华中科技大学

硕士学位论文

基于数据并行的BP神经网络训练算法

姓名：张弦

申请学位级别：硕士

专业：计算机应用技术

指导教师：马光志

20080607

摘 要

BP(Back Propagation)算法，即误差反传训练算法，具有良好的非线形逼近能力，是人工神经网络应用最广泛的训练算法。但是 BP 算法存在训练速度慢、易陷入局部极小值等缺陷。以弹性 BP 算法为代表的 BP 改进算法虽然在一定程度上加快了神经网络的训练，但是对于训练规模巨大的神经网络，这些改进算法仍然不能满足实际应用的要求。

考虑到神经网络本身所具有的并行处理能力，可以利用并行计算来解决大规模神经网络训练问题。BP 网络并行化有结构并行和数据并行两种方法。在基于数据并行的 BP 算法中，训练样本被划分给不同的处理机，各处理机对同样的神经网络进行训练，然后统计所有的训练结果更新神经网络。这种方法的优点是处理机之间的通信量少、并行粒度大。

在基于 MPI(消息传递接口)的并行环境下，通过局域网内互联的 PC 机，组建了一个机群训练平台。采用主/从结构的并行模型，将训练样本数据平均分配到各从节点，由主节点收集并统计训练结果，实现了 BP 神经网络训练的并行化。同时根据神经网络初始权值随机性的特点，在并行 BP 算法的基础上作出了改进。在训练初期，各个节点分别对各自的神经网络进行随机初始化，然后同时对其进行训练，在一定的迭代次数之后筛选出误差最小的神经网络，最后利用筛选出的神经网络进行并行训练。

采用华中科技大学同济医学院提供的高血压调查数据作为训练样本，分别使用串行 BP 算法、并行 BP 算法和改进的并行 BP 算法，建立神经网络并对其进行训练。实验结果显示，并行算法相对于串行算法极大地加快了训练速度。同时改进的并行算法也有效地提高了并行训练的加速比和并行效率。

关键词：人工神经网络，BP 算法，数据挖掘，数据并行

Abstract

BP (Back Propagation) algorithm, also known as the error-propagation algorithm, is a widely used training method in the application of neural networks for its fine capability of non-linear approximation. However, it is known to have some defects, such as converging slowly and falling in a false local minimum. Although some optimization algorithm such as RPROP help to speed up the learning process, for the neural networks with tremendous size and extremely large training set these algorithms could not satisfy the demand of implementation.

The ability of parallel processing is inherent in neural network, so it is feasible to reduce the long training time with the parallel techniques. There are two different parallel implementation schemes for BP networks, the structure parallelism and the data parallelism. In the data parallelism, the training data is distributed to different computing nodes; each node has a local copy of the complete weigh matrices and accumulates weight change values for the given training patterns, and then the weight change values of each node are summed and used to update the global weight matrices. The data parallelism with a large-grain size reduces the communication time. Therefore, it is mostly implemented in the cluster.

By connecting the PCs with a TCP/IP Ethernet local area network, we built up a cluster system with MPI (Message Passing Interface). The parallel BP network is implemented in master/slave mode: The training data is distributed to each slave node; the master node gathers the result processed by the slave and updates the neural network. We also proposed an optimized method which helps to choose better weight matrices to speed up the convergence. In the optimization technique, each node started to train the whole sample set with different initial weights, after several iterations, the node with the minimum error is found and its weight matrices are broadcasted to each node to start the parallel training.

We chose the hypertension data provided by the Tongji Medical College of Huazhong University of Science and Technology as the training data, and implemented the parallel BP network to evaluate the proposed algorithm. The experiment shows that, the parallel algorithm tremendously reduces the training time compared to the sequential BP algorithm, and the optimized method increases the speedup and parallel efficiency.

Key words: neural network, BP algorithm, data mining, data parallelism

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到，本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　月　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于　保密□，在_____年解密后适用本授权书。

　　不保密□。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　月　日　　日期：　　年　月　日

1 绪 论

数据挖掘技术作为解决“数据爆炸”问题而出现的最有效手段，受到了商业、企业界的极大关注。数据挖掘(Data Mining)就是从大量的、不完全的、有噪声的、模糊的、随机的原始数据中，提取隐含在其中的、人们事先不知道的、但又是潜在有用的信息和知识的过程[1]。数据挖掘的目的是提高市场决策能力、检测异常模式、在过去的经验基础上预言未来趋势等。这些知识和规则是隐含的、先前未知的、对决策有潜在价值的有用信息。通过数据挖掘，有价值的知识、规则或高层次的信息就能被抽取出来，为决策提供依据，为知识归纳服务。

数据挖掘有很多研究方向，目前的研究主要集中在分类、聚类、关联规则挖掘、序列模式发现、异常和趋势发现等方面[2]。其中神经网络(Neural Network)在商业等领域的成功应用，使它成为数据挖掘中最成熟、最重要、最活跃的研究内容。

1.1 课题的研究背景和意义

1.1.1 课题的研究背景

本课题的研究背景是国家 863 项目——基于网格的数字化医疗决策支持系统。

近年来，神经网络理论的应用取得了令人瞩目的进展，特别是在人工智能、控制、通信和生物医学等领域得到了广泛的应用。神经网络通过模拟人脑处理信息的方式，将大量的神经元相互连接，通过训练样本对神经网络进行训练，不断更新各个神经元的连接权值，最后得到可以用来求解问题的神经网络。神经网络作为一种重要的数据挖掘方法，很好地应用在医疗数据的挖掘和分析之中。在医疗决策支持系统中，通过对疾病数据建立好相应的神经网络模型，挖掘各种危险致病因素对疾病的影响，总结疾病特征和内在规律，为医生做出正确的预测、预防、早期诊断和治疗决策提供有力支持。

目前，在人工神经网络的实际应用中，BP网络是神经网络里应用最广泛的一种网络，它使用的是梯度下降算法。这种算法通过在误差函数曲面上搜索误差的最小值以求得神经网络的权值。但是误差函数曲面往往是高维的凹凸不平的曲面，使得搜索速度缓慢，神经网络训练时间增长。在实际应用中，特别是在医疗决策支持系

统中，医疗信息数据量巨大结构复杂，对于这种大数据量、大规模神经网络训练的情况，不得不考虑收敛速度和训练时间的问题。

1.1.2 课题的研究意义

随着并行技术的发展，并行计算在解决大运算量的复杂问题上发挥了重要作用。本课题的目的是通过研究并行化技术，将其应用到BP神经网络中，提高大规模神经网络的训练速度。在医疗决策支持系统中，利用并行计算的强大数据处理能力对海量数据进行挖掘与分析，有利于获得更准确的疾病信息、发病规律和致病因素。这对尽早预测、预防、准确诊断并有效治疗各种疾病，提高我国的全民健康和医疗卫生水平具有重大意义。

随着并行计算技术的发展和神经网络的应用越来越广泛，国外许多人开始研究并行神经网络，并且取得了很多有价值的成果。但是这些应用成果多是基于并行阵列计算机，因此需要特殊的硬件支持。随着互联网的普及，分布式计算越来越多的被应用到科学计算中。网络中存在大量的计算机，虽然它们的计算能力各不相同，但可以将这些巨大的计算力资源利用起来进行并行计算。进一步想，如果将分布式并行的技术应用到神经网络训练的过程中，则可以充分发挥分布式并行计算的优势，而且不受硬件的限制，同时又能提高神经网络处理大量数据的能力。因此研究神经网络在分布式网络环境下的并行化更具有实际价值和现实意义。

