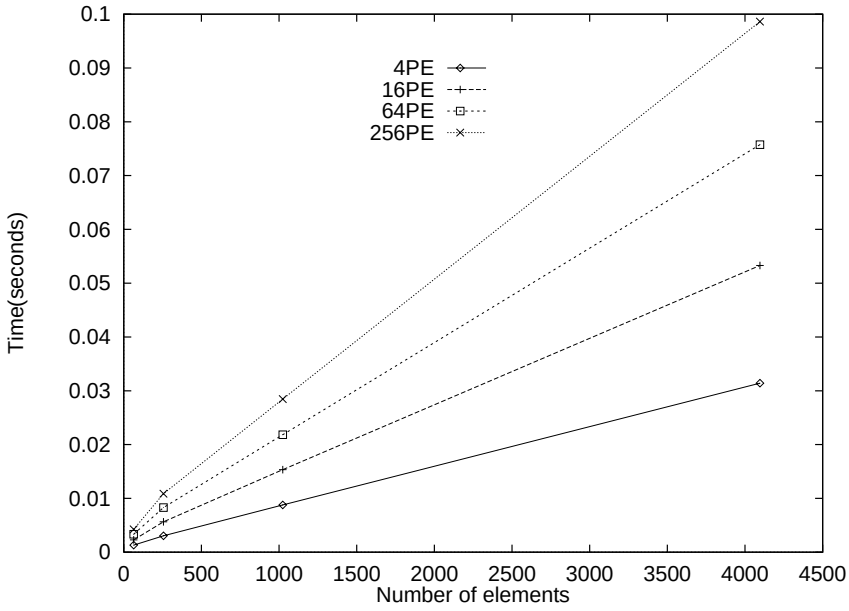

図 3.6: 配列加算通信に要する時間

3.3.2 メッセージ長，PE 数を考慮したモデル

3.3節では，メッセージ長や PE 数に通信時間が依存しない理想並列計算機での各並列学習モデルの比較を行った．本節では，これらのパラメータを考慮した並列学習モデルの比較を行う．こうしたパラメータを決定するにあたり，3.4.1節で述べる超並列計算機 nCUBE/2 における学習セット並列モデルおよびパス並列モデルを用いて検討する．

学習セット並列モデルおよびパス並列モデルでは，各 PE 上にネットワークのコピーを作成して並列に学習を行う．コピー上で計算された重みの修正量は配列加算通信を用いて修正され，結果は全 PE に送信される．nCUBE/2 において，配列加算通信に必要な時間を図 3.6に示す．

図 3.6より，配列加算通信時間はメッセージ長に比例し，同時に PE 数にも比例することが分かる．図 3.6をもとに近似を行うと，配列加算通信で必要となる時間 t'_{comm} は，メッセージ長を M_{len} とおくと（3.29）式で近似できる．

$$t'_{comm} = (0.0005 + 3 \times 10^{-6} M_{len}) \log N_{PE} \tag{3.29}$$

（3.29）式で示した配列加算通信時間を用いた学習セット並列モデルと改良学習セット

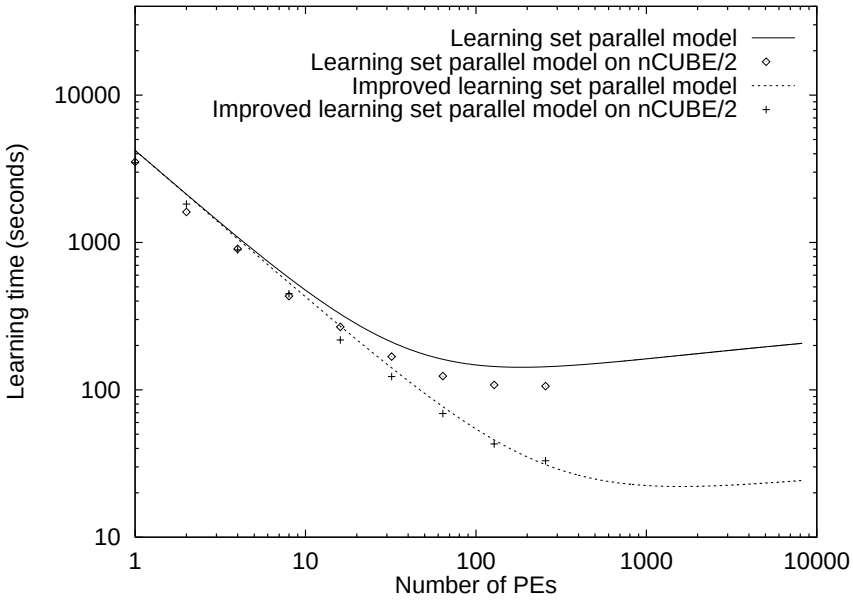

図 3.7: 並列学習モデルと実測値の比較

並列モデルによる学習時間の理論値と，これらのモデルを並列計算機 nCUBE/2 上に実装した時の実測値を図 3.7に示す.

学習時間の測定には，3.4節で用いるのと同じく encode/decode 問題を使用し，学習パターン数は 8192 個，最大学習回数は 250 回，ネットワークの構成は 13-7-13 とした．図 3.7 より，PE 数が増加すると各 PE で行われる処理は減少するものの,（3.29）式により通信時間が増大して高速化率が鈍くなるという傾向がモデルでも再現されている.

3.3.1節と同一の条件を用い，t_{comm} を t'_{comm} としたときの各並列モデルの学習時間を図 3.8 に示す．なお，ユニット並列モデルで発生する PE 間通信は 1 対 1 通信であるため t_{comm} を修正せずに用いている.

図 3.8より，全体の傾向は理想並列計算機を用いた場合と変化しない．しかし，ユニット並列モデルと学習セット並列モデルを比較すると，より少ない PE 数で必要な学習時間の逆転が生じていることが分かる．学習セット並列性を用いるモデルでは，メッセージ長や PE 数を考慮することにより通信時間が増大するためである．改良学習セット並列モデルとパス並列モデルでは，1PE=1 学習パターンとなる 1 万個の PE を用いた場合でもユニット並列モデルよりも高速である．また，パス並列モデルではコピー数が少ないため，学習セット並列モデルよりも通信時間の増加の影響が遅れてあらわれる．したがって，ユニッ

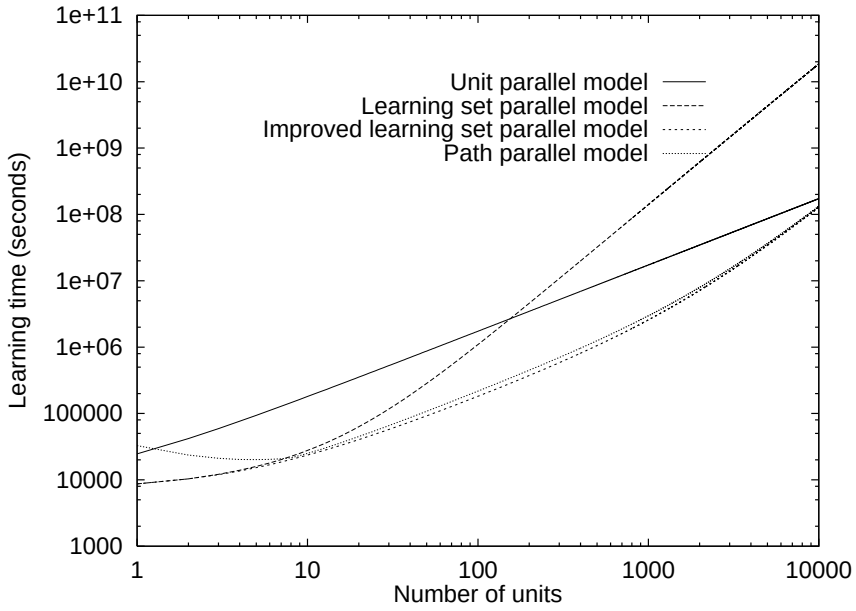

図 3.8: 通信遅延を考慮したときの各並列学習モデルの学習時間

ト数（PE 数）が多い領域では改良学習セット並列モデルとの差が小さくなっている．

図 3.9に，$1000 - 1000 - 1000$ のニューラルネットワークを用い，$N_p = 10000, N_{epoch} = 10000$ とし，最大 1 万個の PE で学習を行ったときの学習セット並列モデル，改良学習セット並列モデル，パス並列モデルの学習時間を示す．図 3.9には参考として 3000 個の PE を用いたユニット並列モデルでの時間も同時に示す．

図 3.9より，学習セット並列モデルでは PE 数が 200 個程度用いたときが最も高速であり，それ以上 PE を増やすと通信時間の増加により学習時間がかえって長くなる．これに対し，改良学習セット並列モデルおよびパス並列モデルでは PE 数が 2000 個程度以上になると学習速度の向上が鈍化しはじめるが，1PE=1 学習パターンとなるまで並列化を行っても学習時間が増大することはない．3000 個の PE を用いたユニット並列モデルと比較しても，同程度の PE 数では改良学習セット並列モデルとパス並列モデルの方が高速であることが分かる．改良学習セット並列モデルとパス並列モデルの比較では，改良学習セット並列モデルの方が高速であり，PE 数が多くなると差が小さくなる．また，今回使用したユニット並列モデルではニューラルネットワークのトポロジが PE 網と同一であると仮定していることを考えると，ユニット並列モデルの学習時間はさらに長くなると考えられる．以上の結果から，誤差逆伝搬学習を並列化する場合は通信回数を削減した改良学習セット並列モデ

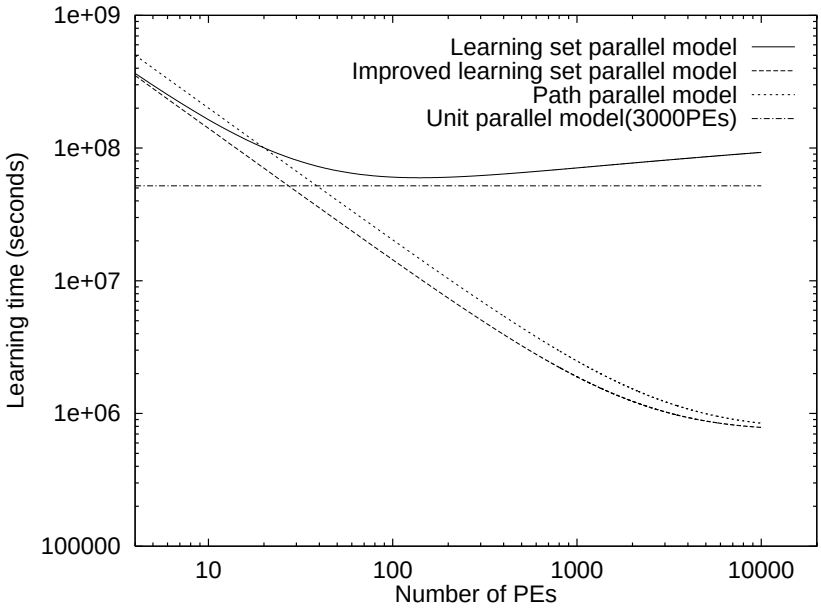

図 3.9: 1000-1000-1000 ネットワークのおける通信遅延を考慮した各並列学習モデルの学習時間

ルが最も有効であると言える．

3.4 並列誤差逆伝搬学習法の並列実装

本節では，3.2節で述べた 3 つの並列誤差逆伝搬学習法を超並列計算機 nCUBE/2 上に実装し，各並列学習モデルの学習速度向上率について検討する．各並列学習モデルの学習速度向上率の測定については，parity 問題と encode/decode 問題を使用し，最大学習回数は 250Epoch とした．

3.4.1 超並列計算機 nCUBE/2

nCUBE/2 は nCUBE 社により商用化された MIMD 方式の超並列計算機である．各プロセシングエレメント（以下 PE と略す）は 64bit の CPU，FPU，メモリ，ネットワークインターフェースからなる，各 PE はプロセッサが持つシリアルリンクを用いて直接ハイパーキューブ網を構成しており，最大 13 次元のハイパーキューブを構成することが可能となっ

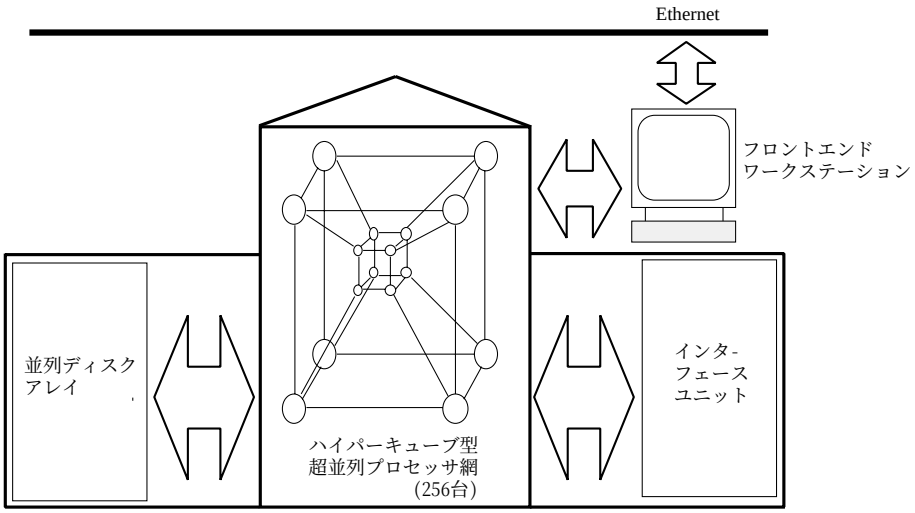

図 3.10: nCUBE/2 システムの概要

ている．nCUBE/2 のプロセッサは 50 万トランジスタの CMOS 型であり，クロック周波数 20MHz で動作する．

ここで用いた nCUBE/2 は PE を 256 台搭載しており，この内 16 台がホストコンピュータと I/O を行う機能を備えている．各 PE が持つメモリは，ホストと通信を行う 16 個の PE が 16MB，その他の PE が 4MB であり，本研究で用いたシステムでは全体で 1.2GB となる．図 3.10に nCUBE/2 システムの概要を示し，表 3.1に nCUBE/2 の諸元をまとめる．

3.4.2 ユニット並列モデルによる並列学習実験

ユニット並列モデルの実装実験では，1 台の PE を 1 ニューロンとしたモデルを超並列計算機 nCUBE/2 上に実装し，速度向上率を測定した．ユニット並列モデルの学習速度向上率の測定には，10bit の parity 問題を用い，nCUBE/2 上のメモリ搭載量の制限により 10，20，30Epoch の学習を行った．ニューラルネットワークの構成を 10-5-1（入力ユニット 10，隠れユニット 5，出力ユニット 1）とし，16PE を用いて並列学習したときの学習に要した時間と WUPS（Weight Update Per Second）値を表 3.2に示す．

表 3.2に示したように，ユニット並列モデルでは WUPS 値が約 100 程度と極めて低速な学習速度しか得られていない．ユニット並列モデルでは比較的小さいメッセージの通信が膨大な回数行われるため，同期通信を行う nCUBE/2 での実行では同期によるオーバーヘッ

表 3.1: nCUBE/2 諸元

PE 数	256PE
CPU	64bit カスタム，10MIPS
FPU	32bit 3.3MFLOPS 64bit 2.4MFLOPS
メモリ	0 ～ 15: 16MB 15 ～ 255: 4MB
通信チャネル	シリアル 2.2MB/s
通信オーバーヘッド	send 140(μsec) receive 55(μsec)
ディスク	16GB, SCSI-2
言語	C，Fortran，Assembler

表 3.2: ユニット並列モデルの学習時間 (パターン数 1024)

Epochs	ユニット並列モデル	
	処理時間 (秒)	WUPS
10	505.8	101.2
20	1015.5	100.8
30	1505.5	102.2

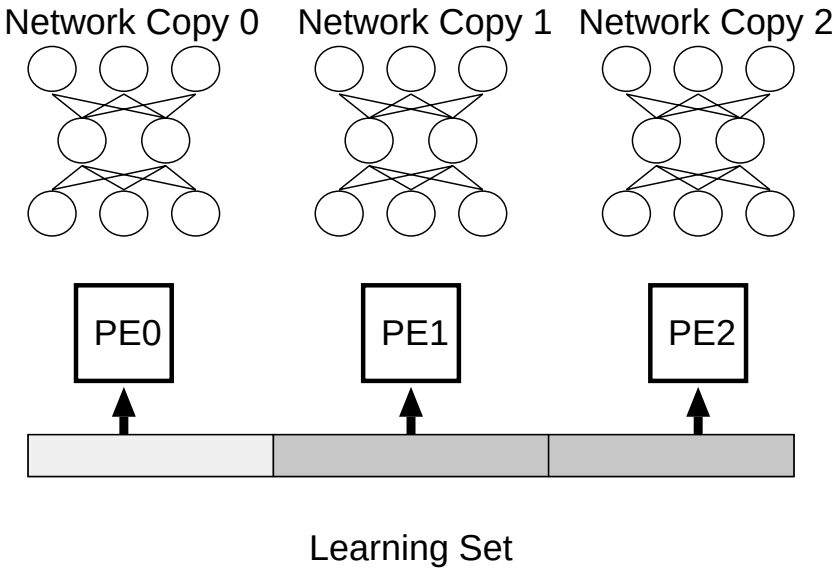

図 3.11: 学習セット並列モデルの実装例

ドが大きい．したがって，並列化による PE の処理の低減以上に通信負荷が大きくなってしまうことが原因である．また，nCUBE/2 の PE 網とニューラルネットワークのトポロジが異なるため，メッセージの中継などに要する時間なども，学習時間増大の一因である．

3.4.3 学習セット並列モデルによる並列学習実験

学習セット並列モデルでは，各 PE が同一構成のニューラルネットワークを持ち，それぞれが学習セットの一部を用いて学習を行う．各 PE は与えられた学習パターンを用いて重みの修正量を計算して加算しておき，全学習セットを処理した後，まとめて重みを更新する．学習セット並列モデルの実装例を図 3.11に示す．

学習セット並列モデルでは，10bit∼13bit の parity 問題および encode/decode 問題を用いて学習速度を計測した．このときのニューラルネットワークの構成を表 3.3に示し，図 3.12に parity 問題を処理したときの，図 3.13に encode/decode 問題を処理したときの学習速度向上率を示す．

図 3.12と図 3.13より，学習パターン数が多いほど高速化していることが分かる．これは，学習セット並列モデルでは PE 間通信は PE 数により一定なので，学習パターンが増えると相対的に PE 間通信のオーバーヘッドが減少するためであると考えられる．学習セット

表 3.3: ニューラルネットワークのユニット数

ビット数	parity			encode/decode		
	入力	隠れ	出力	入力	隠れ	出力
10	10	5	1	10	5	10
11	11	7	1	11	6	11
12	12	6	1	12	6	12
13	13	7	1	13	7	13

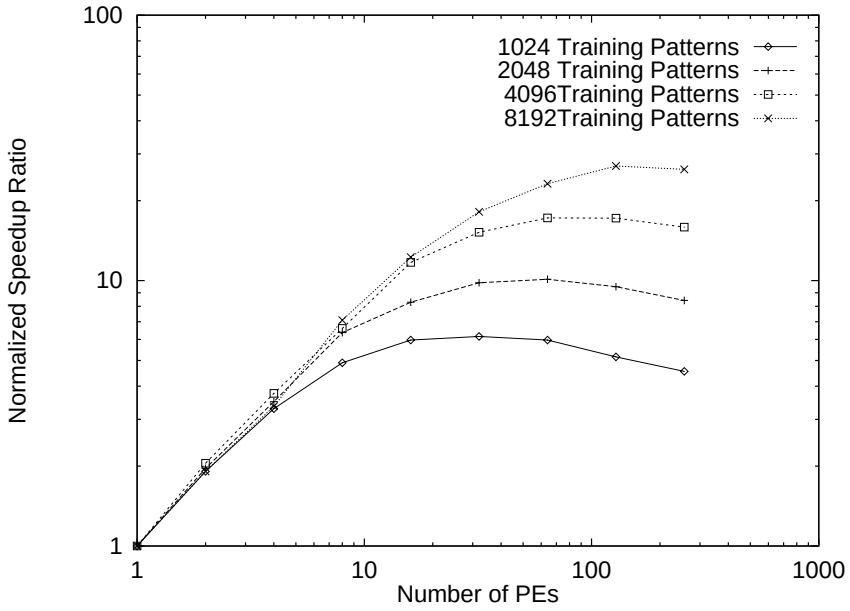

図 3.12: 学習セット並列モデルで parity 問題を学習したときの学習速度向上率

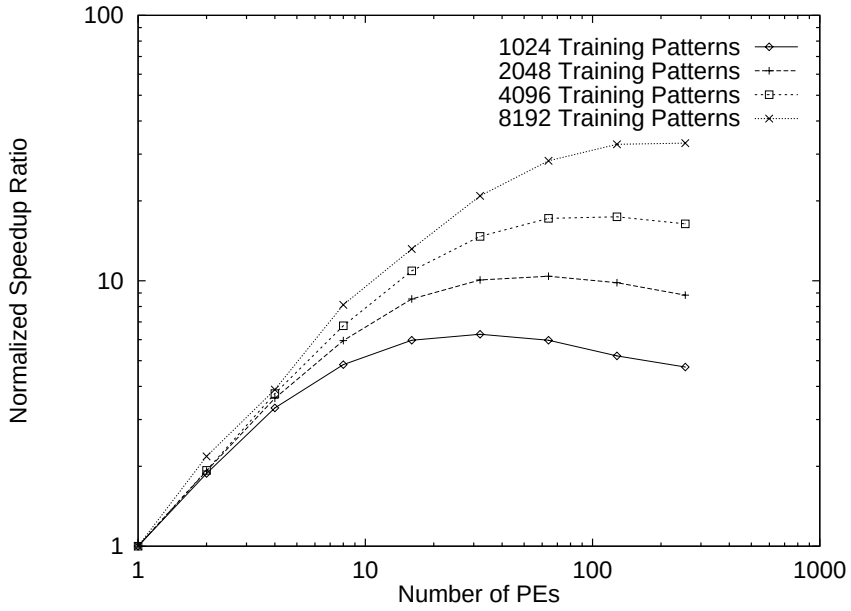

図 3.13: 学習セット並列モデルで encode/decode 問題を学習したときの学習速度向上率

並列モデルでは，encode/decode 問題を 256PE で処理したときに 1PE 時の約 30 倍の処理速度を得ることができた．しかしながら，次のような問題点も明らかになった．

- 使用 PE 数に比べて，速度向上率が小さい．
- PE 数をある数以上に増やすと，かえって処理速度が低下する場合がある．

これは，学習セットを多数の PE に分割することで，1PE 当たりの処理時間は低下するが，相対的に PE 間通信のオーバーヘッドが増大してしまい，処理時間内では通信時間が支配的になるためと考えられる．

3.4.4 改良学習セット並列モデルによる並列学習実験

学習セット並列モデルにおける通信回数を減らすため，3.2.3節で改良学習セット並列モデルを提案した．本節では改良学習セット並列モデルを nCUBE/2 で実行したときの学習速度向上率を示す．図 3.14に parity 問題を処理したときの，図 3.15に encode/decode 問題を処理したときの速度向上率を示す．このとき使用したネットワークの構成は表 3.3と同一である．

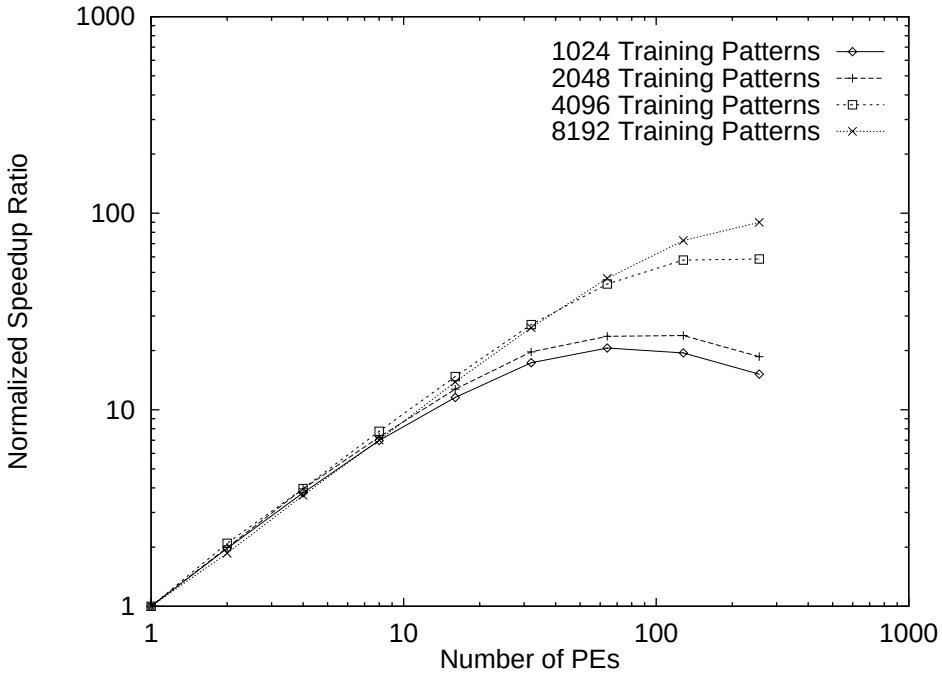

図 3.14: 改良学習セット並列モデルで parity 問題を学習したときの学習速度向上率

図 3.14，図 3.15より，学習セット並列モデルの場合と同様，パターン数が多いほど高速化していることが分かる．学習パターン数 8192 の encode/decode 問題を処理した場合，1PE 時に比べて約 106 倍の速度向上率を得ることができた．同一学習パターン数の parity 問題では，1PE 時に比べて約 90 倍の速度向上率を得た．これらの結果から，改良学習セット並列モデルではネットワーク規模が大きく，学習パターン数も多い大規模ネットワークでは特に有効であると言える．

以上より，学習セット並列モデルは階層型ニューラルネットワークを高速に学習できる並列学習法であることが明らかとなった．また，各 PE に同数の学習パターンを配分することにより，PE 間の負荷の均一化も同時に図ることができる．

3.4.5 パス並列モデルによる並列学習実験

改良学習セット並列モデルとパス並列モデルを組み合わせて parity 問題と encode/decode 問題を処理したときの学習速度向上率を，それぞれ図 3.16，図 3.17に示す．

図 3.16，図 3.17より分かるように，パス並列モデルと改良学習セット並列モデルを組み

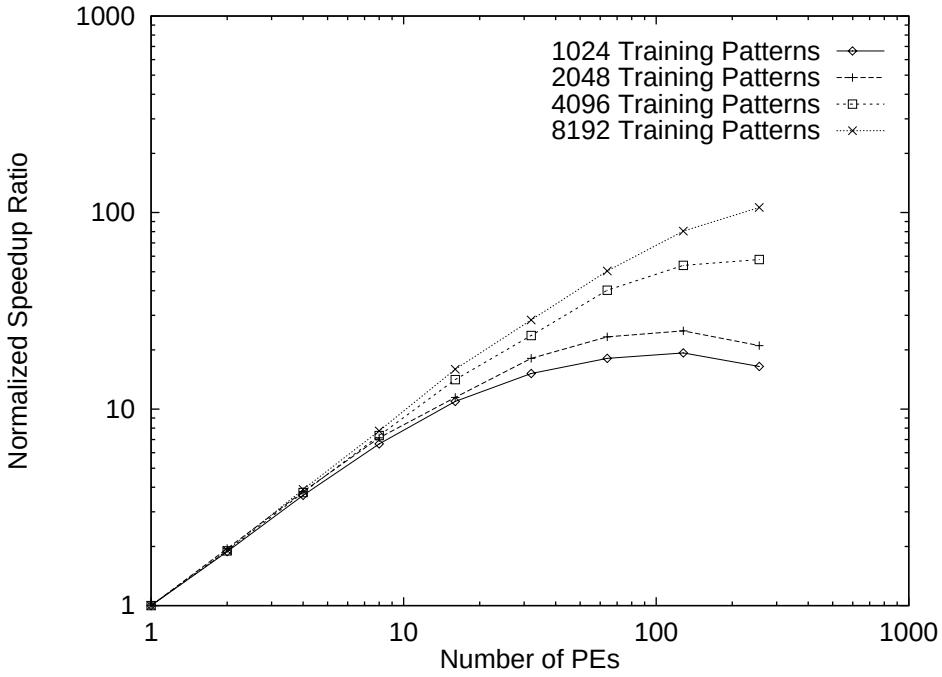

図 3.15: 改良学習セット並列モデルで encode/decode 問題を学習したときの学習速度向上率

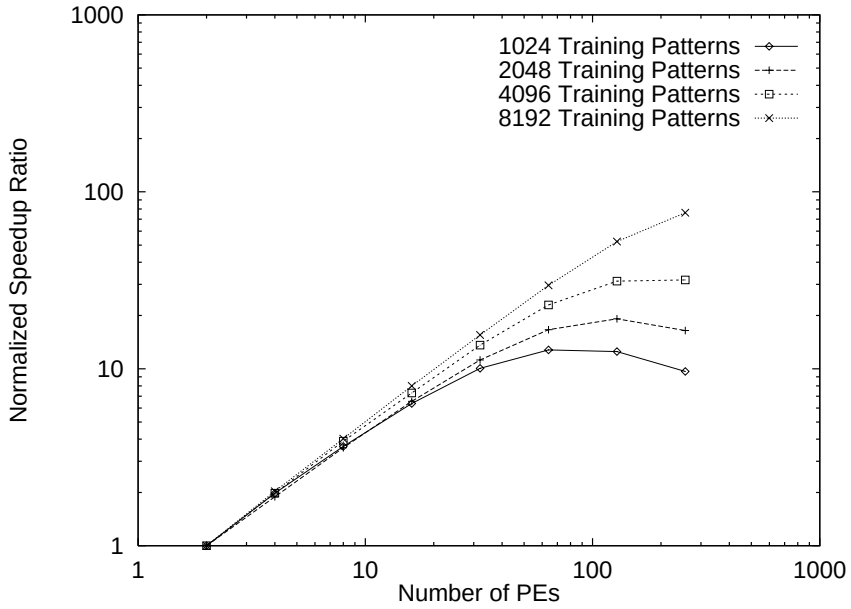

図 3.16: parity 問題をパス並列モデルで学習したときの学習速度向上率

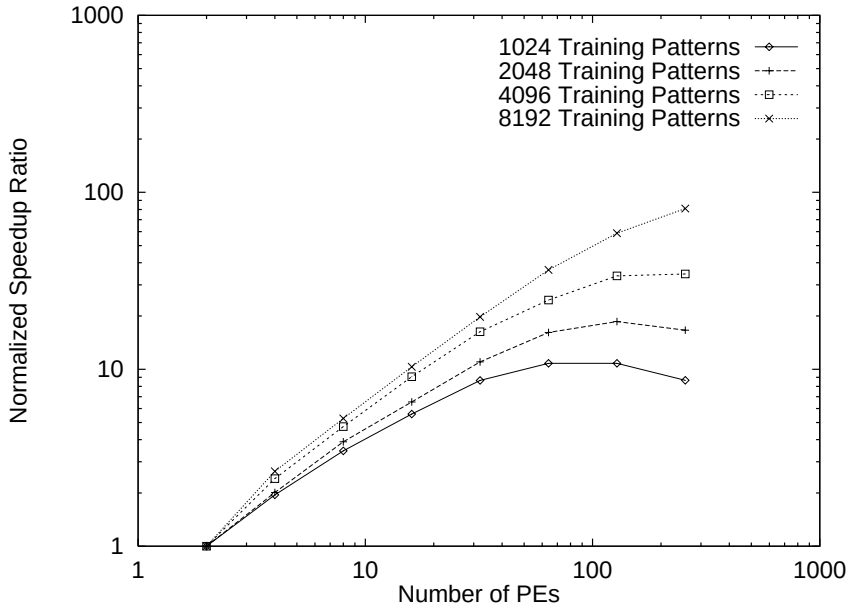

図 3.17: encode/decode 問題をパス並列モデルで学習したときの学習速度向上率

合わせた場合，改良学習セット並列モデルによる高速化が支配的である．パス並列モデル単独の寄与を明らかにするため，学習パターン数 8192 での改良学習セット並列モデルと，パス並列モデル+改良学習セット並列モデルの学習時間を比較する． 図 3.18に parity 問題の場合を，図 3.19に encode/decode 問題の場合をそれぞれ示す．

