2004－HPC－100　(2)
2004／12／17

非均質環境における誤差逆伝播法の矩形分割によるマッピング手法

小澤 洋司 †　　菅谷 至寛 †　　阿曽 弘具 †

階層型ニューラルネットワークは様々な分野で広く応用されているが、その学習法のひとつである誤差逆伝播法の計算量は膨大である。そのため、処理時間短縮のために並列化の研究が数多く行われている。しかし、各プロセッサの能力が均質な環境を対象としたものが多く、能力が非均質な環境では能力の低い計算機に全体の性能が制限される。本研究では、非均質環境における適切な処理の割当てのために誤差逆伝播法の並列化を矩形分割問題としてモデル化する。各プロセッサが担当する処理を矩形で表すことで、能力に応じた正確な計算の分配と通信時間を、容易に考慮することができる。その上で、通信時間が最小となる分割を選択するシンプルな手法を提案する。また、実験によりその有効性を確認する。

A BP Mapping Method Using a Rectangular Partitioning on Heterogeneous Environment

Yoji Ozawa†　　Yoshihiro Sugaya†　　Hirotomo Aso†

Back propagation (BP), which is one of the neural network training algorithms, can be applied to various application. It requires long processing time, so many researchers have tackled this problem using parallel processing. But most of them aim for homogeneous environment only. On heterogeneous environment the lowest processor restricts whole performance. We consider parallelization of BP as a rectangular partitioning model. The model makes it easy to consider an accurate load-balancing and communication time. We propose a simple method to decide a partition which minimizes commnication time. And we show its efficiency experimentally.

1. はじめに

階層型ニューラルネットワークの 1 つである多層パーセプトロンとその学習法である誤差逆伝播法 (以下 BP 法) は、良い識別性能を持つ識別器を、学習サンプルから比較的容易に構成できる。そのため、様々な分野で広く応用されている。しかし、学習に要する計算量は膨大なものとなるため計算時間の短縮が必要とされている。高速化のアプローチとして、並列化の研究が盛んに行われている [1][2]。BP アルゴリズムには 2 種類の並列性（データ並列性、ノード並列性）が内在しており、これらを適切に組み合わせることが必要である。多くの研究は、各プロセッサの能力が均等な環境を対象としている。しかし、本研究で対象としている計算機クラスタのような並列計算機では、各プロセッサの能力が異なることが大いに考えられる。このような各プロセッサの能力が非均質な環境で、既存の均質環境を対象としたアルゴリズムを用いると、最低の能力のプロセッサに全体の性能が制限されてしまう。これを避けるために、各プロセッサの能力に応じて処理を割り当てる必要がある。しかし、2 種類の並列性に関して非均質性を考慮するのは、非常に困難な問題である。また、並列性の組合せの割合により、全体の性能が変化することが知られている。適切な組合せの割合を決定する手法が提案されているが、正確に求めるためには多くのパラメータを必要とし、手間のかかる作業となる。本手法では BP 法の並列化を矩形分割問題としてモデル化することでこれらの問題に対応する。

以下、本論文では第 2 章で BP 法について、第 3 章で BP 法の並列化手法について述べる。第 4 章で矩形分割によるモデル化、そのモデルを用いたマッピング手法の提案を行う。第 5 章で性能評価を行い、第 6 章でまとめを述べる。

†東北大学大学院工学研究科
Graduate School of Engineering, Tohoku University

2. 誤差逆伝播法（BP 法）

BP 法は多層パーセプトロンにおいて、ある入力パターンから所望のパターンが出力されるように、学習サンプルを用いて結合重みを調整するアルゴリズムである。本研究では 3 層パーセプトロン（図 1）を対象とする。

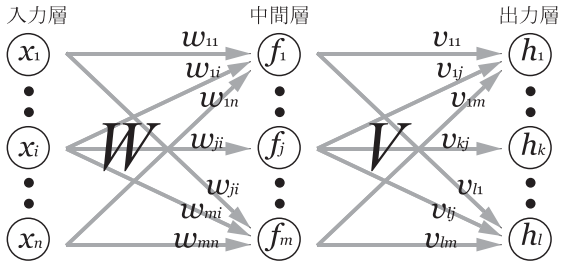


図 1: 3 層パーセプトロン

n、m、l をそれぞれ入力層、中間層、出力層のユニット数とし、入力層と中間層を結合する重み行列を $\boldsymbol{W}(m \times n$ 行列)、中間層と出力層を結合する重み行列を $\boldsymbol{V}(l \times m$ 行列) する。入力を $\boldsymbol{x}$、学習サンプル集合を X（サンプル数 s）とする。