1.2 国内外研究概况

人工神经网络(Artificial Neural Network, ANN)，是在模拟生物神经网络的基础上构建的一种信息处理系统，具有强大的信息存贮能力和计算能力。人工神经网络在1943年由McCullo和PittS提出，在20世纪80年代进入了一个发展高潮，至今已开发出误差反向传播网络、对向传播网络、径向基函数网络、自组织映射网络、Hopfield网络、Kohonen网络、Elman网络等30多种典型的网络模型，其中以误差反向传播网络，即BP神经网络模型应用最广。由于BP神经网络具有自组织、自适应、非线性和容错性等特性，使其已成为人工智能领域的前沿技术，在模式识别、函数逼近、联想记忆、复杂控制、信号处理等领域应用广泛。特别是在体现知识的复杂性和微观认识不完备性的环境科学领域，BP神经网络方法已初步显示出其广阔的应用前景[3]。下面从BP算法的改进算法，BP算法并行化方法两个方面来介绍BP神经网络的研究现

状。

1.2.1 BP 算法改进方法的研究进展

BP算法，也叫反向传播算法，是由Paul Werboss于1974年首次提出来的。BP算法以其良好的非线性逼近能力、泛化能力以及实用性成为了人工神经网络训练算法中应用最为广泛的算法。据统计，约有80%的神经网络训练算法采用BP算法。但是它也有非常明显的缺点。如：(1)网络结构设计困难，特别是隐含层数和隐层节点数难以确定。(2)训练易陷入瘫痪，收敛速度较慢。(3)易陷入局部极小值。这些缺点极大地影响了BP算法的推广应用。因此很多专家学者对BP算法的改进方法进行了深入的研究。这些改进方法可以分为基于启发式的改进方法和基于数值优化技术的改进方法。启发式方法的代表是可变步长算法、带动量项BP算法和弹性BP算法。基于数值优化技术的方法有：牛顿法、共轭梯度法和Levenberg-Marquardt法等。

(1) 可变步长BP算法

Magoulas等[4]提出了可变步长的BP算法。因为标准BP算法收敛速度慢的重要原因是学习速率选择不当。学习速率选得太小，收敛太慢；学习速率选取得太大，则有可能修正过头，导致发散。在可变步长法中，根据误差的变化来调整学习步长，有效的提高了训练速度。

(2) 附加动量项BP算法

鉴于传统BP算法在训练过程中容易陷入局部极小值以及经常出现的振荡现象，Yu Xiaohu等[5]提出了附加动量项的BP算法。附加动量法使BP神经网络在修正其权重和阈值时，不仅考虑误差在梯度上的作用，而且考虑在误差曲面上变化趋势的影响，允许忽略网络上的微小变化特性。这种方法所加入的动量项实质上相当于阻尼项，减小了学习过程的振荡趋势，从而改善了收敛性，容易找到更优的解。但是这种方法的缺点也是明显的，参数的选取只能通过实验来确定，而且学习速度还不能满足实时的工作需要。文献[6]中将可变步长法和附加动量法结合起来，既有效的限制陷入局部极小值，又有利于缩短学习时间。以上这些算法都是一个高度启发式的过程，必须慎重地选择学习率和动量参数。

(3) 弹性BP算法

1993年，德国的Martin和Heinrich提出了Resilient BP[7]方法，即弹性BP算法。由于标准的梯度下降算法的权值更新由学习速度和梯度的大小共同决定。而随着网络

误差的下降，误差对权值的偏导数不断减小；当网络误差下降到一定精度时，梯度也减小到一定程度，导致权值的修正值也很小，从而网络的调整能力不断减小，最后网络的调整达到几乎停滞的地步。弹性BP算法抛弃了由学习速度和梯度的大小来确定权值的更新量，权值改变的大小仅仅由专门的“更新值”$\Delta_{ij}(t)$来确定。从而消除了梯度的大小对网络收敛速度的影响。由于学习规律的清楚和简单，和最初的反传算法比较，在计算上仅有少量的耗费，但是训练速度和精度却有了很大的提高。文献[8~10]中通过对各种BP改进算法的对比研究，均充分肯定了弹性算法在收敛速度、稳定性、鲁棒性等较其他大多数算法有突出的优势。Christian[11]等在原始弹性算法的基础上又作了相应的改进，进一步提高了该算法收敛速度。

(4) 牛顿法

牛顿法最初由Issac Newton提出，是用来求解非线性方程根的方法。Dennis. J.E和R.B. Schnabel在论文[12]中提出了基于牛顿法的BP算法，从而将求函数最小值的过程转换成求梯度函数等于0时的根的过程。牛顿算法虽然收敛速度快，但是需要额外的空间用来存储赫森矩阵，而随着网络规模的扩大，赫森矩阵所需的存储空间也变得非常大，这时牛顿法是不太现实的。

(5) 共轭梯度法

J .Leonard 和M. A.Kramer在论文[13]中提出了基于共轭梯度法的BP算法。在一般的BP算法中，权值搜索的方向是沿着梯度的反方向，虽然在这个方向上，误差函数的值减少得最快，但这并不是收敛得最快的方向。而在在共轭梯度法中，沿着共轭方向进行搜索，收敛速度将比一般的梯度下降法要快得多。并且可以利用行搜索策略来找到共轭梯度方向上误差函数的最小值点，从而可以决定最佳的权值改变步长。共轭梯度法与牛顿法相比，它的计算代价很低，因此共轭梯度法适用于处理较大规模的问题。

(6) Levenberg-Marquardt算法

Levenberg-Marquardt[14](LM)算法是由Levenberg与Marquardt提出的最小二乘拟合算法，它是牛顿法的改进。该方法应用在BP网络的训练中，它既具有牛顿法的快速收敛性，又结合了最速下降法的全局收敛性。但是其缺点是仍需要较大的存储空间，在网络规模较大的时候是不实用的。近年来，随着模拟退火算法，遗传算法和混沌算法的兴起，有人将这些算法与BP网络结合，利用它们的全局特性来优化神经

网络算法，取得了不错的成果。文献[8~10]中分别对各种BP网络改进算法的性能和优缺点做了比较详尽的描述。

1.2.2 BP 算法并行化的研究进展

随着并行计算技术的发展和神经网络的广泛应用，国内外许多专家学者开始了并行神经网络的研究，并且取得了很多有价值的成果。BP网络的并行化主要有两种思路：结构并行和数据并行。

