Proceedings of the IV Brazilian Conference on Neural Networks - IV Congresso Brasileiro de Redes Neurais
pp. 231-236, July 20-22, 1999 - ITA, Sãp José dos Campos - SP - Brazil

Paralelização do Algoritmo "Backpropagation" em Clusters de Estações de Trabalho

Wagner Melcíades [1], Mario Fiallos [1], Carlos Pimentel [1]
Depto. de Engenharia Elétrica
Universidade Federal do Ceará
Caixa Postal 6001,
Campus do Pici, s/n, Cep 60455-760
Fortaleza, Ceará, Brasil, 1999
E-mails: wagner, mario, pimentel@dee.ufc.br

Abstract

This work describes the parallelization of the algorithm of learning "backpropagation" in clusters of workstations. The studied modes are: online, batch e block [8]. The experimental results show that the parallelization in the batch mode is the best alternative.

1. Introdução

O algoritmo do backpropagation é um dos métodos mais utilizados para treinamento de redes neurais artificiais (RNA) [11]. Infelizmente em muitas situações, seu tempo de treinamento se torna inaceitável, mesmo com a utilização de PCs e estações de trabalho poderosos.

Para tentar contornar este problema muitos estudos tem sido realizados com o objetivo de utilizar máquinas paralelas e/ou clusters de computadores para acelerar os [1]cálculos envolvidos no algoritmo do backpropagation [1] e [4].

De fato, ao longo dos últimos anos, clusters de estações de trabalho e de PCs tem mostrado também sua efetividade na paralelização de outros tipos de aplicações em Ciência e Engenharia [13].

Os clusters de computadores representam hoje em dia uma alternativa bastante consolidada e espera-se que este tipo de máquina seja uma das plataformas mais utilizadas na aceleração de cálculos [13].

A programação desses clusters pode ser realizada com várias ferramentas entre as quais PVM (*Parallel Virtual Machine*) [6] e MPI (*Message Passing Interface*) [12].

Desta forma é possível se construir um ambiente paralelo virtual constituído de máquinas heterogêneas ligadas em rede. O algoritmo pode ser descomposto em várias partes a serem executadas simultaneamente em diferentes processadores.

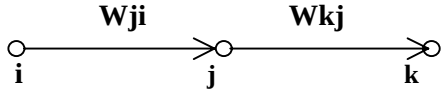
Este artigo está organizado da seguinte forma: na seção 2 apresentamos sucintamente 3 diferentes modos de atualização dos pesos da rede neural, isto é: modo online, modo batch e modo block. O PVM é descrito na seção 3. A seção 4 descreve a paralelização dos algoritmos. Os resultados experimentais e as conclusões do trabalho são apresentados nas seções 5 e 6 respectivamente.

2. Variantes do Backpropagation

As três variações do algoritmo do backpropagation discutidas neste artigo são: modo online, modo batch e modo block [10]. A seguir descreveremos cada um destes modos.

2.1. Backpropagation em Modo Online

Uma descrição completa do algoritmo pode ser encontrada na referência [11]. As relações de precedência e de atualização dos pesos entre os diferentes neurônios são mostradas de forma simplificada na figura 1.



$$Wji(n+1) = Wji(n) + \Delta Wji(n)$$
$$Wkj(n+1) = Wkj(n) + \Delta Wkj(n)$$

Figura 1: Relações de precedência e atualização de pesos entre neurônios de diferentes camadas

[1] Apoiado pelo projeto CNPq PROTEM ICOM

O algoritmo pode ser resumido da seguinte forma:

Algoritmo 1: Treinamento no modo online (resumo)

Inicialização
Repita
Para *cada par de vetores no conjunto de treinamento* ***Faça***
1. aplique à entrada da RNA, o próximo vetor de entrada
2. calcule a saída obtida
3. calcule o erro entre a saída obtida da RNA e a saída desejada
4. calcule Δw para toda a RNA [11]
5. ajuste os pesos da rede
Fim para
Enquanto *o erro estiver fora da tolerância desejada*

