社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－HPC－106 (5)
2006／6／12

ヘテロなクラスタ環境に適した並列誤差逆伝搬アルゴリズム

渥美 清隆† 佐々木健吾††

† 鈴鹿工業高等専門学校 〒510-0294 三重県鈴鹿市白子
† 電気通信大学 〒182-8585 東京都調布市調布ヶ丘 1-5-1
E-mail: †kiyotaka@ka-lab.jp

あらまし ニューラルネットワークの学習法である誤差逆伝搬アルゴリズムには様々な並列アルゴリズムが存在するが，どの方法を見ても，各計算機内の計算処理の重さと計算機間の通信量によって実行速度が決まると言える．一般的に誤差逆伝搬アルゴリズムをそのまま並列化すれば膨大な通信量を避けられない．本論文で提案するアルゴリズムは，中間層のニューロンを基準に負荷分割をすることで，能力の異なる計算機からなる並列計算機でも効率的に計算出来る．また，一部の通信を省略することで高速化を図る．通信の一部を省略しても，学習可能であることを実験的に示す．

キーワード ニューラルネットワーク，誤差逆伝搬アルゴリズム，クラスタ型並列計算機

A parallel algorithm for backpropagation using Heterogeneous Cluster Type Parallel Computer

Kiyotaka ATSUMI† and Kengo SASAKI††

† Suzuka National College of Technology Shiroko, Suzuka, 510-0294 Japan
† The University of Electro-Communications Chofugaoka 1-5-1, Chofu, 182-8585 Japan
E-mail: †kiyotaka@ka-lab.jp

Abstract There are various parallel algorithms for the backpropagation algorithm that is learning method of neural networks. Execute time for these parallel algorithms are fixed by computational complexity in each computer and quantity of communication in a parallel computer. Generally, an parallel algorithm generates enormous communication quantity if you just code a backpropagation algorithm. An parallel algorithm proposed by this paper can efficient parallel compute backpropagation by deviding middle neurons in a parallel computer that ability consists of a different computer. Moreover, this parallel algorithm is fast because a part of communication is omitted. We show experimentally that this algorithm is feasible even if some communication is omitted.can comput

Key words Neural Networks, Backpropagation Algorithm, Cluster Type Parallel Computer

1. はじめに

ニューラルネットワークは，ニューロンと呼ばれる単純な計算処理機構が多数配置され，それらをシナプスと呼ばれる一方向辺によって結合されている．結合度合いを示すため，シナプスには重みという情報が加えられ，送信元となるニューロンから受信先となるニューロンへの信号伝達の強弱を付ける役割を果たす．フィードフォワード型ニューラルネットワークとは，シナプスのループが無く，センサーからの入力を受けて出力信号を得る一方向性のニューラルネットワークである．

フィードフォワード型ニューラルネットワークの学習方法として誤差逆伝搬アルゴリズム [1] が広く利用されている．また，これに関する並列アルゴリズムについての研究が Sundararajan ら [2] によって行なわれている．誤差逆伝搬アルゴリズムは大規模な行列乗算が含まれるので，並列コンピュータ上で負荷分散しやすいという特徴を持っている．しかしながら，単一ニューロンは非常に簡単な計算しか行なわず，ニューロン同士の通信が大量に発生することから，計算機の筐体を越えて誤差逆伝搬アルゴリズムを並列

化する場合，この通信を相対的にどれだけ抑制できるかによって，アルゴリズムの実行速度が決まると言える．また，近年のキャッシュ依存のプロセッサでは，キャッシュヒットの可否によって，速度が大幅に異なるため，行列乗算の式に出来る限り修正し，BLAS(Basic Liner Algebra Subprograms)[注1]などの行列乗算パッケージを利用してキャッシュヒット率を上げることが重要である．