(1) 结构并行

结构并行是对网络结构进行分割，将节点划分给不同的处理机进行并行的训练。按结构分割的方法包括按层分割[15]，横向分割[16]，systolic阵列法和波前阵列法[17]。按层分割是将隐含层的神经元和输出层的神经元分别划分给不同的处理机，采用流水线的方式通过任务之间时间的重叠来达到加速的目的。这种分割方式由于隐含层处理机运算量大，成为了提高计算效率的瓶颈，因此难以实现负载平衡，只有在特定结构的并行机下才可以实现；横向分割是根据处理机数量，把每层神经元都平均分配给各个处理机。这种分割方式较按层分割的方式更容易实现负载平衡，但是处理机之间的通信量大，对于内部通信部件并行度较低的并行机，其效率的提升有限；systolic阵列法和波前阵列法是基于矩阵的并行计算，而神经网络的计算可以归为矩阵的计算，因此这种方式实现起来比较灵活，也可以充分发挥矩阵并行计算的优势。国外在结构并行方面上研究的比较多，Chinn[18]和Kamil[19]分别在Maspar公司的并行机MP-1上实现了这种结构并行的神经网络；Jackson [20]给出了基于结构并行的BP网络在Intel iP2SC/860超立方体结构上的实现。Wayne Allen等人[21]在Symult s2010的并行系统上实现了按纵向分割和按层分割的混合型BP网络模型。

(2) 数据并行

数据并行是将训练样本平均分配到各个处理机，在每个处理机上分别对这些样本进行训练。每次迭代后将个处理机的计算结果进行收集以获得更新之后的网络权值。然后再将新的网络权值更新给各个处理机进行下次迭代。Michael[22]等在IBM公司的GF11上实现了这种并行方式。由于在结构并行法中，处理机的通行开销非常大，网络消息传递复杂，为了不影响计算的速度，通常只在大型的并行机上采用；而基于数据并行的方法可以很有效的减少网络通信开销，有利于在分布式的网络环境下实现[23]。近年来国内在这些方面开展了一些研究，武汉理工大学的高曙[24]、北京大

学的宋国杰[25]分别在PVM机群环境下实现了基于这种数据并行的BP算法。胡月等[26]针对数据并行的BP算法作了进一步改进，提出了二次并行策略，提高了并行计算的效率。文献[27]中将并行神经网络应用在模式识别上，实现了一个汉字识别系统；文献[28]中通过数据并行的神经网络实现了对甲状腺素数量的预测。

如今，很多学者开始了对机群环境下并行神经网络的研究。但是，如果机群系统中各个处理机计算能力不同，就会造成负载不平衡，从而降低了并行效率。针对这个问题华中科技大学的程荣[29]提出了一种数据并行BP神经网络的动态负载平衡方案。郑庆伟[30]等根据处理机的速度来划分任务的大小，提高了在异构网络环境下并行计算的效率。

文献[31]中分别对基于结构并行和数据并行的BP算法进行了对比，得出数据并行的方式适合在机群系统下面实现。

1.3 论文的主要内容

BP 神经网络在对海量数据处理的过程中，训练速度成为了其被进一步应用所面临的主要问题。如何缩短 BP 算法的训练时间，提高训练的精度也成为了研究中的焦点。本文的主要目标是将并行计算结合到 BP 神经网络的训练之中，充分发挥并行处理的优势，提高神经网络训练的速度。

(1) 通过对 BP 算法的改进算法进行研究。找到一种性能相对较优，较适合采取数据并行方式的改进算法。

(2) 研究 BP 神经网络并行化的方法，找到一种适合于机群环境下的 BP 并行算法。并在机群环境下，基于 MPI 并行环境，结合弹性 BP 算法实现基于数据并行的 BP 算法。

(3) 在数据并行的 BP 算法的基础上，改进了样本数据的存储方式。同时提出筛选初始权值的改进方法。

(4) 通过实验，将标准 BP 算法和并行 BP 算法的训练时间进行对比，验证并行算法的有效性和高效性；通过比较不同处理机个数的并行网络，总结出并行 BP 网络中处理机个数与加速比以及并行效率之间的关系；对改进的并行算法进行实验，并对其性能进行评估。

1.4 论文的组织结构

论文共分六个部分，各部分的内容组织如下。

第一章介绍了课题的研究背景和意义、国内外研究现状、课题的主要研究工作和论文的组织结构。

第二章介绍了课题相关的基本理论和技术，包括 BP 神经网络的结构、神经网络原理和 BP 算法的原理。针对 BP 算法的局限，介绍了 BP 的改进算法。

第三章对并行计算和并行计算环境 MPI 进行了介绍。同时给出了神经网络的并行化策略，并对各种并行方式的优缺点进行分析。

第四章给出了基于数据并行的 BP 神经网络的设计与实现，从样本数据的预处理，网络结构的确定和并行计算环境的建立三个方面介绍了系统的实现过程。并在并行算法上进行了改进。

第五章为基于数据并行 BP 算法的实验结果，并对实验结果进行了对比与分析。

第六章对论文中所作的工作做出总结，同时对下一步要进行的工作和可能的研究方向进行了展望。

2 BP 神经网络算法及其改进

2.1 BP 神经网络结构

2.1.1 神经元模型

神经元是神经网络的基本组成单元，图 2.1 为一个神经元的模型。$(x_1,x_2,...,x_n)$ 为神经元的输入信号，$(w_1,w_2,...,w_n)$ 为各个输入与神经元之间的连接权值。神经元接受来自外部的输入信号，将信号与各个边上的权值相乘并求和即 $\sum_{j=1}^{n} w_j x_j$，将求得的加权减去阈值 θ，再将这个结果传递给函数 $f(u)$，得到最后的输出。其中函数 $f(u)$ 称为传递函数，通常为 sigmoid 函数：$f(u)=\frac{1}{1+e^{-u}}$。$f(u)$ 函数图形如图 2.2 所示。因此神经元的作用就是将来自外部的多个输入进行处理得到相应的输出。

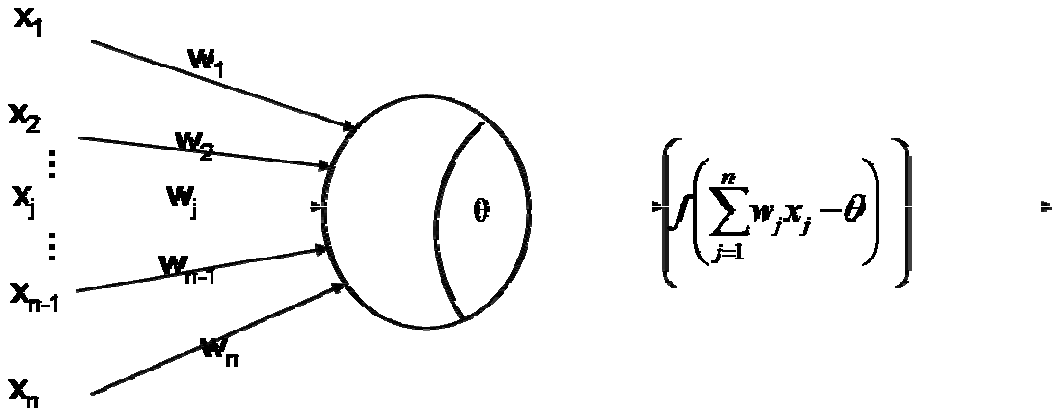


图 2.1 神经元模型

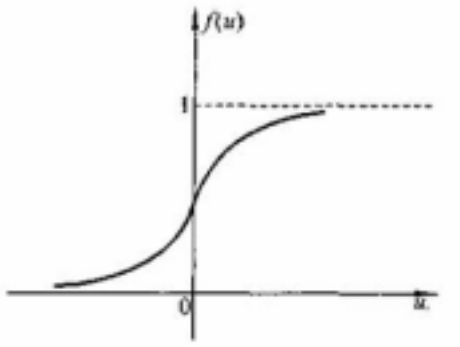


图 2.2 Sigmoid 函数的函数图形

2.1.2 神经网络模型

神经网络是由多个神经元相互连接组成的网络。按照连接方式可分为前馈型和反馈型神经网络。图 2.3 是一个典型的前馈型网络。网络分为三层结构，分别为输入层 a，隐含层 b，输出层 c。输入信号通过输入层向隐含层再向输出层传递，同一层之间的神经元没有信息传递。对于每个神经元而言，它只接收上一层神经元的输出，通过处理之后得到该神经元的输出，并将输出作为输入传递给下一层的神经元。