2.2. Backpropagation em Modo Batch

No modo batch a atualização dos pesos ocorre apenas quando todo o conjunto de treinamento é apresentado à rede. Isto é, a atualização é realizada apenas uma vez a cada *N* iterações, sendo *N* o número de pares do conjunto de treinamento. Desta maneira o algoritmo pode ser resumido da seguinte forma:

Algoritmo 2: Treinamento no modo batch (resumo)

Inicialização
Repita
Para *cada par de vetores no conjunto de treinamento* ***Faça***
1. aplique à entrada da RNA, o próximo vetor de entrada
2. calcule a saída obtida
3. calcule o erro entre a saída obtida da RNA e a saída desejada
4. calcule Δw para toda a RNA acumulando os resultados
Fim para
5. ajuste os pesos da rede
Enquanto *o erro estiver fora da tolerância desejada*

2.3. Backpropagation em Modo Block

No modo block a atualização dos pesos será realizada uma vez em cada bloco de *M* amostras ou pares do conjunto de treinamento, sendo $1 < M < N$. O algoritmo pode ser resumido da seguinte forma:

Algoritmo 3: Treinamento no modo block (resumo)

Inicialização
Repita
Para *cada bloco do conjunto de treinamento* ***Faça***
Para *cada par de vetores no bloco do conjunto de treinamento* ***Faça***
1. aplique à entrada da RNA, o próximo vetor de entrada
2. calcule a saída obtida
3.calcule o erro entre a saída obtida da RNA e a saída desejada
4. calcule Δw para toda a RNA acumulando os resultados.
Fim para
5. ajuste os pesos da rede
Fim para
Enquanto *o erro estiver fora da tolerância desejada*

3. PVM

O PVM (*Parallel Virtual Machine)* facilita tanto a criação quanto a coordenação de aplicações paralelas/distribuídas para uma grande diversidade de arquiteturas de computadores. Existem versões do PVM para UNIX [6] e Windows [14].

É de responsabilidade do usuário, no entanto, a escolha das máquinas que compõem este ambiente, bem como a criação e a sincronização dos processos.

3.1. Componentes do PVM

O PVM possui dois componentes básicos:

- Os *daemons (pvmd3)* que realizam as funções de comunicação e sincronização entre os processos.
- Uma biblioteca de funções.

A figura 2 mostra uma divisão de tarefas seguindo o modelo de programação mestre-escravo [6]. O processo mestre é responsável pela criação dos demais processos. Cabe ao mestre também realizar a divisão de tarefas entre os escravos.

Estes processos podem rodar num mesmo computador ou em computadores diferentes interligados por algum meio de comunicação.

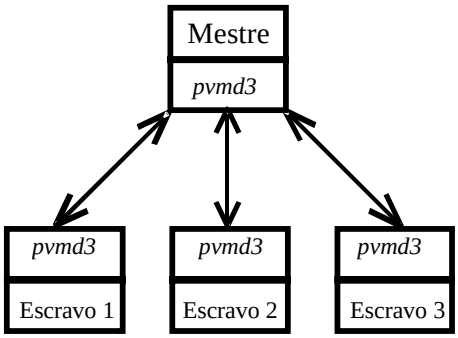


Figura 2: Implementação do modelo mestre-escravo no PVM

4. Paralelização

4.1. Metodologia

A metodologia de paralelização do algoritmo do backpropagation baseou-se nas seguintes etapas:

- Encapsulamento dos procedimentos básicos associados com o modelo a ser paralelizado em uma biblioteca de funções.
- Análise do domínio de variáveis e tarefas.
- Distribuição das diferentes tarefas na arquitetura host.
- Avaliação do desempenho de cada paralelização realizada.

A metodologia de paralelização confirmou o grande "overhead" provocado pela troca freqüente de mensagens no modo online, quando a plataforma é um cluster de estações de trabalho.

Os modos batch e block, por outro lado, apresentam uma troca de mensagens reduzida, o que permite explorar eficientemente o seu paralelismo inerente. A seguir descrevemos a paralelização destes dois modos de treinamento.

4.2. Paralelização no Modo Batch

4.2.1. Descrição

De acordo com a metodologia de paralelização (seção 4.1.) é necessário distribuir as diferentes tarefas entre os diferentes processadores que formam o cluster de computadores.