本論文では，実行速度の向上のために，(1) 行列乗算を多用するため，ブロック単位学習 [2] と呼ばれる方法を用いると同時に，一部の通信を省略することにより，実行速度を向上させることが可能であることを示す．また，一部の通信の省略があまり影響を与えないだろうということを実験的に示す．また，本論文で提案するアルゴリズムの負荷分割のポイントが中間層のニューロンを中心に行なうので，計算機間の処理能力が異なっていたとしても，処理時間平等となるように負荷分割できる．

2. 誤差逆伝搬アルゴリズム

本論文でフィードフォワード型ニューラルネットワークを扱うが，話を単純にするため 3 層ニューラルネットワークを仮定する．ただし，ここで述べる手法は多層への拡張は容易にできる．ニューラルネットワークは複数の入力信号を処理して出力信号を得るニューロンと呼ばれる節点と 2 つの節点を片方向接続する辺からなる．節点は入力層，中間層，出力層にそれぞれ属しており，入力層に属する節点では，外部からの入力をそのまま出力する．中間層および出力層では，それぞれの節点に接続されている入力辺から与えられる重み付信号値の和をシグモイド関数にかけた結果を出力とするようなネットワークを仮定する．このようなニューラルネットワークを学習させるために，教師信号付パターンデータを用いる．これから誤差逆伝搬アルゴリズムを説明するために，いくつかの変数を定義する．

3 層ニューラルネットワークの各層の節点数が入力層に l 個，中間層に m 個，出力層に n 個含まれ，教師信号付パターンデータのデータ数を p とすると，入力層から得られる信号は $S^I_{p,l}$ の行列で表現する．同じく中間層，出力層から得られる信号は $S^M_{p,m}, S^O_{p,n}$ の行列で表現する．入力層から中間層へ与えられる信号に掛けられる重みは $W^M_{l,m}$ の行列で表現する．中間層から出力層へ与えられる信号に掛けられる重みは $W^O_{m,n}$ の行列で表現する．シグモイド関数 $f(x) = 1/(1+\epsilon^{-x})$ で行列 M について，$f(M) = M'$ と表現した場合，行列の各要素について関数 f を実行し，その要素を行列 M と同じ順番に列挙したものを行列 M' とする．この記号を利用して教師付パターンの 1 番目について前向き演算を表現すると，

$$S^M_1 = f(W^M \times S^I_1) \tag{1}$$

で中間層の信号値が得られ，

$$S^O_1 = f(W^O \times S^M_1) \tag{2}$$

で出力層の信号値が得られる．

教師付パターンの教師信号は $T_{p,n}$ と表現する．このとき，教師信号とニューラルネットワークから得られた信号の差を $E = T - S^O$ で表現する．ここではパターン 1 のみ考えているので，$T_{1,n}$ のベクトルに対して E を計算する．次にこの差分情報を用いて重みの調整を行なう後向き演算を行なう．詳細は割愛するが，$W^O_{j,k}$ の修正値 $\Delta W^O_{j,k}$ は次のように計算できる．

$$\Delta W^O_{j,k} = \eta \delta^O_k S^M_j \tag{3}$$
$$\delta^O_k = E_k S^O_k (1 - S^O_k)$$

ここで，η は学習速度を表わし，0 より大きく 1 以下を定義する．1 に近いほど学習速度は上るが，振動することで学習が収束しないこともある．一般には 0.25 から 0.75 を指定する．同様に $W^M_{i,j}$ の修正値 $\Delta W^M_{i,j}$ は次のように計算できる．

$$\Delta W^M_{i,j} = \eta \delta^M_j S^I_i \tag{4}$$
$$\delta^M_j = S^M_j (1 - S^M_j) \sum_{k=1}^{n} W^O_{j,k} \delta^O_k$$