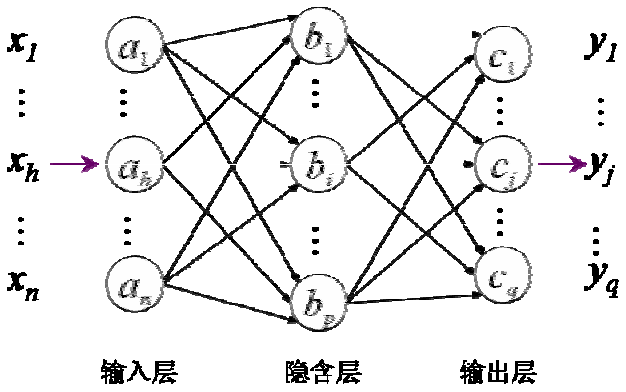


图2.3 神经网络模型

2.2 BP 神经网络算法

2.2.1 神经网络的学习过程

首先介绍神经网络是如何解决实际问题的。如果把我们要求解问题的已知条件记为 X，问题的结果记为 Y，将 X 和 Y 分别作为某个神经网络的输入和输出，当输入 X 经过这个神经网络之后可以得到对应的结果 Y，那么这个神经网络就可以求解我们需要的问题。显然这里的主要问题是找到这种满足条件的网络。而网络的结构是可以根据所研究的问题而确定下来，只有网络的权值是可以改变。因此我们的关键任务是要找到网络的权值 W。实际情况是，问题的部分已知条件 X 和对应的结果 Y 已经被给出了，那么就可以通过已知的（X，Y）来求得这个神经网络。而（X，Y）被称作训练样本。

因此，神经网络的训练就是这样一种过程。通过已知的样本对神经网络进行训练，得到需要的权值和阈值。当有新的问题 X^{new} 来临时，可以通过训练得到的神经网络求得新问题的答案 Y^{new}。

在神经网络训练的初期，网络的权值是经过随机的初始化的，因此样本中的输入 X 经过这个随机初始化的网络后得到的输出很可能与已知的期望输出 Y 不相符，因此需要对网络的权值进行调整。而调整的依据就是得到的输出与期望输出之间的误差。将调整后的网络再次应用于输入样本 X，就可以得到新的误差，从而继续调整网络。通过不断的迭代这个权值调整的过程，当最后的输出误差满足一定要求时，可以认为我们需要的网络已经找到，即网络训练结束。图 2.4 显示了神经网络的训练过程。

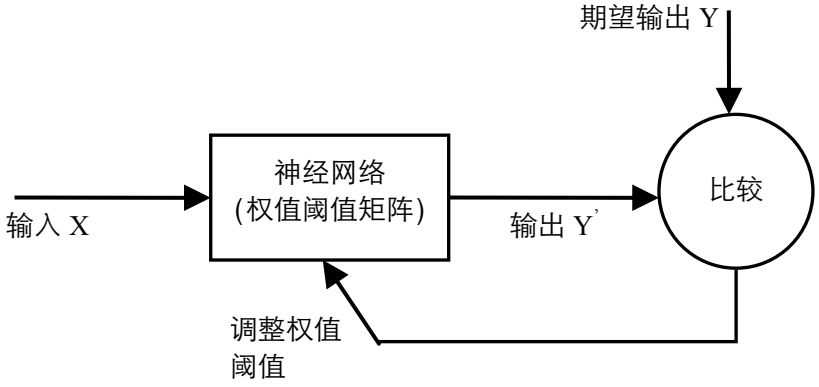


图 2.4 神经网络的训练过程

2.2.2 BP 算法的数学推导

在BP算法中，网络的权值是根据网络的误差来调整的。网络的误差为由输出值与期望值之间的均方误差表示，记为

$$E=\frac{1}{2}\sum_{j}(y_j^t-y_j)^2 \tag{2.1}$$

其中 y_j^t 为第j个输出节点的期望输出， y_j 为第j个节点的实际输出。如果将网络的所有权值组成的向量记为 $[w_{11},w_{12},...w_{ij},...]$，那么在训练样本确定的条件下，网络的误差 E 只与网络的权值有关，因此可以认为误差 E 是关于权值变量 $[w_{11},w_{12},...w_{ij},...]$ 的函数。训练网络就是使误差 E 最小化，因此可以将训练过程转化成函数求最小值问题。

传统的BP算法是基于梯度下降法来求最小值的。梯度计算公式为

$$\nabla E=\nabla f(w_{11},w_{12},...w_{ij},...)=\left[\frac{\partial f}{\partial w_{11}},\frac{\partial f}{\partial w_{12}},...,\frac{\partial f}{\partial w_{ij}},...\right]^T \tag{2.2}$$

