第 7 卷第 2 期
1996 年 2 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 7, No. 2
Feb. 1996

动态环拓扑多机系统上 DBP 学习算法并行计算模型的构造与实现*

管惠维

(上海大学计算机系　上海　200072)

摘要　人工神经网络模型的软件模拟,其并行算法的设计、实现及性能评价对于神经网络计算机和各种专用神经网络 VLSI 芯片的研制具有十分重要的意义. 本文首先构造了一个分布式存储器、信息传递方式的多机系统作为软件模拟人工神经网络的平台,并用一个环拓扑结构的多 Transputer 网络予以实现. 接着提出并实现了一个适用于动态环拓扑形式的 DBP 并行计算模型,它主要包括神经元的划分和映射策略;DBP 中活性值、误差反向传播及权值修改的多机并行算法. 然后讨论该 DBP 算法的时间复杂度和加速比.

关键词　人工神经网络,BP 学习算法,并行算法,环拓扑.

人工神经网络 ANN(artificial neural network)的一个显著特性是它含有大量可并行计算的单元. 一个 ANN 可以完全用硬件实现,也可以用软件虚拟实现. ANN 的硬件实现是为每一个神经元提供一个处理器,各个神经元之间用通道互连,这已有多种VLSI实现方案.[1,2]然而,ANN 的硬件实现往往适合一些专门的应用领域,一但固化就难以重构. ANN 的软件虚拟实现是在若干处理器上模拟 ANN 的行为特征. 在各个处理器上可映射许多神经元,即可将 M 个神经元的 ANN 驻留在 N 个处理器上,建立必要的并行计算模型加以实现. ANN 的软件模拟便于在不同的互连拓扑结构上重构,以探索不同 ANN 的特性. 此外,对在不同拓扑结构的多机网络上实现 ANN 的性能评价也是十分重要的. 因此 ANN 的软件模拟是 ANN 硬件实现的基础.

本文首先构造一个基于分布式存储器的、信息传递方式的多机系统 DMMPMS(distributed—memory, message passing multiprocessor system)作为模拟 ANN 的硬件平台,然后提出了一个在动态环拓扑结构上分布式反向传播 DBP(distributed backpropagation)学习算法的并行计算模型,分析了其时间复杂度.

1　模拟 ANN 的多机系统及其映射策略

1.1　模拟 ANN 的多机系统 DMMPMS 设计方案

* 本课题受上海市重点学科发展基金、国家教委博士基金资助. 作者管惠维,1954 年生,副教授,主要研究领域为并行处理与网络分布式计算,多媒体技术,人工智能.

本文通讯联系人:管惠维,上海 200072,上海大学计算机系

本文 1994—12—29 收到修改稿

首先我们提出一个基于分布式存储器结构的、信息传递方式的多机系统DMMPMS的设计方案，如图1所示. 这个系统能方便地使用一个多transputer网络加以实现.[3]其中每一个处理器即可采用一片Transputer芯片，局部存储器对应Transputer的片上的存储器，网络接口对应Transputer的通信链Link，通过交叉开关Coo4可实现网络中各个处理器之间的互连.

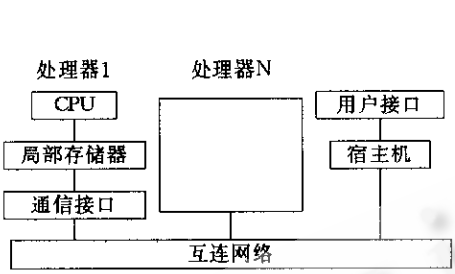

图1 DMMPMS多机系统

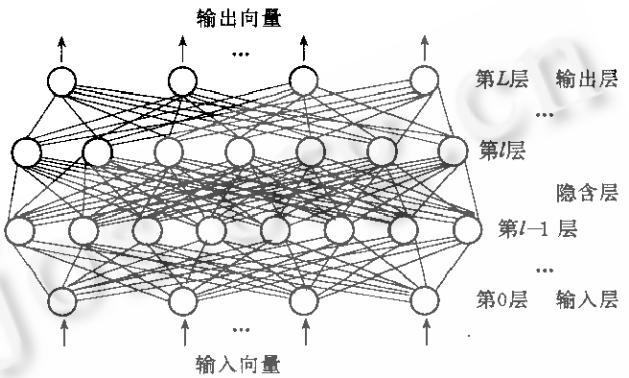

图2 全互连ANN

1.2 多层ANN的BP算法

一个$L+1$层的全互连ANN如图2所示. 其中$L=0$为输入层，含有n_0个神经元，第i层($1\leqslant i\leqslant L$)含有n_i个神经元. 各层中的神经元分别连接到其相邻的所有神经元.