上記の前向き演算および後向き演算を繰り返し行なうことにより，教師信号の値に近くなるよう重みが修正されていく．

さて，Sundararajan はで誤差逆伝搬アルゴリズムの様々な並列化および高速化を紹介あるいは提案している [2] が，その中でブロック単位学習を本論文で利用するので，それについても紹介する．ブロック単位学習とは，複数のパターンについて同時に計算をして，計算の省力化を図る方法である．前向き演算 (1) 式，(2) 式は以下の式となる．

$$S^M = f(S^I \times W^M) \tag{5}$$

$$S^O = f(S^M \times W^O) \tag{6}$$

これは簡単に言えば，行列・ベクトル積が行列・行列積に変っただけである．また，後向き演算 (2.) 式，(2.) 式は以下の通りである．上付きの t は転置を表す．

$$\Delta W^O = \eta (S^M)^t \delta^O \tag{7}$$
$$\delta^O_{p,k} = E_{p,k} S^O_{p,k} (1 - S^O_{p,k})$$

(注1): http://math-atlas.sourceforge.net

$$\Delta W^M = \eta (S^I)^t \delta^M \tag{8}$$

$$\delta^M_{p,j} = S^M_{p,j}(1 - S^M_{p,j}) \sum_{k=1}^{n} W^O_{j,k} \delta^O_{p,k}$$

ブロック単位の逐次改善的な方法とは異なる結果となるが，計算量を改善できる他，BLAS(Basic Liner Algebra Subprograms) などの行列演算パッケージを活用することが可能となり，速度の大幅な向上が期待できる．

3. 並列計算のための負荷分散方法

並列計算機には様々なタイプのものが存在するが，本研究で対象とするものは，計算速度が異なる複数の計算機が単一ネットワーク上に接続され，MPI(Message Passing Interface) による並列計算プログラムを実行可能であるようなクラスタ型並列計算機を考えることにする．また，各計算機はベンチマークテストなどによって，行列乗算の相対的速度差が分っているものとする．

Sundararajan [2] は並列計算機のための負荷分散方法についてもいくつか紹介している．その中で彼が述べている節点分割による並列化と辺分割とによる並列化を合わせたような分割を提案する．まず，図 1 のようなニューラルネットワークを分割することを考える．

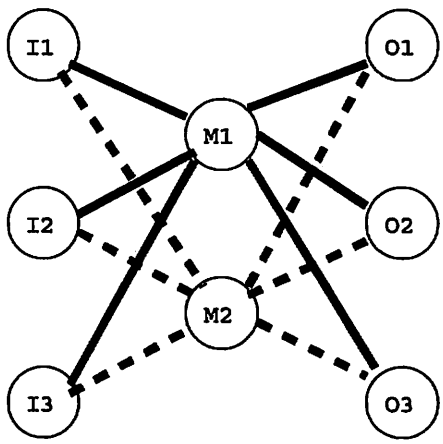

図 1 ニューラルネットワークの分割

節点 M1 に係る辺の処理を全て計算機 1 が実行し，節点 M2 に係る辺の処理を全て計算機 2 が実行すると考える．さて，O1, O2, O3 の節点での計算は，どちらの計算機で行うのか？それは，両方の計算機で同じ計算をさせることにする．また，各出力節点の出力信号を計算するためには互いの計算機が持つ中間層の出力信号を必要とするが，ここは，過去に計算された結果を互いの計算機間で通信しておき，それを活用する．一見無駄に思える計算であるが，出力層の各節点の重みを更新する際に，全ての計算機上で出力層の各節点の出力信号を必要としている．なぜならば，各プロセッサの処理担当である，M1 または M2 に接続されて辺の重みは，それぞれの計算機で処理するためである．出力信号を通信で補うことも考えられるが，大抵場合，通信で補うよりも，各計算機で計算させた方圧倒的に速い．

もう少し具体的に書き下すことにしよう．計算機 1 から計算機 q までの q 台の計算機があり，それぞれを $pr_1, pr_2, \ldots, pr_q$ で表す．各計算機は能力が異なっているが，事前に能力の比は分っている．まずは，各計算機の能力に比例した中間層の節点の分配数を決定する．分配数が決まれば，次に具体的に処理を担当する設定を各計算機に割当てる．これは，行列 S^M，行列 W^M，行列 W^O を分割する作業に他ならない．そこで，pr_h が担当している行列であることが分かるように S^{Mh}, W^{Mh}, W^{Oh} とする．ここまで定まった時点で以下のアルゴリズムを実行する．