以二维的情况为例，图2.5为函数 $E = f(w_{11}, w_{12})$ 的曲面图形，图2.6为该函数图形的平面等高图。

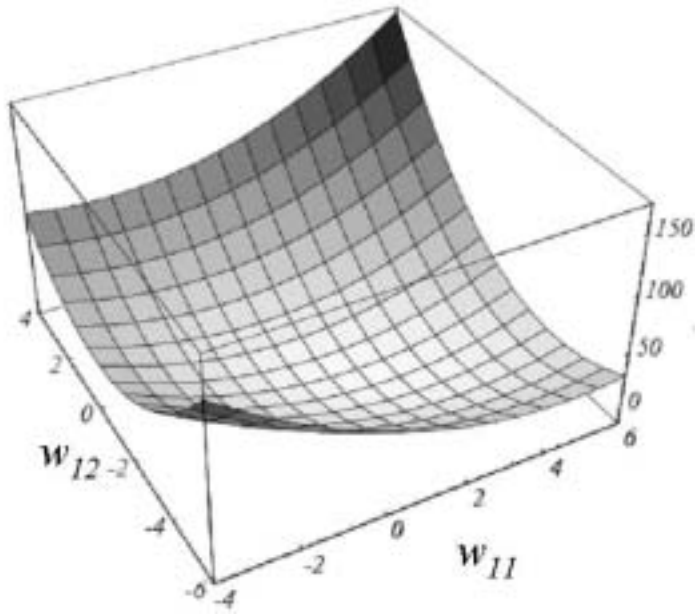


图2.5 误差函数曲面

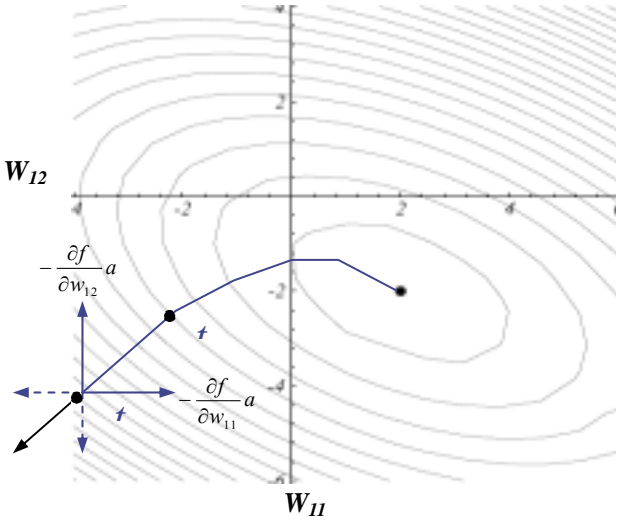


图2.6 函数的平面等高图

图2.6显示了梯度下降法的过程，即在初始点处求得该点的梯度，在梯度的负方向上通过固定的步长a找到下一个点，并重复以上过程最终找到了该函数的极小值点。由于梯度的方向是函数值增长最快的方向，那么沿梯度相反的方向即为函数减少最快的方向，如果每次都在梯度相反的方向上搜索，经过若干步之后可以到达极小值点。

对于梯度下降法，首先要计算梯度，因此需要求得误差对各个权值的偏导数 $\frac{\partial E}{\partial w_{ji}}$。以图2.3中的神经网络为例，对BP算法进行数学推导[28]。设a、b、c三层对应网络的输入层，隐含层和输出层。每层的节点个数分别为n、p、q。a_h为输入层的第h个神经元的输出，b_i为隐含层第i个神经元的输出，c_j为输出层的输出。$[x_1,x_2,...,x_n]$，$[y_1^t,y_2^t,...,y_q^t]$分别为训练样本的输入和输出。

对于输出层c，根据复合函数求导公式，偏导数计算公式为

$$\frac{\partial E}{\partial w_{ji}}=\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial w_{ji}} \tag{2.3}$$

误差的表达式为

$$E=\frac{1}{2}\sum_j (y_j^t-y_j)^2 \tag{2.4}$$

于是

$$\frac{\partial E}{\partial y_j}=-(y_j^t-y_j) \tag{2.5}$$

因为隐含层的输出为

$$y_j=sig(\sum_i w_{ji}b_i) \tag{2.6}$$

则

$$\frac{\partial y_j}{\partial w_{ji}}=sig'(\sum_i w_{ji}b_i)b_i \tag{2.7}$$

将(2.5)式和(2.7)式代入(2.3)式中可得

$$\frac{\partial E}{\partial w_{ji}}=\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial w_{ji}}=-(y_j^t-y_j)sig'(\sum_i w_{ji}b_i)b_i=\delta_j^c b_i \tag{2.8}$$

其中 δ_j^c 为输出层的敏感度

$$\delta_j^c=-(y_j^t-y_j)sig'(\sum_i w_{ji}b_i) \tag{2.9}$$

对于隐含层，敏感度为

$$\delta_i^b=\frac{\partial E}{\partial b_i}=\frac{\partial E}{\partial \sum_h w_{ih}a_h}\cdot\frac{\partial \sum_h w_{ih}a_h}{\partial b_i}=\frac{\partial E}{\partial \sum_h w_{ih}a_h}\cdot sig'(\sum_h w_{ih}a_h) \tag{2.10}$$

其中 $\frac{\partial E}{\partial \sum_h w_{ih} a_h}$ 为误差 E 关于隐含层 b 的第 i 个神经元输入的偏导，由于误差从该神经元传播到下一层与其相连的所有神经元，因此需要求得下一层的偏导之和，即

$$\frac{\partial E}{\partial \sum_h w_{ih} a_h} = \sum_j \frac{\partial E}{\partial c_j} \cdot \frac{\partial c_j}{\partial \sum_h w_{ih} a_h} = \sum_j \delta_j^c w_{ji} \tag{2.11}$$

又因为

$$\frac{\partial E}{\partial w_{ih}} = \delta_i^b a_h \tag{2.12}$$

将式(2.11)和式(2.10)代入式(2.12)中，可以求得隐含层的偏导数

$$\frac{\partial E}{\partial w_{ih}} = \sum_j \delta_j^c w_{ji} sig'(\sum_h w_{ih} a_h) \cdot a_h \tag{2.13}$$

再根据公式

$$w(t) = w(t-1) - a\frac{\partial E}{\partial w} \tag{2.14}$$

即可求得每一次权值的改变量。

2.2.3 BP 算法的局限

BP网络的训练是一种最优化问题，通过样本集来对网络权值进行调整，直到使输出误差最小。但是基于梯度下降法的BP算法存在着限制与不足。

(1) 在陡峭的波谷中振荡。误差曲面通常也存在陡峭的波谷，如果训练步长选择不当，网络的权值调整过大，网络不能收敛到极小点上，而在极小点周围来回振荡。

(2) 在平坦区域连接权值调整缓慢。误差曲面往往存在一些平坦区，由于传递函数的导数趋于零，使得修正量 Δw 趋于零，因而网络权值的调整过程几乎处于停顿状态。即出现所谓的“网络麻痹现象”。

(3) 收敛到局部极小点。由于BP算法采用的是梯度下降法，训练是从某一初始点沿某一方向逐渐达到误差的最小值。而网络误差曲面是高维的凹凸不平的复杂曲面，因此，在学习过程中可能陷入某个局部极小点[32]。

2.3 BP 算法的改进

BP改进算法主要分为两类，第一类是基于启发式改进方法，这源于对标准反向

传播算法特定性能的研究，包括可变的学习速度，使用动量项等。另一类是基于数值优化技术，包括共轭梯度算法，牛顿法，LM算法等。虽然可变学习速度和动量法可以在一定程度上加速网络的训练，但是训练的结果对算法参数的选择十分敏感，需要通过通过多次实验来确定一个较好的参数；对于牛顿法和LM等算法，虽然训练速度很快，但是需要额外的存储空间，且不易实现并行化，因此本文采用弹性算法这种改进的BP算法作为串行算法，并在其基础上并行化。

2.3.1 弹性 BP 算法

标准的梯度下降算法的权值改变量由学习速度和梯度的大小决定，其公式为 $\Delta w = -a\frac{\partial E}{\partial w}$。而随着网络误差的下降，误差对权值的偏导数不断减小；当网络误差下降到一定精度时，偏导数也减小到一定程度，导致权值的修正值也很小，从而算法对网络的调整能力不断减小，最后网络的调整达到几乎停滞的地步。

弹性算法不用误差函数对网络权值的偏导数来调整网络，也不用学习速率来调整网络，而是另外设置一个“权更新值Δ”来调整网络，网络权值的修正值 Δw 由权更新值Δ计算。为了叙述方便，以BP 网络中某一权值 w_{ij} 的训练为例，来说明弹性算法的网络权值的训练过程，权值 w_{ij} 的修正值 Δw_{ij} 计算公式[7]为