BP 法は Forward フェーズ、Backward フェーズ、Modify フェーズで構成される。Forward フェーズでは、入力されたサンプルに対して中間層、出力層の各々の出力 $\boldsymbol{f}, \boldsymbol{h}$ が順に計算される。

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\sigma}\boldsymbol{W}\boldsymbol{x} \tag{1}$$
$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{\sigma}\boldsymbol{V}\boldsymbol{f}(\boldsymbol{x}) \tag{2}$$

ここで、$\boldsymbol{\sigma}$ はベクトルの各要素に非線型関数 $\sigma(u)$ をかける演算である。ここでは、シグモイド関数 $\sigma(u) = \frac{1}{1+e^{-u}}$ を用いることとする。

Backward フェーズでは、重み更新量 $\Delta\boldsymbol{W}, \Delta\boldsymbol{V}$ を以下の式より計算する。

$$\Delta\boldsymbol{V}(\boldsymbol{x}) = \boldsymbol{\delta}^{(2)}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})^T \tag{3}$$
$$\Delta\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{\delta}^{(1)}(\boldsymbol{x})\boldsymbol{x}^T \tag{4}$$

ここで、$\boldsymbol{\delta}^{(1)}$、$\boldsymbol{\delta}^{(2)}$ は修正誤差と呼ばれ、その各要素は以下のようにして求める。

$$\delta_k^{(2)}(\boldsymbol{x}) = (h_k(\boldsymbol{x}) - d_k(\boldsymbol{x}))h_k(\boldsymbol{x})(1 - h_k(\boldsymbol{x})) \quad (1 \le k \le l) \tag{5}$$

$$\delta_j^{(1)}(\boldsymbol{x}) = f_j(\boldsymbol{x})(1 - f_j(\boldsymbol{x}))\sum_{k=1}^{l} v_{kj}\delta_k^{(2)} \quad (1 \le j \le m) \tag{6}$$

ここで、$\boldsymbol{d}(\boldsymbol{x})$ は教師信号である。

Modify フェーズでは、重み更新量を用いて、重みを更新する。

$$\boldsymbol{V}^{new} = \boldsymbol{V}^{old} - \epsilon \sum_{\boldsymbol{x}\in X} \Delta\boldsymbol{V}(\boldsymbol{x}) \tag{7}$$
$$\boldsymbol{W}^{new} = \boldsymbol{W}^{old} - \epsilon \sum_{\boldsymbol{x}\in X} \Delta\boldsymbol{W}(\boldsymbol{x}) \tag{8}$$

ϵ は学習定数と呼ばれる、微小係数である。

全ての学習サンプルに対して、Forward、Backward フェーズを行った後、一括して Modify フェーズを行う。この一連の計算をイタレーションと呼び、誤差が閾値以下になるまで繰り返す。

3. 誤差逆伝播法の並列化

BP 法に内在するデータ並列性とノード並列性による基本的な並列化手法 [1] について述べる。さらに、2 種類の並列化手法を組み合わせた手法について述べる。

3.1 データ並列

学習サンプル集合 X（学習サンプル数 s）を分割して、その部分集合を各プロセッサに割り当てる。各プロセッサは割り当てられた学習サンプル（プロセッサ P_i のサンプル数 s_i）について重み更新量の部分和を求める。各プロセッサが保持する部分和の総和が重み更新量となるので、通信を行い部分和の総和を求める。その結果を各プロセッサに配り直す。

重み更新量 $\Delta\boldsymbol{V}, \Delta\boldsymbol{W}$ を通信するので、データ並列による通信要素数は $(n + l)m$ となる。

3.2 ノード並列

重み行列を分割し、重み更新量の計算を並列化する [3]。重み行列 $\boldsymbol{W}(m \times n$ 行列) を**列方向**に、重み行列 $\boldsymbol{V}(l \times m$ 行列) を**行方向**に分割する。つまり、ノード並列は中間層のユニット数 m を分割することに相当する。プロセッサ数が 3 の場合を以下に示す。

