5K-1

対象のニューラルネットに合わせたプロセッサ自動配置する並列学習アルゴリズム

佐々木 健吾　　渥美 清隆
鈴鹿工業高等専門学校

1. はじめに

現在主流のニューラルネット学習法[1]である誤差逆伝搬法は非常に多くの繰り返し学習を必要とする．そのため学習終了までに非常に多くの時間がかかる．文献[3]では並列アルゴリズムを提案しているが，プロセッサ数の増大により学習時間が短縮できないことが示されている．そこで私達は大規模なクラスタ型並列計算機を用いてニューラルネット学習の高速化を目指している．ここで各クラスタは1ノードに1プロセッサがインストールされ，全てGbEで接続されているようなクラスタシステムを仮定する．プロセッサのニューロンへの配置方法はニューラルネットの構造によって大きく異なる．また，各プロセッサの性能が全て均一とも限らないので全てのノードに均一な負荷が最適とも限らない．そこで私達はニューラルネット学習にかかるプロセッサ自動配置も提案する．

2. 誤差逆伝搬法の並列化における問題点

誤差逆伝搬法においてニューラルネットワークを構成するユニットは入力を与えられたら出力を返し，出力が逆伝搬して来たら入力重みを変更するという動作を繰り返すだけであり，動作は非常に単純である．もし，OpenMPで記述すれば簡単に並列化できるが，それでは大規模な並列化は出来ない．なぜならOpenMPは共有メモリ型並列計算機を想定していて，プロセッサ数の大幅な増加は難しいからである．そこで本研究ではMPI[2]を用いることにした．

また，並列化を行うとき，用意できるプロセッサの性能やメモリーへのデータ転送の性能等が均一とは限らず，それぞれのプロセッサにかかっている負荷が均一とも限らない．だから各プロセッサに均一な仕事を与amazonえたとしても同じ時間では処理できない．もし，その後に同期をとる通信をする必要があれば早く終ったプロセッサを待たせる事になってしまいプロセッサの使用率を下げる事になる．

3. 並列化の方法

3.1 アルゴリズム

先程も述べたようにニューラルネットワークの学習にかかる各ユニットの計算は非常に単純である．だから各ユニットにプロセッサを与えて，それらが計算したものを毎回通信していると今のプロセッサの計算速度とプロセッサ間の通信速度を考慮したら確実に遅くなる．

そこで私達は入力−中間層の重みの一度当たりの変化が中間−出力層の重みに比べて少ないことに注目して入力−中間層の重みの通信を制限し並列化することで高速化を目指した．通信しない間は更新前のデータを使用する．たとえば図1のように白と横線のユニットを二つのプロセッサで同時に学習する時，点線の重みは通信するまで更新前のものを使う．

これは，オリジナルの方法[1]とは異なるので結果が正しくなる保証はない．正しい方法と比べるた時にどれくらい誤差がでてくるのかも検討課題である．

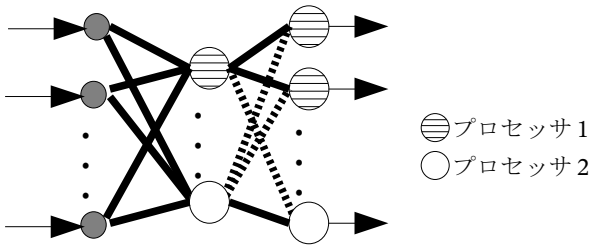


図1 プロセッサの配置とプロセッサ間通信

3.2 プロセッサの自動配置

先程も述べたように並列化をする時，常にノードの性能が均一とは限らず，またそれぞれに同じ負荷がかかっているとも限らない．そこで私達はプロセッサの性能や負荷状況を考慮して，各プロセッサに与える計算の量を調整することとした．性能，初期の負荷状況に基づいて各プロセッサに負荷を分割し，途中で大きく負荷状況が変化した時にも使用する．

4. 実験

4.1 概要

図4(a)のような曲線によって分類された二つの

領域を判断するようにニューラルネットに学習させ，その学習が並列化したものとしないものでどれくらい速度と精度に差が出るのかを調べた．

4.2 ニューラルネットの構造

図 2 のように領域を x 方向に 100, y 方向に 100 づつ合計で 10000 の領域に分解し，そこが 1 か 0 かを入力にする．つまり入力のニューロンは 10000 存在する．中間のニューロンは曲線で分類された二つの領域を判別するということを考えて 15 とした．出力はその座標が 1 ならば「10」, 0 ならば「01」としたので，出力のニューロンを 2 とした．

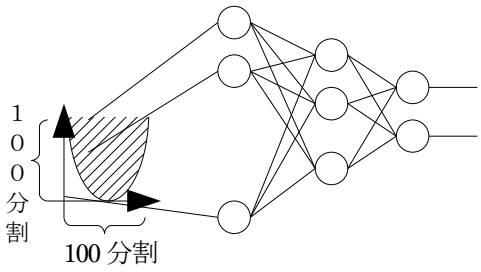


図 2 ニューラルネットワークの構造

4.3 学習の方法

図 3 の 10000 に分解された領域の入力の値が 1 であるところが斜線部ならば 10 を白い領域ならば 01 を出力するという規則を元にして学習セットを作る．X 方向について 4 点に 1 つ,Y 方向について 4 点に 1 つずつ規則正しくとった，全体の 1/16 である 625 の点が学習セットである．

ただし，ニューラルネットの汎化能力を生かすために，入力の 1 の周りに図 4 のようなぼかしを加えた．

			0.25			
		0.25	0.5	0.25		
	0.25	0.5	1	0.5	0.25	
		0.25	0.5	0.25		
			0.25			

図 3 1 の周りのぼかしとして加える入力

4.4 自動化の方法

それぞれのプロセッサに処理時間(集団通信をするための待ち時間を除く)を計算させてそれの逆数に比例させてニューロンを割り付けた．アルゴリズムには DDA(Digital Differential Analyzer)を用いた．

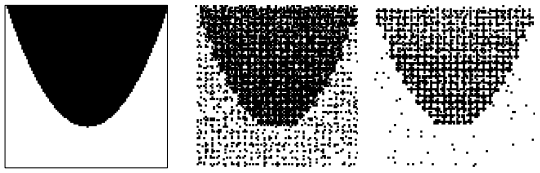
4.5 実験結果

2 台の Pentium4/2.4GHz の計算機を使った実験結果を表 1 に示す．これによると並列化したものの方が速度が速くなり誤りが増えていることがわかった．ただし並列度 4,8 は環境が用意できなかったので，誤り率を仮想的な環境で測定しただけである．図 4(b)，図 4(c)は実際の学習結果である．

表 1 並列度の違いによる誤り率や速度の変化

並列度	*誤り率*	*学習時間(s)*	*速度向上率*
1	0.21	215	1
2	0.30	122	1.76
4	0.31	–	–
8	0.26	–	–

※速度向上率は学習時間の短縮割合を示す



(a)サンプル (b)並列度 1 (c)並列度 2

図 4 学習の結果

5 考察

乱暴な計算ではあるが，並列化有りの学習時間が 122 秒で無しが 215 秒なので，2CPU で並列化したならば速度は倍になるはずなのに 215 秒の半分に比べると 14.5 秒の差がある．それが通信時間であると考えられる．

誤り率は並列化すればするほど増えていくというわけではなく，一定の値の範囲に収束する可能性がある．

参考文献

[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] 青山 幸也, “並列プログラミング入門 MPI 版”, 理化学研究所, 1995.
[3] 山本 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”, ハイパフォーマンスコンピューティング, 1997.