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} & (\frac{\partial E^{(t)}}{\partial w_{ij}} > 0), \\ +\Delta_{ij}^{(t)} & (\frac{\partial E^{(t)}}{\partial w_{ij}} < 0), \\ 0 & (\text{其他}) \end{cases} \tag{2.15}$$

式(2.15)中，t为训练次数。弹性算法应用批处理[33]训练方式。$\frac{\partial E^{(t)}}{\partial w_{ij}}$ 是第t次训练时训练集的所有训练样本的梯度累加和。弹性BP算法的原理是：如果误差函数对网络权值偏导数为正，说明网络误差增大，那么网络权值应该减小，因此，权值修正值 Δw_{ij} 取 $-\Delta_{ij}$；反之，如果偏导数为负，说明网络误差减小，为了加快网络的收敛，网络权值可以增大，修正值 Δw_{ij} 取 Δ_{ij}。

存在一种特殊情况，如果前后两次训练的误差函数对网络权值的偏倒数改变符

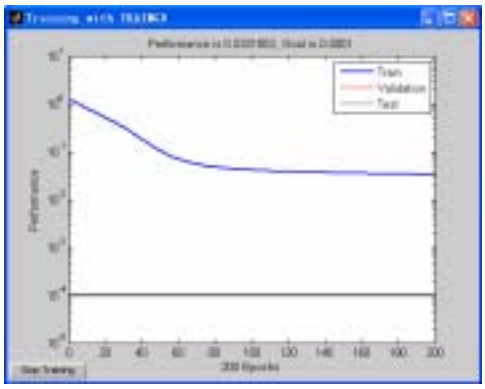
号，说明前一次训练的网络权值调整太大，使$\frac{\partial E^{(t-1)}}{\partial w_{ij}} \times \frac{\partial E^{(t)}}{\partial w_{ij}} < 0$，从而某个极小值点$\frac{\partial E^{(t-1)}}{\partial w_{ij}} = 0$被丢失，则前一次训练权值的修正值应该恢复[34]。

权更新值Δ_{ij}根据前后两次训练产生梯度信息的符号进行相关的自适应的学习更新，其更新公式[7]为

$$\Delta_{ij}^{t} = \begin{cases} \eta^{+} \times \Delta_{ij}^{(t-1)} & (\frac{\partial E^{(t-1)}}{\partial w_{ij}} \times \frac{\partial E^{(t)}}{\partial w_{ij}} > 0), \\ \eta^{-} \times \Delta_{ij}^{(t-1)} & (\frac{\partial E^{(t-1)}}{\partial w_{ij}} \times \frac{\partial E^{(t)}}{\partial w_{ij}} < 0), \\ \Delta_{ij}^{(t-1)} & (\text{其他}) \end{cases} \tag{2.16}$$

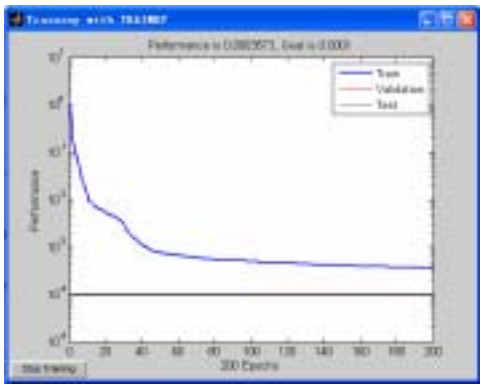
2.3.2 弹性 BP 算法实验结果

本实验利用BP神经网络解决函数逼近的问题。逼近的函数为正弦函数。样本数据为51组；网络结构为，输入层1个节点，隐含层8个节点，输出层1个节点；最小误差为0.0001；最大迭代次数为200。原始算法和弹性算法的误差曲线如图2.7所示。



(a)

图 2.7 (a)原始 BP 算法的训练误差曲线



(b)

(b)弹性 BP 算法的训练误差曲线

从图 2.7 中可以看出，弹性 BP 算法在训练速度和精度上较原始 BP 算法有较大的提高。在相同的迭代步数之内，弹性 BP 算法达到了 10^{-3} 的精度，而原始 BP 算法只达到了 10^{-1}。在最初 40 步迭代内，弹性算法已经具有非常快的训练速度，如果精度要求不是很高，则只需要很少的步数即可收敛。而原始 BP 算法的下降速度则非常缓慢，在 60 步之后已经趋于平缓。

同时因为该算法相对于牛顿法和 LM 算法对存储需求不大，对于规模较大的神经网络更加适用。又因为其对参数的设置不敏感，不会因为参数的不同而变得不稳定。因此本文以弹性 BP 算法为基础，研究该算法的并行化。

2.4 本章小结

本章主要讲述了BP神经网络的结构，BP算法的学习过程。在深入分析BP算法的基础上，得出BP算法存在训练时间长、收敛速度慢等局限性。针对这些局限性，介绍了改进的BP算法。BP改进算法可以分为基于启发式的改进方法和基于数值优化技术的改进方法。其中弹性BP算法作为一种较为优秀的算法既利于实际应用，而且也适用于并行实现。于是，本文以弹性BP算法为基础，研究将BP算法并行化的方法，提高神经网络训练的速度。下一章介绍神经网络的并行化方法。

3 神经网络并行化

神经网络固有的并行性为并行计算提供了基础，然而由于硬件的限制，在单机上的神经网络往往通过串行化的程序来实现，这样大大限制了神经网络的并行特点，因此有必要研究 BP 神经网络的并行化方法。

3.1 并行计算介绍

随着计算机应用的不断深入和扩展，需要速度更快、性能更高的计算机系统。虽然超大规模集成电路技术的发展极大地提高了计算机器件的速度，从而使计算机系统的性能大幅度提高。但是，仍然满足不了当前对更高性能计算机系统的需要。人们基于并行处理的思想，开辟了并行处理和并行处理计算机的研究领域。本节对并行计算及相关知识作概念性的介绍。

3.1.1 并行计算类型

并行是指在同一时刻或在同一时间间隔内完成两种或两种以上性质相同或不相同的工作。提高计算机系统处理速度的重要措施是增加处理的并行性。而指令和数据是计算机解决问题所涉及到的两个基本方面，即让计算机“执行什么”样的操作和对“什么对象”执行相应的操作。因此并行计算的类型分为两种：任务并行和数据并行[35]。

(1) 任务并行

将总任务划分成多个子任务，由不同的处理机分别处理不同的子任务，通过任务的重叠以达到并行的目的。而每台处理机可以处理相同的或不同的数据。这种方式的好处是：即使各个子任务之间相互依赖也可以实现并行，比如流水线方式。

(2) 数据并行

将数据划分给不同的处理机，每个处理机执行相同的任务来对数据进行处理，以达到减少计算时间的目的。例如要求数组 a[100]的和，按照数据划分的方式将数组的前 50 个元素 a[0]到 a[49]分配给处理机 A，将后 50 个元素 a[50]到 a[99]分配给处理机 B，然后让 A 和 B 同时对 50 个数求和，最后将两边的结果相加。可以看到 AB 处理机执行相同的任务，而所处理的数据不同。这种并行方式要求所处理的数据之