BP学习算法是一个非常典型的ANN计算模型.[4-6]它能根据给定的输入、输出模型，寻找出一组合适的权重集，以达到ANN的输出接近给定的模式. 鉴于这一特性，它被广泛地应用于具有模式识别特征需求的领域.

多层ANN的BP基本算法由下列3个方程组成：

$$act(l,k)=f\left(\sum_{j=0}^{n_l-1}weight(l-1,j,k)\cdot act(l-1,j)\right)\quad k=0,\cdots,n_l,l=1,\cdots,L \tag{1}$$

$$backerr(l,j)=\begin{cases}[tar(l,j)-act(l,j)]\cdot[act(l,j)\cdot(1-act(l,j))], & l=L\\ \left[\sum_{k=0}^{n_l+1}backerr(l+1,k)\cdot weight(l,j,k)\right]\cdot[act(l,j)\cdot(1-act(l,j))], & l=L-1,\cdots,1\end{cases} \tag{2}$$

$$dweight(l-1,j,k)=\eta\cdot backerr(l,k)\cdot act(l-1,j) \tag{3}$$

其中f为一个形式为$f(x)=(1+e^{-x})^{-1}$的非线性函数，$act(l,k)$为第l层上第k个神经元的活性值；$weight(l,i,j)$为在l层第i个神经元与第$l+1$层上第j个神经元相连的权重值；$backerr(l,j)$是l层上第j个神经元的反向误差值；$tar(l,j)$是输出层第j个神经元的目标输出值；$dweight(l-1,j,k)$为$l-1$层上第j个神经元与l层上第k个神经元之间权重的修改值；η为学习率. 式(1)代表ANN中神经元活性值的前向计算过程；式(2)代表误差的反向传播；式(3)代表权重修改.

1.3 DMMPMS上多层ANN的划分映射

将ANN映射到多机DMMPMS的关键问题是：(1)如何合理划分数据，在BP算法中是如何划分和存放活性值，权重值和误差值. (2)如何在各个处理器之间有效地传送这些数

据.(3)如何设计一个高效的广播算法.

在我们的实验模型中,采用这样的映射策略,即将一个多层 ANN 纵向划分为 N 个子网,而各个子网被映射在一个 N 处理器构成的 DMMPMS 的一个处理器上,各个处理器之间以点对点的通信链连接方式进行通信.假设第 L 层有 n_l 个神经元,则它被划分成 N 个部分,该层 n_l/N 个神经元被赋给一个处理器.图 3 描绘了一个全互连的 ANN 被映射到 4 处理器 DMMPMS 的情况,为简便起见,我们假设该网有 4 层,每层有 8 个神经元.(实际上层次和每层神经元个数均可变)其中各个处理器维护驻留在其局部存储器的各个神经元的活性值、权重值和反向误差值.