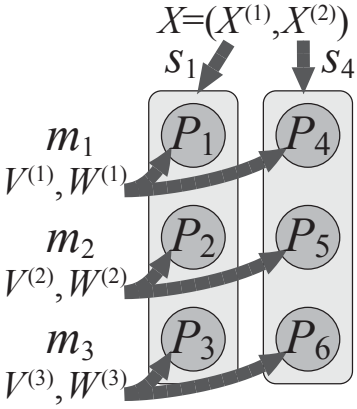
$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}^{(1)} \\ \hline \boldsymbol{W}^{(2)} \\ \hline \boldsymbol{W}^{(3)} \end{pmatrix}, \quad \boldsymbol{V} = \left(\begin{array}{c|c|c} \boldsymbol{V}^{(1)} & \boldsymbol{V}^{(2)} & \boldsymbol{V}^{(3)} \end{array} \right)$$

このように分割することで、プロセッサ間通信は Forward フェーズ計算時の 1 度のみとすることができる。通信要素数は ls となる。

3.3 データ並列とノード並列の組合せ

2 種類の並列化手法は組み合わせることができる（図 2）。幾つかのプロセッサからなる並列グループを作成し、分割した学習サンプル集合を割り当てる。（並列グループの数を N_{DP} とし、データ並列度と呼ぶ。）そして、各並列グループ内でノード並列を行う。プロセッサ P_i に割り当てる学習サンプル数を s_i、分割された重み行列の中間層のユニット数を m_i とする。プロセッサ P_i の BP 法の計算量は表 1 のようになる。ls_i は全体の計算量に比べて小さいため、P_i の計算量は $m_i s_i$ にほぼ比例するとみなせる。

ノード並列による通信は並列グループ内のプロセッサ間通信となる。データ並列による通信は、各並列グループで求めた重み更新量の部分和を通信するため、異なる並列グループのプロセッサ間の通信となる。つまり、重み行列の同じ部分を保持しているプロセッサ間での通信となる。



プロセッサ数 N=6　データ並列度 $N_{DP} = 2$ の場合

図 2: 2 種類の並列化手法の組合せ

4. マッピング手法

各プロセッサの能力の非均質性を考慮したマッピング手法を提案する。まず、BP 法の並列化を矩形分割問題としてモデル化する。これにより、計算量と通信時間を容易に考えることができる。次に、非均質性を考慮した過去の並列化手法を矩形分割モデルを用いて再考し、その問題点を明らかにする。計算量を適切に割り当て、通信時間を最小にする分割が最適となるが、最適な分割を選択するのは非常に困難な問題となる。従って、幾つかの条件を加えた分割方法を提案する。そして、その分割の中で最適な分割を選択するアルゴリズムを提案する。

4.1 矩形分割問題へモデル化

非均質環境では、プロセッサの能力に応じて割り当てる処理の量を変える必要がある。

表 1: 各プロセッサにおける BP 法の計算量

	加減算	乗除算	シグモイド演算
Forward フェーズ	$(n+l-1)m_is_i - ls_i$	$(n+l)m_is_i$	$m_is_i + ls_i$
Backward, Modify フェーズ	$(n+l)m_is_i + 2ls_i$	$(n+2l+1)m_is_i + 2ls_i$	—

n:入力層のユニット数, l:出力層のユニット数, m_i:プロセッサ P_i に割り当てられた中間層のユニット数, s_i:プロセッサ P_i に割り当てられた学習サンプル数

Beaumont ら [6] は、非均質環境における行列乗算の並列化を矩形分割を用いて行った。本研究ではこの矩形分割の考え方を BP 法の並列化に応用する。3.3 節で述べたようにプロセッサ P_i の計算量が m_is_i に比例することから、P_i の計算量は $m_i \times s_i$ の矩形で表現でき、BP 法の並列化を矩形分割問題としてモデル化することができる。これにより、データ並列とノード並列を両方考慮した適切な割当てを容易に考えることができる。