间没有相互依赖关系。比如求 Fibonacci 数列时，后一个数是前两个数之和，这种情况下无法实现数据并行。

在实际应用中，通常将任务并行和数据并行结合起来实现并行化。

3.1.2 并行计算机

根据计算机硬件支持并行方式的不同，将并行机分为多核计算机，对称多处理机，分布式处理机。多核处理机和对称多处理机采用的的是一台机器上的多个处理部件或芯片来完成并行任务；而包括网格以及机群在内的分布式处理机是通过多台处理机来执行相同的任务[35]。

1．多核处理器(Multi-core computer)

多核处理器是指在一个处理器上集成多个运算核心（内核），从而提高计算能力。其实现方式是将多个内核封装在一个 Die（晶元）上，通过直连架构连接起来。处理单元之间共享总线，二级缓存；或者将放在不同 Die（晶元）上的多个内核封装在一起，各处理单元通过直接构架与外部 I/O 相连，并且使用独立的缓存。这两种方式的典型代表是 Intel 的 Core 2 和 AMD 的 dual-core 处理器。多核处理器在进行多任务处理时较单核更高效，但是对单任务而言需要程序或软件的支持才能更好的发挥多核的效率。多核的技术主要面向个人电脑，如今已经成功的应用在游戏机 (比如 PS3 的 8 核 Cell 处理器和 Xbox 360 中的 3 核 Xenon 处理器)、网络处理器、数字信号处理器以及 GPU 中。

2．对称多处理机(SMP, Symmetric Multiprocessor)

最主要的特征是系统的对称性。在这种架构中，多个处理器运行操作系统的单一副本，并共享内存和一台计算机的其他资源。所有的处理器都可以平等地访问内存、I/O 和外部中断。优点是并行度很高，工作负载能够均匀地分配到所有可用处理器之上，但是由于系统总线的带宽是有限的，故处理器的数目是受限的。

3．分布式计算(Distributed computing)

(1) 网格(Grid)

网格是以 Internet、Web 技术和分布计算技术为基础，将地理上分散的各类计算资源、存储资源、数据资源、应用资源、仪器设备等构成统一的虚拟环境，采用开放标准的协议，实现资源的有效共享，为动态参与的、由多机构所形成的虚拟组织协同完成高性能计算、信息处理等各类应用，提供可扩展的、安全的、一致的、不

同等级质量的服务。

(2) 大规模并行处理机(MPP, Massive parallel processor)

MPP 由许多松耦合的处理机组成，采用并行的方式来完成任务。MPP 与 SMP 类似，不同点是每个处理机的 CPU 都有自己私有的资源，如总线、内存、硬盘等。每个处理机中都有独立的操作系统。这种结构最大的特点在于不共享资源，所以不会存在 SMP 中各 CPU 争夺资源而造成的瓶颈问题，但是也给编程带来了困难。

(3) 机群系统(Cluster)

机群系统是利用高速通用网络将一组高性能工作站或高档 PC 机，按照某种结构连接起来，并在并行程序以及可视化人机交互环境支持下，统一调度，协调处理，实现高效并行处理系统。从结构和结点间的通信方式来看，它属于分布式存储系统，主要利用消息传递方式实现各主机之间的通信，由建立在一般操作系统之上的并行编程环境完成系统的资源管理及相互协作，同时也屏蔽工作站及网络的异构性，对程序员和用户来说，机群系统是一个整体的并行系统。

机群系统的特征是：①机群系统的每个节点都是一个完整的工作站；②每个节点通过低成本的网络互连；③机群系统的各个节点具有本地内存和磁盘；④每个节点都有完整的操作系统。

近几年来，随着计算机运算速度大幅提高，硬件价格低廉，同时局域网技术已十分成熟，高性能互联网的拓扑结构和处理器之间的距离已经不再是影响并行机系统性能的关键因素，为利用 PC 机组建并行计算集群提供了高性价比条件，机群从而被各种用户所接受，正逐步成为并行编程的主要平台[36]。

3.1.3 并行算法

传统的计算机程序是通过串行算法来实现的，而采用并行算法设计的程序，可以同时在多个处理机上同时运行。通过创建一些可以同时执行的进程，这些进程之间进行相互协作和消息传递，从而达到共同求解问题的目的。

(1) 并行算法的分类

并行算法是给定并行模型的一种具体明确的解决方法和步骤。按照不同的划分方法，并行算法有多种不同的分类。

根据运算的基本对象的不同可以将并行算法分为数值并行算法和非数值并行算法。这两种算法也不是截然分开的，比如在数值计算的过程中会用到查找匹配等非

数值计算的成分，当然非数值计算中也一般会用到数值计算的方法。划分为什么类型的算法主要取决于主要的计算量和宏观的计算方法。

根据进程之间的依赖关系可以分为同步并行算法、异步并行算法和纯并行算法。对于同步并行算法，任务的各个部分是同步向前推进的，有一个全局的时钟来控制各部分的步伐，而对于异步并行算法各部分的步伐是互不相同的，它们根据计算过程的不同阶段决定等待、继续或终止。纯并行算法是最理想的情况，各部分之间可以尽可能快地向前推进不需要任何同步或等待，但是一般这样的问题是少见的。

根据并行计算任务的大小，可以分为粗粒度并行算法、细粒度并行算法以及介于二者之间的中粒度并行算法。一般而言并行的粒度越小就有可能开发更多的并行性提高并行度，这是有利的方面。但是一个不利的方面，就是并行的粒度越小通信次数和通信量就相对增多这样就增加了额外的开销，因此合适的并行粒度需要根据计算量通信量计算速度通信速度进行综合平衡这样才能够取得高效率[37]。

(2) 机群算法的设计原则

对于机群计算，有一个很重要的原则就是设法加大计算时间相对于通信时间的比重，减少通信次数甚至以计算换通信。这是因为，对于机群系统，一次通信的开销要远远大于一次计算的开销，因此要尽可能降低通信的次数，或将两次通信合并为一次通信。基于同样的原因，机群计算的并行粒度不可能太小，因为这样会大大增加通信的开销。如果能够实现计算和通信的重叠，那将会更大地提高整个程序的执行效率。

3.2 并行编程环境 MPI 介绍

MPI(Message Passing Interface) 是一个消息传递接口的标准，用于开发基于消息传递的并行程序。其目的是为了提供一个实际可用的、可移植的、高效的和灵活的消息传递接口标准。MPI 以语言独立的形式来定义这个接口库，并提供了与 C 和 Fortran 语言的绑定。这个定义不包含任何专用于某个特别的制造商、操作系统或硬件的特性。因此 MPI 在并行计算界被广泛地接受，并被实现在 PC/Windows、所有主要的 Unix 工作站以及所有主要的并行机上。这就意味着一个用标准的 C 或 Fortran，加上 MPI 实现的消息传递并行程序，可以不作修改地运行在单台 PC 机、单台工作站、工作站网络和 MPP 系统上，无论这些机器由谁制造，用的是什么操作系统[37]。

3.2.1 MPI 的特点

(1) MPI 是一个库，而不是一门语言。因此它本身不能自编译。因此需要其它的编程语言来调用。现在它支持 C、C++、FORTRAN77、FOTRAAN 90 四种编程语言。MPI 必须和具体的编程语言结合才能使用。