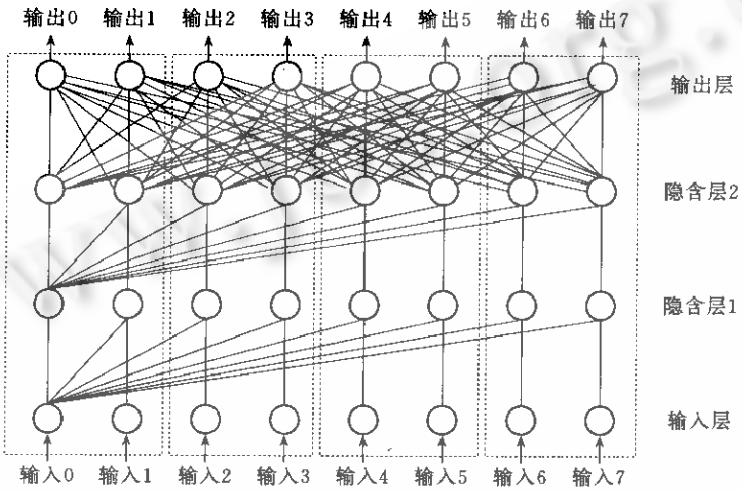

图3 DMMPMS上多层ANN的划分映射

2 动态环拓扑多机网络上的 DBP 并行算法

基于上述映射策略和神经元的划分,BP 算法应加以修改成为一个分布式算法 DBP,以便实现在各个处理器上并行协同计算和传送数据.下面分别论述我们提出的在动态环拓扑结构多处理器网络上的 DBP 算法及其实现.该算法不仅适合环拓扑多机网络中处理器个数的动态变化,而且也能满足 ANN 中层数和神经元个数的动态变化.

2.1 DBP 算法中活性值的前向计算

设 $NS(l,i)$ 为第 l 层上映射在处理器 PE_i 上的神经元集合,则 $|NS(l,i)|=n_l/N$.从式(1)可知,为了计算各个神经元的活性值 $act(l,k)$,必须知道存放在不同处理器上的所有低一层上的神经元的活性值 $act(l-1,j)$.这就需要对各个处理器 PE_i 实施这样的操作:即对各个神经元 $j\in NS(l-1,i)$,广播 $act(l-1,j)$;而对各个神经元 $j'\notin NS(l-1,i)$,接受 $act(l-1,j')$.这个过程通常可采用 All－to－All 广播算法[7]来实现.一但完成上述操作,每个处理器均已获得 $l-1$ 层上所有别的神经元的活性值,这样式(1)就可以在各个处理器上并行地独立执行.下面以抽象的 Occam 形式[8]给出在 $n.pe$ 号处理器上式(1)的分布式并行算法.

```
SEQ l=0 FOR L−1
  SEQ n=0 FOR n.processor−1
```

$all-to-all-broadcast(act(l,j))$

SEQ $k=n.neu.per.pe*n.pe$ FOR $n.neu.per.pe$

$$act(l+1,k)=f(\sum_{j=0}^{n_l-1}weight(l,j,k)*act(l,j))$$

其中 $n.processor$ 表示连在环拓扑上的处理器个数，$n.neu.per.pe$ 为每个处理器上第 l 层的神经元个数，$n.pe$ 为处理器号.

2.2 DBP 算法中误差的反向传播

同理，为了在 $n.pe$ 号处理器上计算第 j 个神经元的反向误差值 $backerr(l-1,j)$，$j\in NS(l-1,i)$，需要存放在其它处理器上和所有 l 层上的反向误差值 $backerr(l,k)$. 类似于 act 的计算，在 $n.pe$ 号处理器上实施如下操作：对于神经元 $k\in NS(l,i)$，$n.pe$ 号处理器向其它处理器广播 $backerr(l,k)$；而对于神经元 $k'\notin NS(l,i)$，则接受 $backerr(l,k')$. 因此，在各处理器上并行计算分布划分后的式(2)的算法实现如下：

PAR $j=n.neu.per.pe*n.pe$. FOR $n.neu.per.pe$

$backerr(L,j)=[tar(L,j)-act(L,j)]*[act(L,j)*(1-act(L,j))]$

SEQ $l=L-1$ TO l

SEQ $n=0$ FOR $n.processor-1$

$all-to-all-broadcast(backerr(l+1,j))$

SEQ $j=n.neu.per.pe*n.pe$ FOR $n.neu.per.pe$

$$backerr(l,j)=[\sum_{k=0}^{n_{l+1}-1}abckerr(l+1,k)*weight(l,j,k)]*[act(l,j)*(1-act(l,j))]$$

2.3 DBP 算法的权重值修改和优化

据 BP 学习算法式(3)，计算 l 层上 j 神经元与 $l+1$ 层上 k 神经元之间的权重修改值需要 $l+1$ 层上 k 神经元的反向误差和 l 层上 j 神经元的活性值，若这两个神经元恰好不在同一处理器上，若仍按式(3)来计算，必然引起附加的通信开销. 为了提高并行计算加速比，在我们的 DBP 算法中，使用了在各个处理器上分别独立计算权重修改值的新方法.

