

6R-7

8隣接プロセッサ・アレイによる ニューラルネットワークの並列処理

武田 利浩、 田中 昭吉、 丹野 州宣
(山形大・工)

1 はじめに

これまでにさまざまなモデルのニューラルネットワークが提案され、基礎的な研究や解析が行われる一方で、パターン認識や連想記憶など多くの分野に渡ってアプリケーションへの応用も行われるようになってきた⁽¹⁾⁽²⁾。ニューラルネットワークは学習に膨大な時間を必要とするので、改良や試行錯誤をとまなう研究やアプリケーションへの応用などにおいては、高速なシミュレータを必要とする。

本論文は、大規模なBPモデル及びKohonenの自己組織化特徴マッピングモデルを8隣接プロセッサ・アレイ上で高速にシミュレートする並列処理アルゴリズムを提案する。本アルゴリズムは、ニューラルネットワークの本来持つ並列性を生かし、ニューロン間の"通信"を極力抑えることで高速な実行を実現している。

2 ニューラルネットワーク

2.1 BPモデル

BPモデル⁽¹⁾⁽²⁾は、図1のような階層型ニューラルネットワークに、誤差逆伝播法を用いて学習を行う。学習は、前進処理と誤差逆伝播処理の2つのフェーズを繰り返すことによって行われる。前進処理では、入力層に与えられた入力データに対する出力層での出力を求める。誤差逆伝播処理では、出力と教師信号との誤差を求め、出力層から入力層へ逆向きに伝播させて、ニューロンの間の結合荷重としきい値を更新する。

以下にL層のBPモデルでの処理を示す。

前進処理

$$y_i(k+1) = \sum_j x_j(k) w_{ij}(k) \quad (1)$$

$$x_i(k+1) = f(y_i(k+1), \theta_i(k+1)) \quad (2)$$

誤差逆伝播処理

<出力層>

$$\delta_i(L) = (T_i - x_i(L)) f'(y_i(L), \theta_i(L)) \quad (3)$$

<中間層>

$$\alpha_i(L) = \sum_j \delta_j(k+1) w_{ji}(k) \quad (4)$$

$$\delta_i(k) = \alpha_i(k) f'(y_i(k), \theta_i(k)) \quad (5)$$

$$w_{ij}(k) = w_{ij}(k) + \alpha \delta_i(k+1) x_j(k) \quad (6)$$

$$\theta_i(k) = \theta_i(k) + \beta \delta_i(k+1) \quad (7)$$

ただし、 $f()$ は入出力関数、 α, β は結合荷重、しきい値の更新定数、 k は層番号を表す。

2.2 Kohonenの自己組織化特徴マッピング

各ニューロンは、図2のように2次元に配置され、それぞれ入力ベクトル X に対する参照ベクトル m_i を持つ⁽¹⁾⁽²⁾⁽³⁾。教師なしで、次々と入力ベクトル X を提示して、距離計算、競合、更新を繰り返す。距離計算によって入力ベクトル X と参照ベクトル m_i との距離が計算され、競合によって一番近いものを選ばれる。更新では、選ばれたニューロンとその近傍のニューロンの参照ベクトルを更新する。距離計算と競合、更新の処理は次のようになる。

距離計算

$$d_i = \|x - m_i\| \quad (8)$$

競合

$$d_c = \min_i d_i \quad (9)$$

更新

$$\left\{ \begin{array}{ll} m_i(t+1) = m_i(t) + \alpha(t)[x(t) - m_i(t)] & \text{if } i \in N_c(t) \\ m_i(t+1) = m_i(t) & \text{if } i \notin N_c(t) \end{array} \right\} \quad (10)$$

$\alpha(t)$ は、 $0 < \alpha(t) < 1$ を満たす、時間と共に減少する更新係数。

3 8隣接プロセッサ・アレイ上での並列処理

3.1 8隣接プロセッサ・アレイの構成と機能

本論文で扱う8隣接プロセッサ・アレイは、図3に示すように隣接する8方向のPEとの間に通信路を持つトラス構造である。各PEは、加減算、乗算、入出力関数演算の機能を持ち、演算に必要なデータを保持する為にローカルメモリを持つ。各PEは、図4のように隣接する8方向のPEとの半2重通信機能を持つ。ここで、プロセッサ・アレイのサイズを $P \times P$ 個とし、PEを $PE(i,j)$ ($i,j=0,1,2,\dots,p-1$)の形で表わす。