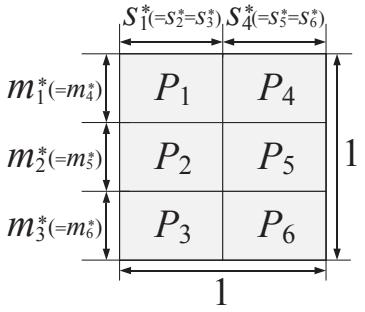


図 3: 矩形分割問題にモデル化

図 3 はプロセッサ数 $N = 6$、データ並列度 $N_{DP} = 2$ の時の矩形分割である。全計算量を表す単位正方形を、1 つのプロセッサに割り当てられる処理の計算量を表す矩形に分割する。プロセッサ P_i の処理を表す矩形の幅を s_i^*、高さを m_i^* とする（$s_i = s_i^* s$、$m_i = m_i^* m$ となる）。このモデル化により各プロセッサの学習サンプルと重み行列の割当てが表現できる。また、図 3 のようにプロセッサを並べる順番を指定する。

矩形の面積 $m_j^* s_i^*$ が計算量にほぼ比例するので、各プロセッサの能力に応じて、矩形の面積を変えることで正確な負荷分散を行うことができる。

プロセッサ P_i のデータ並列に関する通信の要素数は $(n+l)m_i$、ノード並列に関する通信の要素数は ls_i であり、通信量は s_i と m_i により決まる。つまり、矩形の幅と高さ（形状と呼ぶ）で通信量を考慮することができる。

従って、このモデルを考えることで各プロセッサへの計算量と通信量、通信を行うプロセッサのグループを考えることができる。

4.2 過去の研究

非均質環境に対応した過去の研究 [3][4][5] について考える。これらの問題点として、2 つあげられる。まず、矩形分割の視点で考えると、処理を割り当てる時に格子を維持している点である。つまり、計算量を表す矩形の面積は正確に各プロセッサの能力に対応していない。従って、各プロセッサの能力を十分に活かすことができない。

また、プロセッサ全体に対して可能なデータ並列度は数通り存在する。これは、データ並列とノード並列の組合せ方が複数考えられることを意味している（図 4:プロセッサ数が 4 の時、$N_{DP} = 1, 2, 4$）。ニューラルネットワーク (NN) のサイズ、学習サンプル数などの条件により、それぞれ処理時間が異なるため、最適なデータ並列度を選択する必要がある。過去の研究 [3][4] において、並列度選択手法が提案されている。しかし、正しく選択するためには計算能力や通信性能等、多くのパラメータが必要となり、それらのパラメータを事前に正しく測定するのは手間がかかる作業となる。

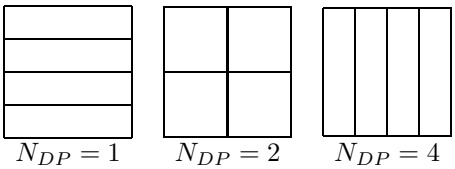


図 4: プロセッサ数 $N = 4$ の時の分割の仕方

4.3 分割方法

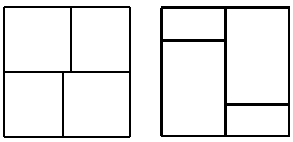
本研究では矩形分割モデルを利用し、処理の割り当てを次のような考えのもとで決める。

- 面積：面積はプロセッサの能力に比例するため、各プロセッサの能力により決める。プロセッサ P_i の能力を $p_i(\sum_{i=1}^{N} p_i = 1)$ とし、矩形の面積 $m_i^* s_i^* = p_i$ とする。これにより、計算を適切に分配でき、各プロセッサの能力を十分に利用することができる。
- 矩形の形状と配置：矩形の形状（幅 s_i^*、高さ m_i^*）と配置を決めることで通信時間を概算することができる。（概算の方法については後で詳しく述べる。）通信時間が最小のものが最良の分割であると考え、通信時間が最小になる形状と配置にする。

矩形の面積、形状と配置を決めることにより、分割が決まる。しかし、分割の仕方は非常に多数存在し、その中から上記の 2 つの条件を満たす分割を選択するのは、困難である。従って、本研究ではさらに以下の条件を加え、その中で通信時間が最小となる最適なものを選択する。