设 l 层 j 神经元和 $l+1$ 层 k 神经元分别驻留在处理器 PE_i 和 $PE_{i'}$ 上，如图4. 如果 PE_i 上的 j 神经元知道存放在 $PE_{i'}$ 上的 $backerr(l+1,k)$，则它就可得出权重修改值 $dweight(l,j,k)$；同理，在 $PE_{i'}$ 上的神经元 k 若知道存放在 PE_i 上的 $act(l,j)$，则在 $PE_{i'}$ 上同样可独立地计算 $dweight(l,j,k)$. 由于在 DBP 算法的活性值前向计算中，各层上的所有神经元的活性值均已广播到各个处理器上(见2.1)，而在误差的反向传播计算中，各个层上的所有神经元的误差值也已广播到各个处理器上(见2.2)，因此我们可以直接地使用它们在各个处理器上独立计算权重修改值 $dweight(l,j,k)$，避免了大量通信开销. 在 $n.pe$ 号处理器上，我们的权重值修改的并行算法描述如下：

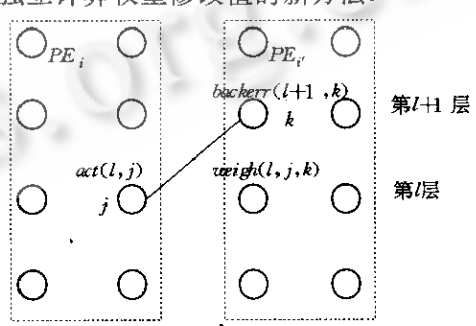

图4 不同处理器上神经元的分布和连接

SEQ $l=L-1$ TO 1

SEQ $j=n.neu.per.pe*n.pe$ FOR $n.neu.per.pe$

SEQ $k=0$ FOR $n.layer.l$

$dweight(l,j,k)=learn.rate*backerr(l+1,k)*act(l,j)$

$weight(l,j,k)=weight(l,j,k)+dweight(l,j,k)$

SEQ $j=n.neu.per.pe*n.pe$ FOR $n.neu.per.pe$

SEQ $k=0$ FOR $n.layer.l-1$

$dweight(l-1,j,k)=learn.rate*backerr(l,k)*act(l-1,j)$

$weight(l-1,j,k)=weight(l-1,j,k)+dweight(l-1,j,k)$

其中 $n.layer.l$ 为 l 层上神经元的个数，$n.layer.l-1$ 为 $l-1$ 层上神经元的个数. 进一步分析，权重值修改和计算的过程可以合并到误差反向传播计算过程中，而使 DBP 算法得到优化. 下面我们给出分布在 $n.pe$ 号处理器上误差反向传播和权重修改的并行算法：

PAR $j=n.neu.per.pe*n.pe$ FOR $n.neu.per.pe$

$backerr(L,j)=[tar(L,j)-act(L,j)]*[act(L,j)*(1-act(L,j))]$

SEQ $l=L-1$ TO 1

SEQ $n=0$ FOR $n.processor-1$

$all-to-all-broadcast(backerr(l+1,j))$

SEQ $j=n.neu.per.pe*n.pe$ FOR $n.neu.per.pe$

$$backerr(l,j)=\left[\sum_{k=0}^{m_{l+1}-1} backerr(l+1,k)*weight(l,j,k)\right]*[act(l,j)*(1-act(l,j))]$$

SEQ $j=n.neu.per.pe*n.pe$ FOR $n.neu.per.pe$

SEQ $k=0$ FOR $n.layer.l$

$dweight(l,j,k)=learn.rate*backerr(l+1,k)*act(l,j)$

$weight(l,j,k)=weight(l,j,k)+dweight(l,j,k)$

SEQ $j=n.neu.per.pe*n.pe$ FOR $n.neu.per.pe$

SEQ $k=0$ FOR $n.layer.l-1$

$dweight(l-1,j,k)=learn.rate*backerr(l,k)*act(l-1,j)$

$weight(l-1,j,k)=weight(l-1,j,k)+dweight(l-1,j,k)$

综上所述，我们提出的在动态环拓扑多机网络上的 DBP 并行算法，既考虑到处理器个数的动态变化(用参数 $n.processor$ 调整)，又实现了分布在各个处理器上的并行计算(用参数 $n.pe$ 表示)，而且适于 ANN 中层数的动态变化和神经元个数的动态变化(分别用参数 $n.layer.l$ 和 $n.neu.per.pe$ 表示). DBP 并行算法在单处理器和多处理器环拓扑结构上的实验结果如下.