図 3.18，図 3.19より，パス並列モデルと組合せるよりも改良学習セット並列モデルを単独で用いた方が高速である．これは，パス並列モデルでは 2 個の PE を組にして用いるために実質的な PE 数が減少するためである．モデルを用いたシミュレーションでは，PE 数を増やしたときに改良学習セット並列モデルとパス並列モデルの学習時間の差が小さくなるという現象が見られたが，実験ではそうした傾向があらわれていない．これは用いた PE 数よりも学習パターン数がはるかに多く，PE 数増加による通信時間の増大よりも学習セットの分割による学習時間短縮の効果が大きいためである．

図 3.20に，ニューラルネットワークのコピー数で比較したときの改良学習セット並列モデルとパス並列モデルの学習時間を示す．図 3.20より，パスを分割することによりパス並列モデルでは 1 コピー当たりの学習時間が短くなっていることが分かる．しかし，PE 数が多くなり学習処理に必要な時間が短くなると，組になった PE 間での通信時間が無視できなくなり，パス並列モデルの学習時間は改良学習セット並列モデルよりも長くなる．

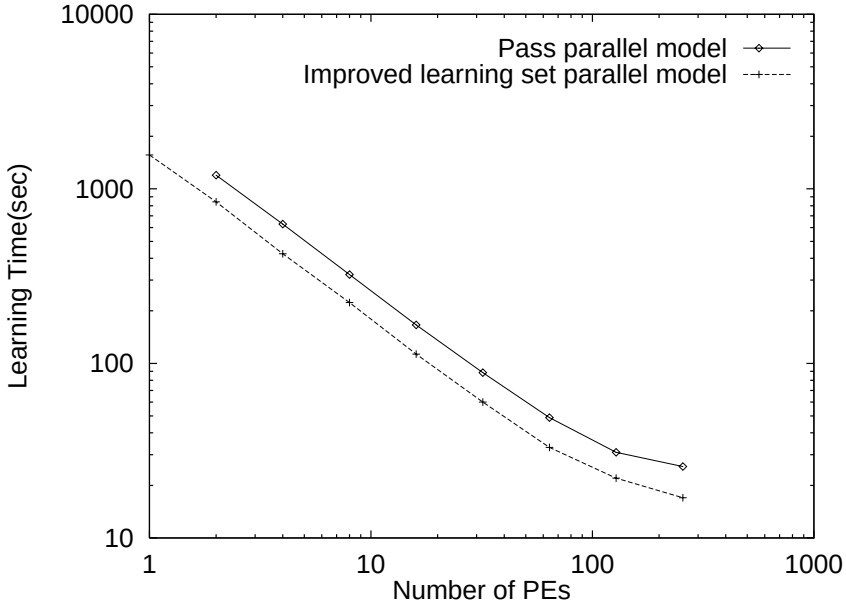

図 3.18: 8192 パターンの parity 問題による改良学習セット並列モデルとパス並列モデルの比較

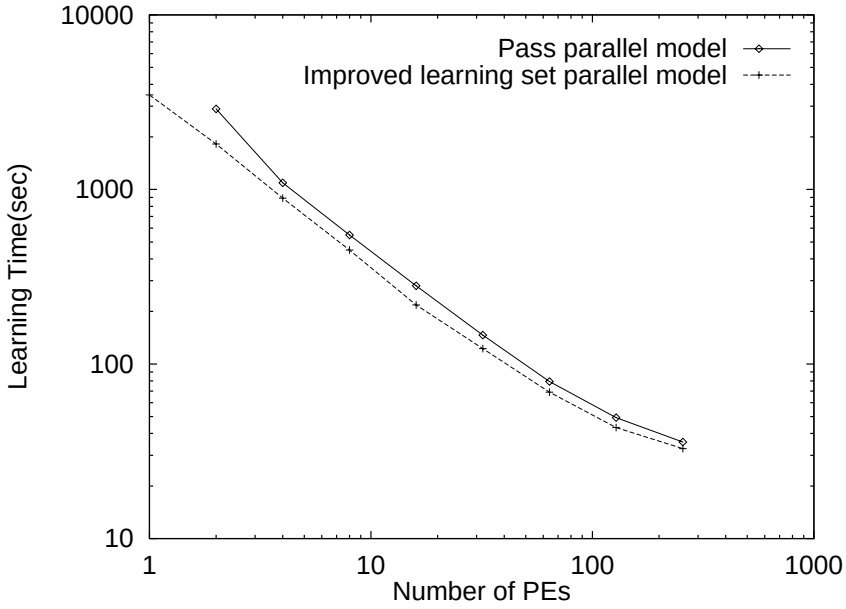

図 3.19: 8192 パターンの encode/decode 問題による改良学習セット並列モデルとパス並列モデルの比較

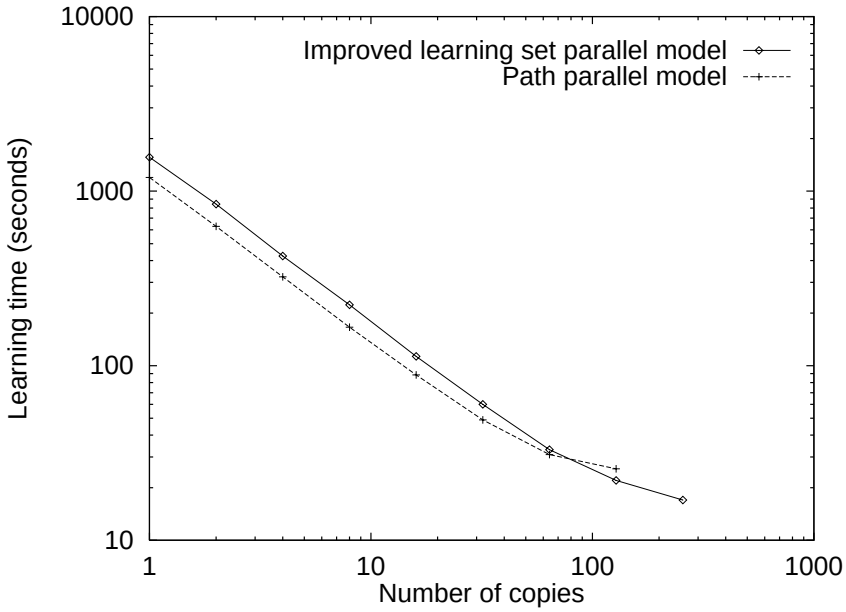

図 3.20: コピー数で比較したときの改良学習セット並列モデルとパス並列モデルの学習時間

以上より，パス並列モデルはネットワークのコピー数で比較すれば改良学習セットモデルよりもわずかに高速であるが，PE 数が増加すると通信オーバーヘッドが相対的に大きくなり効率が悪くなる．また，複数 PE を組にするため作成できるネットワークコピー数も減少し，PE 数で比較したときは改良学習セット並列モデルの方が高速となる．

3.5 まとめ

3.2.1節~3.2.4節において，3 種類の並列誤差逆伝搬学習について述べた．ここでは，得られた結果をもとに各並列学習法の利点と問題点についてまとめる．

ユニット並列モデルでは，モデルにおける解析でも nCUBE/2 による実装でも低い学習速度しか得られなかった．これは，学習に対する通信負荷が高いためである．また，実験で用いた nCUBE/2 では PE 間通信は同期通信となるため，ある PE が通信を行なうとき他の PE の処理がブロックされてしまい，学習速度は極めて低い値に留まった．また，モデルによる解析では PE 間の結合網とニューラルネットワークのトポロジーは同一と仮定したが，実際にはこうした条件は実現が難しい．こうした理由から，ユニット並列モデルを用いて効率的な誤差逆伝搬学習を行うためには，各 PE は単純な機能で良いが，極めて

高速な PE 間通信が可能な計算機が必要であると結論づけられる．

改良学習セット並列モデルは，3 種類の並列学習法の中で最も高速な学習が可能であった．これは複数の重みを同時に通信・加算することにより通信の回数を大幅に減少させることができたためである．学習セット並列モデルをさらに高速化する場合では，大きなメッセージバッファを持ち，一度に大きなメッセージを受信でき，集計した新しい重みを他の PE へ高速にブロードキャストできるメカニズムを持つ計算機を用いることが考えられる．

パス並列モデルでは，作成されるコピー数で比較すれば改良学習セット並列モデルよりもやや高速であることが分かった．しかし，PE を組にして用いる都合上みかけの PE 数が減少し，組になった PE 間での通信オーバーヘッドも発生するために nCUBE/2 上での実験では大幅な高速化を行うことができなかった．パス並列を効率的に用いるには，組になった PE 間での通信時間のオーバーヘッドが小さいことが重要である．これが大きくなると，改良学習セット並列モデルに対する学習時間の優位が小さくなってしまう．そこで，同一 PE 内にフォワードパス，バックワードパスを行う演算ユニットを持ち，各演算ユニットが重みを保持する単一のメモリ領域にアクセスする分散共有メモリ型 PE を多数備える並列計算機などがパス並列を効率的に実行できると考えられる．

第 4 章

部分再学習法

4.1 はじめに

3章において誤差逆伝搬学習を並列化し，高速な学習を行う手法について検討した．その結果，PE 間通信回数を減らした改良学習セット並列モデルが最も高速な並列学習手法であることが明らかとなった．しかし，ネットワークや学習セットが大規模になる実用問題では，さらなる高速学習が求められている．Fahlman[7] は誤差逆伝搬学習法を改良して QuickProp と呼ばれるアルゴリズムを提案した．塚本 [9] らは一度の提示で学習が完了する瞬時学習法を提案している．また，ハードウェアを用いた高速化としては並列計算機への実装が挙げられる．例えば，Muller ら [13] は DSP を多数用いた並列ニューラルネットワークシミュレータを構築し，高速な学習を実現した．

学習のさらなる高速化手法として，ニューラルネットワークのハードウェア実装が考えられる．特に，ニューロンに相当するデバイスを直接 VLSI チップ上に実装することにより，大規模かつ高速なニューラルネットワークの構築が期待できる．しかし，VLSI 上にニューラルネットワークを実装する場合，チップ製造時の欠陥によりニューラルネットワークが正常に動作しない場合があるため，これを回避する技術が必要となる．

Khunasaraphan ら [23] は，故障したリンクが持つ重みを健全なリンクに分担させて故障を回避する手法について報告している．丹ら [22] は，階層型ニューラルネットワーク自体が持つ耐故障性について注目し，故障の影響を受けないようにネットワークを学習させるアルゴリズムを提案している．當麻 [24] はニューラルネットワークの一部が故障しても，冗長性を用いて他の最適解を探索できることを指摘した．また，伊勢崎 [32] は再学習を用

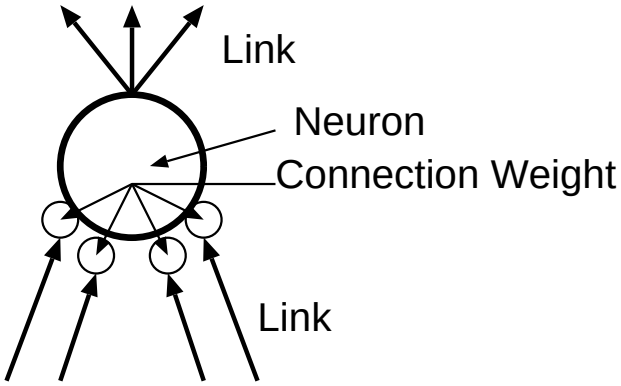

図 4.1: 故障を生じうる箇所

いることにより階層型ニューラルネットワークが数回の故障に耐えられることを示している．しかしながら，これらの研究ではすべての故障パターンに対する重みをあらかじめ用意しておく必要があったり，故障回避を実現する学習プロセスに長い時間が必要な場合があるという問題点がある．また大規模なネットワークにおいて実用的な時間で故障補償が可能か十分検討されていない．

本章では，階層型ニューラルネットワークが本来持つ冗長性を利用し，故障部分のみに誤差逆伝搬学習を適用することにより高速かつ容易に故障回避が可能となる部分再学習法を提案する．部分学習法は，ニューラルネットワークの構造や学習アルゴリズムに変更を加える事なく，故障部分の補償が可能である利点を有している．また，部分再学習法における初期重みによる学習速度の違いについても検討する．更に，大規模ニューラルネットワークに関して部分再学習法を適用し，効果的な故障保障の可能性を示す．

4.2 ニューラルネットワークにおける故障

VLSI や WSI に代表されるハードウェアには，製造上の欠陥が避けられない．特に大規模なニューラルネットワークの実装ではチップ面積の増大を伴ない，製造欠陥の発生率は指数関数的に増大する．このため，ニューラルネットワークをこれらのハードウェア上に実装するときには，欠陥があっても正常な動作を保つような故障補償能力を持たせることが不可欠である．安永ら [33] は，ニューラルネットワーク中で生じうる故障を分類し，中間層のニューロンに故障が生じた場合の故障補償法について報告している．本節ではニュー

表 4.1: ニューロン故障，結合重み故障とリンク故障の関係

故障個所	故障	対応するリンク故障
ニューロン	0-スタック	複数リンクの 0-スタック
	1-スタック	複数リンクの 1-スタック
重み	$+\infty$-スタック	出力リンクの 1-スタック
	0-スタック	単一入力リンクの 0-スタック
	$-\infty$-スタック	出力リンクの 0-スタック

ラルネットワークをハードウェア上に実装する場合に考えられる故障について考察し，本論文で対象とする故障について述べる．

4.2.1 故障の定義

誤差逆伝搬学習を用いる事により，ニューラルネットワークは提示されるすべてのパターンに対して最大誤差がある一定値 e 以下になるように重みがチューニングされる．

$$\max_{i,n}(t_n^{(i)} - o_n^{(i)})^2 < e^2 \tag{4.1}$$

ここで，n はニューラルネットワークの出力素子の番号，i は学習パターン番号，$t_n^{(i)}$はパターン i が提示されたときに出力素子 n が出力すべき値（教師信号），$o_n^{(i)}$はパターン i が提示されたときに出力素子 n が実際に出力した値である．本論文では，何らかの理由により式 (4.1) を満たすことができなくなった場合をニューラルネットワークの故障と定義する．

4.2.2 リンク故障モデル

本節では，部分再学習法で扱う故障モデルについて検討する．階層型ニューラルネットワークは，ニューロンの機能を持つ素子とその間の結合から構成される．故障を起こしうる箇所は，図 4.1に示すニューロン，リンク，結合重みの 3 つが考えられる．

ニューロンの故障は，その出力が $[0, 1]$ であることから最も影響が大きく，出力値が 0 または 1 へのスタック故障について詳しく検討されている [33]．スタック故障とは，そのニューロンへの入力に関わらず出力が 0，または 1 で変化しない故障をいう．また，出力が

ランダムに変化する故障も考えられる．結合重みの故障は，その値が $[-\infty, +\infty]$ であることから，$\pm\infty$ へのスタック故障および 0-スタック故障が考えられる．リンクについては断線故障がある．出力層のニューロンにおける出力リンクの故障は，もはや（4.1）式で規定される仕様を決して満たすことができないので，本論文では考慮しない．また，出力がランダムに変化するニューロンに対しても故障補償が困難であるため，ここでは議論しない．

さて，ニューロンや重みの故障は基本的にリンクの故障に置きかえて考えることができる．ニューロンのスタック故障は，故障ニューロンからの入力を持つ上位層のニューロンからみれば入力リンクの故障とみなすことができる．また，結合重みの$\pm\infty$ スタック故障は，入力値が 0 でなければそのニューロンの出力を 0 か 1 に固定することになり，結果的に出力リンクの 0-スタック故障あるいは 1-スタック故障となる．これを上位層のニューロンからみれば，入力リンクの故障と考えることができる．これらの故障とリンク故障の関係を表 4.1に示す.

表 4.1に示したリンク故障には，出力リンクの故障と入力リンクの故障の 2 つの場合が考えられる．さらに，それぞれについて 0-スタック故障，1-スタック故障がある．出力リンクのスタック故障は，上位層のニューロンから見れば単一，あるいは複数の入力リンクのスタック故障と見なせることは先に述べたとおりである．入力リンクの 0-スタック故障は，入力リンクの断線として考えることができる．同様に，入力リンクの 1-スタック故障はそのリンクが持つ重みがそのまま入力となることを考えると，入力リンクの断線と同時に重み分だけのバイアスが常にかかることであり，しきい値が変化したことと等価である．次章では，代表的な故障として入力リンクの断線故障を対象とした学習法について検討する．

4.3 部分再学習法

4.3.1 故障を考慮した学習法

ニューラルネットワーク内に故障が生じ，式（4.1）で定められる条件を満たすことができなくなった場合，なんらかの故障補償が必要になる．Khunasaraphan ら [23] は，故障を起こしたリンクを正常なリンクで代替するように重みを決定する手法を提案している．代替リンクの重みの計算は解析的に行われ，単一あるいは複数のリンクを用いて故障補償を行う．しかし，この手法をハードウェア上で実行する場合，起こりうるすべての故障パターンに対応する重みをあらかじめ決定しておく必要があり実用的ではない．丹ら [22] は，

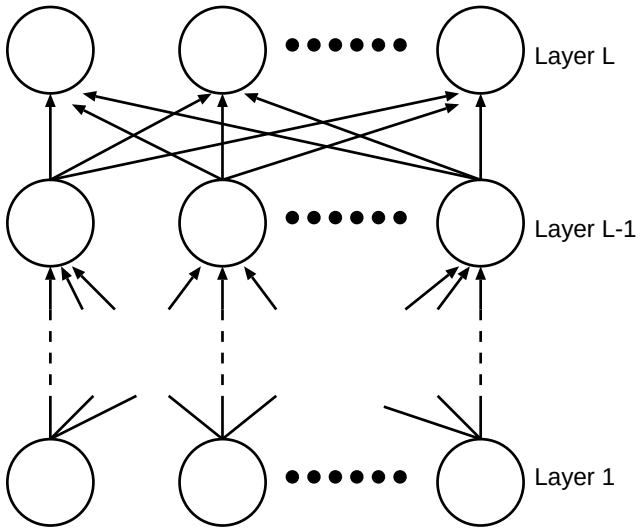

図 4.2: 再学習対象とする階層型ニューラルネットワーク

ニューロン間のリンクに故障が生じても，ニューラルネットワークの出力が影響されないように重みを決定する手法について述べている．しかし，この手法では故障補償を実現する重みの決定に多大な時間が必要であるという問題がある．當麻ら [24] は，ニューラルネットワークの内包する冗長性を利用して再学習を行うことにより，故障補償が可能である点を指摘している．すなわち，学習が完了しているニューラルネットワークの一部を切断した後，学習パターンを提示して再度の学習を行うことにより，故障前の機能が回復できることを述べている．

そこで，本論文ではリンクの故障を対象とし，故障部分のみで誤差逆伝搬学習を行うことにより故障を高速に補償する部分再学習法を提案し，その性能について評価する．

4.3.2 リンク故障の補償

図 4.2で示すような，L 層の階層型ニューラルネットワークを考える．ここで，第 1 層を入力層，第 L 層を出力層とする．層間のリンクが持つ結合重みは誤差逆伝搬学習により決定され，学習に用いる学習パターンを $\{s^{(i)}, t^{(i)} : i = 1, \ldots, T\}$ とする．ここで，$s^{(i)}$，$t^{(i)}$はそれぞれ i 番目の入力ベクトルと理想出力ベクトルであり，以下で定義される．

$$s^{(i)} = \{s_1^{(i)}, \cdots, s_M^{(i)}\}$$

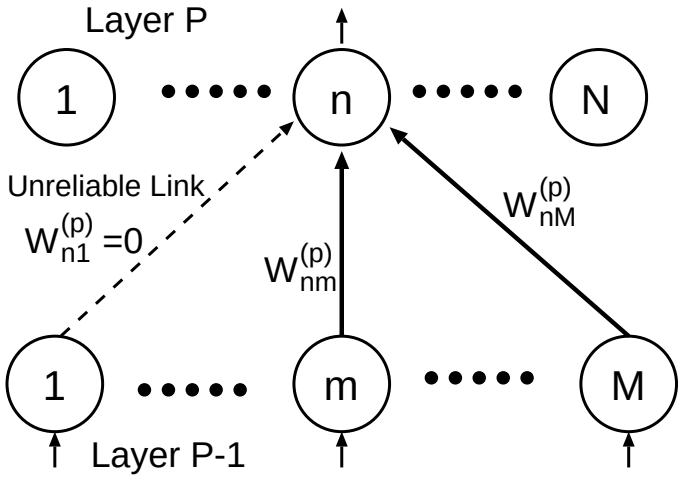

図 4.3: 部分再学習の対象範囲

$$t^{(i)} = \{t_1^{(i)}, \cdots, t_N^{(i)}\}$$

入力リンクの単一断線故障が発生し，そのリンクからの入力が途絶えた場合を考える．入力リンクを M本持つ，第 p 層の n 番目のニューロンのリンクが故障したと仮定し，第 p 層には N個のニューロンがあるとする．ここで各リンクの持つ結合重みを，$w_{nl}^{(p)},\ l = 1, \ldots, M$ とする．一般性を失うことなく，1 番目のリンクが断線故障を起こしたと仮定できる．断線故障を起こしたリンクからの入力がなくなることは，その入力がニューロンの出力に影響を与えることがないということであり，これは結合重み $w_{n1}^{(p)}$を 0 とすることに等しい．ここで学習パターン $s^{(i)}$がニューラルネットワークに入力されたとする．このとき，第 p 層への入力を $s^{(i)(p)}$，第 p 層の n 番目のニューロンの出力を $o_n^{(i)(p)}$とする．断線故障により，$w_{n1}^{(p)}$が 0 となることは，n 番目のニューロンの入力空間が M次元から $M-1$ 次元に減少することを意味している．従って部分再学習法では，故障の影響を受ける上位層のニューロンのみに注目し，正常なリンクの結合重み $w_{nl}^{(p)},\ l = 2, \ldots, M$を調節して，故障リンクを持つニューロン n の出力を T個の学習パターンすべてに対して故障前の出力 $o_n^{(i)(p)}$と同等にすることにより高速に故障回避を行う．

図 4.3に示すように，部分再学習の対象となるニューロンは故障リンクを持つニューロン一つでよい．

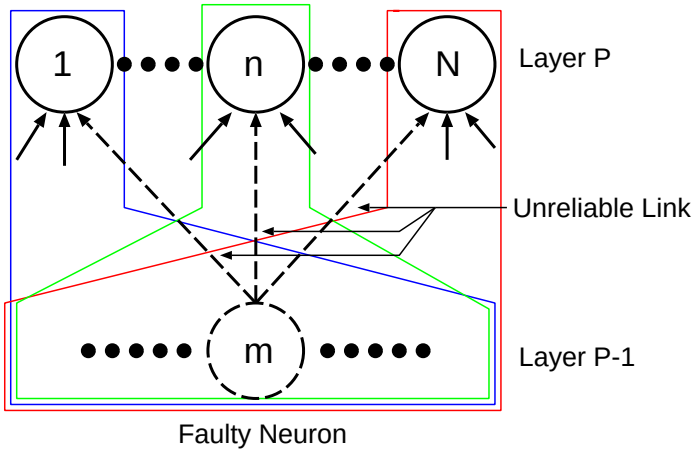

図 4.4: ニューロン故障

$w_{n1}^{(p)}$が常に 0 であるニューロン n は，T個の学習パターン

$$\{(y^{(i)(p)}, o_n^{(i)(p)}) : \ i = 1, \ldots, T\} \tag{4.2}$$

により誤差逆伝搬学習を行う．ここで，$y^{(i)(p)}$は p 層の N個のニューロンに対する $M-1$ 次元の入力ベクトルであり，式（4.3）の関係が成り立つ．

$$y^{(i)(p)} = \{s_2^{(i)(p)}, \cdots, s_M^{(i)(p)}\} \tag{4.3}$$

一つのニューロンが持つ複数のリンクが同時に故障をした場合も同様に,（4.2）式により故障補償を行う．このときの $y^{(i)(p)}$は，式（4.4）で示され，$\boldsymbol{H}$ は正常なリンクからの入力の集合である．

$$y^{(i)(p)} = \{s_x^{(i)(p)}; x \in \boldsymbol{H}\} \tag{4.4}$$

4.3.3 ニューロン故障の補償

ニューロン一つが故障を起こし，その出力が 1-スタック，あるいは 0-スタックした場合を考える．4.2.2節で述べた通り，出力リンクの 1-スタック故障は上位層からみると入力リンクの 0-スタック故障としきい値変化が同時に起こったことと等しいので，ここでは 0-スタック故障についてのみ述べる．

図 4.4に，第 $p-1$ 層のニューロン m が 0-スタック故障を起こした場合を示す．通常，階層型ニューラルネットワークは完全結合を用いているので，第 $p-1$ 層のニューロンの故

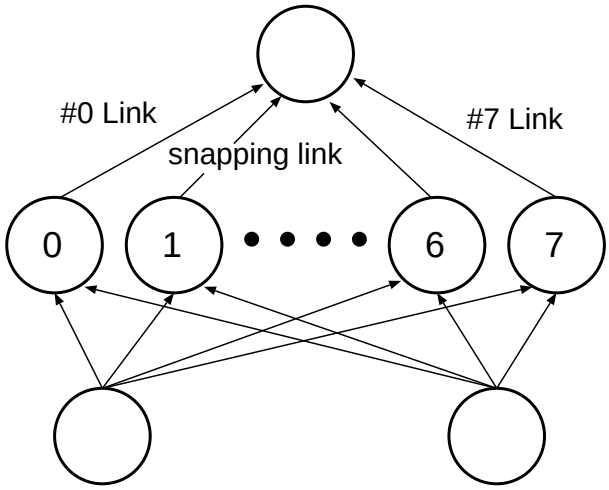

図 4.5: XOR 問題学習時の 2-8-1 ネットワーク

障は p 層のニューロンすべてに影響を及ぼすことになり，結合重みは式（4.5）に示すように変化する．

$$w_{nm}^{(p)} = 0, \quad n = 1, \ldots, N \tag{4.5}$$

図 4.4で示すように，第 $p-1$ 層のニューロンの故障は，そのニューロンからの出力を受ける上位層のニューロンから見れば，単一のリンクの故障と同じである．したがって，この故障を補償するためには，p 層の全ニューロンを対象として部分再学習を行えばよい．したがって学習に用いる学習パターンは（4.6）式で表される．

$$\{(y^{(i)(p)}, o_u^{(i)(p)} : \quad u = 1, \ldots, N, i = 1, \ldots, T\} \tag{4.6}$$