As características do modo batch apontam naturalmente para a utilização do paralelismo de dados no lugar do paralelismo funcional [5]. Desta forma, cada um dos processadores escravo realizará o mesmo cálculo (treinamento de uma cópia da RNA original) mas com subconjuntos de treinamento (dados) diferentes. Estes subconjuntos são na realidade extraídos do conjunto de treinamento original da RNA. Entretanto, cada processador escravo utilizará sempre o mesmo conjunto de pesos sinápticos criados inicialmente pelo mestre e ajustados ao longo do treinamento.

A figura 3 mostra como a RNA é espalhada, a partir do processador mestre, em três redes idênticas alocadas em três processadores escravos.

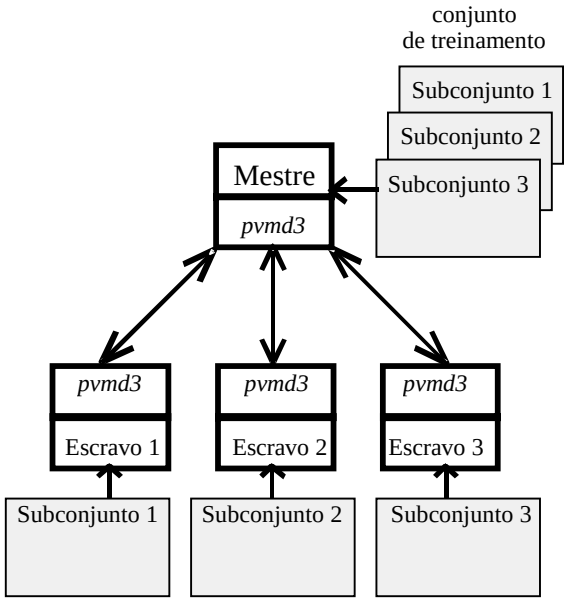


Figura 3: Metodologia de paralelização

Cada processador escravo inicia seu processamento de forma assíncrona e independente dos demais e treina a sua RNA utilizando seu subconjunto de treinamento. A cada iteração (apresentação de um par de treinamento) são calculadas variações de pesos (Δw) para todos os pesos sinápticos da rede e estes valores são acumulados. O fim de uma etapa do treinamento (época *n*) corresponde à apresentação de todos os pares do conjunto de treinamento. Quando isto ocorre então cada processador escravo envia para o processador mestre o valor final de Δw correspondente à etapa em questão. O processador mestre atualizará os pesos sinápticos da rede original a partir da média aritmética da contribuição de todos os *N* processadores escravos de acordo com as eq. (1) e (2). Caso a convergência do erro calculado não seja satisfatório, então um novo espalhamento da RNA ocorrerá (já com os pesos sinápticos atualizados) e uma nova etapa do treinamento será iniciada.

Observa-se que a existência de uma barreira global ao final de uma época é necessária devido ao fato que o processo mestre precisa testar a convergência do erro em cada época e comparar este valor ao valor predefinido, o que só pode ser feito após o recebimento da contribuição do erro de todos os processos escravos.

$$\Delta W(n) = \frac{1}{N}\sum_{k=1}^{N} \Delta w_k(n) \qquad (01)$$

$$w(n+1) = w(n) + \Delta W(n) \qquad (02)$$

4.2.2. O Algoritmo Paralelizado no Modo Batch

Para os processos mestre e escravo no modo batch são mostrados os algoritmos resumidos 4 e 5.

Algoritmo 4: Processo mestre (resumo)

Inicialização
1. crie processos escravos
2. envie pesos iniciais e subconjunto de treinamento para os processos escravos
Repita
3. receba variações de pesos dos processos escravos
4. ajuste os pesos da RNA
5.envie novos pesos para os processos escravos
Enquanto *o erro estiver fora da tolerância desejada*

Algoritmo 5: Processos escravos (resumo)

Inicialização
1.receba pesos iniciais e subconjunto de treinamento do processo mestre
Repita
Para *cada par de vetores no conjunto de treinamento* ***Faça***
2. aplique à entrada da RNA, o próximo vetor de entrada
3. calcule a saída obtida
4. calcule o erro entre a saída obtida da RNA e a saída desejada
5. calcule Δw para toda a RNA acumulando os resultados
Fim para
6. envie variações de pesos (Δw) para mestre
7. receba os novos pesos do mestre
Enquanto *o erro estiver fora da tolerância desejada*

4.3. Paralelização no Modo Block

4.3.1. Descrição

O paralelismo descrito para o modo batch (seção 4.2) se aplica da mesma forma para o modo block. A diferença é que a comunicação entre os processos escravos e o processo mestre ocorrerá sempre ao final de um bloco.