(1)3层神经网络上72个神经元

	单处理器	3个处理器	4个处理器
时间	2503368	2202017	2194348
加速比		1.1369	1.1408

(2)4层神经网络上96个神经元

	单处理器	3个处理器	4个处理器
时间	3356283	3116047	2935834
加速比		1.077	1.143

单位时间为 T800的内部时钟50ns.

3　DBP 算法的时间复杂度和加速比

基于动态环拓扑多机系统的 DBP 并行算法的时间复杂度和加速比分析如下. 为简便起

见，设 ANN 中各层的神经元个数为 n，均在一层上讨论计算时间，则在单处理器上执行由式(1)～(3)所给出的 BP 算法需要的时间为： $T_1=tact+terr+tuw$

其中 $tact$ 是执行活性值前向计算的时间；$terr$ 是计算误差反向传播的时间；而 tuw 是计算权重修改值的时间. 因此，它们可分别用下列公式表示：

$$tact=n*(n*MAT+FT)$$

$$terr=n*(n*MAT)$$

$$tuw=n*(n*MAT)$$

所以 $$T_1=tact+terr+tuw=n*(3*n*MAT+FT) \tag{4}$$

其中 MAT 是两个浮点数之间相乘后再相加所花费的时间；FT 为计算非线性函数 f 所需的时间. 若我们的 DBP 并行算法在由 N 个处理器相连的环拓扑上执行，则其花费的时间分别为：$tdact=(N-1)*DBT+n*(n*MAT+FT)/N$

$$\begin{aligned} tdew &= DET+DWT \\ &=[(N-1)*DBT+n*(n*MAT)/N]+[n*(n*MAT)/N+n*(n*MAT)/N] \\ &=(N-1)*DBT+3n(n*MAT)/N \end{aligned}$$

其中 $tdact$ 表示活性值的分布前向计算时间；DBT 表示在环拓扑上完成广播算法的时间；$tdew$ 表示计算误差的反向传播和修改权重值花费的总的时间；而 DET 和 DWT 则分别表示计算误差反向传播和修改权重值的时间. 因此，在 N 个处理器构成的环拓扑上，在 ANN 的一层上并行执行 DBP 算法所花费的总的时间 T_N 可用式(5)表示如下：

$$\begin{aligned} T_N &= tdact+tdew \\ &=[(N-1)*DBT+n(n*MAT+FT)/N]+[(N-1)*DBT+3n(n*MAT)/N] \\ &=2(N-1)*DBT+n(n*MAT+FT)/N+3n(n*MAT)/N \end{aligned} \tag{5}$$

在计算加速比时，我们还必须考虑各个处理器的通信机制. 在我们的 DBP 算法实现中，采用点对点通信方式，即在单位通信时间内，各个处理器能在其互连的通信链上并行地发送和接收一个单位信息. 若一个处理器发送和接收一个单位信息的单位通信时间被定义为 μ，则它包含建立必要的通信机制的时间和实际传送一个单位信息的时间，其下限为 n/N. 因此，根据式(4)和式(5)，我们可以获得在 N 个处理器环拓扑上 DBP 算法的加速比为：