ここで $y^{(i)(p)} = \{s_j^{(i)(p)} : j = 1, \ldots, M,\ j \neq m\}$ である．

4.4 部分再学習法の性能評価

4.4.1 XOR 問題

4.3節で述べた部分学習法を XOR 問題に適用してその有効性を示し，部分学習法によって得られたニューラルネットワークの性質について検討する．対象とするネットワークは図 4.5に示すように入力層に 2 個，出力層に 1 個，隠れ層に 8 個のニューロンを持つ 2-8-1 ネットワークとする．

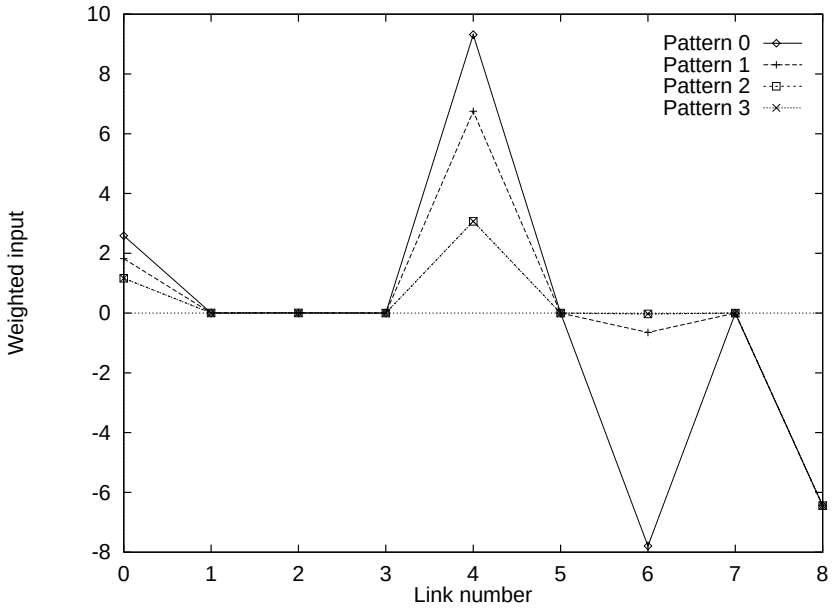

図 4.6: 各パターンによる出力ニューロンへの重み付け入力

XOR 問題は 2-2-1 ネットワークで学習できることが既に知られている．故障回避を実現するための冗長性として，隠れ層の 6 個のニューロンを使用する．Sun4/5 上で 2-8-1 ネットワークに学習パターンを 10,000 回提示して XOR 問題を学習するのに，3.5 秒の時間が必要であった．このときの学習率は 0.6，モメンタム項は 0.4 とし，終了条件として学習パターンの最大提示回数 50,000 回，各出力ニューロンでの誤差が 0.1 以下を用いた．

図 4.6に，学習完了後の出力ニューロンへのリンク毎の入力を示す．図 4.6 のリンク 8 は，しきい値のために常に 1 を出力するニューロンと接続しており，図 4.6に示される値はリンクの重みそのものである．

図 4.6より，ニューロン 0，4，6，およびしきい値ニューロンからの入力が大きく，他のニューロン 1，2，3，5，7 からの入力はほとんど 0 となっている．図 4.7に示した，各パターンに対する隠れ層のニューロンの出力を見ると，ニューロン 1，2，3，5，7 の出力自体が全学習パターンに対し小さい．つまり，これらの重みが故障してもネットワークの出力には影響をほとんど与えないことが予想される．

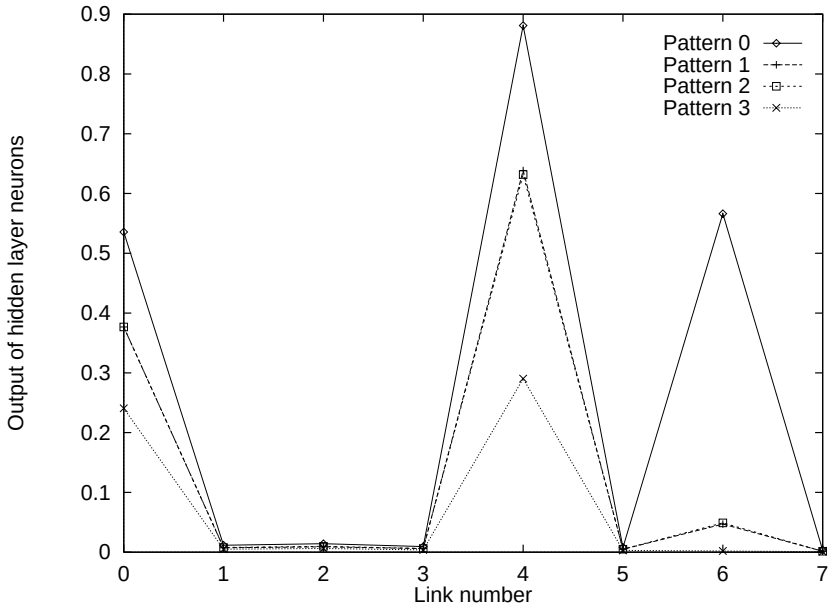

図 4.7: 隠れ層のニューロンの出力

4.4.2 乱数を初期重みとした部分再学習法

4.4.1節と同様の条件で，隠れ層から出力層へのリンク 8 本がそれぞれ単独で断線を起こした場合を想定して，部分再学習法による故障補償と誤差逆伝搬学習による故障補償を行い，必要となった学習パターン提示回数と処理時間を比較する．

表 4.2に，2-8-1 ネットワークを再学習したときの学習パターン提示回数と学習時間を示す． このときの初期重みは共に乱数を使用し，学習時間は異なる初期値からの学習を 5 回行なった場合の平均である．また，故障リンクの番号は図 4.5中のニューロン番号に一致する．ここでは，しきい値ニューロンからのリンクは故障しないものとした．

表 4.2で示すように，すべての単一リンク故障に対する再学習に対して，部分再学習法により高速に故障補償を行うことができた．4.4.1節で予想したように，リンク 4，6 の故障に対する補償は他のリンクの故障補償に比べて長い時間が必要となった．

リンク 6 の故障補償では，故障を補償するのに必要となった学習パターン提示回数は，3 層の誤差逆伝搬法を用いた時と比較して約 3 倍となった．しかし，学習時間では約 60%の時間で故障補償が終了している．これは部分再学習法が故障に関係するネットワークの一部のみを用いて再学習を行うため，重み更新を行うリンク数を少なくできるからである．

表 4.2: XOR 問題における学習パターン提示回数と再学習時間

故障リンク	誤差逆伝搬法		部分再学習法	
	提示回数	時間（秒）	提示回数	時間（秒）
0	9792	3.45	1090	0.067
1	12304	4.36	1058	0.070
2	11524	4.10	1066	0.077
3	10735	3.80	1064	0.070
4	11501	4.07	5347	0.350
5	10402	3.68	1066	0.070
6	10340	3.65	33073	2.18
7	10354	3.67	1062	0.067

ネットワークの規模が大きくなればなるほど，一部の重みだけを更新対象として故障補償を行うことによる高速化が期待できる．

4.4.3 故障前の重みを初期重みとした部分再学習法

誤差逆伝搬学習ではネットワークが最適解に収束することを保証しておらず，極小値に収束してしまう場合がある．このため学習を開始するときには重みを乱数で初期化することが一般に行われる．しかしながら，元のネットワークが冗長性を持つものであれば，故障によるネットワークの出力変化がそれほど大きくならず，微小な重みの修正で故障を補償できる可能性がある．つまり，再学習を開始するときの初期重みとして故障前の値を用いることで収束すべき値に近い地点からの再学習が可能と考えられる．そこで，初期重みとして故障を起こす前の重みを用いて故障補償を試みた．

4.4.1節と同様のネットワークを用い，故障リンク以外の重みは初期重みとしてそのまま使用して再学習を行なったときの学習パターン提示回数と学習時間を表 4.3に示す．

表 4.3より，リンク 1，2，3，5，7 の断線故障に対する再学習時間は，初期重みとして乱数を用いた場合に比べ大幅に高速化することができた．これは図 4.6で示したように，これらのリンクは冗長ニューロンからのリンクでネットワークに与える故障の影響が小さく，

表 4.3: 初期重みに故障前の重みを用いた時の XOR 問題におけるパターン提示回数と再学習時間

故障リンク	パターン提示回数	再学習時間（秒）
0	416	0.03
1	1	0.00
2	1	0.00
3	1	0.00
4	4967	0.33
5	1	0.00
6	41833	2.76
7	1	0.00

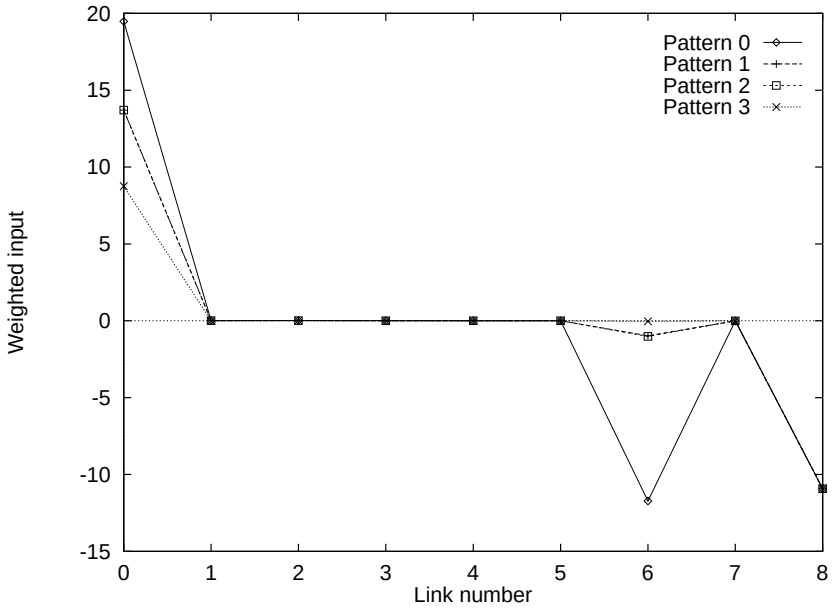

図 4.8: リンク 4 の故障で故障前の重みを初期重みとして部分再学習を行ったときの各リンクからの入力

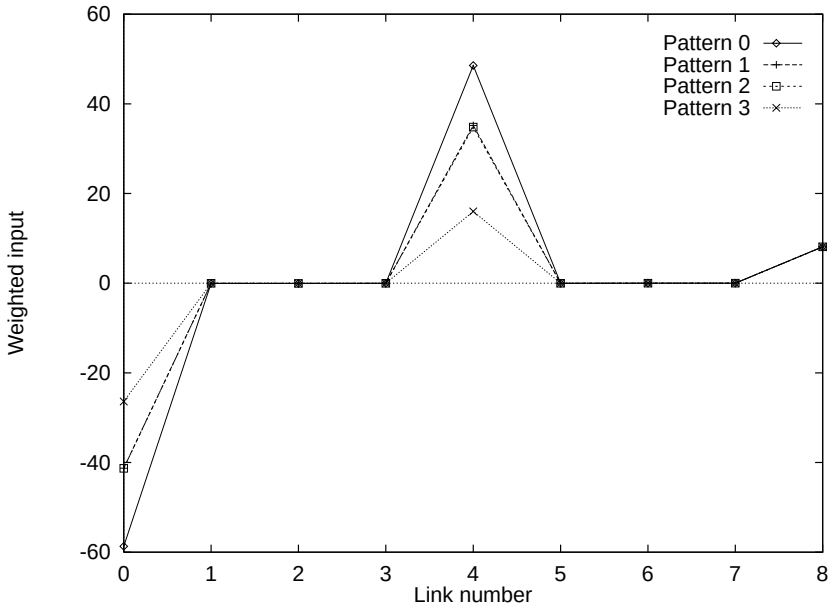

図 4.9: リンク 6 の故障で故障前の重みを初期重みとして部分再学習を行ったときの各リンクからの入力

重みの修正がほとんど不要であったことを示している．また，リンク 4 では乱数を初期値として用いた場合と同等，リンク 6 では収束にやや時間がかかった．

リンク 4 の再学習に際し，重みを乱数で初期化した場合と，故障前の重みを初期重みとして用いた場合の出力ニューロンへの入力を図 4.8に示す．このとき，乱数を初期重みとして用いたときも同じ重みに収束した．図 4.8より，リンク 4 の故障ではリンク 0，6 としきい値の変更でその補償を行なっている．

同様に，故障前の重みを初期重みとして部分再学習によりリンク 6 の故障補償を行なったときの，出力ニューロンへの各リンクからの入力を図 4.9に示す．このとき乱数を初期重みとして用いた場合も同じ重みに収束した．

図 4.9より，リンク 6 の故障をリンク 0，4 としきい値の調整により補償しているのが分かる．また，リンク 4 の故障補償時に比べ，出力ニューロンへの入力の絶対値はかなり大きな値となっている．隠れ層のニューロンからの出力は図 4.7のまま変化していないので，これは隠れ層-出力層間の重みが故障前の値から大幅に変化したことを意味している．このため，乱数を初期重みとした部分再学習に対して高速化できなかったと考えられる．

XOR 問題の学習により，単一リンクの断線故障に対する部分再学習法の性能を評価して

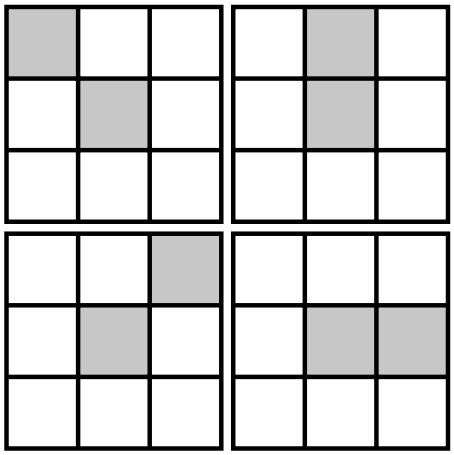

図 4.10: ブロック内の線分の方向別割合を抽出するマッチングパターン

きた．これにより，リンク故障に対しては初期重みとして故障前の重みを用いた部分再学習法が有効であることが分かった．しかしながら，XOR 問題でリンク 0，4 やリンク 0，6 が同時に故障したと想定して部分再学習を行ったところ，収束させることができなかった．これは図 4.7に示すように，学習に用いた 2-8-1 ネットワークが実際には 2-3-1 ネットワークの形に収束しており，2 本のリンクの同時故障の場合は XOR 問題の学習に必要な 2-2-1 ネットワークを構成できないためである．こうした問題を避けるためには，情報を多重化するような学習をあらかじめ行うか，下層ニューロンも含めた再学習が必要であると考えられる．

4.4.4 大規模問題への応用

部分再学習法を大規模なネットワークに応用し，その性能について検討する．ここでは顔画像認識を行う 64-9-32 のネットワークを用いる [34]．用いる画像は 256×256 ピクセル，25 階調のグレースケール画像である．ニューラルネットワークに画像を入力するにあたり，次の処理を行った．まず平滑化フィルタを画像全面にかけたあと，Hilditch 法によりエッジ抽出を行い．得られたエッジ画像を 4×4 に分割する．その後各ブロック内に図 4.10に示すマッチングパターンを左下→右上に走査し，ブロック内に存在する線分量を 4 方向に分けてカウントする．これにより 16 個のブロック内に含まれる線分の量が，縦，横，左上，右上の 4 方向それぞれについて分かる．得られた線分の量を定数で正規化し，64 次元の入力ベクトルを生成する．出力は各人に対応する出力ニューロンを用意し，ある人物の顔画

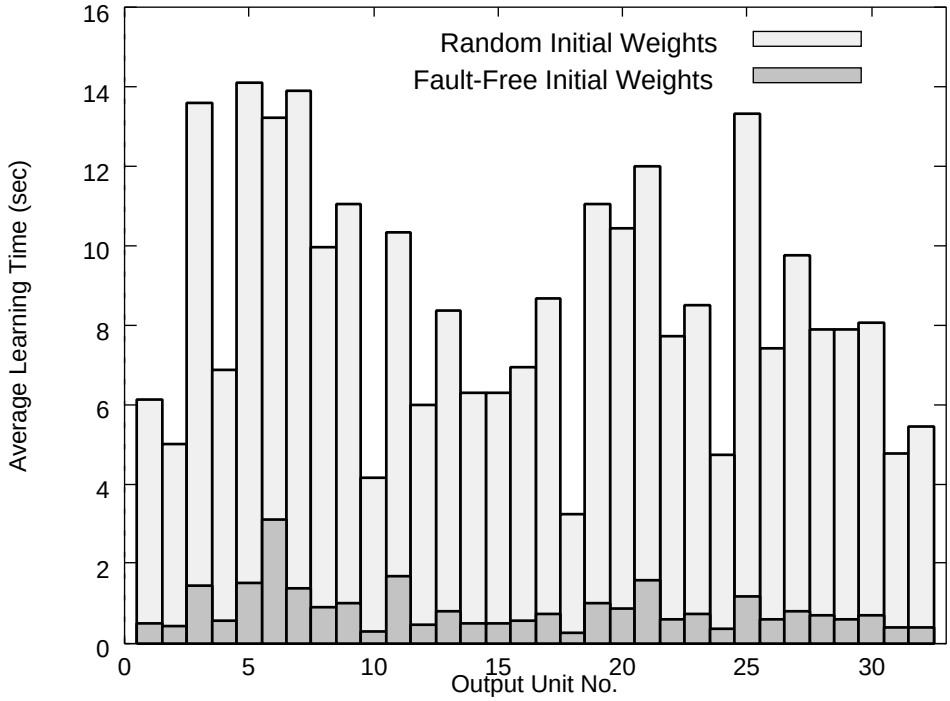

図 4.11: 部分再学習による大規模ネットワークの故障補償に必要な平均時間

像に対応する出力ニューロンのみが 1 を出力し，その他のニューロンは 0 を出力することとした．

以上のように生成した 64 次元の入力ベクトルを 32 人の顔画像から生成し，64-9-32 のニューラルネットワークにより学習を行った．このときの学習率は 0.7，モメンタム項は 0.3 とした．32 個の出力ニューロンそれぞれが持つ 9 本のリンクのうち 1 本が故障したとして再学習を行い．これを 9 本のリンクすべてについて行った．図 4.11に各出力ニューロンにおける平均再学習時間を，図 4.12に隠れ層のニューロンの入力パターンに対する最大，最小および平均出力を示す．

図 4.12より，XOR 問題の場合と異なって入力パターンは隠れ層のニューロンで均等に表現されていることが分かる．図 4.11より，入力を分散表現させた大規模ネットワークでは，初期重みとして故障前の重みを用いることにより，乱数初期重みを用いた場合より高速に故障補償を行うことができた．また，故障補償に必要な時間はどのニューロンのどのリンクの故障補償を行った場合でも大きく異なることはなかった．

複数リンクが同時に故障した場合として，第 1 出力ユニットの 2 本のリンクに故障が発

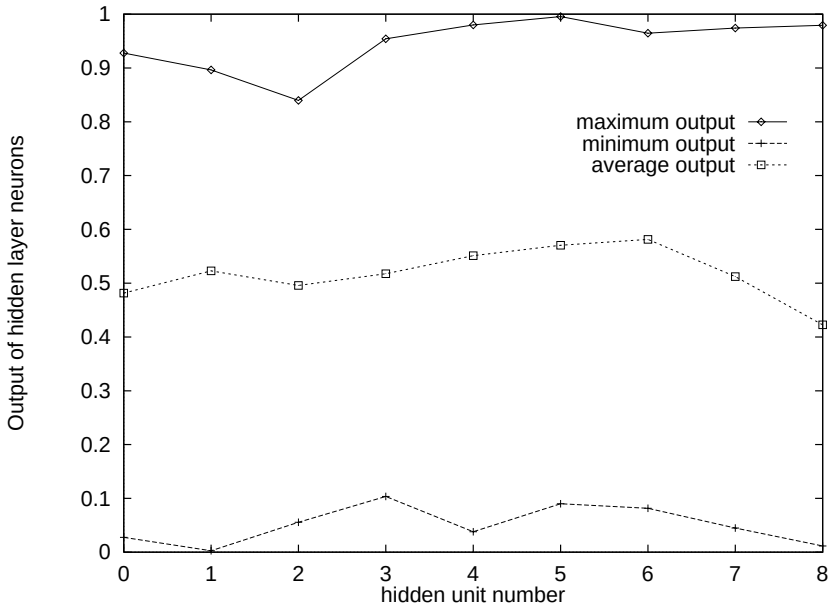

図 4.12: 入力パターンに対する隠れニューロンの最大，最小，平均出力

生したと仮定し，30,000 回を限度として部分再学習法による故障補償を行った．その結果，部分再学習法により 36 組の故障に対して 24 組の故障を補償することができた．さらに故障前の重みを初期重みとする場合，乱数を初期重みとして用いた場合の約 64%の時間で故障補償を行うことができた．以上より，実用規模の問題で適切な数の隠れニューロンを用意することで，部分再学習法により効果的な故障補償が可能と考えられる．

4.5 まとめ

本章では，階層型ニューラルネットワークにおける断線故障を高速かつ単純な方法で補償する部分再学習法について提案し，その有効性を XOR 問題と顔画像認識問題を用いて調べた．その結果，故障のない健常なリンクが持つ重みをそのまま初期重みとして用いる部分再学習法により，高速に再学習を行い故障補償を行うことができた．

XOR 問題を用いた小規模ネットワークでの実験では，3 つの隠れ層のニューロンが情報のほとんどを表現しており，これらのニューロンからのリンクが同時故障した場合には故障補償を行うことができず，より下層からの再学習が行うか，隠れ層による情報の多重化を行う必要があると考えられる．

実用規模の問題の例として顔画像を用いた画像認識問題をとりあげた．XOR 問題と異なり隠れ層ニューロンに情報が分散して表現されたため，どのニューロンのどのリンク故障についても故障補償にかかる時間が大きく異なることはなく，故障前の重みを初期重みとした部分再学習法が有効であることが示された．

第 5 章

自己組織化ニューラルネットワークの並列学習

5.1 はじめに

Kohonen により提案された SOM は，ベクトル量子化や制御などの分野での応用が提案されつつある [16]．しかしながら，SOM が平面上に広がった入力層，競合層を用い，各層間は完全結合で結ばれるため，2.3.3節で述べたようにユニット数を増すと計算量の爆発が起こる．そのため，並列処理による高速化手法 [17, 18, 19] が提案されているが，SOM の学習アルゴリズムが持つ特性のため，PE の負荷が不均一となり効率的ではない．また，画像データなどの巨大な入力層を必要とする問題では，従来の並列化手法では十分な学習速度を得ることができない．

次節では従来の並列化手法を概観し，その問題点について議論する．5.3節では，競合層ではなく入力層を分割して並列化を行う入力層分割法を提案し，学習時間をモデルを用いて導出する．同時に競合層分割法による並列 SOM の学習時間を同様に導出して比較検討を行う．また，入力層分割法を並列計算機に実装し，PE 間の負荷について議論する．

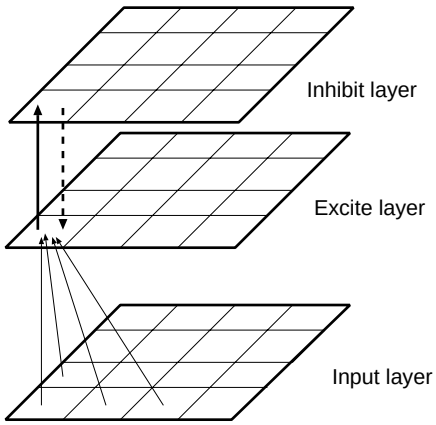

図 5.1: Obermayer による 2 層競合層モデル

5.2 SOM の並列学習法とその問題点

5.2.1 Obermayer による並列学習法

Obermayer ら [17] は，ニューロン数約 1 万 6 千，入力パターンが 900 次元という大規模な SOM を，Thinking Machines 社製超並列計算機 CM-2 と自作のトランスピュータにより並列実行する手法を提案した．

脳内のニューロン間に見られる側抑制を実現するために同一層内の結合を設けると計算量がさらに増加するため，SOM では通常最大活性値を出力しているノードからの距離に反比例した係数を重み更新時に用いる．Obermayer らはより生物の脳に近く，かつ計算量を削減するため競合層を興奮性，抑制性の係数を持つ 2 層とした．Obermayer らによる 2 層モデルを図 5.1に示す．

図 5.1を格子網を持つ CM-2，リング網を持つトランスピュータ上へ実装し，それぞれ細粒度，粗粒度処理として双方の性能を比較している．CM-2 では，競合層の 1 ユニットを CM 上の 1PE に割り当て，トランスピュータでは PE 数が少ないため，競合層を分割して実装している．Obermayer はこれらの手法により，入力次元数と競合層のユニット数にほぼ比例する処理時間で SOM を実行できることを報告している．

5.2.2 Chwan による並列化

Chwan ら [19] は，直鎖網上と 2 次元トーラス網上への SOM の実装について述べている．直鎖網上へは，競合層の 2 次元配列で垂直位置が重なるユニットを同一 PE に割り当てている．学習パターンは各 PE に入力された後，PE 上でのローカルウィナーが決定される．各 PE はローカルウィナーのユニット番号をローテーションし，グローバルウィナーが決定される．その後，グローバルウィナーの近傍が決定され，近傍内に位置するユニットを持つ PE が重みを更新する．

2 次元トーラス網上への実装は競合層をブロックに分割し，各ブロックをトーラス網上の PE に割り当てる．直鎖網への実装時と同じく，入力パターンが全 PE に与えられた後，各 PE は担当するエリア内でのローカルウィナーを決定する．各 PE でのローカルウィナーから，競合層でのグローバルウィナーを決めて近傍を計算し，近傍内のユニットを持つ PE により重みを更新する．

Chwan によるこの手法は，同一 PE に異なったエリアの競合層が割り当てられるため，PE の利用効率が良好であるという利点がある．たとえば直鎖網では，近傍の直径が PE 数より大きければ，すべての PE がビジー状態となる．しかし，学習終盤になり近傍直径が小さくなると，アイドルとなる PE が増えるという問題がある．

5.2.3 Myklebust による並列化

Myklebust ら [18] は，5.2.1節や 5.2.2 節のモデルがいわゆるユニット並列モデルであることを指摘した．ユニット並列モデルは 3章で示したように，特殊な場合を除いて通信量などの問題のため効率の良い並列化手法とは言えない．そこで Myklebust はユニット並列モデルと学習セット並列を組み合わせて SOM を並列化する手法を提案している．

Myklebust は RENNS という演算ユニット，通信マネージャ，メモリを持つモジュール 16 個を 2 次元トーラス網に接続した並列計算機を用い，その性能について評価している．トーラス網への実装は図 5.2に示すような形に行う．ネットワークのコピー一つに複数のモジュールを割り当て，各コピーの中ではユニット並列モデルを用いている．また，同時にユニット並列モデル，学習セット並列モデルも単独に実装し，それぞれの学習速度を報告している．

Myklebust らは，ユニット並列モデルよりも学習セット並列モデルやそれらを組み合わせたモデルの方が高速に学習が可能であることを報告している．しかし，3.2.2節で用いた

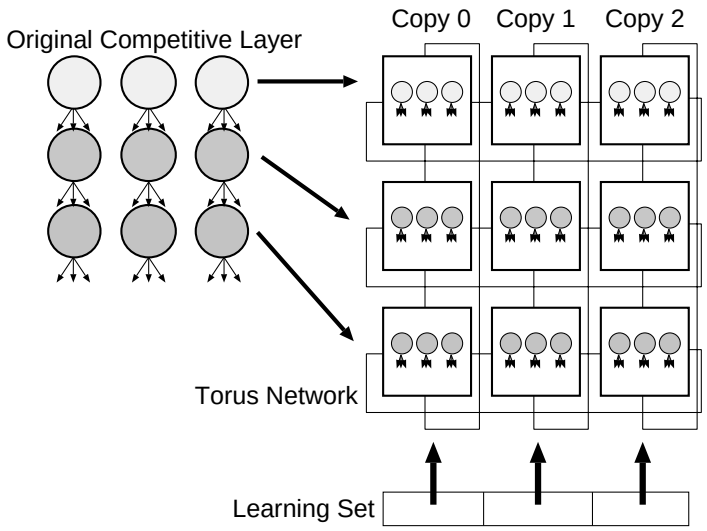

図 5.2: Myklebust によるユニット並列と学習セット並列を組み合わせた実装

ような重みのバッチ更新を用いるとトポグラフィックマップが生成されず，少数の学習パターンを処理した後に重みの更新操作が必要であると報告している．

5.2.4 従来の並列化手法の問題点

ユニット並列モデルを主体にした Obermayer，Chwan のモデル，およびユニット並列モデルと学習セット並列モデルを組み合わせた Myklebust のモデルについて概説した．

ユニット並列モデルでは，5.2.2節で述べたように学習が進むにつれて重み更新を行うユニット数が減少するため，競合層を分割するモデルでは学習後半の PE の使用効率が悪化し，負荷が不均一になるという問題がある．図 5.3に，入力層 2×1，競合層 10×10 の SOM に，ランダムに発生させた 100 個の学習パターンを入力して 15000 回の学習を行ったときの，各競合層ユニットの重み更新回数を示す．

図 5.3から分かるように，ウィナー，あるいは近傍ユニットとして重み更新を行った回数が最も多いユニットは，約 145,000 回，最も少かったユニットは約 54,000 回であり，約 2.6 倍の開きがある．したがって，Obermayer の並列モデルのような競合層分割によるユニット並列学習手法では，これがそのまま PE 間の処理時間の差となり並列化効率が低下してしまう．また，Chwan らの実装では，1 つの PE が複数のユニットを担当するため学習初期で近傍が大きいときの負荷は均一化されるが，学習が進んで近傍が小さくなると Obermayer