No modo block cada subconjunto de treinamento recebido pelos escravos é subdividido em uma certa quantidade de sub-blocos. Ao final da apresentação de todos os pares do sub-bloco os processadores escravo entram em contato com o processador mestre para envio dos resultados. O tamanho do bloco para uma época é portanto o somatório dos tamanhos de todos os sub-blocos referentes a esta época.

4.3.2. O Algoritmo no Modo Block

O processo mestre realizará as mesmas tarefas apresentadas no algoritmo 4.

Algoritmo 6: Processos escravos (resumo)

Inicialização
1. receba pesos iniciais subconjunto de treinamento do processo mestre
Repita
Para *cada bloco do conjunto de treinamento* ***Faça***
Para *cada par de vetores no bloco do conjunto de treinamento* ***Faça***
2. aplique à entrada da RNA, o próximo vetor de entrada
3. calcule a saída obtida
4. calcule o erro entre a saída obtida da RNA e a saída desejada
5. calcule Δw para toda a RNA acumulando os resultados.
Fim para
6. envie as variações de pesos (Δw) para o mestre.
7. receba os novos pesos do mestre
Fim para
Enquanto *o erro estiver fora da tolerância adotada*

5. Testes Experimentais

5.1 Cluster

O cluster foi formado por estações de trabalho heterogêneas SUN e IBM, rodando UNIX e PVM. As estações de trabalho são interligadas através de uma rede ethernet de 10Mb/s.

5.2 Descrição do Problema

Para testar o desempenho da paralelização foi escolhido o exemplo simplificado de classificação de regiões em um plano mostrado na figura 4 [11].

A RNA classificará as duas regiões mostradas na figura 4 a partir de um número igual de padrões obtidos

para cada uma delas. A estrutura da rede compõe-se de duas entradas que correspondem a um par de coordenadas e de duas camadas ocultas, sendo a primeira com 12 neurônios e a segunda com 4 neurônios. A saída é constituída por 2 neurônios. Neste caso a rede fornecerá apenas dois tipos de respostas, uma para cada região reconhecida.

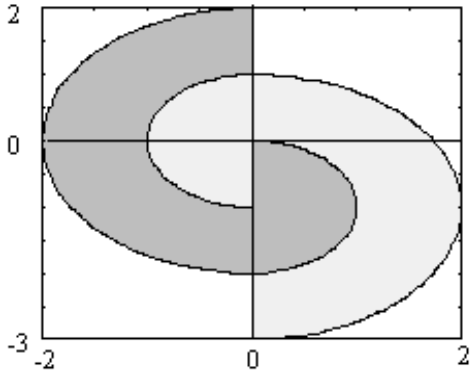


Figura 4: Exemplo de regiões no plano

A função de ativação utilizada para todos os neurônios é a tangente hiperbólica. A tabela 1 mostra os parâmetros da RNA utilizados para os treinamentos seqüencial e paralelo.

Tabela 1: Parâmetros da RNA

Modos	NP	α	η	tol.
Online	120	0,01	0,05	9e-06
Batch	120	0,05	0,2	9e-06
Block	120	0,05	0,2	9e-06

Onde NP : Número de padrões por região
α : Constante associada ao momentum
η : Taxa de aprendizagem
tol.: Tolerância desejada

5.3. Comparação entre os modos de atualização dos pesos

A tabela 2 mostra o número de épocas resultantes do treinamento da RNA.