Step 1. pr_1 のみで全ての重みをランダムに決定するなど，必要なパラメータを設定する．
Step 2. pr_1 のみでランダムに設定された重みを用いて初期の S^M, S^O を計算する．
Step 3. pr_1 から全ての計算機に対して，W^M, W^O, S^I, T, S^M を送信する．
Step 4. 以下の文を繰り返す．

> Step 4.1. 各 pr で前向き演算をする．ただし，各 pr_h は中間層の出力値について，S^{Mh} のみを計算する．また，S^O を計算する場合は，S^{Mh} および足りない中間層の出力分については，S^M を利用して計算する．
> Step 4.2. 各 pr は各自で計算した S^{Mh} を互いに交換し，S^M を再構成する．
> Step 4.3. pr_1 のみ出力層の出力と教師信号との誤差を調べ，一定以下であれば，Step 5 へ行く．
> Step 4.4. 各 pr で後向き演算をする．ただし，各 pr_h は W^{Oh} および W^{Mh} のみを更新する．
> Step 4.5. 各 pr は各自で計算した W^{Oh} を互いに交換し，W^O を再構成する．

Step 5. pr_1 は全 pr から W^{Oh}, W^{Mh} を回収し，W^O, W^M を再構成する．

ここで，Step 4.1 の動作に注目する．1 回前の前向き演算の結果と，今計算した前向き演算の結果を合成しているのは問題かもしれない．しかし，これのためだけにもう一度 S^M の回収と再構成をすれば，大幅に計算時間が増えてしまう．そこで，我々はこの問題について，2 つの方法で答えることにする．1 つは，実験をすることにより，期待した動作を示すことであるが，これは次節に譲る．もう一つは，中間層の出力と出力層の出力の違いについてである．図 2 は，

誤差逆伝搬アルゴリズムを逐次実行した場合に，各層の重みの修正量を示したものである．縦軸は修正量，横軸は学習回数を表わす．これを見る限り，入力層-中間層間の重み変化が少ないということは，中間層出力値の変化は少なく，中間層-出力層間の重み変化が大きいということは，出力層出力値の変化は大きいということが分かる．つまり，中間層の出力値を 1 回前のものを利用したとしても，大きくは変わらないだろうという期待して良いだろうと考えている．

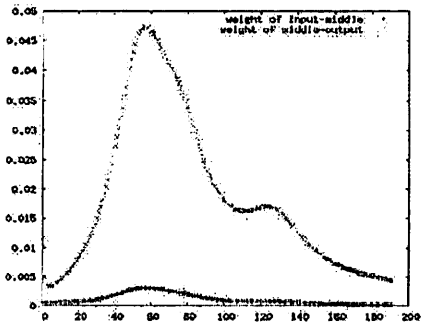

図 2 各層の重み修正量

4. 実　　験

実験では，図 3 の図を記憶させることを行なった．具体的には，図 3 の 5 つの領域において，ある点が指定されたら，それが何番目の領域であるかを答える問題である．この画像を 100×100 の画素に分割し，教師付パターンとして，規則的に取り出した 625 点を用いて学習させることにした．並列計算機環境としては，鈴鹿高専の教育用端末 (CPU:Pentium4/2.8GHz, 主記憶:512MByte) をネットワークブートして利用することにした．今回の実験では最大 16 台の計算機を利用しているが，全て同じ性能である．プログラムは C 言語で作成し，LAM-MPI, Atlas3 のパッケージを利用した．ソースコードは 700 行程度である．学習するニューラルネットワークは入力層が 10000 節点，中間層が 30 節点，出力層が 5 節点である．学習速度を決める η は 0.01, 収束条件は，教師信号との誤差平均が 0.01 以下とした．なお，η は通常よりもかなり低めであるが，これはブロック学習による影響で，一般的な 0.3 程度の大きさでは，正しい解に行き着かなかったので，経験的に定めた．