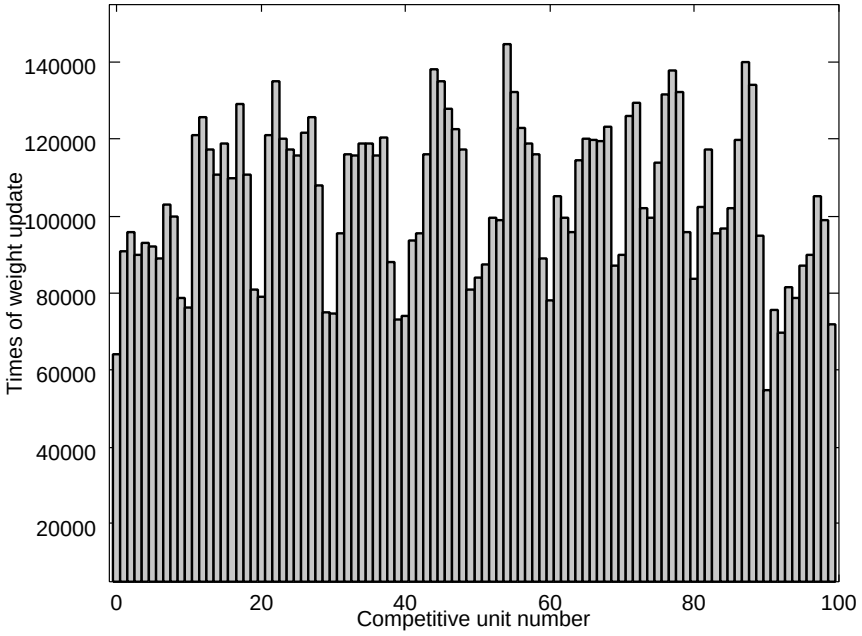

図 5.3: 競合層ユニットの重み更新回数

によるモデルと同様の問題が発生する．また，ウィナーの競合層上での位置が PE 間の境界に近く，近傍関数や学習率関数を距離の関数としている場合には，PE 間通信が余分に発生するという問題がある．Myklebust による並列化手法についても，競合層分割によるノード並列を基本にしているため，PE 間の負荷が不均一という問題は解決されていない．また，競合層のサイズが大きい大規模 SOM では，PE 間の通信が大幅に増加するために十分な高速化を行うことができないという問題もある．

以上の問題点を踏まえ，原理的に負荷の不均一が発生せず，大きな競合層を用いる大規模 SOM を高速に学習させる入力層分割法を次節で提案し，その評価を行う．

5.3 入力層分割による SOM 並列化

5.3.1 入力層分割法

SOM は入力層，競合層を 2 次元上に配置するため，ユニット数の増加は大幅な計算量増加をもたらす．そこで並列化による処理の高速化が求められているが，5.2.4節で述べたような問題が存在する．以下では，PE 間の負荷の不均一が原理的に発生しない入力層分割

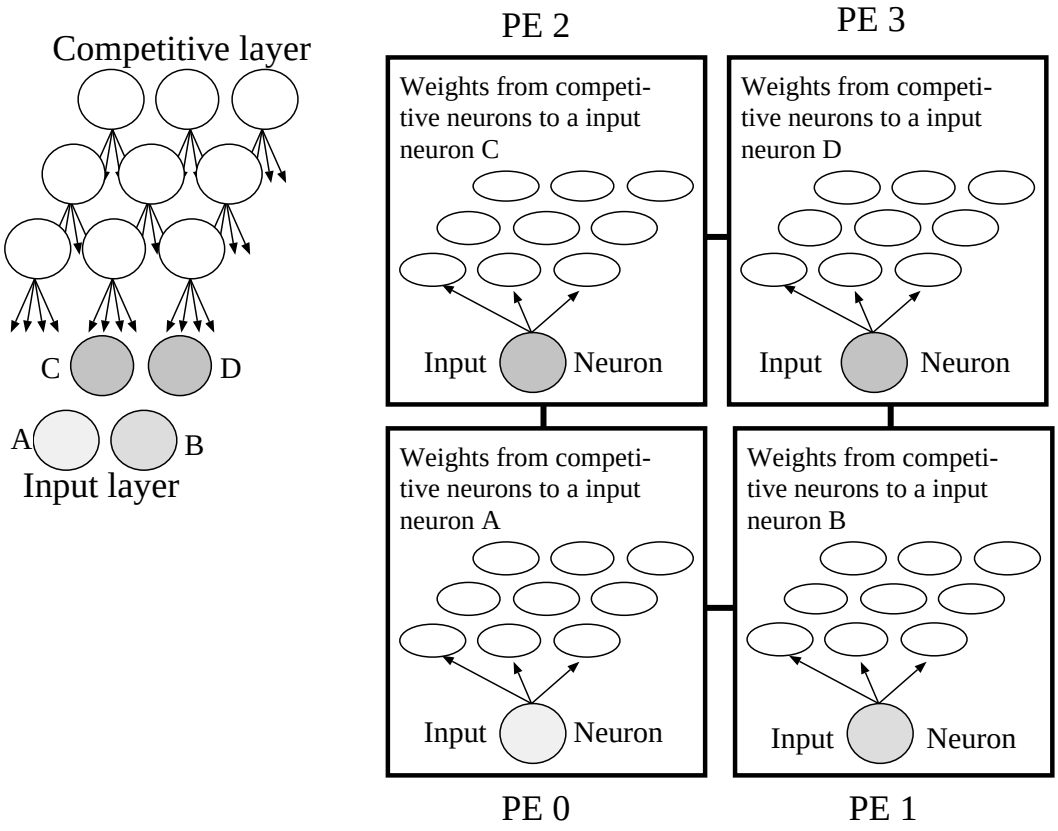

図 5.4: 入力層分割法の概要

による並列化を提案し，その性能を評価する．

図 5.4に，入力層分割の概念を示す．各 PE は入力層のユニットを担当し，競合層の各ユニットへの重みを持つ．したがって，競合層のどのユニットがウィナー，あるいは近傍となって重み更新を行っても，すべての PE が同等の処理をするため負荷の不均一は発生しない．

対象とする SOM は入力層，競合層の 2 層とし，それぞれ $M \times M$，$N \times N$個のユニットからなり，ある時刻 t における入力ユニット i と出力ユニット j間の重みを W_i^j $(i = 1, \cdots, M^2,\ j = 1, \cdots, N^2)$ とする．この SOM を $M \times M$個の PE に分割し，各 PE は N^2 次元の重みベクトルを持つ．学習パターン $T = \{t_1, \cdots, t_{M^2}\}$ が入力層ユニットに入力された場合を考える．各 PE は学習パターンで対応する要素を用い，入力ベクトル要素 t_iと重みベクトル W_i^jとの間のユークリッド距離 d_i^jを（5.1）式に基づき計算する．

Algorithm 5.1 入力層分割法

1 Initialize $W_i^j(0)$.
2 Distribute t_i to each PE i.
3 **do** on each PE i,
4 **for** $j = 0$ to N^2,
5 $d_i^j = ||W_i^j(t) - T_i||$.
6 broadcast d_i^j and calclate $d^j = \sum_i d_i^j$, then send it to PE i.
7 decide winner $c = \arg\min_j d^j$ and neighbours $N_c(t)$ on each PE.
8 $W_i^j(t+1) = W_i^j(t) + h_{cj}(t)(W_i^j(t) - t_i)\ j \in N_c(t)$
9 **while** $d^c < LIMIT$, or learn for max iterations.

図 5.5: 入力層分割学習法のアルゴリズム

$$d_i^j = \sqrt{(t_i - W_i^j)^2},\ j = 1, \cdots, N^2 \tag{5.1}$$

各 PEi が $d_i^j,\ j = 1, \cdots, N^2$を計算した後で，$i$ について d_i^jを（5.2）式により集計し，結果を全 PE にブロードキャストする．

$$d^j = \sum_{i=1}^{M^2} d_i^j \tag{5.2}$$

その結果，競合層ユニット jと学習パターン間の距離 d^jをすべての PE が持つことになる．各 PE は独立に d^jが最も小さいものをウィナー c として選び,（2.32）式，あるいは（2.33）式で示される近傍関数 h_{cj}を用いて c とその近傍 $N_c(t)$ の重みを更新する．したがって，各 PE は同一個数の重み要素を更新することになり，負荷の不均一は原理的に発生しないことになる．図 5.5に，入力層分割法による SOM の並列学習アルゴリズムを示す.

5.3.2 入力層分割法の学習時間

2.3.2節と同様に，入力層分割を用いた並列 SOM の学習時間について考察する．入力層，競合層はそれぞれ $M \times M$，$N \times N$の 2 次元格子を構成しており，$N_{PE} = M^2$個の PE を持つ並列計算機に実装するとする．近傍関数は（2.32）式，近傍範囲 $N_c(t)$ は（2.34）式とした．

学習パターンは各要素に分解され，各要素が PE に分配される．各 PE では学習パターン要素と競合層ユニットへの重みとの 2 乗誤差を計算する．これに必要な時間 T_d'は次式で

表される．

$$T'_d = N^2(t_{add} + t_{multi}) \tag{5.3}$$

次に競合層中のユニットで最も学習パターンに近いユニットを決定するが，これには各 PE が持つ誤差要素を集計する必要がある．この集計には簡単化のため，各 PE が一つのホストに誤差要素を送り，ホストで加算処理後各 PE にブロードキャストするというモデルを用いる．すべての PE がホストにデータを送るのに必要な時間が M^2t_{comm}，ホストが送られたデータを加算する時間が $(M^2-1)t_{add}$，加算した結果の平方根を計算するための時間が N^2t_{sq}，計算結果を全 PE に送信するのに必要な時間が M^2t_{comm}であるので，誤差集計に必要な時間 T'_c は次式のようになる．

$$T'_c = (M^2-1)t_{add} + N^2t_{sq} + 2M^2t_{comm} \tag{5.4}$$

各 PE はホストから受信した集計後の誤差をもとにウィナーと近傍を決定するが，これは逐次処理を行ったときと同じ T_s, T_nである．近傍 $N_c(t)$ 内の競合層ユニット数は $N_c^2(t)$ なので，これの重み変更に必要な時間 T'_{uw}は (5.5) 式で表される．

$$T'_{uw} = (2t_{add} + t_{multi})N_c^2(t) \tag{5.5}$$

以上より，学習パターン一つを入力層分割 SOM で学習するのに必要な時間 T'^{som}_{total}は（5.3）式,（5.4）式,（5.5）式と T_n, T_sとの総和であり,（5.6）式で表される．

$$\begin{aligned} T'^{som}_{total} &= T'_d + T'_c + T_n + T_s + T'_{uw} \\ &= \left\{M^2 + N^2 + 2N_c^2(t) + 1\right\} t_{add} + \left\{N^2 + N_c^2(t) + 4\right\} t_{multi} \\ &\quad + N^2t_{sq} + T_s + 2M^2t_{comm} \end{aligned} \tag{5.6}$$

入力層分割法を用いて，N_p個の学習パターンを N_{epoch}回学習するのに必要な時間は,（5.6）式に N_pを乗じ，それを N_{epoch}まで加算した（5.7）式で表されることになる．

$$T'^{som} = \sum_{t=1}^{N_{epoch}} N_p T'^{som}_{total} \tag{5.7}$$

5.3.3 競合層分割法の学習時間

本節では従来の競合層分割法による学習時間について議論する．入力層分割法と同じ条件を用い，1PE に競合層の 1 ユニットを割り当てると仮定する． 学習パターンは 1 パター

ン毎にすべてのPEにブロードキャストされ，並列に競合層ユニットの誤差を計算する．これに必要な時間T''_dは（5.8）式で表される．

$$T''_d = M^2(t_{add} + t_{multi}) + t_{sq} \tag{5.8}$$

各PEは単一の競合層ユニットを割り当てられているので，競合層全体でのウィナーを決定するためには一旦計算した誤差をホストに送る必要がある．ホストはウィナーを決定後，各PEにウィナーをブロードキャストする．これに必要な時間T''_cは（5.9）式となる．

$$T''_c = 2N^2 t_{comm} + t_s \tag{5.9}$$

ウィナー決定後，近傍半径および学習率の決定には，他のモデルと同様にT_nの時間が必要である．重みの更新は近傍範囲内のユニットを割り当てられたPEのみが行い，他のPEはアイドル状態で処理の終了を待つとすると，重み更新に必要な時間T''_{uw}は（5.10）式で表される．

$$T''_{uw} = M^2(2t_{add} + t_{multi}) \tag{5.10}$$

以上より，競合層を分割したときの並列モデルにより学習パターン1つを処理するのに必要な時間T''^{som}_{total}は（5.11）式となる．

$$\begin{aligned} T''^{som}_{total} &= T''_d + T''_c + T_n + t_s T''_{uw} \\ &= (3M^2 + 2)t_{add} + (2M^2 + 4)t_{multi} \\ &\quad + t_{sq} + t_s + 2N^2 t_{comm} \end{aligned} \tag{5.11}$$

以上よりN_p個の学習パターンをN_{epoch}回学習するのに必要な時間T''^{som}は（5.12）式で表される．

$$T''^{som} = N_{epoch} N_p T''^{som}_{total} \tag{5.12}$$

5.3.4 入力層分割法と競合層分割法の比較

（2.38）式による逐次処理SOMと,（5.6）式による入力層分割，および（5.11）式による競合層分割による並列SOMの処理時間を比較する．$M = N = n$，$N_{epoch} = 1000$,$N_p = 100$，$t_s = n \times t_{add}$とし，理想並列計算機上での各学習法による学習時間を図5.6に示す．並列学習における学習時間の計算では，ユニット数と同数のPEを使用すると仮定している．ま

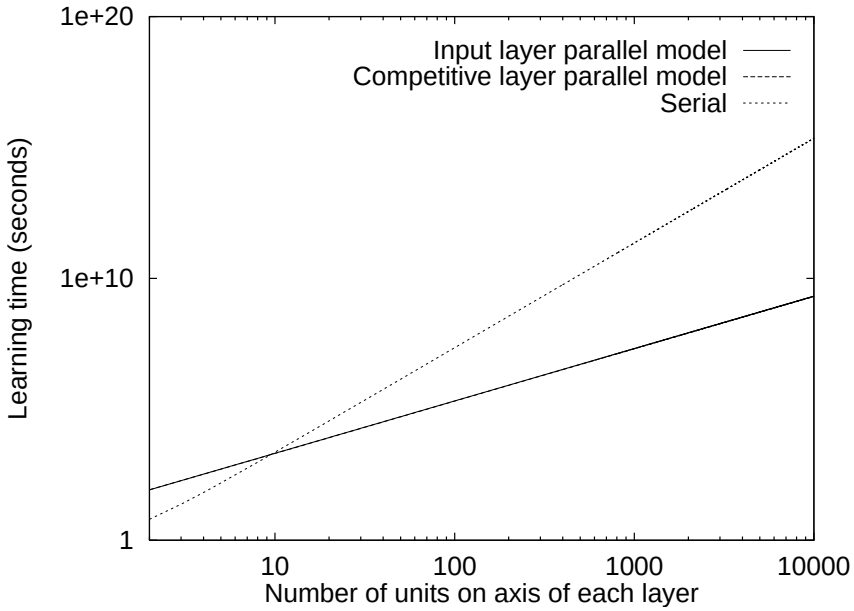

図 5.6: 逐次処理および理想並列計算機上での入力層分割と競合層分割学習法の学習時間

た，各学習時間の計算における各処理時間は，nCUBE/2 での実測により以下のように設定した．なお，通信時間 t_{comm} は 1 対 1 通信のときの値である．

$$
\begin{aligned}
t_{add} &= 1\mu sec. \\
t_{multi} &= 1\mu sec. \\
t_{sq} &= 10\mu sec. \\
t_{comm} &= 100\mu sec.
\end{aligned}
$$

図 5.6に示すように，逐次処理と比較して並列処理を用いた学習は大幅な高速化が達成できる．これは，逐次処理では通信時間は各層の軸上のユニット数 n に対して $\mathcal{O}(n^4)$ で処理時間が増加するのに対し，並列化時には $\mathcal{O}(n^2)$ であり，ネットワークのサイズが大きくなるにつれ並列化による効果が大きくなる．モデルによるシミュレーションでは，100×100 の SOM では 1 万個の PE を用いた並列処理により約 100 倍の，1000×1000 の SOM では約 10000 倍の高速化が達成できる．入力層分割法と競合層分割法を比較した場合，競合層分割法の方がわずかに高速である．これは競合層分割法での 2 乗誤差の計算が並列で t_{sq} で行うことができるのに対し，入力層分割法は $N^2 t_{sq}$ となることが一因と考えられる．

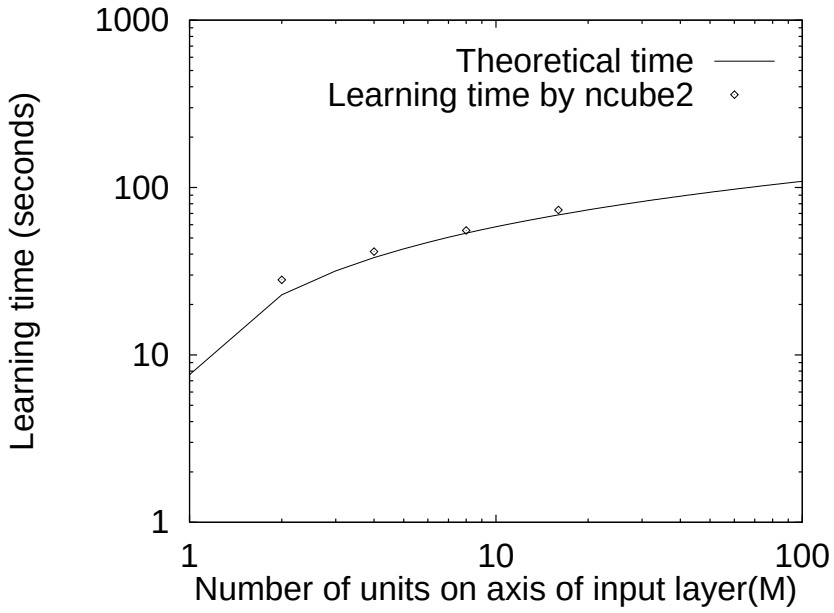

図 5.7: 並列 SOM 学習法のモデルと実装結果の比較

図5.7に，入力層分割法を nCUBE/2 に実装したときの学習時間とモデルによる計算時間を示す．このとき，入力層分割法での通信時間は配列加算通信を用いることにより，$2M^2t_{comm}$ から $t'_{comm}(N_{PE}, Mlen)$ となる．t'_{comm}において，$N_{PE} = M^2, Mlen = N^2$である．また，SOM の規模は競合層を 10×10 とし，入力層のサイズを $2 \times 2, 4 \times 4, 8 \times 8, 16 \times 16$ とし，$N_p = N_{epoch} = 100$ とした．図 5.7より，モデルによる計算時間は実装による実験結果と誤差 10%以内で一致している．この結果から，SOM の学習時間検討の手段として（5.7）式，（5.12）式を用いることは妥当と考えられる．

さて，図 5.8に PE 数とメッセージ長を考慮した通信時間を用いた場合の並列学習時間を示す．このときの通信時間は，3.3.2節で示した（3.29）式を用いた．図 5.8より，理想並列計算機上での比較では差が見られなかったが，メッセージ長や PE 数による通信遅延を考慮すると競合層分割法に比べて入力層分割法の学習時間が長くなっている．競合層分割法では通信で送られるメッセージ長は一定であるが，入力層分割法では競合層のユニット数に比例してメッセージ長が増加し，ネットワークの規模が大きくなるにしたがって一回の通信時間が増大する．このことから，各層で同一規模のユニットを持つ SOM では，競合層分割法の方がやや高速となる．

次に，入力層と競合層のユニット数が異なる場合について検討する．図 5.9に入力層を

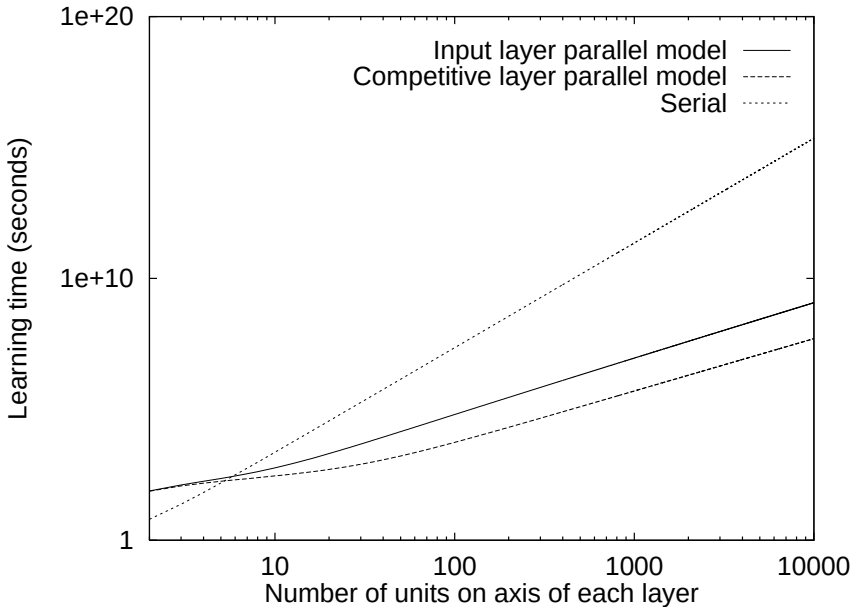

図 5.8: 通信遅延を考慮した並列 SOM の学習時間の比較

100×100 として競合層のユニット数を変化させたときの学習時間を，図 5.10に競合層を 100×100 に固定して入力層のユニット数を変化させたときの学習時間を示す．このとき，$N_p = 100, N_{epoch} = 1000$ である．

入力層を 100×100 に固定した場合，図 5.9より競合層ユニット数が少ない間は入力層分割法の方が高速である．しかし，競合層のユニット数が増加するにしたがって入力層分割法の学習時間は長くなり，競合層分割法の方が高速になる．入力層分割法では，(5.6）式より学習時間が競合層サイズ N^2により主として支配されているため，競合層のサイズが大きくなると学習時間が長くなる．これに対し，(5.11）式で示される競合層分割法の学習時間では Nを含む項はなく，通信時間が $t'_{comm}(N^2, 1.0)$ で Nを影響を受けるのみである．

図 5.10では競合層を 100×100 に固定し，入力層のサイズを変化させたときの学習時間を示す．入力層分割法では（5.6）式から分かるように t_{add} の係数と通信時間 $t'_{comm}(M^2, N^2)$ に Mが含まれているため，入力層サイズが大きくなると学習時間が長くなっていく．しかし，競合層分割法では（5.11）式より学習時間は Mに支配されるため，入力層分割法よりも学習時間の増加が大きくなり，入力層のサイズがおよそ競合層の 4 倍以上では入力層分割法よりも学習時間が長くなる．

以上より，提案した入力層分割法は画像符号化などで多く用いられている大規模な入力

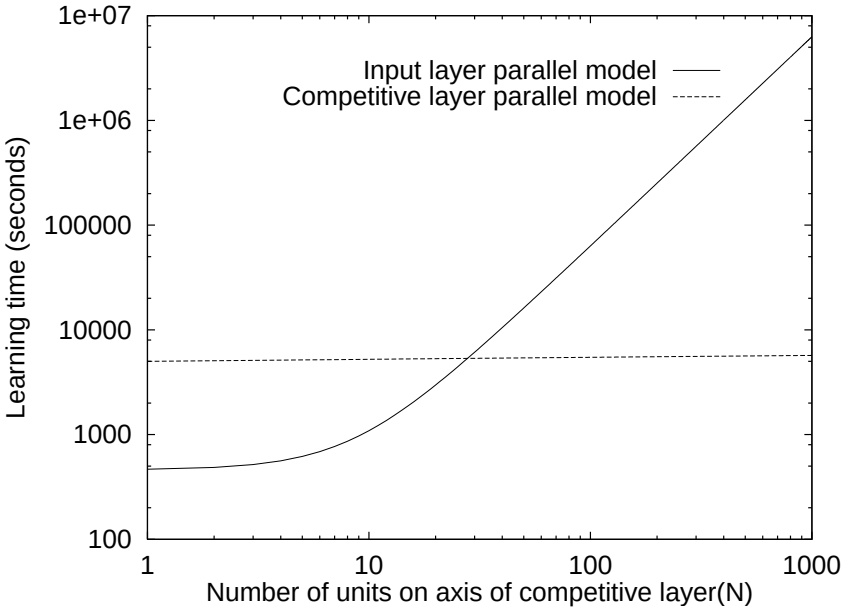

図 5.9: 入力層を 100×100 としたときの学習時間

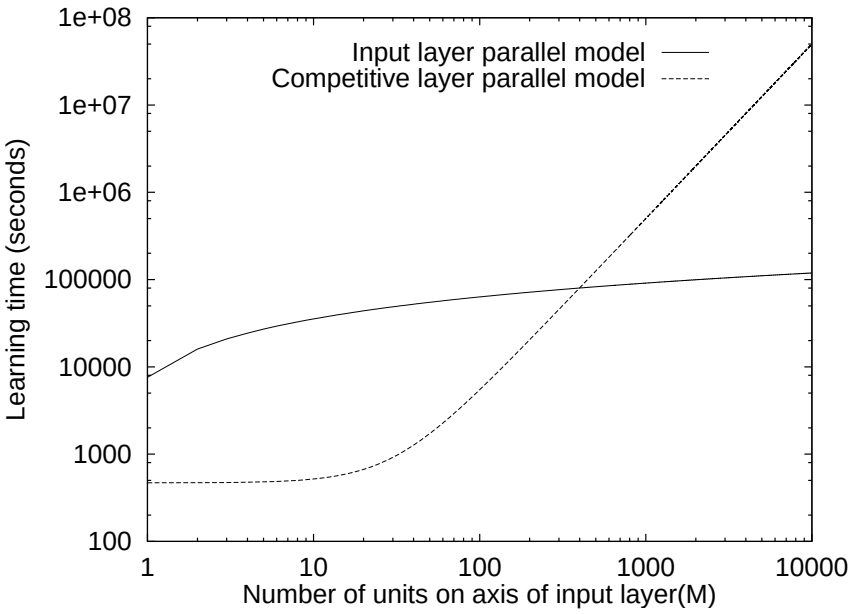

図 5.10: 競合層を 100×100 としたときの学習時間

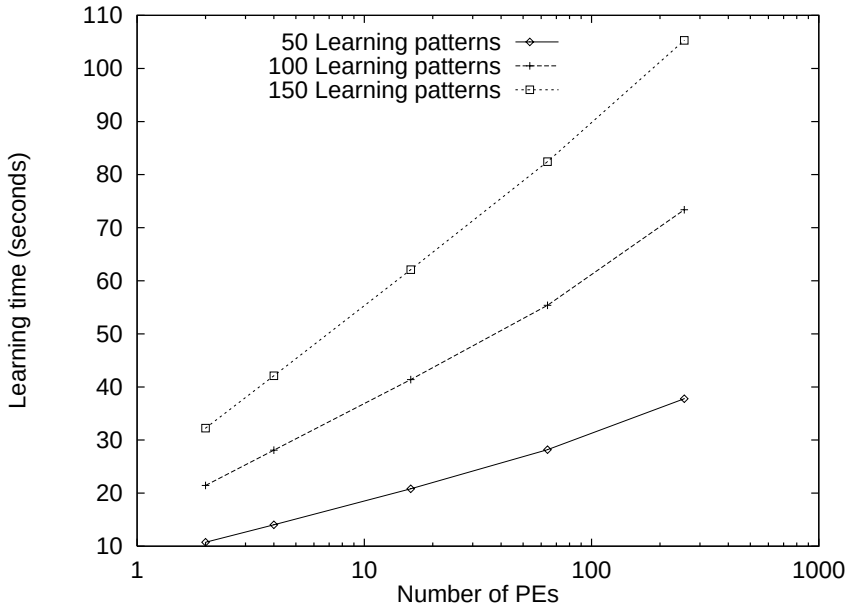

図 5.11: 競合層サイズを 10×10 として入力層のサイズを変化させたときの入力層分割法の学習時間

層を持つ SOM に適した学習法であるといえる．

5.3.5 並列学習実験

学習時間に関する考察

本節では，並列計算機 nCUBE/2 を用いて入力層分割法の評価を行う．用いる SOM のサイズは入力層を $2 \times 1, 2 \times 2, 4 \times 4, 8 \times 8, 16 \times 16$，競合層を 10×10 とし，学習パターン数を $50, 100, 150$，最大学習回数を 100 として学習を行った．このときの学習時間を図 5.11 に示す．

図 5.11より，学習時間は学習パターン数に比例して増加しており,（5.7）式で示したモデルと同様の結果となっている．PE 数の変化による学習時間の増加を見ると，PE 数の log，すなわちハイパーキューブ網の次元数に比例して学習時間が増加している．これは，入力層分割法での通信時間が（3.29）式のようにハイパーキューブ網の次元数に比例して増加するためである．