$$\begin{aligned} S_N &= T_1/T_N \\ &=\frac{n(3*n*MAT+FT)}{2(N-1)*DBT+n(n*MAT+FT)/N+3n(n*MAT)/N} \\ &=\frac{n(3*n*MAT+\alpha*MAT)}{2(N-1)*DBT+n(n*MAT+\alpha*MAT)/N+3n(n*MAT)/N} \qquad (\text{设 } FT=\alpha*MAT) \\ &=\frac{n(3*n*MAT+\alpha*MAT)}{2(N-1)*n*\mu/N+n(n*MAT+\alpha*MAT)/N+3n(n*MAT)/N} \qquad (\text{设 } DBT=n*\mu/N) \\ &=\frac{n(3*n*MAT+\alpha*MAT)}{2(N-1)*n*\beta*MAT/N+n(n*MAT+\alpha*MAT)/N+3n(n*MAT)/N} \qquad (\mu=\beta*MAT) \\ &=\frac{n(3n+\alpha)N}{2(N-1)*n*\beta+n(n+\alpha)+3n^2} \end{aligned}$$

$$=\frac{(3n+\alpha)N}{2(N-1)\beta+4n+\alpha}$$

当 n 非常地大于 N 时，DBP 算法的加速比为：$\lim_{n\to\infty} S_N=3N/4$.

由此可见，当 ANN 规模很大，即各层上的神经元个数大大高于环拓扑中互连的处理器个数时，我们的 DBP 算法具有很高的效率.

4　结束语

ANN 的软件模拟，其并行算法的设计与实现以及性能分析对于神经网络计算机及各种专用 ANN 芯片的研制具有十分重要的意义. 本文构造了一个基于分布式存储器的点对点信息传递方式的松散耦合的多机系统 DMMPMS 作为软件模拟 ANN 的平台，用一个动态环拓扑结构的多 Transputer 网络加以实现. 我们以典型的 BP 学习算法为例，提出并实现了一个基于动态环拓扑的 DBP 并行可执行模型，它主要包括 ANN 中各层上神经元的划分和映射策略；DBP 的活性值、反向误差传播以及权值修改的并行算法. 最后，我们讨论了 DBP 算法的时间复杂度，并给出了加速比. 现在，我们正在进一步研究各种 ANN 并行算法的设计与实现，以及它们和不同拓扑形式的关系.

致谢　作者感谢上海交通大学计算机系孙永强教授对于本项研究工作的关心和指教.

参考文献

1　Ghosh J, Hwang K. Mapping neural networks onto message—passing multicomputers. Journal of Parallel and Distributed Computing. 1989, **6**(2): 291～330.

2　Mead C A. Analog VLSI and neural systems. Addison—Wesley, 1989.

3　Inmos Limited. Reference manual of transputer. Bristol: Prentice Hall, 1987.

4　MeClelland J L, Rumelhart D E. Explorations in parallel distributed processing. Massachusetts: MIT Press, 1989. 83～120.

5　Rumelhart D E, Hinton G E, Williams R J. Learning internal representations by error propagation. Parallel Distributed Processing: Explorations in The Microstructure of Cognition, Massachusetts: MIT Press, 1987. 318～362.

6　Anderson J, Rosenfeld E. Neurcomputing. Massachusetts: MIT Press, 1989.

7　Johnsson S L, Ho C T. Optimum broadcasting and personalized communication in hypercubes. IEEE Trans. Comput. 1989, **38**(9): 1249～1268.

8　Inmos Limited. A tutorial introduction to occam programming. Bristol: Prentic Hall, 1987.

CONSTRUCTION AND IMPLEMENTATION OF THE PARALLEL COMPUTING MODEL OF DBP LEARNING ALGORITHM

Guan Huiwei

(Department of Computer Science and Engineering　Shanghai University　Shanghai　200072)

Abstract The software simulation of artificial neural networks, and the design, implementation and evaluation of parallel algorithm about artificial neural networks are highly important to the research and production of neural computers and the various of special neural VLSI chips. In this paper, a DMMPMS (distributed—memory, message—passing multiprocessor system) is constructed as a multicomputer architecture for simulating the artificial neural network at first, which is implemented by a multi—transputer system of a ring topology form. Next, an executable parallel computing model of DBP algorithm of multilayered neural network on dynamic ring topology is proposed and implemented, which mainly includes the strategy of partition and map for neural cells; the parallel algorithms of the forward computation of activations, the backpropagation of errors and the update of weights. Then the time complexity and speed—up ratio of the DBP algorithm are investigated.

Key words Artificial neural network, BP learning algorithm, parallel algorithm, ring topology.