3.2 BPモデル

BPモデルでは、その学習において式(1)~(7)の処理を行うが、これらは他のニューロンすなわちPEとの通信を必要とするものではないものに分けられる。式(2)、(3)、(5)、(7)は、そのニューロン内で完結しておりPE間の通信が必要である。式(1)、(4)、(6)の処理にはPE間の通信が必要である。これらの通信が必要な処理について通信時間を小さくするために、1つのニューロンの演算を他のニューロンへ分散させる。ここではこの3つの処理について、ニューロンのマッピングと各PEでの処理を述べる。

BPモデルでは、L層の階層型ニューラルネットワークを使うので、ここでは各層のニューロン数を N_k ($k=1,\dots,L$)と表すことにする。一般に、 N_k と P は、等しくないが、ここでは説明を簡単にするために $P=N_k$ と仮定する。

3.2.1 ニューロンのマッピング

各PEに対し各層のニューロンを1つずつマッピングする。したがって、各PEは層数(=L)のニューロンを持ち、入力層への入力 $x_i(1)$ 、各層のニューロンのしきい値 $\theta_i(k)$ 、出力層への教師信号 T_i が用意される。また、プロセッシングエレメント $PE(i,j)$ にマッピングされたニューロンへの結合荷重は、対角線上のPEに分散してマッピングする。PE(0,0)にマッピングされたニューロンを例に取ると、図5のようになる。k-1層からk層の1番目のニューロンへの結合荷重は、PE(0,0)上にマッピングされ、斜線で示される対角線上のそれぞれのニューロンは、そのニューロンを含むカギ型のニューロンからの $X_{(k-1)}(j=1,2,\dots,N)$ に対する結合荷重を持つことになる。したがって、k層のN番目のニューロンからk+1層への結合荷重は、PE(i,p-1) ($i=0,1,2,\dots,p-1$)及びPE(p-1,j) ($j=0,1,2,\dots,p-1$)上にマッピングされる。本アルゴリズムで用いるプロセッサ・アレイは、トラス状なのでk層の全てのニューロンに対して同様にマッピングできる。

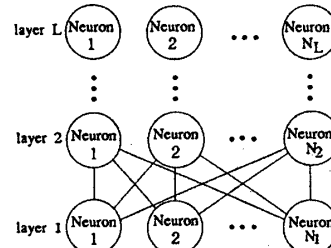


図1 L層のBPモデル

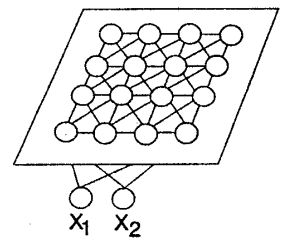


図2 Kohonenモデル

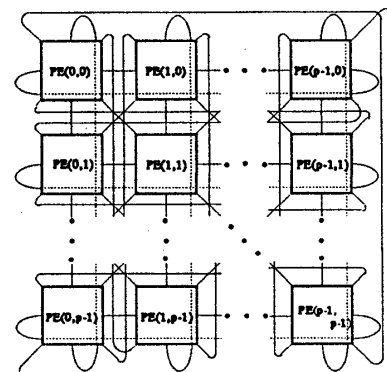


図3 8隣接プロセッサ・アレイ

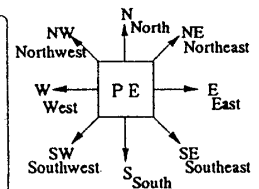
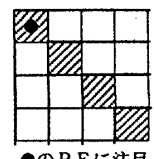


図4 PEの通信リンク



●のPEに注目

図5 結合荷重のマッピング

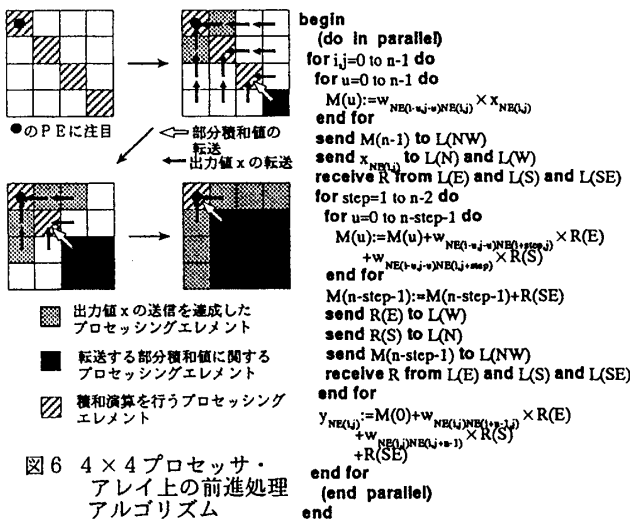


図6 4×4プロセッサ・アレイ上の前進処理アルゴリズム

図7 アルゴリズム1:式(1)