図 5.12に，入力層を 4×4 に固定して競合層のサイズを変化させたときの学習時間を示

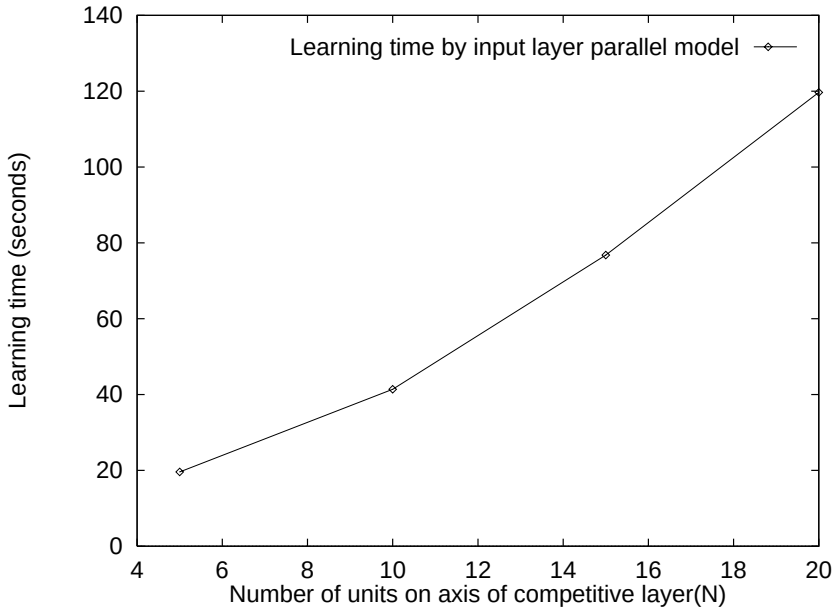

図 5.12: 4×4 の入力層で競合層サイズを変化させたときの学習時間

す.（5.6）式より，入力層分割法では四則演算の処理時間は N^2 に比例し，通信時間もメッセージ長の増加といった形で競合層サイズの影響を受ける．図 5.12より，競合層の軸上のユニット数の変化に対してわずかではあるがほぼ軸上の競合層ユニット数の 2 乗で学習時間が増加していることが分かる．

図 5.13に，入力層のユニット数が変化したときの逐次計算モデルと入力層分割法での相対学習時間を示す．入力層分割法の相対学習時間は 10×10 の競合層で nCUBE/2 で学習したときの値を使用した．図 5.13から，逐次処理では入力層サイズの増加に比例して学習時間が長くなることが分かる．これに対して入力層分割法では，入力層のサイズが約 100 倍となっても学習時間の増加は約 3.5 倍に留まっており，並列化により大幅な学習時間の短縮が実現されていることが分かる．

PE 間の負荷に関する考察

本節では，入力層分割法及び競合層分割法での各 PE 間の負荷状況について検討する．入力層分割法，競合層分割法ともに入力層，競合層のサイズを同一とし，100 個の学習パターンを 100 回学習した．入力層分割法については nCUBE/2 上での各 PE の学習時間を用いて負荷を検討する．競合層分割法については，重み更新対象ユニット以外を受け持つ PE

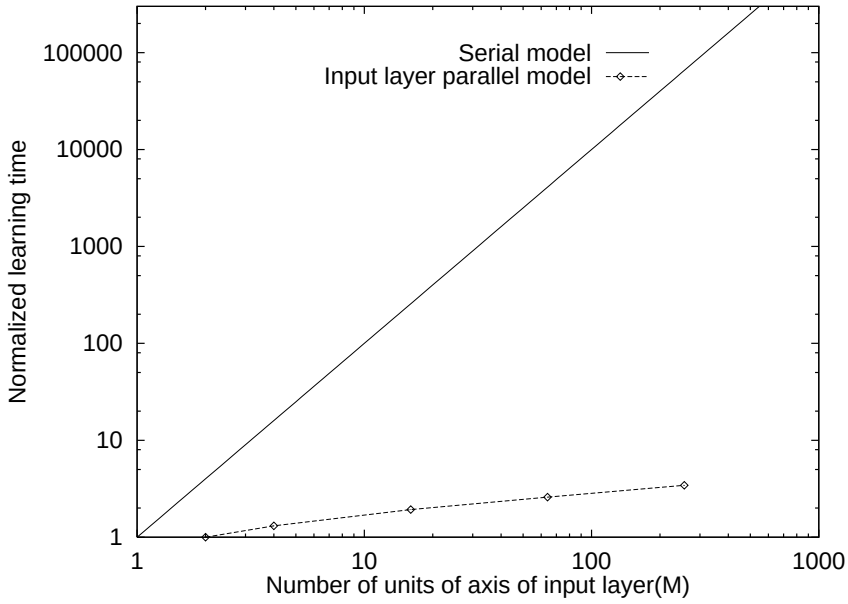

図 5.13: 入力層ユニット数の変化に対する相対学習時間

はアイドルとなるため，各 PE の学習時間は重み更新回数に比例する．したがって，学習時間に代えて各 PE での重み更新回数を元に各 PE での負荷状況を議論する．

図 5.14に，SOM のサイズを 4×4 としたときの入力層分割法および競合層分割法での各 PE の学習時間と重み更新回数を示す．入力層分割法では，PE0 を除きほぼ同一の学習時間となっている．配列加算通信では，各 PE が持つ誤差を PE0 にいったん集めたあとその結果をブロードキャストするため，PE0 での学習時間がやや長くなる．入力層分割法において，負荷の差はおよそ 3%程度であった．これに対して重み更新回数がそのまま学習時間の差として現れる競合層分割法では，負荷の差はおよそ 2 倍となっている．図 5.15で示した 8×8 の SOM でもこの数値はほぼ同一であり，入力層分割法では事実上負荷の不均一が発生していないことが分かる．

以上より，入力層分割法では PE 間の負荷の不均衡が発生しないことが実験的にも確認された．

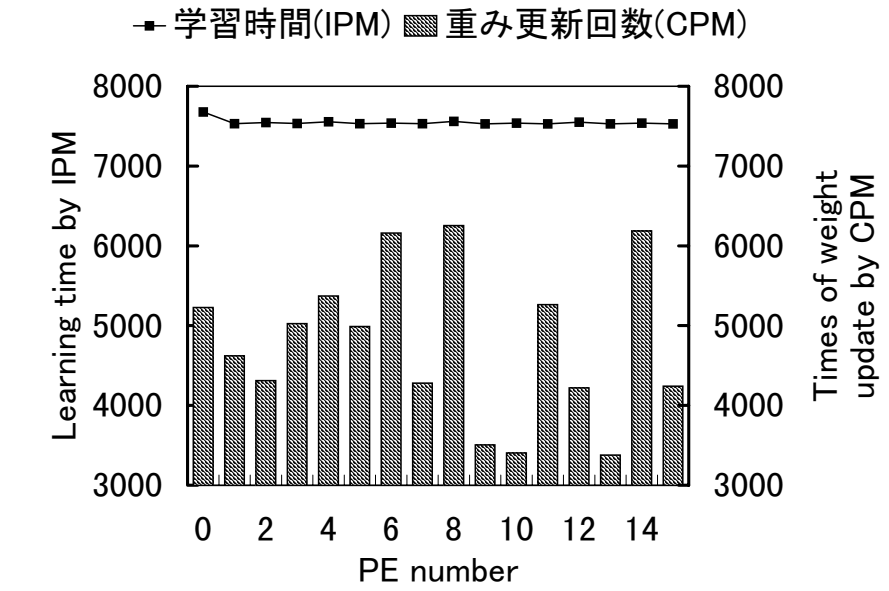

図 5.14: 16PE を使用する SOM での各 PE の学習時間と重み更新回数

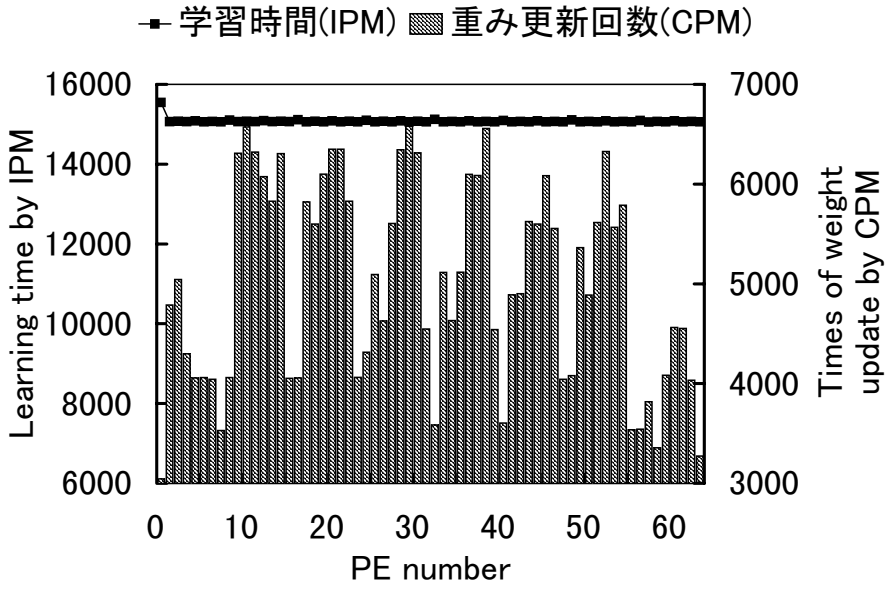

図 5.15: 64PE を使用する SOM での各 PE の学習時間と重み更新回数

5.4 まとめ

Kohonen による SOM は従来の階層型ニューラルネットワークと異なりユニットが 2 次元マップ状に配置されており，類似した入力に対して近傍のユニットが反応するというトポグラフィックマッピングを行うことができるという特徴がある．こうした特徴は，ベクトル量子化などの分野で応用され，相応の成果をあげつつある．しかし，多数のユニットとその相互結合を用いるため，ネットワークが大きくなると学習時間が膨大になるという，ニューラルネットワークに共通した大きな問題がある．

こうした問題を解決するため，並列処理によるいくつかの高速学習法が提案されている．しかしながら，SOM の学習アルゴリズムのため従来の競合層分割による並列学習では PE 間の負荷が不均衡になり，学習後半にアイドルとなる PE 数が増えてしまう．画像などのユニット数が膨大になる問題や，多数の入力ユニットを必要とする多次元情報の分類を行う問題では十分な高速化が行えないという問題があった．

本章では，PE 間の負荷の不均一が原理的に発生しない入力層分割法による SOM の並列学習法を提案し，その学習速度について検討した．その結果，256PE（16×16 の入力層）を使用した時の各 PE の負荷のばらつきは約 3%となり，優れた負荷分散性を示した．これに対し，従来の競合層分割法では，PE 間の負荷は最大で約 2 倍のばらつきとなることを示した．

学習時間の解析では，100×100 程度の SOM の学習においてユニット数と同数の PE を用いることにより，逐次処理の場合と比較して約 3000 倍の高速化が達成できることを示した．また，ネットワークの規模が大きくなればなるほど，並列化による高速化の効果は大きくあらわれる．ここで提案した入力層分割法は，入力層と競合層のユニット数が同程度では従来の競合層分割法ほぼ同じ程度の学習時間である．競合層のユニット数が入力層のユニット数よりも多くなると，入力層分割法の方が高速となる．したがって，入力層分割法は膨大な数の入力層ユニットを必要とする画像符号化などの分野や，多数の入力情報を分類・統合するようなクラスタリングなどの問題に対して効果的であると考えられる．

第 6 章

忘却学習法

6.1 はじめに

Kohonen による SOM では与えられた入力に対して最も近い重みを持つ競合層のユニットがウィナーとして選択され，近傍とともに重みを更新する．学習の進展にともない，近傍を縮小していくことでボロノイ領域を形成し，隣接するユニットが類似した重みをもつように学習されていく．このとき，学習結果に影響を及ぼさないデッドノードが発生する場合があることが報告されている [16]．特に，学習パターンの分布に突出部があるような場合では，マップの一部にウィナーが集中してしまい，全体的なマップピングが崩れてしまう場合がある．Kangas ら [20] は MST（Minimal Spanning Tree，最小結合木）を用いた新しいマッピング方法を提案し，ユニットの空間的配置でなくユニットが持つ重みベクトルの相対的な違いに基いた隣接関係の定義を行っている．また，Hendtlass[35] は，ユニットの自己生成・消滅により，3 次元空間中にユニットを自動配置するアルゴリズムを提案している．しかしながら，これらの手法ではユニットの空間的配置による位相関係がなくなってしまうという問題がある．また，並列化による高速学習という観点からも，ユニットの割り当てが困難になる．そこで本章では，石川 [36] により階層型ニューラルネットワーク用に提案されている忘却学習を SOM に取り入れることにより，学習に関与しないデッドーノードを削除する手法を提案し，その性能を評価する．

4章で述べたように，ニューラルネットワークを高速に実行するためにはハードウェア実装による高速化が有効である．このためには，ハードウェア上に存在する故障 PE を補償する学習法が不可欠となる．階層型ニューラルネットワークに対しては様々な故障補償法

が提案されているが，SOM のような自己組織化ニューラルネットワークを対象とした故障補償法は従来ほとんど検討されていない．そこで本章では，デッドノードを削除する忘却学習法を故障 PE を含む SOM の学習に応用し，トポグラフィックマップの生成を行う手法を検討する．作成されたトポグラフィックマップの評価を行うために，格子状のマップの品質，および学習パターンの近似精度といった評価尺度を用いて，故障を含む SOM の忘却学習法の評価を行う．

6.2 忘却学習アルゴリズム

6.2.1 忘却の定義

SOM の忘却学習で用いる「忘却」とは,「記憶していたことを忘れる」という意味ではない．脳がどのような機能を用いて「記憶」を実現しているのか分かっていない以上,「記憶」が消滅・想起不能となる「忘却」のメカニズムも定かではない．そこで，本節では抽象度の高い「記憶」と対になる「忘却」ではなく，ニューロンあるいはシナプスレベルでの「忘却」について定義し，その応用について検討する．

ニューロンはシナプスにより情報を伝達し，高度の情報処理を行っている．ニューロンやシナプスが破壊されたときに起こることは，SOM における「忘却」を考える上での基礎となる．

人間の脳の重量は生後6ヶ月の間急速に，その後緩やかに増加する．この重量増加はニューロンの増大によるものではなく，樹状突起の進展やグリア細胞と呼ばれる信号の伝達機能を持たない細胞の増加によるものであることは知られていたが，この時期に過剰神経細胞の死滅が起こっていることが最近明らかになってきた [37]．また，成熟した脳では各部位の神経細胞がどの標的細胞に軸索を伸ばすかは正確に定まっているが，新生仔期の哺乳動物ではこの対応が正確でない．すなわち，成長過程でニューロン間の対応に淘汰が行われており，これは新生仔期の異所性軸索側枝の退行であると考えられている．

以上のことから，脳で行われているらしい淘汰は以下の 2 点のようにまとめることができる．

ニューロン: 本来の機能を実現できない細胞は死滅する．

シナプス: 情報を表現するのに不要な結合はなくなる．

SOM では，ニューロンの淘汰は競合層のユニットを淘汰することに相当する．すなわち，SOM では学習プロセスを通じて入力パターンに関係を持たないユニットを削除し，ユニット数を減少させることなる．また，シナプスの淘汰は入力層-競合層間のリンクのうち，入力パターンを表現するのに不必要なリンクがなくなることと考えられる．リンクの存在自体は重みの値として表現できる（重みが常に 0 のリンクは，そのユニットの出力に影響を与えない）ため，SOM に Hebb 学習を用いることでリンクの淘汰は始めから獲得されていると考えることができる．そこで，以降ではユニットの淘汰を対象とした忘却を以下に定義する．

定義 6.1 (忘却) 平面上に配置された競合層を持つ SOM において，学習に関与しない競合層のユニットを削除することを忘却と定義する．

以降の節では SOM の競合学習に忘却を導入し，偏った分布を持つ学習セットを用いてのトポグラフィックマッピング作成実験を行い，その結果について検討する．

6.2.2 SOM の忘却学習

忘却学習は，1990 年に石川 [36] により階層型ニューラルネットワークでの不要なリンクを削除するために提案された手法であり，6.2.1節での分類ではシナプスによる忘却と考えることができる．

階層型ニューラルネットワークでは，全部の層間結合を用いた密結合学習が行われる傾向がある．しかし，入力パターンを表現するのにすべてのリンクが必要であるとは限らない．できるだけ少ないリンクとユニットで学習セットの骨格構造を形成するため，誤差逆伝搬学習の損失関数（2.5）式に忘却項を追加し，(6.1）式により重みの修正を行う．

$$\Delta w'^{L,j}_{L-1,i} = \Delta w^{L,j}_{L-1,i} - \varepsilon' \mathrm{sgn}(w^{L,j}_{L-1,i}) \tag{6.1}$$

ここで$\Delta w^{L,j}_{L-1,i}$は L 層のユニット i から $L-1$ 層のユニット jへの結合重みの誤差逆伝搬学習による修正量，ε'は忘却係数である．石川の忘却学習では，学習の間は常に忘却が行われる．忘却係数のために誤差最小値からずれた位置に重みが収束しようとするために，通常の誤差逆伝搬学習よりも学習時間が必要となる．論理関数の学習などでのシミュレーションにより，明確な骨格構造が忘却学習により獲得可能なことが報告されている．

石川による階層型ニューラルネットワークの忘却学習法はリンクが持つ重みの忘却に相当する．ところで，SOM では競合層のユニットが持つ重みはそのまま信号空間を表現している．したがって石川の手法のように重みを忘却により変更すると，SOM におけるトポグラフィックマッピングの生成ルールから逸脱してしまう．たとえば，SOM では学習が進むにしたがって重み更新対象となる近傍範囲が縮小していき，最終的にはウィナーのみが重みの更新を行う．したがって，学習後半にデッドノードとなったユニットの重みを忘却により変更することは完成しつつあるトポグラフィックマッピングを破壊してしまう可能性が考えられる．したがって，SOM における忘却学習として定義 6.1で述べたように忘却は競合層のユニットを単位とし，提示されるすべての学習パターンに対して重みの更新を行わなくなったユニットをデッドノードとして削除する．

入力層 $M \times M$，競合層 $N \times N$のユニットを持つ 2 層の SOM を考える．ある時刻 t における入力層のユニット i から競合層のユニット jへの結合重みを $W_i^j(t)$ とし，学習セットは M^2次元の要素を持つ学習パターン $T_p = \{t_1^p, \cdots, t_{M^2}^p\}, p = 1, \cdots, N_p$とする．競合層の各ユニットは，学習パターンと重みの間のユークリッド距離 d^jを次式で定義する．

$$d^j = ||t_i^p - W_i^j||, \qquad (i = 1, \cdots, M^2) \tag{6.2}$$

最も距離 d^jが小さいものをウィナー c として近傍 $N_c(t)$ を決定する．近傍 $N_c(t)$ 内にある競合層ユニットは（2.33）式で表される学習率を用いて次式により重みの更新を行う．

$$W_i^j(t+1) = W_i^j + h_{cj}(t)\left\{W_i^j(t) - t_i^p\right\}, \quad j \in N_c(t) \tag{6.3}$$

学習セットに含まれる学習パターンを学習する間に，一度も重みの更新を行わなかったユニット群 kの全ての重み W_i^kを全て 0 として忘却を行い，これらのユニットの隣接関係を定義しない．

Kohonen の SOM 学習に対し忘却を導入した忘却学習法の概要を図 6.1に示す．はじめに，標準的な SOM の学習アルゴリズムに従い，重みの更新を行う．次に一通りの学習セットの学習の間に一度もウィナー，あるいは近傍として重みの修正を行わなかったユニットを忘却対象とし，競合層から削除する．次節以降ではシミューレーションにより忘却学習の有効性について検討する．

Algorithm 6.1 SOM の忘却学習法

1 Initialize W_i^j,
2 Do
3 **for** $p = 0$ to N_p,
4 **for** $j = 0$ to N^2,
5 $d^j = ||t_i^p - W_i^j||, i = 1, \cdots, M^2$.
6 Decide winner c and neighbours $j \in N_c(t)$.
7 $W_i^j(t+1) = W_i^j(t) + h_{cj}(t)(t_i^p - W_i^j),\ j \in N_c(t)$.
8 $W_i^k = 0$ and extinguish this unit.
9 **repeat** $d^c < LIMIT$ or learn for max iterations.

図 6.1: 忘却学習のアルゴリズム

6.3 忘却学習法の性能評価

忘却学習の効果を確認するためにシミュレーションによる実験を行う．忘却学習は学習に関与しないデッドノードの削除を目的としている．そこで，デッドノードを発生しやすい状況として構造分布入力を持つ学習パターンを使用し，通常の Kohonen による学習を行った場合と比較する．学習セットは，結果を視覚的に確認するため 2 次元の入力とし，初期学習率 0.4，最大学習回数 15,000，近傍直径初期値は競合層各軸のユニット数の 60%の値とし，学習パターンとそのパターンのウィナーの重みのユークリッド距離が 0.03 以下となったところで学習終了とした．競合層のサイズは 10×10， これに 150 個の学習パターンを提示して忘却学習を行った．

6.3.1 一様分布入力の学習

本節では，信号空間に一様に分布した学習パターンを用いて，忘却を行った場合と行わない場合との比較を行う．学習パターンは 2 次元ベクトルとし，各要素は $[0.1, 0.9]$ の範囲でランダムに発生させて学習セットとした．学習パターンの値は，両要素とも 0.1 以上であるので，忘却により削除されたユニットの重みは 0.0 とし，削除されたユニットはデッドノードとして隣接関係の再構築を行った．

実験に用いた一様分布の 150 個の学習パターンを持つ学習セットの例を図 6.2に示す．また，この学習セットを 10×10 の競合層を持つ SOM に入力し，15,000 回の学習を行った

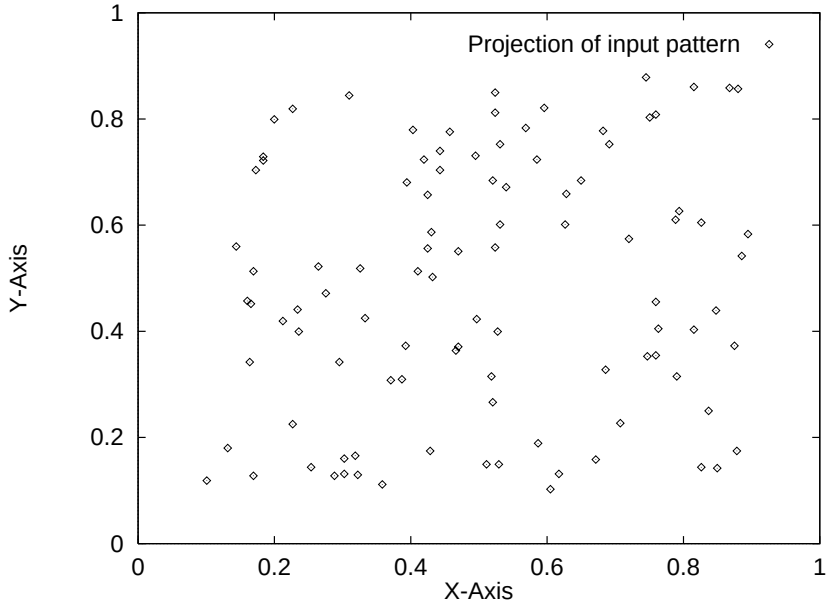

図 6.2: 一様分布入力の例

ときの各ユニットが持つ重みと学習パターンを図 6.3に示す．このとき，初期学習率は 0.4，近傍の初期半径は 6，近傍直径 Dは（6.4）式， 学習率αは（6.5）式によりウィナーからの距離 d と近傍直径 Dに応じて減少するとした．

$$D = 6 \times \left(1.0 - \frac{t}{T}\right) \tag{6.4}$$

$$\alpha = 0.4 \times (1.0 - \frac{t}{T}) \exp\left(-\frac{d^2}{\lceil D/2 \rceil^2}\right) \tag{6.5}$$

（6.4）式,（6.5）式で，t は現在の学習回数，Tは最大学習回数である．図 6.3において，緑の点はあるユニットが持つ重みの位置を示し，各点を結ぶ線はその重みを持つユニット間の隣接関係を表したものである．橙の点は学習パターンを示す．

図 6.3より，類似した入力に反応するユニットが近傍にあり，乱数入力による格子状のトポグラフィックマッピングが完成している．しかしながら，学習パターン位置と重み位置を比較すると，学習パターンの存在しない部分にもユニットが存在している．これらのユニットは学習パターンではなく，近傍ユニットの隣接関係により配置されており，本来存在しない情報を表現している．また，図 6.4に各ユニットの重みが最後に更新された Epoch 数，図 6.5 に近傍の直径を示す．

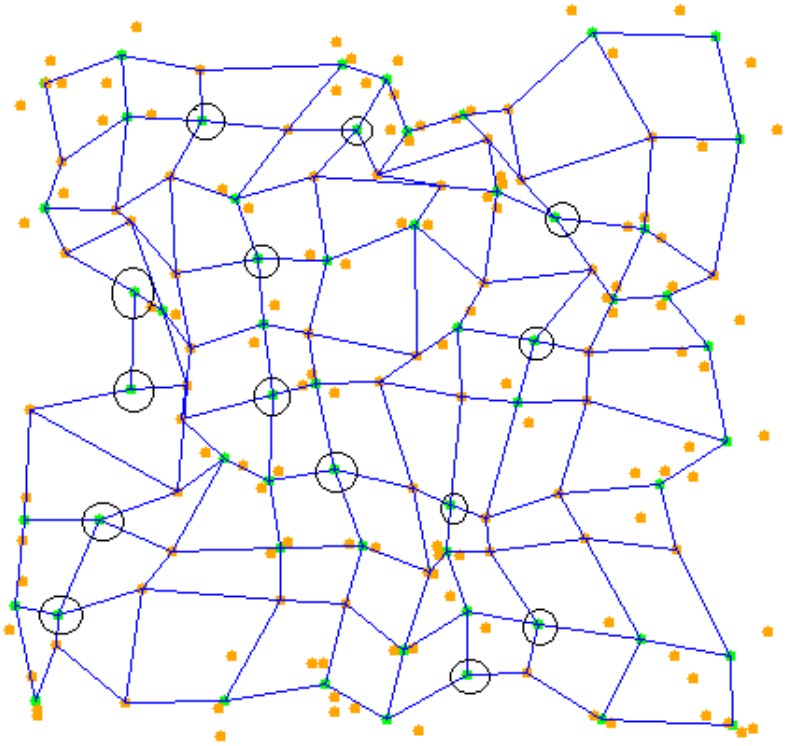

図 6.3: 一様分布入力を学習したときの SOM の重みとユニット間隣接関係

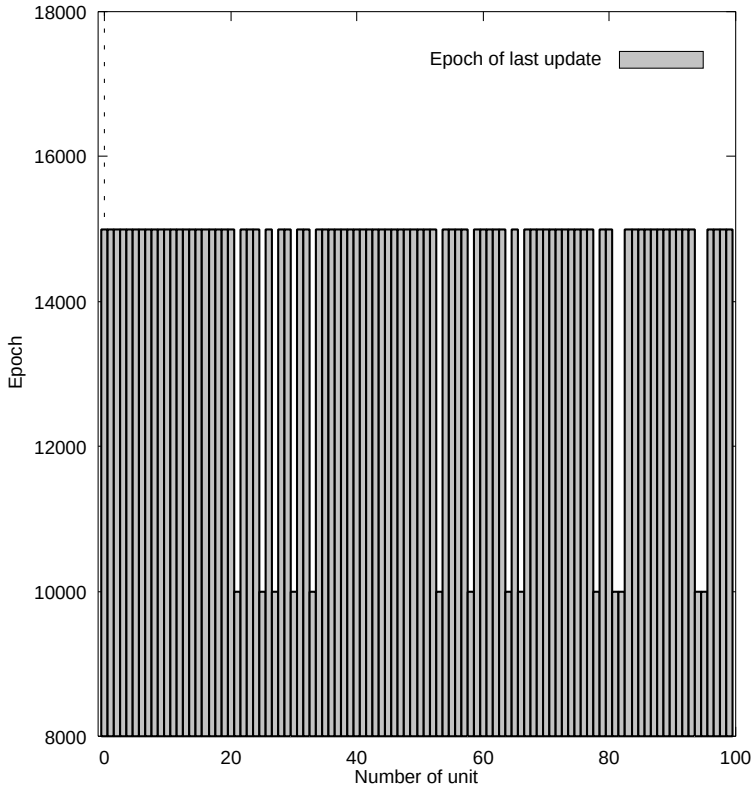

図 6.4: 一様分布入力学習時の最終重み更新 Epoch 数

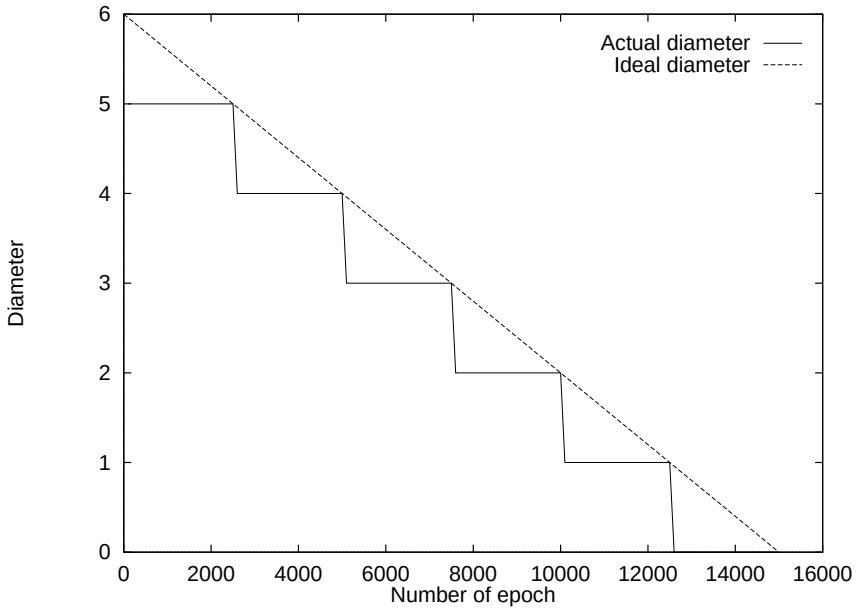

図 6.5: 10×10 の競合層 SOM の近傍直径

図 6.4より，図 6.3で丸く囲った 14 個のユニットが約 10000 回の学習以降重みの更新を行っていないことが分かる．10000 回以降の近傍直径は図 6.5より 1 であり，これら 14 個のユニットは約 10,000 回の学習以降学習パターンに反応することがないデッドノードとなる．