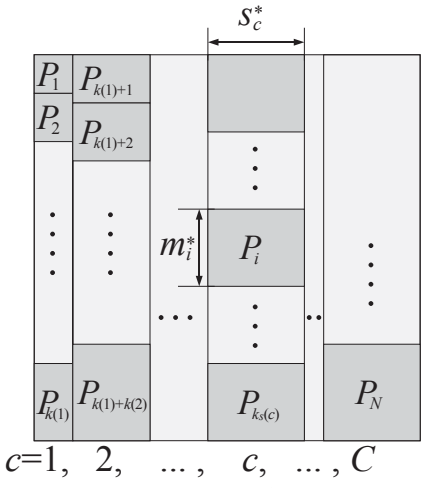
- 列を固定する。図 5（右）で示すように、垂直線は途中で曲がることはなく、直線である。
- 列毎に能力の低いプロセッサから並べる。

これらの条件を加えると、矩形分割の仕方は図 6 のようになる。ここで、列数を C、第 c 列に割り当てら



列が固定されていない分割（左）、
固定されている分割（右）

図 5: 列を固定した分割



ここで、$k_s(c) = \sum_{j=1}^{c} k(c)$

図 6: 分割方法

れたプロセッサ数（垂直方向の分割数）を $k(c)$ とする。第 c 列の幅 s_c^* は $k(c)$ 台のプロセッサの総能力とする。すなわち、s_c^* は式 (9) のようになる。

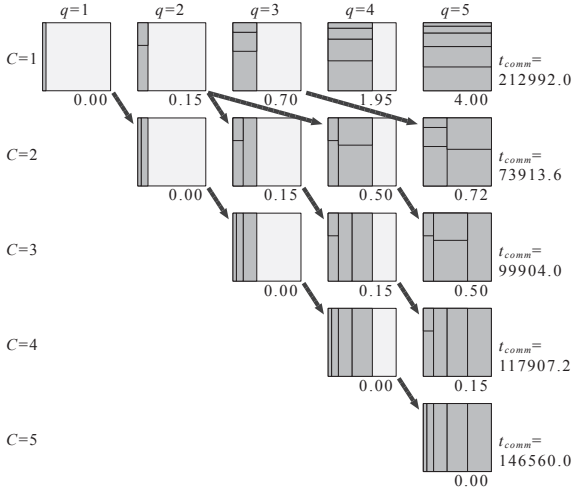
$$s_c^* = \sum_{i=i_c+1}^{i_c+k(c)} p_i \ , \quad i_c = \sum_{j=1}^{c-1} k(j) \tag{9}$$

この分割をしたときの、通信時間は式 (10) で概算できる。

$$t_{comm} = 2ls \cdot \max_c(s_c^*(k(c)-1)) + 2(l+n)m(C-1) \tag{10}$$

第 1 項は垂直方向の通信時間を表している。垂直方向の通信はノード並列に関する通信であり、同じ列の中（並列グループ内）のプロセッサが出力層の値を求めるための値を共有する通信である。この通信は、1 つのプロセッサに値を集め、足し合わせた後、その結果をブロードキャストする。本研究ではプロセッサ間通信は、1 対 1 通信で行っている。つまり、1 列に $k(c)$ 台のプロセッサがあるので、通信回数は $2(k(c)-1)$ となる。また、列幅は固定されており、各列は同時に通信をすることが可能であるため、通信時間は列の中で最大のものとなる。従って、通信回数 $2(k(c)-1)$ に通信要素数 $ls \cdot s_c^*$ を乗じたものが各列の通信時間となり、その中で最大のものを垂直方向の通信時間としている。

第 2 項は水平方向の通信時間を表している。水平方向の通信は同時に行えない場合があるので、通信要素数は $(l+n)m$ とする。水平方向の通信を行うプロセッサ数は C 台であるので、第 1 項の場合と同様に考え、通信時間は $2(l+n)m(C-1)$ となる。



プロセッサ数 $N = 5$、プロセッサの能力 (0.05, 0.10, 0.20, 0.30, 0.35)、NN サイズ n-m-l は 203-80-26、学習サンプル数 $s = 1024$ の場合の最適な分割を計算する過程を示している。各正方形の右下の値は $t_C(q)$ であり、右端に t_{comm} の値を記した。また、矢印の始点の分割に新しい列を加えた場合が $t_C(q)$ が最小になることを表している。