9.2.2 PEでの処理

前進処理の式(1)の処理は、k層のニューロンに着目すれば、k-1層からの出力 $X(k-1)$ 全てを必要とする。ここで、結合荷重を対角線上に分散して配置しているため、出力 $X(k-1)$ に対応する対角線上のニューロンに渡し積和を行い、積和値を集めればよい。図6に示すように、出力 $X(k-1)$ はN,W方向に送信し、積和値は対角線上のNW方向に対して遠いPEの分から順に送信する。受け取った出力 $X(k-1)$ は、入力リンクと反対の向きにスルーする。

ここで、送信にはN,W,NWの3方向しか使っていないので半2重でもコンフリクトすることもなく、PEの配置はシメトリックなので、全てのPEで同様の処理を平行に行うことができる。このアルゴリズムを図7に示す。ただし、アルゴリズム中のsend及びreceiveは送受信の命令で、L()は送信リンクを表し、R()は受信レジスタを表す。また、NE(i,j)はPE(i,j)にマッピングされたニューロン番号を表す。

逆伝播処理の式(4)は、前進処理の式(1)と逆向きのデータの流れとなる。したがって図8のように、誤差 δ をSE方向に流し、カギ型に配置された結合荷重との積和を求め積和値を、SE方向に流すことで処理が行われる。

逆伝播処理の式(6)は、k層の結合荷重を更新するのにk+1層の誤差 δ とXを必要とする。誤差は、結合荷重のある対角線上のPEに渡し、対角線上の結合荷重に対するXをSE方向に流すことで処理が行われる。

式(4)と(6)についてのアルゴリズムは、割愛する。

9.2.3 任意形のBPモデルへの拡張

ここでは、各層のニューロン数とプロセッサ数が一致しない($P \neq N, (k=1,2,3,\dots,L)$)ようなBPモデルへの拡張を考える。前進処理の式(1)については、次のようになる。

(a) $P \leq N, (k=1,2,3,\dots,L)$ 他のニューロンに影響を与えない常態に0を出力するニューロンを加えて、 $p=N_k$ となるようにする。
(b) $P < N, (k=1,2,3,\dots,L)$ $m_k P = N_k (m_k \text{は自然数})$ となるようにニューロンを加え、大きさ $P=p \times p$ の m_k 個の部分に分ける。それぞれを各PEに割り付け、1つのPEに m_k 個のニューロンを持たせる。

各部分は、他の部分の $X(k)$ 及び $w_{ij}(k)$ を必要としないので、分けた部分をそれぞれ独立して計算することが可能である。したがって、k+1層での出力を求めるには、大きさ $p \times p$ の部分の演算を m_k 回繰り返す、最後に和を求めればよい。

同様にして、式(4)、(6)も処理することができる。

9.3 Kohonenの自己組織化特徴マッピング

9.3.1 ニューロンのマッピング

Kohonenの自己組織化特徴マッピングは、図2に示したような2次元のニューロンの配置を用いるので、そのまま8隣接プロセッサ・アレイへとマッピングする。

このとき、式(8)の距離計算について、入力データXの持たせ方によって2つのアルゴリズムが考えられる。全てのニューロンに必要な入力Xをマッピングする方法と、各ニューロンに分散して持たせる方法である。前者は、Xと m_i の距離を求めるのに通信を必要としないが、各ニューロンにXの次元だけのメモリを必要とする。後者では、距離を求めるのに通信を必要とするが、メモリは次元数/ニューロン数で済む。PE数と次元数が近い時有利である。これは、先に示したBPモデルの式(1)の処理と同様にして実現できるので説明は割愛する。