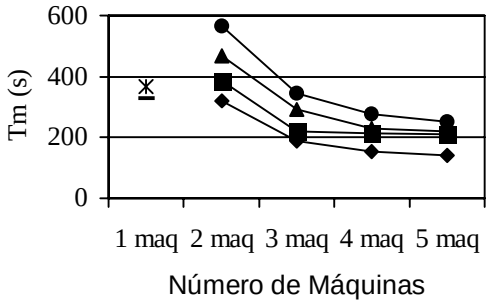
Tabela 2: Número de épocas do treinamento

Modos	Número de épocas
Online seqüencial	3139
Batch seqüencial	4501
Batch paralelo	4501
Block paralelo (1 bloco =1/2 do conjunto de treinamento)	2956
Block paralelo (1 bloco =1/3 do conjunto de treinamento)	2537
Block paralelo (1 bloco =1/4 do conjunto de treinamento)	2332

A implementação paralela do modo online não é mostrada na tabela 2 devido ao motivos descritos na seção 4.1.

Os resultados da tabela 2 mostram que o modo batch requer maior número de épocas de treinamento. Entretanto, como mostrado no gráfico 1, os tempos de treinamento no modo batch são menores do que os correspondentes no modo online .

O gráfico 1 comprova os tempos médios entre os algoritmos batch e block para um número variável de máquinas no cluster. Note que o algoritmo batch apresenta melhores resultados.



◆ batch
■ block 1/2
▲ block 1/3
● block 1/4
✶ seqüencial online
– seqüencial batch

Gráfico 1: Programa paralelo – batch, block e online com variações no tamanho do bloco para o block

6. Conclusão

A paralelização do algoritmo do backpropagation para RNA permitiu obter uma aceleração dos cálculos bastante satisfatória, especialmente no modo batch.

A paralelização do algoritmo no modo block mostrou um desempenho menor do que o do modo batch para o caso do exemplo das regiões no plano, contudo também permitiu obter uma aceleração satisfatória dos cálculos.

Sabe-se ainda que a metodologia empregada não é adequada para um número excessivo de máquinas no cluster. Com o aumento do número de máquinas aumenta-se a comunicação entre os processadores e reduz-se a quantidade de cálculos realizado por cada processo escravo. O resultado é que o tempo de processamento total aumenta comprometendo o desempenho do programa mesmo numa rede considerada rápida.

A técnica de paralelização descrita neste trabalho está sendo empregada em outros problemas computacionalmente intensivos, como por exemplo a classificação de impressões digitais.

Referências

[1] N. Chaves, P.Martins e C. Fernando. Um Estudo de Paralelismo em Redes Backpropagation. II Simpósio Brasileiro de Redes Neurais, 1995.

[2] T. Resk. Distributed Simulation of Synchronous Neural Networks. Invited Article, Neural Network, Vol 7, No 1, USA., 1994.

[3] P. Leray, P. Gallinari e E. Didelet. A Neural Network Modular Architecture for Network Traffic Management. CESA, Lille, 1994.

[4] T. Resk. Mapping and Parallel Simulation of Synchronous Neural Network on Multiprocessors. Procedings of Euromicro Conference. Barcelona, 1993.

[5] K. Hwang. Advanced Computer Archtecture Parallelism-Scalability-Programmability. Mc Graw Hill, 199

[6] PVM 3 Users Guide and Reference Manual. Oak Laboratories. Tennessee USA, 1994.

[7] B. Kruatrachue e T. Lewis. Gran size Determination for Parallel Processing. IEEE Software, New York, 1998.

[8] A. Gerasoulis e T. Yang. On the Granularity and Clustering of Directed Acyclic Task Graphs. IEEE Transations on Parallel and Distributed System. 1993.

[9] L. Fausett. Fundamentals of Neural Networks. Prentice Hall, 1994

[10] L. Coetzee. Parallel Approaches to Training Feedforward Neural Nets. Faculty of Engineering, University of Pretoria, 1996.

[11] S. Haykin. Neural Networks. A Comprehensive Foundation. Macmilian College Publishing Company, 1991

[12] P. Pachecko. Parallel Programming with MPI. Morgan- Kaufmamm, 1996.

[13] G. F. Pfister. In Search of Clusters. Prentice Hall, 1998.

[14] M. Fischer e J. Dongarra. Another Architecture: PVM on Windows 95/NT, 1994.