図 7: 分割決定アルゴリズム

この通信時間 t_{comm} が最小となる C と $\{k(c)\}$ を求め、最適な分割として、採用する。

4.4 分割決定アルゴリズム

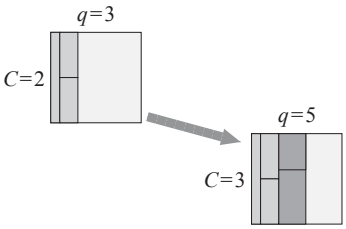
4.3 節で述べた分割の中で通信時間 t_{comm} が最小になる分割を選択する。つまり、何列目に幾つプロセッサを割り当てればよいかを考える。分割を決定するアルゴリズムは、動的計画法を用いる。C を列数、q を割当て済みプロセッサ数とする。

図 7 のように、各 C、q で通信時間が最小となる分割を決めていく。この時、式 (10) の第 1 項に相当する $t_C(q)$（式 (11)）を計算し、保持する。式 (10) の第 2 項は q には依存しないため、$q = 1, 2, \ldots, N-1$ では、第 1 項に相当する値を計算すれば十分である。

また、t_{comm} の第 1 項は垂直方向の通信を表しており、各列の最大値である。従って、$t_C(q)$ は以下のようにして計算する。q' 台のプロセッサが列数 $C-1$ に分割されているとして、新しい列 C を加える。その列には、$k(C) = q - q'$ 台のプロセッサを割り当てる。つまり、垂直方向に $q - q'$ 分割する。その新しい列に関する通信時間は $t_{0C} = (\sum_{i=q'+1}^{q} p_i)(q - q' - 1)$ である。t_{0C} と $t_{C-1}(q')$ の大きい方がその分割での垂直方向の通信時間となる。図 8 に $C = 3, q = 5, q' = 3$ の時の例を示す。そして、各 q' $(q' = C-1, \ldots, q-1)$ において、同様にそれぞれの分割をしたときの通信時間を求め、その中で最小となるものを $t_C(q)$ とする。

全て計算した後、$t_C(N)$ の値を用いて、t_{comm} を計算し、その中で最小となる分割を決定する。

$$\begin{aligned} t_C(q) &= \max_{c \le C}(s_c^*(k(c)-1)) \\ &= \min_{q'}\{\max\{t_{0C}, t_{C-1}(q')\}\} \end{aligned} \tag{11}$$



第 3 列には $q-q'$=5−3=2 台のプロセッサを割り当てる。

図 8: $t_C(q)$ の求め方

より具体的なアルゴリズムは以下の通りである。

1. プロセッサを能力 p_i に関して昇順に並べる。
2. $t_C(q)$ を順に求める。

for $C = 1$ to N
 for $q = C$ to N

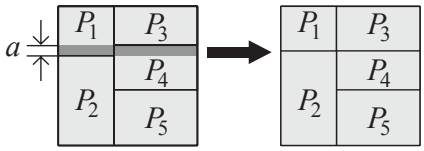
$$t_C(q) = \min_{r\in[C-1,q-1]}\left(\max\left((\sum_{i=r+1}^{q} p_i)(q-r), t_{C-1}(q)\right)\right)$$

 end for
end for

3. $t_{comm}(C) = ls \cdot t_C(N) + (l+n)m(C-1)$ を $C = 1, 2, \ldots, N$ について計算し、最小となる時の分割を選択する。

4.5 小さな通信グループの削除

水平方向の通信を行うプロセッサの組を通信グループと呼ぶ。例えば、図 9(a) のように分割した場合、通信グループは 4 つある。このとき、プロセッサ 1 はプロセッサ 3 とプロセッサ 4 と通信をするが、プロセッサ 4 との通信要素数（高さ）は小さい。このような小さな通信グループができることが考えられる。この時、図 9(b) のように、1 列目と 2 列目で分割する位置を同じにする。こうすることで、通信グループが 3 つに減るため通信に伴うオーバーヘッドが削減でき、全体の性能は向上すると考えられる。この小ささの判定閾値を 0.06 とし、$a \leq 0.06$ の時、通信グループの削除を行う。