実験プログラムは 1 台の計算機から実行可能となるように作成したので，1 台から 16 台まで計算機を増やした場合について調査した．

表 1 は実験結果である． Pr は計算機台数，Time は実行時間，Rate は 1 台で実行した場合の実行速度を 1 とした場合の速度向上率，Rip は学習の繰り返し回数，Comm は通信時間に相当するだろう予想時間で，実際には，$Comm = (Pr * Time - 3862)/Pr$ で計算した値である．

まず，注目したいのは，学習の繰り返し回数である．計算機の台数が増えてもほとんど 7100 前後で終了している．これは計算機台数が収束速度に影響を与えないことを示しており，安定して学習できていると言うことができる．速度向上率については，十分かどうか判断できない．今回の実験では問題が軽すぎたのではないかと考えている．特に通信相当時間を見てみると，一部を除いて 500 秒前後かかっていることが分かる．実行時間が 1000 秒前後にまでなっている時に，逐次実行部分が 500 秒も占めていては，これ以上の速度向上は望めない．それでも，学習問題をほとんど行列乗算に帰着させてしまい，並列実行しやすい状況を作り出したので，16 台の計算機の場合においても速度向上しており，問題の規模が大きくなれば速度向上も期待できる．

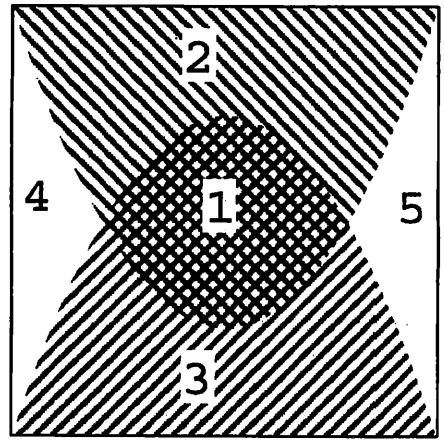

図 3 実験用パターン

表 1 実験結果

Pr.	Time(s)	Rate	# of Rip.	Comm.(s)
1	3862	1.00	7103	0
2	2314	1.67	7096	383
3	1807	2.14	7100	520
4	1431	2.70	7106	466
5	1296	2.98	7105	524
6	1171	3.30	7103	527
7	1079	3.58	7103	527
8	962	4.01	7101	479
9	963	4.01	7097	534
10	957	4.04	7095	571
11	875	4.41	7104	524
12	864	4.47	7103	542
13	866	4.46	7103	569
14	849	4.55	7103	573
15	838	4.61	7099	581
16	744	5.19	7098	503

5. 考　　察

クラスタ型並列計算機のように，ネットワークトポロジの自由度が高い並列計算機では，様々な並列アルゴリズムの速度がさらに向上できる可能性がある．また，一部の計

算の順序を入れ変えたり，省略したりすることで，数学的な保証から外れてしまうかもしれないが，実質的に適切な精度を保ったまま，大幅に速度向上できる可能性がある．以前の計算機と異なり，現在の計算機は主記憶とキャッシュメモリとの速度差が大きく，BLASのような行列乗算パッケージを利用できるかどうかは，実行速度に大きな差を与えることになる．本論文で提案方法においても，ブロック学習を積極的に利用することで，BLASを利用できるようになり，速度向上に寄与している．また，前向き演算時の通信の一部を省略することにより，通信時間の削減や同期待ち時間を削減している．

本論文では，中間層節点の割当てを静的に行なっているが，動的にすることは難しくない．ただし，動的にすると，通信による情報交換の必要がなかった W^M の再構成と分配が必要になる．この量は決して小くないので，どのタイミングで中間層節点の動的再割り当てを行うのかは，慎重に検討する必要があるだろう．

今後は様々な問題規模のものを適用し，アルゴリズムの安定性を確認したい．

文　献

[1] R.Beal, T.Jackson: "ニューラルコンピューティング入門", 海文堂, 1993.

[2] N.Sundararajan, P.Saratchandran: "Parallel Architectures for Artificial Neural Networks", IEEE Computer Society, 1998.