ここでは、競合による勝者を定めるアルゴリズムを述べる。説明を簡単にするために、ニューロン数をNと表すことにすると、 $P=N$ と仮定する。

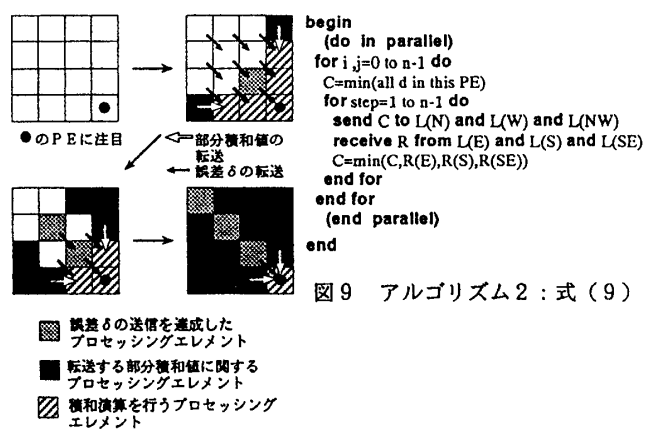


図9 アルゴリズム2:式(9)

図8 4×4プロセッサ・アレイ上の逆伝播処理アルゴリズム

9.3.2 PEの処理

8隣接の特徴を生かしたブロードキャストによって最小の距離 $d(i,j)$ を持つものをえらぶことができる。アルゴリズムは図9のようになる。

ただし、Cはニューロン番号と最小の値を持つ構造体、min()は最小値を選びCにニューロン番号と最小の値をセットする関数である。

4 処理時間の評価

BP及びKohonenの自己組織化特徴マッピングの1回の学習に要する処理時間を見積もる。その際の諸条件は3節の仮定に従う。

4.1 BPモデル

式(2)、(3)、(5)、(6)、(7)の演算に要する時間をそれぞれ T_1, T_2, T_3, T_4 とする。式(1)、(4)での単位通信時間を T_c 、積の単位時間を T_m 、和の単位時間を T_s とすると、前進処理時間と逆伝播処理時間の和Tは

$$T = 2n^2(L-1)T_c + (2n^2+3n-3)(L-1)T_1 + (3n-3)(L-1)T_2 + (L-1)(T_3+T_4+n^2T_m+T_s) + T_3$$

と表わすことができる。これにより積和処理時間のオーダーは $O(n^2L)$ であり、通信時間のオーダーは $O(nL)$ であることがわかる。

4.2 Kohonenの自己組織化特徴マッピング

距離計算の時間を T_d 、競合における比較時間を T_c 、通信時間を T_s 、ベクトル m_i の更新に要する時間を T_u とする。処理時間Tは、 $T = T_d + (T_c + T_s)(n-1) + T_u$ と表わすことができる。競合の処理と通信時間は共に $O(n)$ であることがわかる。

競合において、 $P < N$ の時は、1つのPEに割り当てられた、ニューロンのうちの最小値を求めるステップが増えるだけで、通信は増えない。したがって、処理時間Tは次式で表せる。

$$T = mT_d + T_c + (T_s + T_u)(n-1) + mT_s$$

ただし、 $m = \lceil N/P \rceil$ 、 T_c は m 個の $d(i,j)$ の比較時間である。

5 むすび

本文ではBP及びKohonenの自己組織化特徴マッピングモデルについて8隣接プロセッサ・アレイ上で実現する並列処理アルゴリズムについて述べた。

本アルゴリズムは、処理の分散による通信時間のオーバーヘッドを最小化した並列処理を実現することが可能である。また、隣接した8つのプロセッサ間で通信できる最新の商用マシン⁽⁴⁾上に容易にインプリメントできると思われる。

問題点としては、プロセッサ・アレイにマッピングするニューロン数がPE数の整数倍でないとき効率が落ちてしまう点が挙げられる。

今後は、対象とするニューラルネットワークのモデルを広げていき、処理時間の詳細な検討も行いたい。

文献

- (1) Richard P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP MAGAZINE, April 1987.
- (2) ニューロコンピューティングの基礎理論, 海文堂, 1990.
- (3) Teuvo Kohonen, "The Self-Organizing Map," Proc.IEEE, vol.78, no.9, pp.1464-1480, September 1990.
- (4) "MasPar Parallel Application Language(MPL) Reference Manual," Document Part Number 9302-0000, Revision A4, MasPar Computer Corporation, March 1991.