(a) 削除前　(b) 削除後

図 9: 小さな通信グループの削除

5. 性能評価

提案手法を計算機クラスタ上に実装し、性能を評価する。

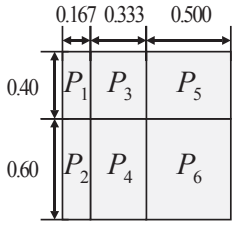
5.1 従来手法

本手法と比較する手法として、過去の研究 [3] を改良したものを用いる。

過去の研究では、データ並列に関してのみ非均質性を考慮し、割り当てを変更していた。今回はノード並列に関しても同様に非均質性を考慮する。つまり、以下のように分割、及び割り当てを行う。

- 矩形分割の格子は維持。（列、行共に固定）
- 列毎に能力の低い順にプロセッサを並べる。
- 学習サンプル数 s の分割比（データ並列）: 各並列グループ内で最低のプロセッサの能力比
- 中間層ユニット数 m の分割比（ノード並列）: 最低の並列グループ内でのプロセッサの能力比

図 10 に従来手法の分割の例を示す。



プロセッサ数 $N = 6$、データ並列度 $N_{DP} = 3$
各プロセッサの能力比 (1, 1.5, 2, 2.5, 3, 3.5)

図 10: 従来手法の分割例

5.2 実験条件

実験条件は表 2 の通りである。BP 法で扱うタスクは NETtalk[7] とした。

表 2: 実験条件

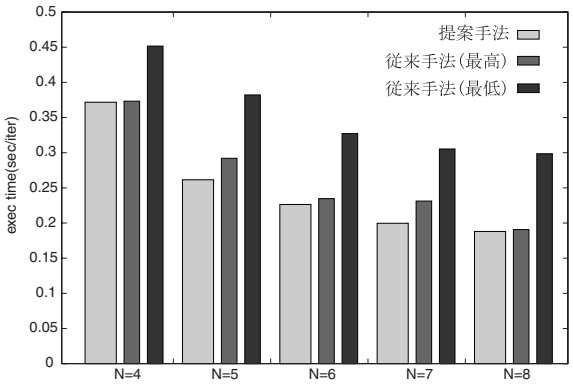
CPU	Intel Pentium III 800MHz,500MHz
OS	Linux 2.4
通信ライブラリ	LAM/MPI version7.0
NIC	1000base-T
NN サイズ	*n-m-l*: 203-80-26
学習サンプル数	$s = 1024$

2 種類の非均質環境で実験を行った。2 種類のプロセッサ（800,500MHz）を用い、さらに、複数のジョブをバックグラウンドで実行し、見かけの能力を落とすことで非均質環境とした。

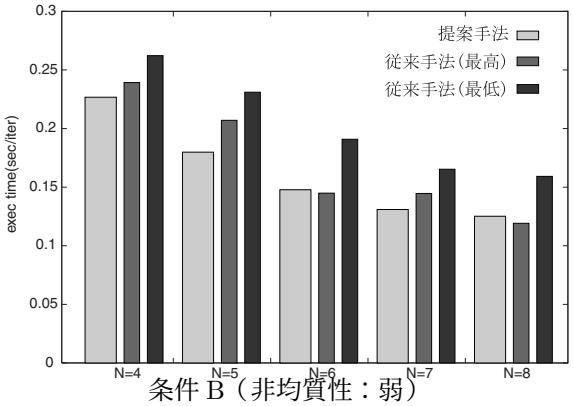
各プロセッサの能力比は次のようになっている。条件 A（非均質性：強/ 0.25, 0.31, 0.63, 1.0, 1.0, 0.42, 0.67, 0.63）。条件 B（非均質性：弱/ 0.63, 0.63, 0.63, 1.0, 0.63, 1.0, 1.0, 0.63）。プロセッサは 4 〜 8 台使用した。8 台未満の場合は、左から順に使用する。つまり、条件 A でプロセッサ数が 5 台の時に使用するプロセッサの能力は 0.25, 0.31, 0.63, 1.0, 1.0 である。