そこで，忘却学習により不要となったユニットをデッドノードとしてトポグラフィックマップから削除する．忘却を行わない場合と同一の条件で忘却学習を行ったときの競合層ユニットの重みとその隣接関係を図 6.6に示す．図中，赤い点がデッドノードとなったユニットの重みを示しており，これらは近傍を表す線では結んでいない．

図 6.3と図 6.6を比較すると，入力パターンが存在しない領域のユニットは忘却により消滅し，重み空間の原点に置かれていることが分かる．このとき削除されたユニットは 10,000 回目の学習以降ウィナーとならなかった 14 個のユニットである．この結果から，学習に関与しないデッドノードの削除が忘却により可能であることが分かる．

生成されたトポグラフィックマッピングに対しては,（6.6）式で表される tpg 値 [31] により評価を行う．

$$\mathrm{tpg} = \frac{1}{N} \sum_{i=1}^{N-1} \left\{ \frac{1}{M_{i,j}} \sum_{j=0}^{M_{i,j-1}} d(w_i, w_{i,j}) \right\} \tag{6.6}$$

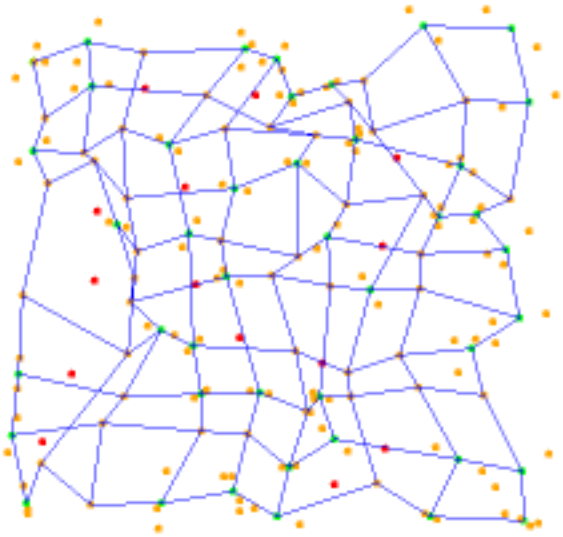

図 6.6: 一様分布入力の忘却学習を行った時の SOM の重みとユニット間隣接関係

表 6.1: 一様分布入力学習時の生存ユニット数と tpg 値，及び平均最小距離

	生存ユニット数	tpg 値	平均最小距離
忘却なし	100	0.007621	0.015159
忘却あり	86	0.008171	0.009718

（6.6）式において，Nは生存しているユニット数，$M_{i,j}$はユニット i の近傍（上下左右）に生存しているユニット数，$d(w_i, w_{i,j})$ はユニット i とその近傍ユニット jの重みのユークリッド距離である．したがって，tpg 値はあるユニットが近傍と平均してどの程度離れているかを表しており，値が小さいほど品質のよいマッピングが行われていると言える．tpg 値を評価に用いる妥当性は 6.3.3節で考察する．

表 6.1に忘却を行ったときと行わないときの tpg 値を示す．同時に，各ユニットが持つ重みと最も近い学習パターンとの平均距離を示す．

表 6.1より，14 個のユニットが忘却により消滅し，生存ユニットが持つ重み間の平均距離は Kohonen の SOM とほとんど同等であることが分かる．また，各ユニットが最も反応

表 6.2: T 字分布入力学習時の生存ユニット数と tpg 値，及び平均最小距離

	生存ユニット数	tpg 値	平均最小距離
忘却なし	100	0.003263	0.009108
忘却あり	80	0.003172	0.005088

した学習パターンとの平均距離においても，忘却により不要なユニットが削除されたためにより小さい値となり，学習パターンに即したトポグラフィックマップの生成が行われていることが分かる．

6.3.2 構造分布入力の学習

本節では，よりデッドノードが発生しやすい構造分布入力に対して忘却学習を行い，その効果を評価する．学習実験を行った入力分布は Kohonen が [16] で用いたものに独自のものを加えて，図 6.7で示す 4 種類とした．それぞれ学習パターン数は 150 とし，各ブロック内の学習パターン数は同じであるとした．学習対象とする SOM は，入力層 2×1，競合層 10×10 として，全学習パターンに対するウィナーの誤差が 0.03 以下となるまで，15,000Epoch を最大学習回数として学習を行った．各ブロック内において学習パターンはランダムに置かれているため，ブロック内では格子型のトポグラフィックマッピングが行われることになる．

(a)T 字型分布入力

図 6.7（a）で示した T 字型分布入力を用いて SOM の学習を行ったときの各ユニットの重みと隣接関係，および学習パターンを図 6.8に示す．

図 6.8より，T 字型分布の偏りのある学習セットでもおおよそのマッピングは完成しているが，一様分布入力の学習時と同じく学習パターンが存在しない領域に配置されるユニットが一部存在する．

図 6.9に，忘却学習を行ったときの重みとユニットの隣接関係を，表 6.2に忘却を行った場合と行わない場合の tpg 値，および各ユニットの重みと学習パターンとの平均最小距離を示す．

表 6.2より，20 個のユニットが忘却により削除された結果，生存している各ユニットの

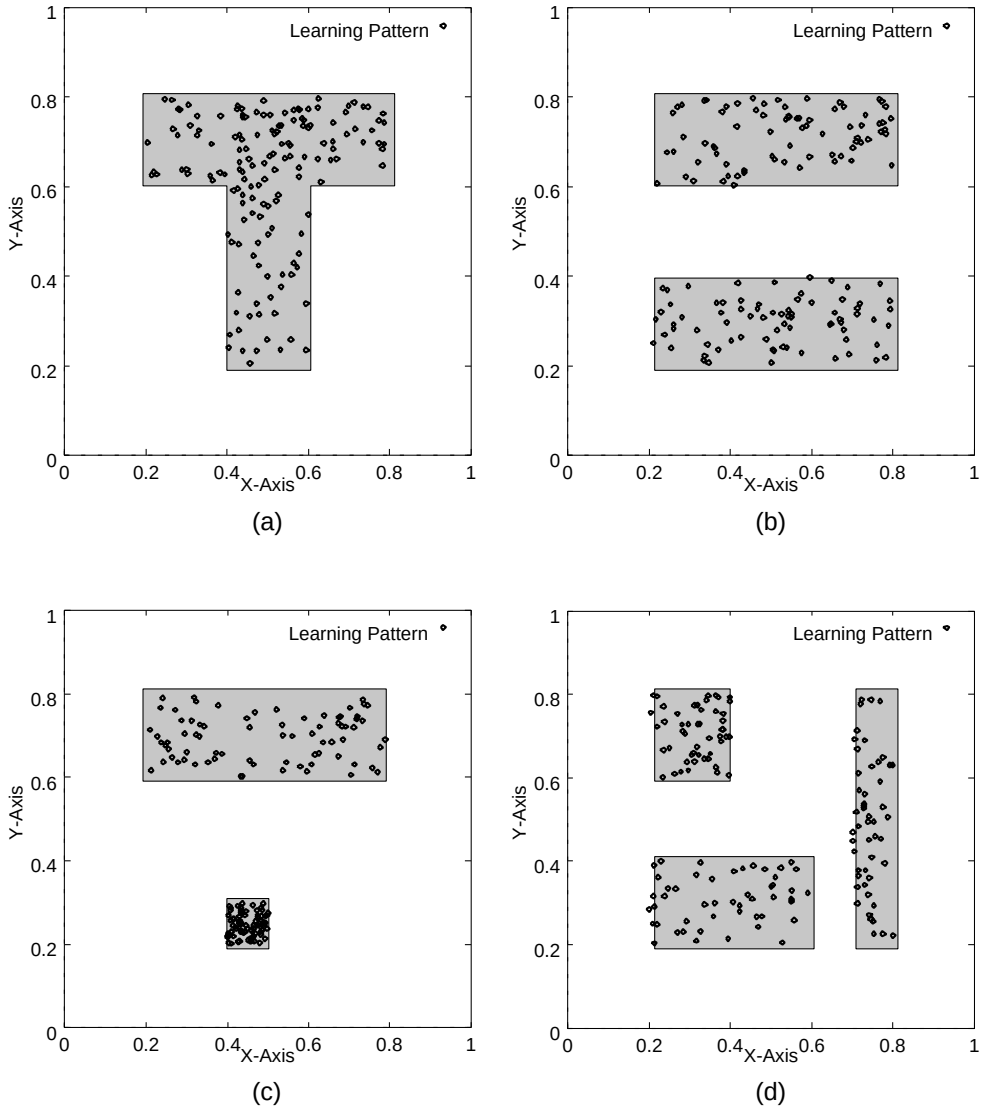

図 6.7: 忘却学習の対象とした構造分布入力

図 6.8: T 字型分布入力を学習した SOM の競合層ユニットの重みと隣接関係

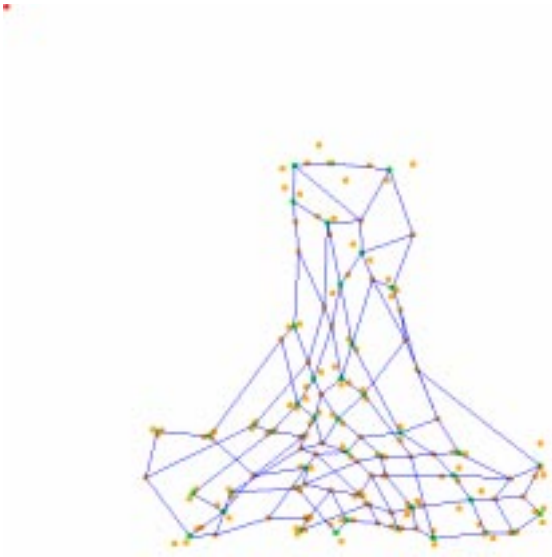

図 6.9: T 字型分布入力を忘却学習したときのユニットの重みと隣接関係

重みはより学習パターンに近いものとなっている．また，tpg 値も忘却を行わない場合とほとんど同等であり，トポグラフィックマッピングの質を落とすことなく，より学習パターンに則したマッピングが行われている．

(b) 二並行分布入力

図 6.7（b）で示した二並行分布では，学習パターンが存在する 2 つの帯の間に不要なユニットが発生し，他のユニットに影響を与えることが考えられる．Kohonen のアルゴリズムにより作成したトポグラフィックマッピングを図 6.10に示す．このとき，10969 回の学習で近似精度 0.03 以下となった．

図 6.10より，学習パターンが存在しない領域に複数のユニットが存在している．これらのユニットは学習パターンが存在する領域のエッジ部分に存在するユニットに影響を与えるため好ましくない．そこで忘却によりこうしたユニットの削除を行う．図 6.11に忘却学習を行ったときの競合層ユニットの重みと隣接関係を示す．

図 6.11より，学習パターンが存在しない重み空間にはユニットが存在しておらず，忘却によりマップから削除されていることが分かる．二並行分布入力で学習したときの tpg 値と，各ユニットが持つ重みと最も近い学習パターンとの平均距離を示す．

表 6.3より，19 個のユニットが忘却により削除された一方で，tpg，平均最小距離とも小

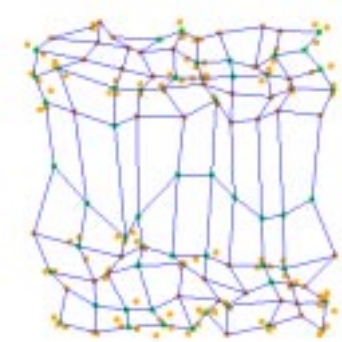

図 6.10: 二並行分布入力学習時のユニットの重みと隣接関係

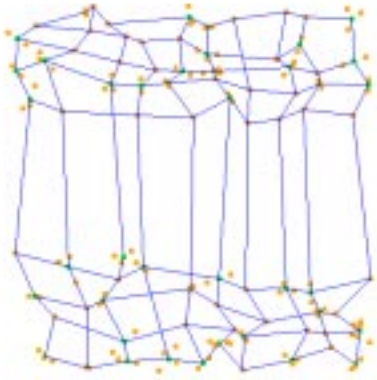

図 6.11: 二並行分布入力を忘却学習した時のユニットの重みと隣接関係

表 6.3: 二並行分布入力学習時の生存ユニット数と tpg，及び平均最小距離

	生存ユニット数	tpg 値	平均最小距離
忘却なし	100	0.005230	0.015307
忘却あり	81	0.003435	0.004639

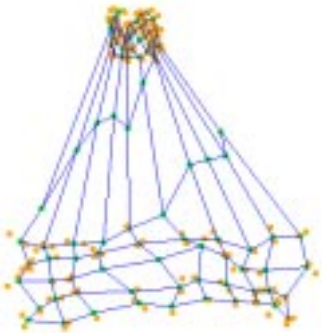

図 6.12: 長短二並行分布入力を学習した SOM の競合層ユニットの重みと隣接関係

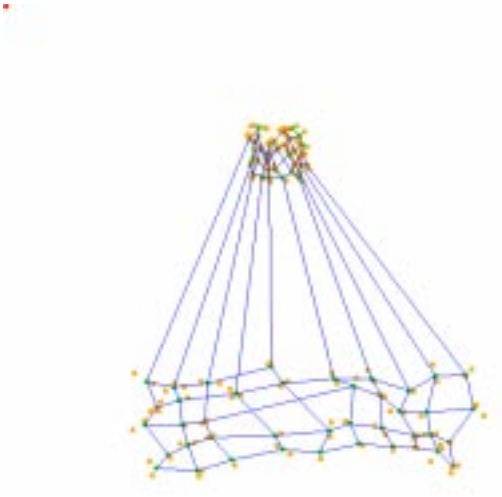

図 6.13: 長短二並行分布入力を忘却学習したときのユニットの重みと隣接関係

さくなり，品質のよいトポグラフィックマッピングが行われていることが分かる．また，忘却を行ったときの行わないときも学習回数は同じ 10969 回であった．これは，すべての学習パターンに対してウィナーが持つ誤差を停止基準に用いているためである．

(c) 長短二並行分布

図 6.7（b）の二並行分布入力で，分布の一方が狭い領域に高密度に分布している場合である．Kohonen の学習アルゴリズムにより学習を行ったところ，10006 回の学習で近似精度 0.03 以下となった．このとき得られたトポグラフィックマップを図 6.12に示す．

図 6.12より，図 6.10と同じく学習パターンが存在しない空間にいくつかのユニットが配置されていることが分かる．競合層サイズが 10×10 である SOM の近傍直径は図 6.5より 10001 回目の学習以降 1 となる（ウィナー以外は重みの更新が行われない）ため，近傍が 8 近傍のときまでにウィナーとなるユニットはほとんどが正しくマッピングされていると考えられる．

図 6.13に忘却学習を行ったときの競合層ユニットの重みと隣接関係を，表 6.4に tpg 値，および各ユニットの重みと学習パターンとの平均最小距離を示す．

表 6.4より，22 個のユニットが忘却により削除されていることが分かる．tpg 値は忘却を行わない場合に比べて約 70%小さくなり，生き残っているユニットはより近傍ユニットの

表 6.4: 長短二並行分布入力学習時の生存ユニット数と tpg 値，及び平均最小距離

	生存ユニット数	tpg 値	平均最小距離
忘却なし	100	0.006029	0.018894
忘却あり	78	0.001578	0.005603

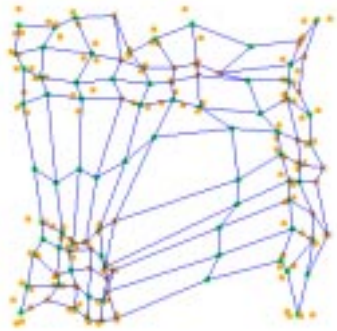

図 6.14: 二並行分布+垂直分布入力を学習した SOM の競合層ユニットの重みと隣接関係

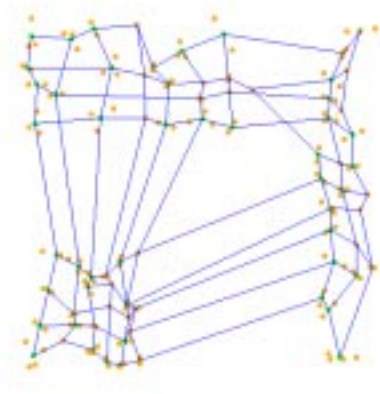

図 6.15: 二並行分布+垂直分布入力を忘却学習したときのユニットの重みと隣接関係

近くにあることが分かる．また，各学習パターンとウィナーとの平均距離も忘却を行わないときの約 30%となり，より厳密に学習パターンを反映したマッピングが行われている．

(d) 二並行+垂直分布

図 6.7（d）は，長短並行分布入力に加え，垂直方向への分布を加えたものである．Kohonen の学習アルゴリズムにより学習を行ったところ，15,000 回の最大回数の学習が行われた．このとき得られたトポグラフィックマップを図 6.14に示す．

図 6.14より，3 つの入力ブロックの中間にユニットが配置され，学習パターンを反映していないことが分かる．（c）の長短二並行分布入力と異り，15,000 回の学習を行っても基準近似精度に達していないことから，ウィナーのみの重み更新（近傍直径が 1 以下）段階と

表 6.5: 二並行+垂直分布入力学習時の生存ユニット数と tpg 値，及び平均最小距離

	生存ユニット数	tpg 値	平均最小距離
忘却なし	100	0.005776	0.021533
忘却あり	73	0.002915	0.006943

なっても一つのユニットが複数の学習パターンに反応していると思われる．

図 6.7（d）の学習セットに対し，忘却学習を行ったときの競合層ユニットの重みと隣接関係を図 6.15に示す．

図 6.15から，入力ブロック間にあったユニットは忘却によりすべて消滅している．表 6.5に二並行+垂直分布入力を学習したときの tpg 値と，競合層の各ユニットの重みと最小平均距離を示す．

表 6.5より，忘却により 27 個の不要なユニットが忘却により削除されていることが分かる．また tpg 値も忘却を行わない場合に比べて約 50%減となり，より均質なマッピングが行なわれていることが分かる．さらに，入力ブロック間に存在するユニットが削除されたため，ウィナーと学習パターンとの距離もより小さくなっている．

6.3.3 考察

トポグラフィックマッピングの品質

忘却学習により不必要なユニットを削除しつつトポグラフィックマッピングを行う忘却学習法を，一様分布入力と構造分布入力を用いて実験を行った．いずれの場合もトポグラフィックマッピングを完成することができたが，その品質を tpg 値で評価することの妥当性について考察する．

tpg 値は（6.6）式で表されるように各競合層ユニットの 4 近傍の平均距離である．tpg 値は小さい値であるほど滑らかであると述べたが，tpg 値が最小となる場合について考える．図 6.16に示すネットワークを仮定し，tpg 値を計算する．

図 6.16の各ユニットに番号を付け，それらを $u_1 \cdots u_9$とする．中央のユニット u_5が図 6.16（b）のようにδだけずれた時の tpg 値を tpg' とする．図 6.16より明らかに，各ユニットが

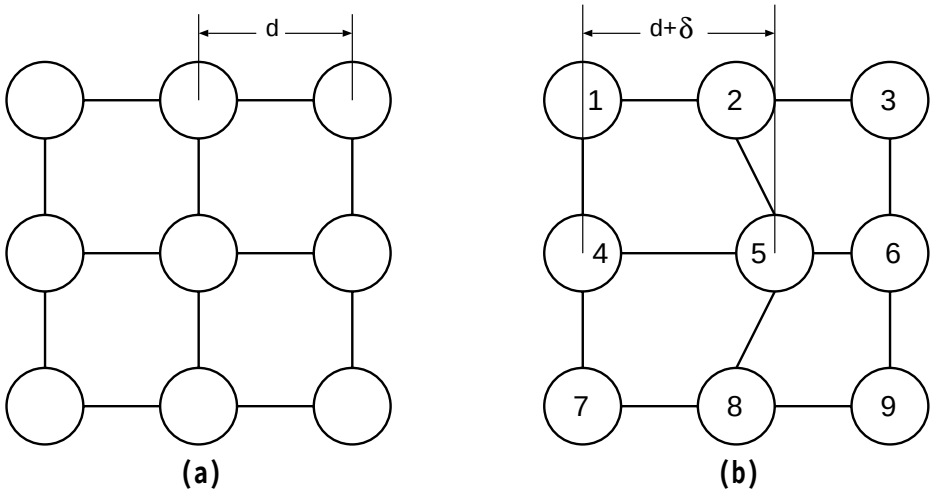

図 6.16: tpg 最小となるネットワーク

持つ近傍ユニットとの平均距離 $d(u)$ は以下の式で表される．

$$d(u_1) = d(u3) = d(u_7) = d(u_9) = d \tag{6.7}$$

$$d(u_2) = d(u_8) = \frac{2d + \sqrt{d^2 + \delta^2}}{3} \tag{6.8}$$

$$d(u_4) = \frac{3d + \delta}{3} \tag{6.9}$$

$$d(u_6) = \frac{3d - \delta}{3} \tag{6.10}$$

$$d(u_5) = \frac{d + \sqrt{d^2 + \delta^2}}{2} \tag{6.11}$$

tpg 値は（6.7) 式～（6.11）式を加えてユニット数で割ったものなので，

$$\text{tpg'} = \frac{1}{9}\left(6d + \frac{4d + 2\sqrt{d^2 + \delta^2}}{3} + \frac{d + \sqrt{d^2 + \delta^2}}{2}\right) \tag{6.12}$$

となる．$d = 1$ として（6.12）式を図示すると，図 6.17となる．tpg' を最小にするのは明らかに$\delta = 0$ のときであり，これは図 6.16（a）で示す正方格子上にユニットが配置されているときである．

以上より，tpg 値は生成されたトポグラフィックマッピング上のユニットがどれくらい均等に配置されているかを示していることが分かる．実験で用いた構造分布入力は，各ブロック内ではランダムに学習パターンが分布するため SOM の特性から格子状にユニットが配

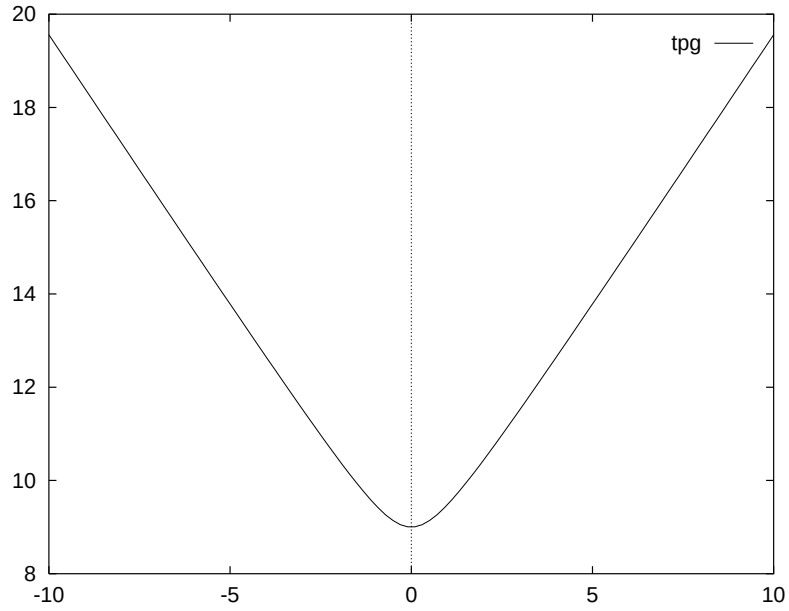

図 6.17: tpg 値の計算値

置される．したがって，tpg 値は SOM のトポグラフィックマッピングを評価する一つの尺度となりうる．

忘却開始の時期

忘却学習法では，学習セットを一通り学習する間にウィナーあるいは近傍とならなかったユニットを忘却により削除する．したがって，近傍直径が大きい学習初期ではユニットが削除されず，大域的なトポグラフィックマッピングが生成されることになる．構造分布入力の学習シミュレーションでは初期近傍直径が軸上ユニット数の 60%としているため，学習初期では全てのユニットがウィナー，あるいは近傍として重み更新の対象となりえ，忘却は行われない．実験で用いた 4 つの学習セットにおいて，各ユニットの最終重み更新 epoch を図 6.18に示す．

図 6.18より，全ての学習セットにおいて最終重み更新回は約 10,000epoch が最小である．(6.4）式より，10,000Epoch 以降は近傍直径が 1 以下（ウィナー以外は重み更新を行わない）である．つまり，学習初期で近傍が大きいときに学習パターンに即したトポグラフィックマップが生成されてしまうと，学習パターンが存在しない領域にユニットが存在するにもかかわらず削除されない場合が考えられる．こうした場合の例として，局所分布入力を

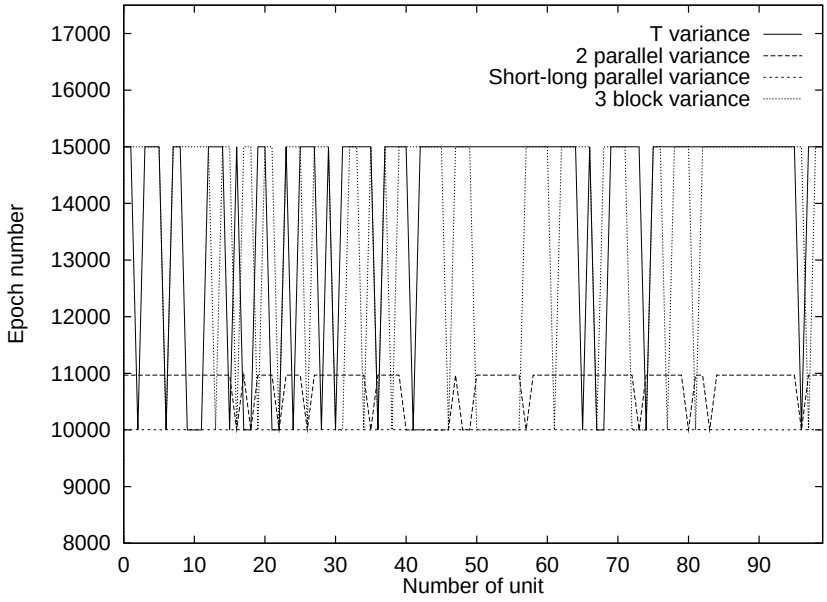

図 6.18: 各学習セットでの最終重み更新 epoch

図 6.19を示す.

図 6.19で示した学習セットでは，極めて小さい領域に学習パターンが集中している．したがって，ごく一部のユニットのみで学習が進行して誤差が小さくなっていく．学習はウィナーとなったユニットの学習パターンとの誤差が一定値（ここでは 0.03）以下で打ち切られるため，近傍直径が大きく忘却が行われないうちに学習が終了する．図 6.19の学習セットを用いて忘却学習を行ったときと忘却を行わなかったときのトポグラフィックマッピングを図 6.21と図 6.20にそれぞれ示す.

図 6.20，図 6.21の学習双方とも，各ユニットの最終重み更新 epoch 数は学習の終了 epoch 数に等しく 5003 回であった．すなわち，局所分布入力を学習したときは近傍半径が大きいうちにウィナーの誤差が小さくなり，忘却が行われない．このため，図 6.21から分かるように学習パターンが存在しない領域にユニットがあり，忘却の効果が現れない.