5.3 実験結果

実験結果を図 11 に示す。横軸がプロセッサ数 N、縦軸がイタレーション毎の処理時間である。各プロセッサ数において左が提案手法、右 2 つが従来手法である。従来手法は 1 つのプロセッサ数において、データ並列度が数通り存在し、性能が異なる。従って、図 11 で真中の値は最高の性能となった処理時間、右の値は最低の性能となった処理時間である。



条件 A（非均質性：強）



条件 B（非均質性：弱）

図 11: 実験結果

ほとんどの場合で、提案手法は従来手法と同等の性能か、それ以上の性能となった。

同等の性能となった原因を考える（条件 A/$N = 4, 6, 8$、条件 B/$N = 6$）。提案手法は従来手法よりも、計算量の割当ては能力を正確に反映しているため、それだけ性能が向上することが期待される。しかし、同等の性能に留まってしまったのは、矩形分割における行が固定されていないために通信グループが増加し通信回数が増えたこと、計算機の能力比を正確に求めることができていなかった可能性があることなどが考えられる。

しかし、従来手法は図 11 で示したようにデータ並列度により大きく性能が変わるため、最適なデータ並列度を選択する必要がある。一方、提案手法は割当てが一意に決まる。従って、同等の性能であっても、提案手法の方が有効な手法であると言える。

さらに、$N = 5, 7$ で特に提案手法の性能がよい。これは従来手法において、データ並列度は 1 か N しかとれないためであると考えられる。学習サンプル数 (s) が大きい時は、データ並列度が大きい方が通信量が小さくなるので、全体の性能は上がる。一方、NN サイズ (m) が大きい時は、データ並列度が小さい方が全体の性能が上がる。つまり、データ並列度が 1 か N しかとれないと、極端な条件の場合にしか対応できない。提案手法ではこういった場合にも柔軟に対応できるため、性能が従来手法に比べ、大幅に勝っている。実際、条件 A,$N = 5$ の時の分割は図 12 のようになっており、データ並列とノード並列の両並列化手法を有効に用いることができる分割となっている。

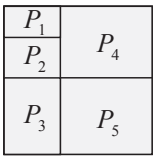


図 12: 分割の様子（条件 A、$N = 5$）

6. まとめ

プロセッサの能力の非均質性を考慮した BP 法の割当て手法を提案した。本手法では、BP 法の並列化を矩形分割によってモデル化することで、プロセッサの能力に応じた処理の分配と通信時間を容易に考慮することが可能となった。そして、通信時間が最小になる分割を最適な分割と考え、その分割を選択するアルゴリズムを提案した。このアルゴリズムが必要とするパラメータは各プロセッサの能力のみであり、非常にシンプルな手法である。また、実際に計算機クラスタ上に提案手法を実装し、本手法の有効性を確認した。

参考文献

[1] A.Singer : Implementation of Artificial Neural Network on the Connection Machine, *Parallel Computing*, Vol.14, pp.305-315, 1990

[2] Rafic A. Ayoubi, Magdy A. Bayoumi: Efficient Mapping Algorithm of Multilayer neural Network on Torus Architecture, *IEEE Transactions on Parallel and Distributed Systems*, Vol.14, No.9, pp.932-943, 2003

[3] 東大亮, 菅谷至寛, 阿曽弘具 : 非均一な並列計算機環境におけるニューラルネットワークの学習, 情処研報 HPC-89-1, pp.1-6, 2002

[4] 小澤洋司, 菅谷至寛, 阿曽弘具 : 非均質環境を考慮した誤差逆伝播法のノード並列アルゴリズム, 情報技術レターズ, pp.5-6, 2003

[5] R. Andonie, A.T. Chronopoulos, D. Grosu, H. Galmeanu: Distributed Backpropagation Neural Networks on a PVM Heterogeneou System, *Proc. Parallel and Distributed Computing and Systems Conf.*, pp.555-560, 1998

[6] O.Beaumont, V.Boudet, F.Rastello, Y.Robert: Matrix Multiplication on Heterogeneous Platforms, *IEEE Transactions on Parallel and Distributed Systems*, Vol.12, No.10, pp.1033-1051, 2001

[7] Terrence J. Sejnowski and Charles R. Rosenberg: Parallel networks that learn to pronounce English text, *Complex Systems*, 1:pp.145-168,1987