6.4 忘却学習法による故障回避法

SOM は人間の脳で見られるようなトポグラフィックマッピングを簡単な学習アルゴリズムから生成できるため，視覚野などの機能を解明する手がかりとして用いられる場合があ

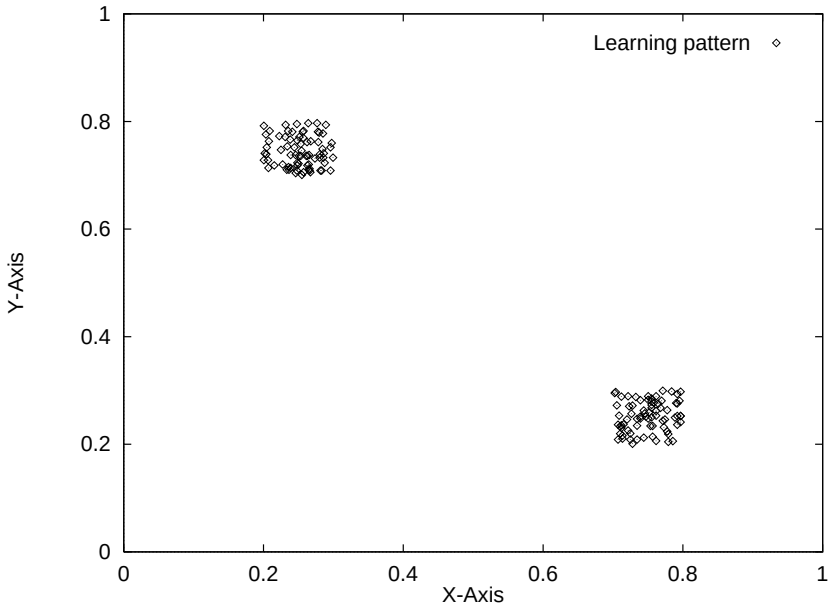

図 6.19: 忘却が行われない局所分布入力の例

図 6.20: 局所分布入力を学習した SOM の競合層ユニットの重みと隣接関係

図 6.21: 局所分布入力の忘却学習を行った時の競合層ユニットの重みと隣接関係

る．しかし，脳の機能を解明するには一層の大規模化，高速化が不可欠である．SOM のより一層の大規模化，高速化を実現する手法の一つとしてハードウェア実装が考えられる．しかしながら階層型ニューラルネットワークの耐故障性については 4章で述べたように研究が行われているが，SOM のハードウェア実装時に必要となる故障回避法については従来ほとんど検討されていない．そこで本節では，忘却学習法による故障回避法を提案し，その性能について議論する．

6.4.1 SOM における故障の定義

Kohonen の SOM において，忘却学習法により不要なユニットを削除しトポグラフィックマッピングが行えることを 6.3節で示した．忘却学習では全ての学習パターンに対して一度もウィナー，あるいは近傍として重み更新の対象にならないユニットを忘却対象とし，競合層から削除を行う．

本節では競合層ユニットに故障が発生している場合を想定し，故障ユニットを忘却により消滅したユニットとして忘却学習を行うことにより故障を回避してトポグラフィックマッピングの生成を試みる．このとき問題となるのは，故障ユニットが重み更新対象範囲に存在する場合である．通常の忘却学習法での忘却対象ユニットは重み空間で学習パターンと十分離れた位置にあるため，重み更新の対象になることはない．しかし，位置を固定した故障ユニットを忘却されたユニットとする場合，重み更新の対象範囲内に故障ユニットが存在する場合が考えられる．そこで，故障ユニットを含む忘却学習法では重み更新範囲に故障ユニットがあっても単に忘却ユニットとして扱う場合と，図 6.22に示すような 4 近傍にもとづく隣接関係の再構築を行って重み更新範囲を広げた場合の 2 つについて比較検討する．

さて，本論文では SOM の並列学習法として入力層分割法を 5章で提案し，PE 間の負荷を均一に保ったままで大規模な競合層を持つ SOM の学習を高速に行えることを示した．しかし，入力層分割法をもとに故障補償を行うことは適当でない．なぜなら，入力ユニットが PE に割り当てられており，PE の故障により入力情報の次元落ちが発生したときには学習パターンの正確な位相が保存できない場合が考えられるためである．そこで，故障補償を行う場合の並列学習法には競合層分割法を想定し，競合層のユニットを PE に割り当てる．PE は 2 次元平面上に格子結合しているとする．忘却学習による故障補償を行うにあたり，以下の条件を仮定する．

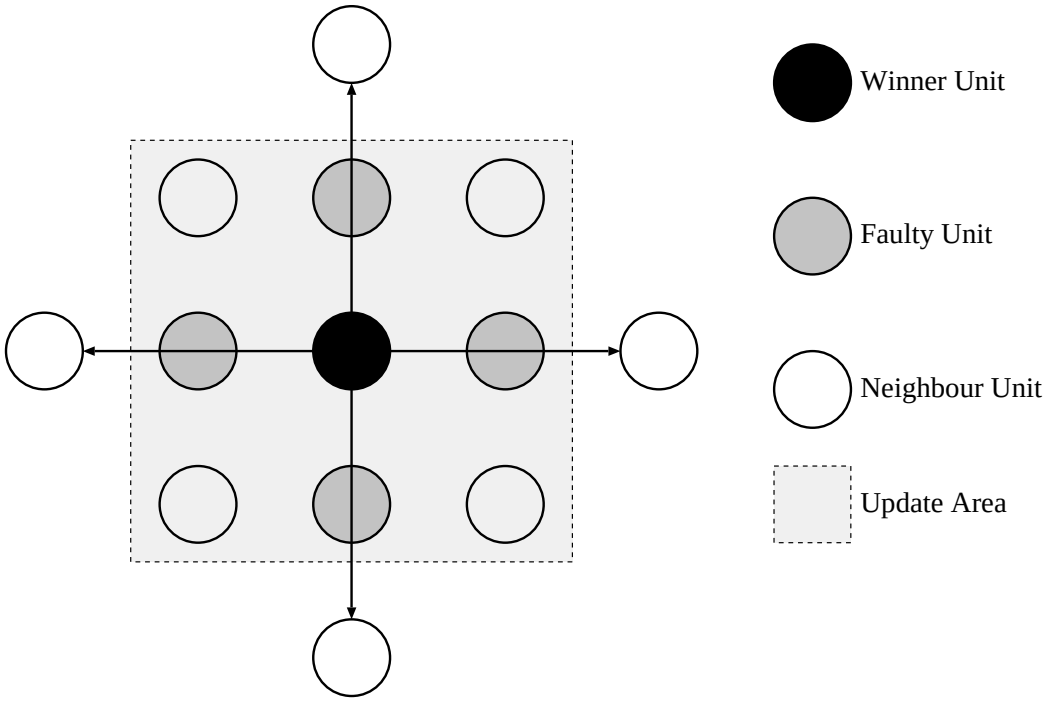

図 6.22: 故障ユニットがある場合の隣接関係の再構築パターン

- 競合層分割法を用いた並列 SOM を対象とする．
- 故障したユニットはすべての学習パターンに対してウィナーとならない．
- 通信路は故障しない．

第 2 の条件は故障ユニットの定義である．忘却学習法ではすべての学習パターンに対して重みの更新対象とならないユニットをデッドノードとして，以降の学習対象としない．したがって，第 2 の条件より故障ユニットは第 1Epoch の学習が終了後にデッドノードと判定され，マップから削除されると同時に，これをスキップするように隣接関係の再構築が行われる．

第 3 の条件は，競合層分割法で SOM を並列化した場合，重みの集計やウィナーの決定時に PE 間通信が行われる．通信路が故障した場合にはこれらの作業に支障が発生して学習を行うことができないため，通信路は故障しないとした．

6.4.2 忘却学習法による故障補償実験

生成されたトポグラフィックマッピングを評価するため，(6.6) 式で定義した tpg 値の他に，(6.13) 式の amd 値，及び (6.14) 式の adw 値を定義して用いる．

$$\mathrm{amd} = \frac{1}{N}\sum_{i=1}^{N} \min\{d(T_i, W_i)\} \tag{6.13}$$

$$\mathrm{adw} = \frac{1}{L}\sum_{j=1}^{L} d(T_j, W_c) \tag{6.14}$$

(6.13) 式において，Nは競合層ユニット数，W_iはユニット i の重み，T_iはユニット i をウィナーとする学習パターン群, $d(\boldsymbol{a}, \boldsymbol{b})$ は$\boldsymbol{a}, \boldsymbol{b}$間のユークリッド距離を表す．また，(6.14) 式において L は学習パターン数，W_cは学習パターン T_jに対するウィナーが持つ重みである．(6.13) 式では，各競合層ユニットが最善で学習パターンをどの程度近似しているかを示し，(6.14) 式では各学習パターンがどの程度の精度で近似されているかを示す．しがたって，amd, adw 値双方とも値の小さい方が精度良くマッピングされていることになる．

一様分布入力による故障補償実験

6.3.1節と同様の条件で，競合層ユニットのうちランダムに 5，10，15 個が故障したとして故障補償実験を行った．なお，以下に示す tpg, amd, adw 値は 3 種類の故障パターンにお

表 6.6: 故障を含む SOM における一様分布入力の学習結果

故障ユニット	忘却	再構築	Alive	Dead	TPG	AMD	ADW
0	故障なし		100	0	1.0	1.0	1.0
5	有	無	85	10	1.09	0.61	0.94
	有	有	85	10	1.09	0.61	0.97
	無	無	95	0	1.06	0.88	0.94
	無	有	95	0	1.08	0.91	0.97
10	有	無	81	9	1.27	0.63	1.07
	有	有	78	12	1.38	0.68	1.13
	無	無	90	0	1.21	0.86	1.07
	無	有	90	0	1.44	1.00	1.13
15	有	無	74	11	1.17	0.78	1.19
	有	有	74	11	1.25	0.81	1.27
	無	無	85	0	1.14	1.03	1.17
	無	有	85	0	1.26	1.13	1.15

ける平均であり，故障の有無による変化を分かり易くするために故障が無い場合の値で正規化して示す.

表 6.6に故障ユニットを含んだ SOM 上における忘却学習結果と，通常の Kohonen 学習による結果を示す．なお，双方とも図 6.22で示した隣接関係再構築を行った場合と行わない場合について実験を行った.

表 6.6より，5 個の故障ユニットを含む学習では忘却学習を用いた場合は 85 個のユニットにより学習パターンをマップしており，10 個のユニットがデッドノードとして消滅している．また，tpg 値は故障がない場合よりもやや大きくなり，amd 値，adw 値は減少している.

tpg 値は故障および消滅ユニットのために，広い空間を少ないユニット数でカバーしなければならないためユニットが持つ重み間の距離が開くため，忘却学習の場合の方がやや大きく，故障がない場合に比べてユニット間の距離がやや大きくなるという結果となった.

amd 値については，忘却により不要なユニットが削除されて計算から除かれているため忘却学習の方が小さい値となり，各ユニットがより学習パターンに近い重みを持っていることが分かる．adw 値では隣接関係の再構築を行うか行わないかにより異なり，重みの再構築を行わない方がやや小さい値となった．また，故障がない場合に比べて amd 値，adw 値はともに小さくなり，故障を含んだ SOM でも，精度を落とすことなくトポグラフィックマッピングの生成が行なえる可能性を示している．

しかし，10 個以上のユニットが故障した場合では，adw 値が全て 1.0 を越え，故障がない場合より悪化した．故障ユニット数の増加は当然正常なユニット数の減少を伴うため，一つのユニットを発火させる学習パターン数が増える，すなわち一つのユニットの担当範囲が広がるために，個々の学習パターンとの距離が大きくなるためと考えられる．実際，tpg 値も同様に 1 を大きく越えており，重み空間におけるユニット間距離が大きくなっていることが分かる．しかしながら，amd 値は他の値ほど悪化していないことから，各ユニットは最も強く発火させる学習パターンから大きくずれてはいない．以上のことから，故障を持つ SOM を忘却学習させたとき，ウィナーとなるユニットは自身の重みに最も近い学習パターンから大きく離れることはないが，ユニット数の減少に伴なって担当する学習パターン数が増加する分，近似精度が悪化する．しかしながら，少数のユニットの故障では近似精度が向上する場合もみられる．

二並行分布入力による故障補償実験

故障ユニットを持つ SOM の構造的入力学習実験として，6.3.2節で用いた二並行分布入力を用いて実験を行った．6.7に故障を 5，10，15 個含む SOM に二並行分布入力を学習させたときの tpg 値，mad 値，adw 値を示す．

故障ユニットを持つ SOM を二並行分布入力により忘却学習させたところ，表 6.7から分かるように tpg 値，amd 値では忘却を行ったときに故障前より小さい結果となった．つまり，各ユニットが持つ重みは学習パターンが存在する空間により平均的に分散し，かつ学習パターンにより接近した位置に置かれているといえる．しかし，忘却を行わない場合は不要なユニットからの影響が残り，これらの値は故障がない場合よりも大きい値となっている．adw 値に関しては，故障前の値よりも大きくなっている．この値は忘却の有無よりは隣接関係再構築の有無により影響を受けており，再構築を行った場合の方がより大きい値を示している．これは再構築を行うことでウィナーから遠距離にあるユニットに影響が

表 6.7: 故障を含む SOM における二並行分布入力の学習結果

故障ユニット	忘却	再構築	Alive	Dead	TPG	AMD	ADW
0	故障なし		100	0	1.0	1.0	1.0
5	有	無	76	18	0.71	0.34	1.09
	有	有	74	21	0.71	0.38	1.17
	無	無	95	0	1.18	1.02	1.07
	無	有	95	0	1.50	1.14	1.13
10	有	無	72	18	0.81	0.38	1.28
	有	有	73	10	0.77	0.36	1.18
	無	無	90	0	1.50	1.07	1.27
	無	有	90	0	1.32	1.07	1.17
15	有	無	70	15	0.74	0.43	1.26
	有	有	69	16	0.97	0.42	1.34
	無	無	85	0	1.14	1.12	1.26
	無	有	85	0	1.47	1.10	1.33

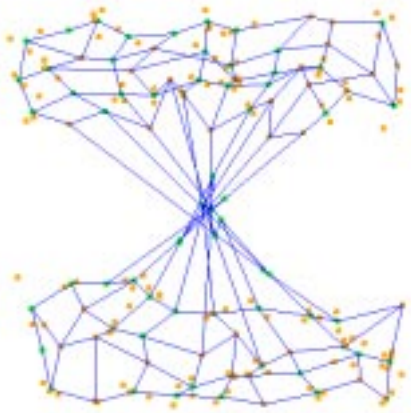

図 6.23: 延長を行わずにねじれたトポグラフィックマップの例

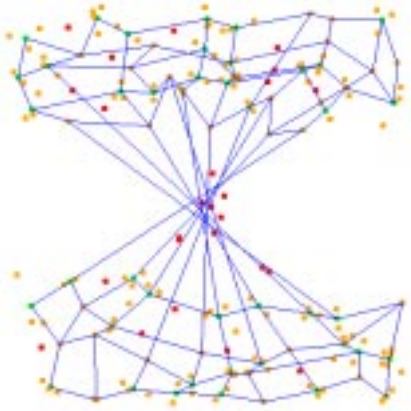

図 6.24: 過剰延長によりねじれたトポグラフィックマップの例

及ぶためと考えられる．これについては次節で議論する．

6.4.3 隣接関係再構築の影響

故障を含む SOM での二並行分布入力の学習で，隣接関係再構築を行わなかったり，逆に近傍の延長を行うことで各学習パターンの近似精度が悪化する場合が見られた．隣接関係の再構築を行わない場合，学習の間に重みの変化が十分にいきわたらない場合があり，トポグラフィックマップのねじれが発生する場合があった．また，隣接関係再構築により故障ユニットを越えて過剰に近傍範囲を延長すると，遠い位置にあるユニットの重みをも変更してしまう場合があるために，やはりねじれが発生する場合が見られた．隣接関係再構築を行わなかったときの，ねじれが発生したトポグラフィックマップの例を図 6.23に示す．また，過剰な延長によって生じたねじれたトポグラフィックマップの例を図 6.24に示す．

また，隣接関係再構築を行った場合，忘却により消滅するユニット数も増大する傾向が見られた．これは図 6.25で示すような場合に中間のユニットが両側のユニットから影響を受けるために重みが中途半端な値になり，近傍範囲縮小により忘却対象となる場合があるためと考えられる．

この問題を解決するためには，次の 2 つの手法が考えられる．

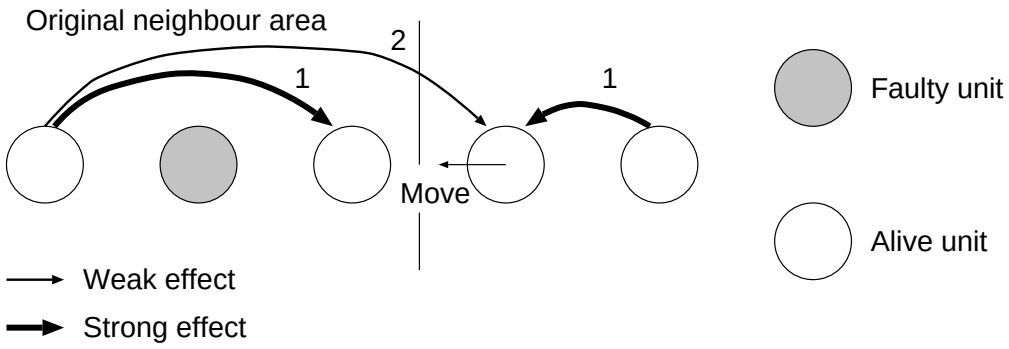

図 6.25: 遠距離ユニットから影響を受ける例

- ウィナーからの距離をスキップしたユニットを含めた距離にする．
- 故障ユニットを越えて影響を受けるユニットを制限する．

重みの更新に用いる学習率は（6.5）式によりウィナーからの距離が遠くなるほど小さくなるよう設定している．実験では故障ユニットをスキップして本来の近傍範囲を越えたユニットとウィナー間の距離は，図 6.25中で示すように故障ユニットを含めない距離として計算している．このため，近傍範囲を外れて重み更新を行うユニットでは，ウィナーからの実距離が大きいにもかかわらず学習率が大きく設定されてしまい，重みが大きく変動してしまう．これを防ぐためには，学習率の計算に用いる距離をウィナーからの実距離とする手法が考えられる．

今回行った実験では，近傍範囲内に故障ユニットが存在する場合，故障ユニットを除いた近傍半径が同数となるまで 4 近傍範囲を延長している．このため複数の故障ユニットが近傍範囲に存在する場合，図 6.26（a）で示したように本来の近傍範囲から大きく外れたユニットまで重み更新対象となる．そこで図 6.26（b）のように故障ユニットを越えて重み更新を行うユニットは 1 つに制限することにより，物理的に離れた位置に置かれているユニットの重みを変更することを避けることができる．

表 6.8に二並行分布入力を 5 個の故障ユニットを含む SOM に忘却学習させたとき，トポグラフィックマップにねじれが発生した場合（図 6.24）について，範囲を制限した隣接関係再構築を行った場合と行わない場合の学習結果を示す．また，制限付き隣接関係再構築を行ったときのトポグラフィックマップを図 6.27にそれぞれ示す．

図 6.24で見られたトポグラフィックマップのねじれが図 6.27では見られない．これは再

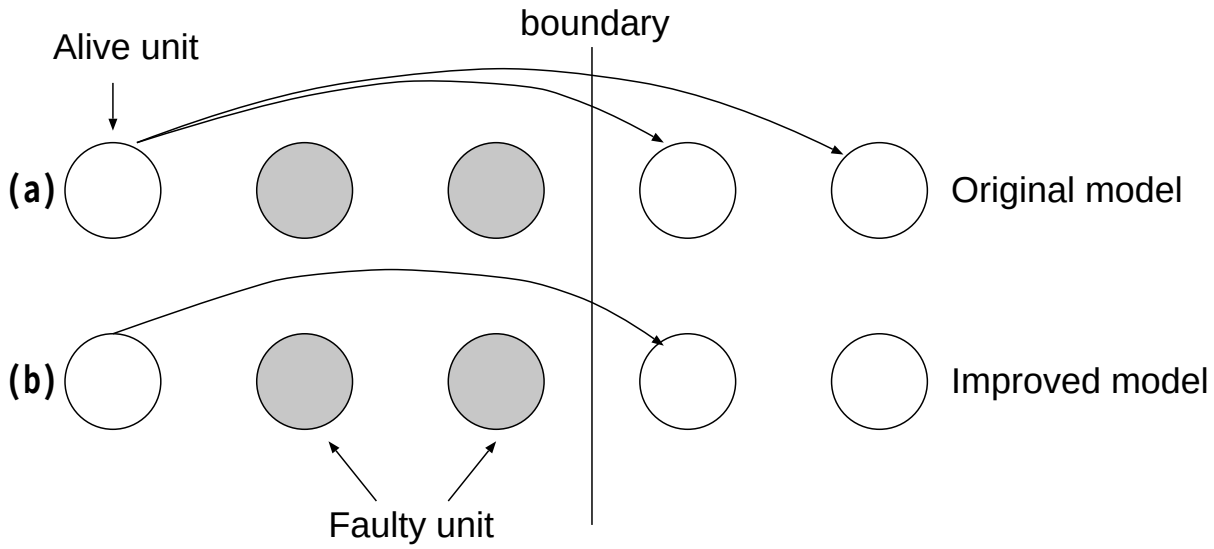

図 6.26: 連続する 2 つの故障ユニットのスキップ

表 6.8: 二並行分布入力を学習した 5 個の故障ユニットを含む SOM の隣接関係再構築法の相違による学習結果

再構築法	Alive	Dead	TPG	AMD	ADW
標準	70	25	0.0037	0.0062	0.011
制限	75	20	0.0039	0.0057	0.010

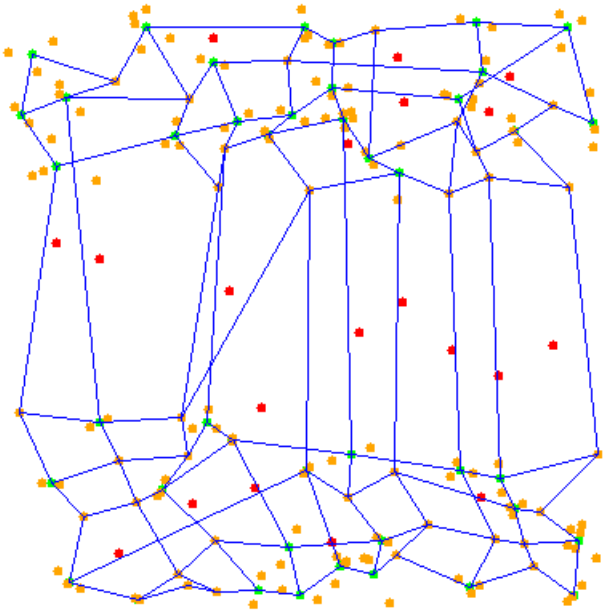

図 6.27: 5 個の故障ユニットを持つ SOM で二並行分布入力により制限付き隣接関係再構築を伴う忘却学習を行ったときのトポグラフィックマップ

構築を行う近傍範囲を限定することによりトポグラフィックマップを乱す要因を取り除いたためであると言える．

他のねじれが発生している場合についても制限付き隣接関係再構築を適用して忘却学習を行ったところ，今回実験を行った範囲内では，隣接関係再構築が原因でねじれが発生したと見られるパターンについて，再構築範囲に制限をつけることで全てねじれを回避することができた．

6.4.4 故障ユニットが集中した時の故障補償

6.4.2節において，ランダムに分布した故障ユニットを含む SOM の故障補償法について検討した．その結果，近傍拡張を制限した忘却学習法を用いることにより，トポグラフィックマップにねじれを発生させることなく，かつ近傍拡張を行わない場合よりも学習パターンの近似精度を良好に保つことが可能であることが分かった．しかし，実際の WSI 製造上での故障ユニットはランダムに分布するばかりでなく，一部に集中して発生する場合も考えられる．本節では，二並行分布入力を例に，部分的に故障が集中した場合の忘却学習による故障補償について考察する．10×10 の競合層を持つ SOM で，競合層にブロック状，またはライン状故障が競合層周辺部のユニットに発生したと仮定して故障補償実験を行った．

ライン状故障補償実験

図 6.28に，ライン状故障の例を示す．ここでは競合層の上端一行が故障した場合，上端から一つ内側のユニットがライン状に故障した場合について実験を行った．学習パターンには二並行分布入力を用い，15000 回を限度として学習を行っている．忘却学習により生成されたトポグラフィックマップを図 6.29に示す．このとき忘却により削除されたユニット数は 15 個，tpg 値は 0.004562，amd 値は 0.005901，adw 値は 0.010194 であり，近傍拡張の有無により変化することはなかった．これは，近傍拡張が故障ユニットをとばして正常なユニットまで近傍範囲を延長するものであるため，競合層端部にあるユニットに対してはそれ以上近傍を延長することができず，近傍拡張の影響を受けないためである．

これに対し，上端から一行内側のラインが故障したときのトポグラフィックマップを図 6.30と図 6.31に示す．図 6.30は近傍拡張を行わなかった場合，図 6.31は制限近傍拡張を行った場合である．また，このときの tpg, amd, adw の各評価値を図 6.32に示す．

図 6.30で発生している隣接関係の交差が，制限近傍拡張を行うことにより消滅している

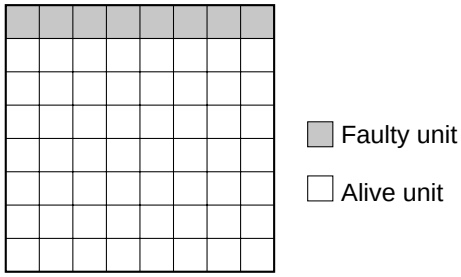

図 6.28: ライン状故障の例

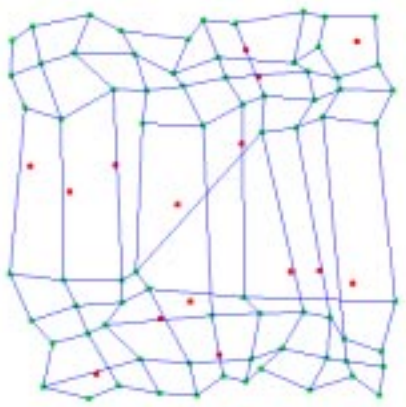

図 6.29: 最上端ユニットがライン状故障した SOM によるトポグラフィックマップ

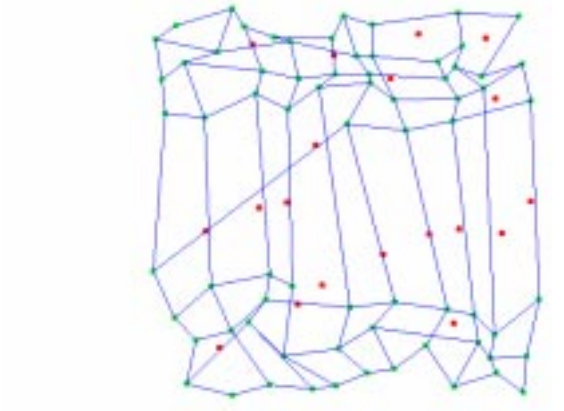

図 6.30: 第二行ユニットが故障し，近傍拡張を行わないときのトポグラフィックマップ

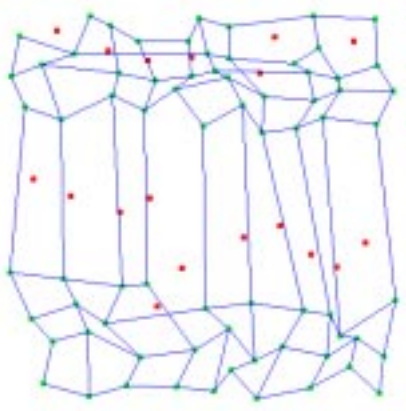

図 6.31: 第二行ユニットが故障し，制限近傍拡張を行ったときのトポグラフィックマップ

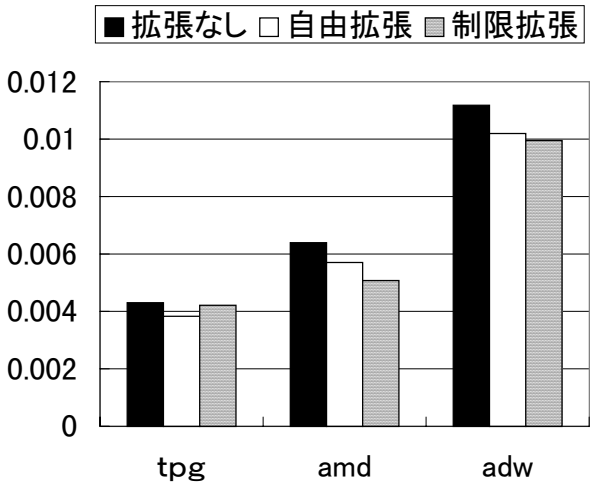

図 6.32: 第二行ユニットが故障したときの各評価値

ことが図 6.31より分かる．また，近傍拡張を行うことにより amd, adwの値が低下していることが図 6.32より分かる．最上端のユニットが故障したときには近傍延長の対象となるユニットが存在しないために各評価値は同一の値となったが，第二行の故障では最上端ユニットが延長対象のユニットとなりえるために，より精度の良いトポグラフィックマップが得られたと考えられる．

ブロック状故障補償実験

ライン状故障の場合と同じく，一部のユニットが集中して故障した場合としてブロック状故障を考える．ブロック状故障とは，$n \times n$ の範囲が同時に故障した場合である．ここでは $n = 3$ とし，二並行分布入力を用いて故障補償実験を行った．近傍拡張による精度改善は，故障部分が競合層の端部にあると期待できないことがライン状故障の補償実験により明らかとなったため，ブロック状故障での故障ユニットの位置は図 6.33に示すような配置とした．

図 6.33に示すような故障を含む SOM において忘却学習により生成されたトポグラフィックマップを図 6.34と図 6.35に示す．ここで，図 6.34は近傍拡張を行わなかった場合，図 6.35は制限近傍拡張を行った場合である．また，図 6.36に近傍拡張の種類による各評価値を示す．

図 6.34では，学習パターンが存在しないマップ中央部においてトポグラフィックマップに

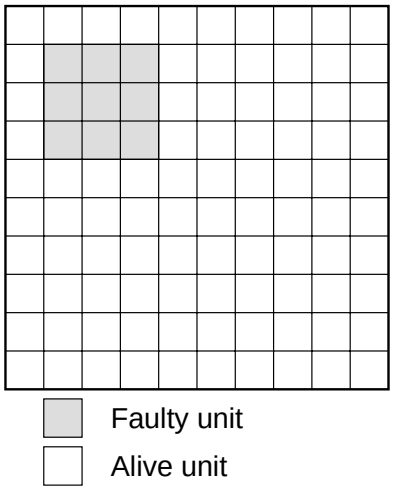

図 6.33: ブロック状故障の補償を行った SOM

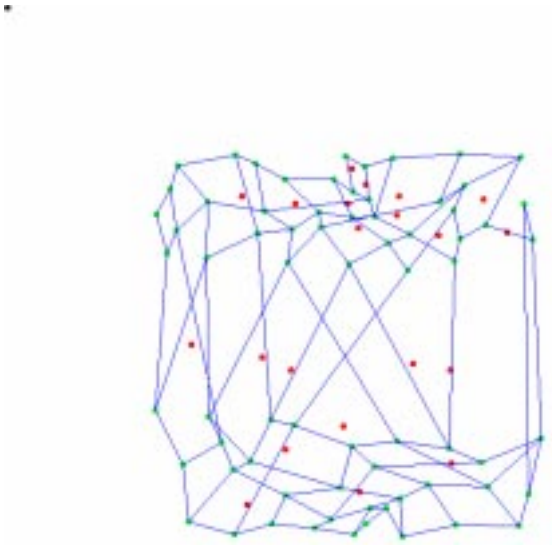

図 6.34: ブロック故障を含む SOM で近傍拡張を行わず忘却学習したときのトポグラフィックマップ

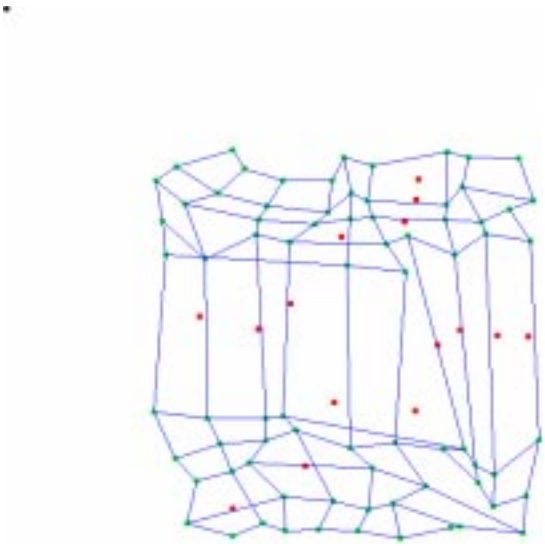

図 6.35: ブロック故障を含む SOM で制限近傍拡張を伴う忘却学習で生成されたトポグラフィックマップ

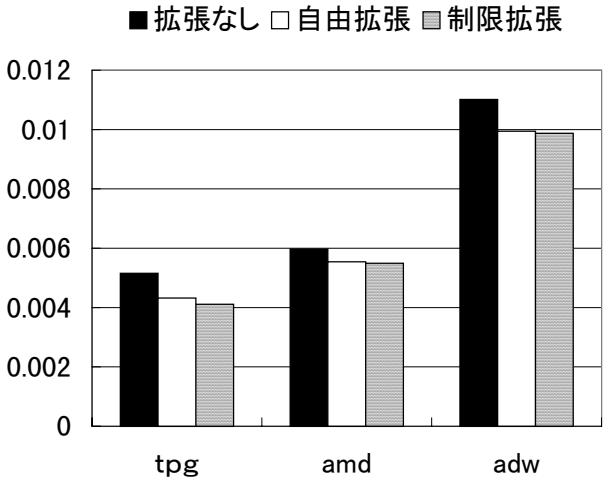

図 6.36: ブロック状故障を含む SOM の忘却学習における各評価値

乱れが生じている．これは，集中的にユニットの故障が生じて近傍内に故障ユニットが数多く含まれてしまい，正しく重みの更新が行われないためであると考えられる．これに対して，制限近傍拡張を行った図 6.35では生成されたトポグラフィックマップに大きな乱れはみられない．また，図 6.36より分かるように，近傍拡張を行うことにより tpg, amd, adw の各評価値は小さくなり，トポグラフィックマップの品質，及び学習パターンの近似精度ともに向上している．

以上より，ライン状故障やブロック状故障など故障ユニットが集中して発生した場合でも，制限近傍拡張により学習パターンの近似精度を低下させることなく，品質の良いトポグラフィックマップの生成が可能であることが示された．

6.5 まとめ

Kohonen による SOM は学習パターンの位相をユニット間の空間的位置に反映させるトポグラフィックマッピングを実現する手法として注目されているが，乱数で初期化された重みのままのユニットが発生したり，学習パターンと関係のないユニットがマップ上に配置されるといったデッドノード問題点が指摘されてきた．これらを解決する手法としては，MST などによる隣接関係の定義などが挙げられるが，ユニットの空間的な位置と学習パ

ターンの位相の類似性が保障できないこと，並列処理による高速化が難しいなどの問題があった．

本章では，はじめに SOM で考えられる忘却について検討した．SOM では重みがユニットの重み空間上での位置を示しており，重みを縮小していくことで実現される忘却が SOM の学習では望ましくないことを説明した．次いで，階層型ニューラルネットワークに提案された忘却学習法について述べ，忘却学習を SOM に応用する場合の問題点について議論した．これらの議論をもとに，ユニットの空間的位相と重み空間での位相を一致させると同時に，デッドノードの削除を自律的に行う忘却学習法を提案した．忘却学習法の効果を確認するために，一様分布，および構造的分布を持つ学習セットを用いて忘却学習実験を行い，トポグラフィックマップの品質を保ちつつ，学習に関与しないデッドノードを削除できることを示した．

さらに，SOM の学習高速化として有力な手法の一つであるハードウェア実装時に必要となる故障保障法について，提案した忘却学習法が応用可能であることを述べた．WSI などのハードウェアには，製造時の故障が避けられない．これらの故障ユニットを忘却により消滅したユニットと同等に扱って学習を進めることで，忘却学習法により故障を含む SOM でもトポグラフィックマップの生成が可能であることを明らかにした．また，近傍内に故障ユニットが存在する場合を考慮して，故障ユニットをとばして隣接関係の再構築をして重みの更新を行う近傍拡張を導入した．このとき，故障ユニット数だけ近傍を延長する自由拡張と，故障ユニット数に関わらず近傍外のユニット 1 個分だけ延長する制限近傍拡張の 2 種類について検討を行った．同時に，故障ユニットが集中して発生するライン状故障時とブロック状故障時についても検討した．その結果，自由近傍拡張を用いた忘却学習ではトポグラフィックマップのねじれが発生する場合が見られたが，制限近傍拡張を用いることで学習パターンの近似精度を悪化させることなく，品質の良いトポグラフィックマップの生成を実現することが可能であることを示した．

忘却学習法により，効率的なトポグラフィックマップが得られることは明らかになったが，消滅するユニットがトポグラフィックマッピングの表現力に与える影響は現在不明である．このため，分類されるべきカテゴリ数をあらかじめ決定した学習セットを用い，これらのカテゴリが忘却学習によるトポグラフィックマッピングでも十分保存されているかどうかを調べる必要がある．

第 7 章

結言

中国では計算機を「電脳」と書く．まさしく，電気で動作する脳であるが，現状の計算機と脳の仕組はまったく異る．現在の多くの計算機は，CPU と呼ばれる VLSI が計算部分を一手に行っている．脳ではニューロンと呼ばれる比較的低速なしきい素子が多数存在し，膨大な数のニューロンが並列に動作している．こうした脳の仕組をモデルに，工学的モデルとして構築されたのがニューラルネットワークである．脳と同じく，単なるしきい素子であるユニットを多数結合し，相互に情報をやりとりすることで期待する機能を実現する．

ニューラルネットワークには教師信号を用いて学習を行うものと，教師信号を用いずに学習を行うものの 2 種類に分けられる．前者の代表例としては階層型ニューラルネットワークがあり，パターン認識などの分野で広く用いられている．後者の例には Kohonen の SOM があり，脳内で見られる機能分化を自律的に実現できる手法として近年注目されている．

こうした機能を実現するためには,「学習」と呼ばれる操作が不可欠である．ニューラルネットワークは多数のユニットとそれを結ぶリンク，および結合重みからなり，学習により結合重みを自律的に獲得していく．学習により自らを問題に対して調節していくため，問題の構造を人間が知ることなく，望む機能を容易にニューラルネットワークに学習させることができる．しかし，ニューラルネットワークの学習は，膨大な数のユニットを用いるため極めて時間がかかり，大きな問題となっている．

本研究では，階層型ニューラルネットワークを学習される効率的な並列学習法について検討を行った．さらに，高速化を行う上で一つの手法となるハードウェア実装時において問題となりうる故障を回避する手法として部分学習法を提案し，その性能について評価を行った．さらに，大規模な競合層を持つ SOM の学習を高速に行うことができる入力層分

割法を提案し，その評価を行った．また，SOM の学習時に発生するデッドノードを取り除くために忘却学習法を提案し，効果的にデッドノードの削除が可能であることを示した．SOM のハードウェア実装では，忘却学習を応用することにより故障 PE を含んだままでトポグラフィックマップの生成が可能であることをあきらかにした．

以下の節では本研究についてまとめ，今後の課題について述べる．

7.1 ニューラルネットワークの並列学習

計算機技術の発展により，コンピュータはますます速く，高機能になっている．しかし，チップ単体での高速化に限界が叫ばれて久しく，計算機機構上の新たな形態として超並列計算機が注目されるようになってきている．超並列計算機は多数のプロセッサを相互結合網により接続し，それぞれが並列に動作することにより高速な処理を実現する．大規模なデータベースマシンなどの分野では一部実用的に用いられており，次世代の計算機アーキテクチャとして期待が高まっている．

特にニューラルネットワークのような並列処理を基本にして構築されているモデルでは，超並列計算機による高速化の期待が極めて大きい．本論文では，2章において代表的なニューラルネットワークの形態である階層型ニューラルネットワークと SOM について，その構造と学習アルゴリズムについて説明し，それぞれの学習時間について検討を行った．また，階層型ニューラルネットワーク，SOM 双方について現在問題となっている点を説明した．

3章において，階層型ニューラルネットワークを学習される代表的な手法である誤差逆伝搬学習法に対してユニット並列モデル，学習セット並列モデル，パス並列モデルの 3 種類の並列化を行い，モデル及び実験によりその学習速度向上率について評価を行った．

ユニット並列モデルはニューラルネットワーク中のユニットが各々並列に動作することを利用した並列学習モデルである．このモデルを nCUBE/2 に実装して学習速度向上率を測定したところ，極めて低い結果しか得られなかった．これは nCUBE/2 のメッセージ受信が同期受信であることや，メッセージの中継などのオーバーヘッドが大きいためと考えられる．モデルを用いた解析でも同様の結果となり，PE が行う処理のほとんどは通信処理であることが分かった．以上から，ユニット並列モデルを高速に実行するには，PE の処理速度は低くても高速な通信を行える計算機を用いる必要がある．

学習セット並列モデルは，誤差逆伝搬学習における重み更新で学習セットを分割して並列に学習できることを利用した並列学習モデルである．nCUBE/2 に実装して学習セット並

列モデルの速度向上率を測定したところ，256PE を使用した encode/decode 問題では 1PE 時と比較して約 30 倍の速度向上率を得た．しかしながら，使用した PE 数に比べて速度向上率が小さく，また PE 数増加でかえって速度向上率が低下した場合も見られた．これは通信回数が多く通信のオーバーヘッドが大きいためであることが分かった．そこで，複数の重みをまとめて一度に通信・加算を行う配列加算通信を用いた改良学習セット並列モデルを提案した．その結果，256PE を用いた encode/decode 問題で約 106 倍の速度向上率を得ることができた．また，モデルを用いて学習セット並列モデルと改良学習セット並列モデルの学習時間と通信時間について考察した．その結果，改良学習セット並列モデルでは通信時間を大幅に減少できると同時に，計算時間に占める通信時間の割合も小さくすることができた．

パス並列モデルは，3 層の階層型ニューラルネットワークにおいて入力層→出力層間でフォワードパスを行うと同時に，出力層→入力層間でバックワードパスを行うモデルである．性質上 2 個の PE を組にして用いる必要があるのと，単体で大幅に高速化できないので，本研究では改良学習セット並列モデルと組み合わせて nCUBE/2 上に実装し，速度向上率を測定した．その結果，パスを分割する以上に学習セットを分割する効果が大きく，改良学習セット並列モデルの方が高速であるという結果が得られた．モデルを用いた解析でも，ある特定の PE 数ではパス並列モデルが改良学習セット並列モデルよりやや高速になる場合があるが，傾向としては改良学習セット並列モデルの方が高速である．

以上の結果から，階層型ニューラルネットワークの並列学習法として改良学習セット並列モデルが優れていることが分かった．

ニューラルネットワークの学習をさらに高速化する場合，VLSI や WSI といったハードウェアにネットワークを直接実装する手法が考えられる．しかし，チップ製造には欠陥が避けられない．こうした欠陥に影響されないようにネットワークをあらかじめ学習させるか，あるいは故障の影響を回避できるようなメカニズムがニューラルネットワークに要求される．

4章では，ユニット並列モデルを例にチップ上に実装したニューラルネットワークで起こりうる故障について考察し，それらの多くがリンク故障として表現できることを指摘した．また，あるリンクの故障により影響を受ける上位層のユニットは唯一つであることから，ネットワーク全体ではなく故障の影響を受ける部分だけを取りだして再学習を行う部分再学習法を提案し，シミュレーションによりその効果を評価した．同時に再学習開始の

重みを故障前の重みとすることでさらなる高速化を図った．小規模ネットワークでの実験としてXOR 問題，大規模ネットワークでの実験として顔画像認識問題によりシミュレーションを行った結果，全ネットワークを対象とした故障補償と比較して大幅な高速化が達成できた．しかし，XOR 問題において隠れ層のユニットの出力によっては部分再学習法では故障補償が行えない場合もあることが明らかとなった．

Kohonen による SOM では，従来競合層の分割による並列化が多く試みられてきた．しかし，競合層分割法では学習後半にアイドルとなる PE が増加する上，入力層ユニット数に比べて競合層ユニットが多くなる大規模問題では十分な高速化が行えないという問題がある．そこで，5章において入力層を分割して PE に割り当てることで常に PE が同程度にビジーとなる入力層分割法を提案し，その性能を評価した．その結果，全ての PE でほぼ均一の負荷となる並列学習が可能であることを明らかにした．このとき，100×100 程度の SOM の学習をユニット数と同程度の PE を用いて並列学習した場合，逐次処理と比較して約 3000 倍の高速化が可能であることモデルによるシミューレーションであきらかにした．

同時に競合層分割法と入力層分割法の学習時間をモデルを用いて比較した．入力層・競合層のサイズが同程度の場合，提案した入力層分割法と従来の競合層分割法の学習時間は同程度である．しかし，画像符号化などで用いられる大規模な入力層を持つ SOM では，競合層分割法では使用する PE 数に比べて SOM 中のユニット数が多くなるために十分な高速化が行えず，学習時間が長くなってしまう．これに対し，入力層分割法では入力ユニット数と同数の PE を使用することから，入力ユニット数の増加に対する学習時間の増加は緩やかである．したがって，入力層サイズ>競合層サイズの SOM では，入力層分割法により高速な学習が行えることがあきらかとなった．

7.2　忘却学習による自己組織化ニューラルネットワーク

Kohonen の SOM では，トポグラフィックマッピングを実現するためにデッドノードと呼ばれる学習パターンを反映しないノードが発生することがある．重みが乱数のまま残されたユニットなどが発生すると，得られるトポグラフィックマップの品質が大幅に悪化してしまう．こうしたデッドノードは，非一様分布した学習パターンを用いた場合に顕著に見られる．こうしたデッドノードを削除するため，6章において忘却学習法を提案した．忘却学習法では，どの学習パターンにもウィナーとならないユニットをマップ上から削除し，除かれたユニットをとばして隣接関係を再構築することによりデッドノードを削除する．一様

分布および構造的分布を持つ学習パターンを用いて忘却学習を行った結果，トポグラフィックマップの品質を保ったまま学習に関与しないデッドノードを削除できることを示した．

階層型ニューラルネットワークの場合と同じく，SOM でも高速な学習にはハードウェア上への実装が有望視されている．このとき問題となる故障 PE の補償法について，従来は代替 PE や通信路を用いて所定の PE 網を実現するアプローチがとられている．しかし，SOM には忘却により削除されるユニットが存在するなど，生来の冗長性を内包している．そこで，忘却学習法を応用することにより故障保障が可能であることを実験により示した．故障 PE 上に割り当てられたユニットは，忘却により消滅したユニットと同等に扱って隣接関係の再構築を行うことでトポグラフィックマップの生成が可能であることを明らかにした．同時に，近傍内に故障ユニットが存在する場合に対して，近傍を越えて重みを更新する制限近傍拡張を導入することで，トポグラフィックマップの品質を良好に保ったまま，学習パターンの近似精度を向上させることができた．

7.3 今後の課題

代表的なニューラルネットワーク形態として階層型ニューラルネットワークと SOM の並列学習法について議論した．人間の脳をモデルにしたニューラルネットワークはその優れた特質から多くの分野で応用され，研究されている．本研究は学習の高速化という観点から並列学習法を取りあげ，モデルによる解析と実装による評価を行った．また，学習高速化の一つであるハードウェア上への実装時に問題となるハードウェア故障を補償する学習法についても考察した．SOM については，学習パターンを反映しないデッドノードを忘却により削除する忘却学習法を提案し，その評価を行った．

階層型ニューラルネットワークにせよ，SOM にせよ，学習を実現するアルゴリズムには様々なものがある．本論文では代表的な学習アルゴリズムとして誤差逆伝搬学習と Kohonen のアルゴリズムを用いたが，他にもより高速なアルゴリズムは様々なものが提案されている．こうしたものを含めて，より並列処理に適した学習アルゴリズムの提案と評価が必要である．

SOM に対する忘却学習法によりデッドノードの削除が可能であることはシミューレションにより示したが，理論的な解析も行う必要がある．また，削除だけでなく入力パターンが集中する空間では新しいユニットを生成するような学習アルゴリズムについても検討が必要ではないか，と考える．提案した忘却学習法では，ユニットごと消滅する忘却を用い

ているが，生理学的研究では結合強度（重み）を変化させている場合が多い．これはSOMではリンク毎の忘却に相当することから，完全結合でなく任意の結合を実現するような学習法が好ましいのではないか，と思われる．

故障を含んだSOMにおける忘却学習では，故障ユニットに加えて忘却による消滅ユニットが発生する．このため，動作しているユニット数が減少し，場合により学習パターンの位相を十分反映できない場合が考えられる．こうした問題を検討するため，表現しているカテゴリ数があらかじめ分かっている学習セットを用い，これがトポグラフィックマップ上で十分表現されているかどうかの検討が必要である．また，忘却により消滅したユニットを担当するPEは普通に動作しており，こうしたユニットが多く発生することはPEの使用効率上好ましくない．そこで，忘却により消滅したユニットを故障ユニット（PE）の代替として使用する手法の可能性について調査する必要がある．

本論文では部分再学習法，および忘却学習法の2つの学習法の提案を行ったが，例題による結果からの考察であり，動作の数学的な解析を行っていない．今後は両学習法の数学的な根拠を与える必要がある．

謝辞

本研究を進めるにあたり，的確な指摘と助言を頂いた北陸先端科学技術大学院大学 情報科学研究科 堀口進教授に心より感謝いたします．

また，本研究を行うにあたり，ゼミ等で熱心に御指導・御助言を頂いた北陸先端科学技術大学院大学 情報科学研究科 阿部 亨助教授に深く感謝の意を表します．

サブテーマを行うにあたり，ニューラルネットワーク全般について熱心に御指導・御討論いただきました Jung H. Kim 博士（現在，Southwest Louisiana University）に厚く御礼申し上げます．

北陸先端科学技術大学院大学 情報科学研究科 日比野 靖教授，松澤 照男教授には研究を進めるにあたり様々な面で御指導を頂きました．また，東北大学工学研究科 阿曽 弘具教授には審査にあたり御指導いただき，ここに深く感謝いたします．

北陸先端科学技術大学院大学 情報科学研究科 丹 康雄助手には，ニューラルネットワークの故障補償ついて助言頂き，多くの文献を紹介して頂きました．ここに感謝いたします．

沼田一成博士（現在，ソニーミュージックエンターテイメント）には，公私にわたりお世話になりました．ここに心より感謝いたします．

北陸先端科学技術大学院大学 情報科学研究科 國藤 進教授には修士課程におけるサブテーマで，人工知能に関する様々な知識を御教授頂きました．ここに御礼申し上げます．

また，同窓の井口 寧氏は計算機環境の整備に多大な努力を払ってくれました．ここに感謝いたします．また，加藤聡君をはじめ，研究室の後輩諸氏は私の毒舌にも耐え，研究内容について様々な質問，鋭い指摘，辛辣な批評をいただき，大いに参考にし，また発奮いたしました．

最後に，本研究を行う機会を与えて下さいました方々に，この場を御借りして御礼を述べさせて頂きます．

参考文献

[1] P. H. Winston, "Artificial Intelligence", chapter 22 – 24, pp. 443 – 504, Addison-Wesley (1992).

[2] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", Bullet. Math. Biophysics, Vol. 5, pp. 115 – 133 (1943).

[3] D. Hebb, "The Organization of Behavior", New York: Wiley (1949).

[4] F. Rosenblatt, "The perceptron: A probablistic model for information sorage and organization in the brain", Psychol. Rev., Vol. 65, No. 6, pp. 386 – 408 (1958).

[5] M. Minsky and S. Papert, "Perceptrons", MIT Press (1969).

[6] D. E. Rumelhart, J. L. McClelland and PDP Research Group, "Parallel Distributed Processing", Vol. 1, MIT Press (1986).

[7] S. E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Network ", CMU-CS-88-162 (1988).

[8] W. Schiffman, M. Joost and R. Werner, "Optimization of the Backpropagation Algorithm for training Multilayer Perceptrons", Technical report, University of Koblentz, Institute für Physics (1993).

[9] 塚本義明, 生天目章, "多層ネットワークの瞬時学習法", 情処学論, Vol. 34, No. 9, pp. 1882 – 1891 (1993).

[10] 曽根原登, 平山亮, "超並列コンピュータによる神経回路モデル処理", 情報処理, Vol. 32, No. 4 (1991).

[11] A. Singer, “Implementation of Artificial Neural Network on the Connection Machine”, Technical report, Thinking Machine Corporation (1990).

[12] X. Zhang, D. L. Waltz, et al., “The Back-propagation Algorithm On Grid and Hypercube Architecture”, Technical report, Thinking Machine Corporation (1990).

[13] U. A. Muller, B. Baumle, P. Kohler, A. Gunzinger and W. Guggenbuhl, “Achieving Supercomputer Performance for Neural Net Simulation with an Array of Digital Signal Processors”, *IEEE Micro Magazine*, Vol. 12, No. 5, pp. 55 – 65 (1992).

[14] M. Witbrock and M. Zagha, “An implementation of backpropagation learning on GF11, a large SIMD parallel computer”, *Parallel Computing*, Vol. 14, No. 3, pp. 329 – 346 (1990).

[15] 田中繁, “一次視覚野のカラム構造形成”, 日経サイエンス別冊 107 脳と心, pp. 44 – 55 (1993).

[16] T. コホネン, “自己組織化マップ”, シュプリンガー・フェアラーク東京 (1996).

[17] K. Obermayer, H. Ritter and K.Schulten, “Large-scale simulations of self-organizing neural networks on parallel computers:application to biological modelling”, *Parallel Computing*, Vol. 14, pp. 381 – 404 (1990).

[18] G. Myklebust and J. G.Solheim, “Parallel Self-organizing Maps for actual applications ”, *Proc. Int. Conf. Neural Networks 1995* (1995).

[19] C.-H. Wu and R. E. Hodges, “Parallelizing the Self-Organizing Feature Map on multiprocessor systems”, *Parallel Computing*, Vol. 17, pp. 821 – 832 (1991).

[20] J. A. Kangas, T. K. Kohonen and J. T. Laaksonen, “Varinats of Self-Organizing Maps”, *IEEE Trans. on Neural Networks*, Vol. 1, No. 1, pp. 93 – 99 (1990).

[21] 久間和生, 田井修市, 太田淳, “ニューラルネットワークのハードウェア”, 信学誌, Vol. 73, No. 7 (1990).

[22] 丹康雄, 南谷崇, “フォールトトレランスを有する階層型ニューラルネットワークとその性質”, 信学論（D-I）, Vol. J76-D-I, No. 7, pp. 380 – 389 (1993).

[23] C. Khunasaraphan, V. Vanapipat and C. Lursinsap, “Weight Shifting Techniques for Self-Recovery Neural Networks”, *IEEE Trans. on Neural Networks*, Vol. 5, pp. 651 – 658 (1994).

[24] 當麻喜弘, “知的情報処理の自己修復・再生”, *Proc. of Artificial Intelligence and Neuro-Computers, 6th Symposium on Universities and Science*, pp. 204 – 208, Kubapro Co. (1992).

[25] 麻生英樹, “ニューラルネットワーク情報処理”, 産業図書 (1988).

[26] 甘利俊一, “ニューラルネット理論の課題”, 信学誌, Vol. 73, No. 7, pp. 680 – 682 (1990).

[27] C. von der Malsburg, “Self-organization of orientation sensitive cells in the striate cortex”, Kybernetik, Vol. 14, pp. 85 – 100 (1973).

[28] 倉田耕治, “神経場の自己組織”, 第 5(1) 章, pp. 129 – 145, サイエンス社 (1993).

[29] E. R. Kandel and R. D. Hawkins, “ニューロンレベルでみた学習”, 日経サイエンス別冊 107 脳と心, pp. 56 – 67 (1993).

[30] T. Kohonen, “Self-Organization and Associative Memory”, Springer-Verlag (1984).

[31] D.-I. Choi and S.-H. Park, “Self-Creating and Organizing Neural Networks”, *IEEE Trans. on Neural Networks*, Vol. 5, No. 4, pp. 561 – 575 (1994).

[32] 伊勢崎剛志, “階層型ニューラルネットワークの故障時における再学習に関する研究”, 東京工業大学 卒業論文 (1992).

[33] 安永守利, 浅井光男, 柴田克成, 山田稔, “ニューラルネットワーク集積回路の自律的な欠陥救済能力”, 信学論（D-II）, Vol. J75-D-I, No. 11, pp. 1099 – 1108 (1992).

[34] 山森一人, 阿部亨, 堀口進, “超並列計算機 nCUBE2 上のニューラルネットワークを用いた 顔画像認識法”, 情処学研報, No. 94-ARC-109, pp. 65 – 72 (1994).

[35] T. Hendtlass, “A Self Organizing Artificial Neural Network with problem dependent structure”, *Proc. Int. Conf. Neural Networks 1995* (1995).

[36] 石川真澄, “忘却を用いたコネクショニストモデルの構造学習アルゴリズム”, 人工知能学会誌, Vol. 5, No. 5, pp. 71 – 79 (1990).

[37] 津本忠治, “脳の発生・発達と可塑性”, 日経サイエンス別冊 107 脳と心, pp. 22 – 31 (1993).

本研究に関する発表論文

査読付き論文

[1] 山森 一人，阿部 亨，堀口 進: “超並列計算機上でのニューラルネットワークの並列学習法”，信学論， Vol.J80-D-II, No.1，pp.350–353，(1997)

[2] 山森 一人，堀口 進: “故障を持つニューラルネットワークの部分再学習法”，信学論（投稿中）

[3] 山森 一人，堀口 進:“並列計算機上の誤差逆伝搬学習法の並列学習モデル”，信学論（投稿中）

[4] 猪瀬博和，山森 一人，阿部 亨，堀口 進: ” 準概略図を用いたハイブリッドニューラルネットワークによる 3 次元物体認識”，信学論，Vol.J78-DII, No.12, pp.1927–1931（1995）

査読付き国際会議発表論文

[5] Jung H. Kim, Kunihito Yamamori, Susumu Horiguchi: “The Fault-Tolerant Design of Artificial Neural Networks”, Joint Technical Conference on Circuit/Systems, Computer and Communications 1995, pp.687-690,(1995)

[6] Kunihito Yamamori, Susumu Horiguchi, Jung H.Kim, Sung-K. Park, Byung H.Ham: “The Efficient Design of Fault-Tolerant Artificial Neural Networks”, Proc.IEEE Int. Conf. Neural Networks, Vol.3, pp.1487-1491,(1995)

口頭発表論文

[7] 山森 一人，阿部 亨，堀口 進: “超並列計算機上のニューラルネットワークシミュレータ”，平成 5 年度電気関係学会北陸支部連合大会論文集，p.399,（1993）

[8] 山森 一人，阿部 亨，堀口 進: “超並列計算機上のニューラルネットワークを用いた顔画像認識に関する研究”，平成 6 年情報処理学会第 49 回全国大会論文集，2-271，(1994)

[9] 山森 一人，阿部 亨，堀口 進: “超並列計算機 nCUBE2 上のニューラルネットワークを用いた顔画像認識法”, 情処研報 (計算機アーキテクチャ)，Vol.94,No.107(94-ARC-109),pp.65–72,(1994)

[10] 山森 一人，堀口 進: “ニューラルネットワークのリンク故障に対する高速な再学習法”，平成 7 年情報処理学会第 51 回全国大会論文集，2-47,(1995)

リサーチレポート等

[11] Kunihito Yamamori, Jung H.Kim, Susumu Horiguchi: “The Efficient Fault-Tolerant Design of Artificial Neural Networks”, JAIST Research Report IS-RR-96-0003,(1996)

[12] 山森 一人， 堀口 進:“自己組織化ニューラルネットワークの並列学習シミュレーション”，JAIST Research Report IS-RR-97-0006,(1997)