(2) MPI 是一种标准或规范的代表，而不是特指某一个对它的具体实现。迄今为止，很多的并行计算机制造商都提供对 MPI 的支持，可以在网上免费得到 MPI 在不同并行计算机上的实现。一个正确的 MPI 程序可以不加修改地在所有的并行机上运行。同时，也有一些商业的 MPI 实现的版本。

(3) MPI 是一种消息传递编程模型，并成为这种编程模型的代表和事实上的标准。MPI 虽然很庞大，但是它的最终目的是服务于进程间通信的。消息传递方式是广泛应用于多类并行机的一种模式，特别是那些分布存储并行机，尽管在具体的实现上有许多不同，但通过消息完成进程间通信的概念是容易理解的[38]。

3.2.2 MPI 的实现

MPICH 是一种最重要的 MPI 实现，它是一个与 MPI-1 规范同步发展的版本，每当 MPI 推出新的版本，就会有相应的 MPICH 的实现版本。它可以免费从 http://www-unix.mcs.anl.gov/mpi/mpich 取得。

LAM (Local Area Multicomputer)也是免费的 MPI 实现，它主要用于异构的计算机网络计算系统。由 Ohio State University 开发，它目前的最新版本是 LAM/MPI 6.3.2，可以从 http://www.mpi.nd.edu/lam/download/ 下载。

3.3 神经网络并行化策略

神经网络本身就具有并行处理的特点，传统的人工神经网络由于硬件条件的限制，仅仅是将这种并行的处理方式以串行化程序模拟出来，因此在计算效率和速度上受到非常大的限制。随着并行计算技术的发展和神经网络的应用越来越广泛，国内外许多专家学者开始了并行神经网络的研究，并且取得了很多有价值的成果。BP 网络的并行化主要有两种思路。

3.3.1 结构并行

根据神经网络自身具有并行处理的特点，将具有 N 个神经元的神经网络分配给

P 个处理机(N>P)，这 N 个神经元被分成 P 组，分在同一组的神经元权值的改变由一个处理机单独计算并与其他处理机通信来实现。按结构分割的方法包括横向分割法，按层分割法。

1．横向分割

如图 3.1 所示，先将网络中每层的神经元按照处理机个数进行划分，比如有三个处理机，则每层神经元都被分为 3 组，然后将各层每组的神经元分配到对应的处理机之中，则每个处理机都含有输入层、隐含层和输出层的神经元。每个处理机中神经元存储的权值是所有与其相连权值。

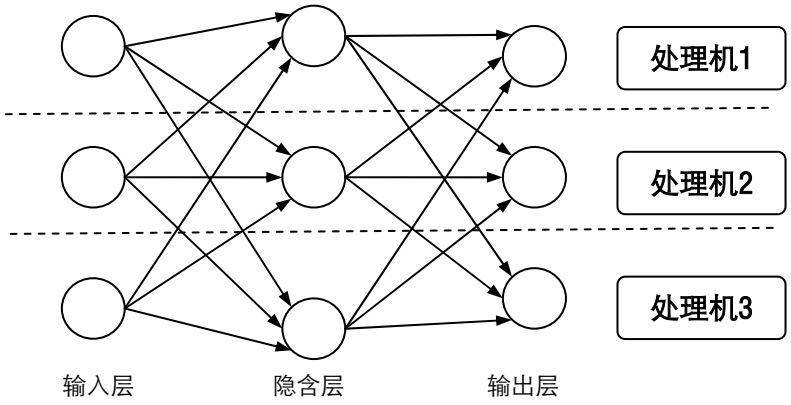


图 3.1 横向分割神经元

在计算时，各个处理机可以通过输入值与该处理机存储的权值单独计算出其隐含层的输出值，然后在各处理机之间相互交换隐含层的输出信息，最后计算出各自输出层的输出值；在误差反传阶段，隐含层的误差是由输出层的误差加权获得的，每个处理机上的神经元通过其隐含层和输出层之间的权值计算出隐含层的误差，然后通过输出层和隐含层的误差就可以计算出各个权值的改变量。

这种并行方式存在以下问题。

(1) 从负载平衡来看，每个处理结点上分配的神经元数应尽量相等，即包含相同个数的输入、隐含和输出神经元。对于动态确定的神经网络结构，意味着无法保证负载的平衡。

(2) 从处理节点之间的通信上来看，隐含层神经元在网络正向传播时都需要相互进行通信，如果隐含层数量增加通信量也会相应加大，由此带来的后果是加速比降低，效率下降。

(3) 从同步的角度看，由于每个神经元只有得到所有输入信号后，才能进行相

应处理，因此不同处理结点中子网层之间必须同步。否则，无法进行下一层神经元的处理。在负载平衡难以实现的条件下，如果某一处理结点的负载量大而引起其他结点的等待，加上结点间的数据通信开销，这种方式在性能上就难以提高。

因此，对于输入、输出层结点数受具体应用限制的前向型神经网络，这种并行方式所得到的加速比也不会很大。即使在训练集较大时，处理结点间的同步也不会使训练速度有明显提高[31]。

2．按层分割

按层分割也叫流水线方式。是将神经元按照不同的层分配到不同的处理机之中的划分方法。如图 3.2 所示，一个含有一个隐含层和一个输出层的网络被划分给两个处理机，每个处理机上的神经元都存储了所有与其相连的权值。

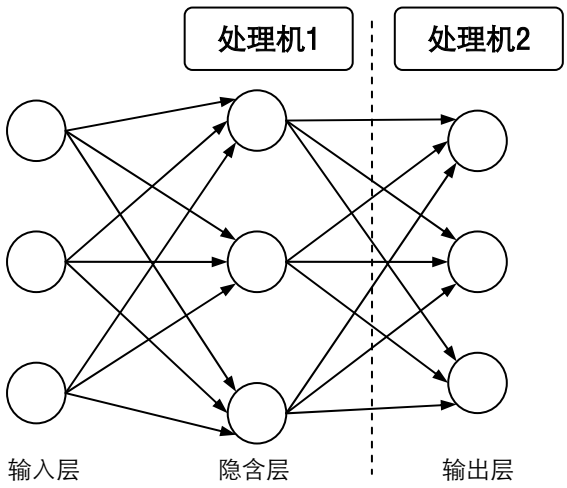


图 3.2 按层分割神经元

其处理过程是：首先，隐含层的处理机计算出训练样本 A 隐含层的输出值。然后输出层处理机读入这个值并且计算出 A 的输出值和误差值。在输出层处理机进行计算的同时，隐含层处理机就会对下一个样本 B 进行处理。接着它又接收到输出层处理机计算出的 A 误差值，然后可以计算出 A 在隐含层的误差值。于是可以计算出样本 A 经过网络后，网络权值的改变量。这个时候输出层处理机又在计算 B 样本的输出值和误差值。图 3.3 中可以清楚的看到，隐含层处理机和输出层处理机采用流水线的方式，通过任务的重叠来达到并行的目的。

这种并行方式存在以下问题。

(1) 负载平衡难以实现。在正向传播的过程中，输入值与各权值的乘积是主要的运算。假设网络的输入层、隐含层和输出层神经元数目分别为 n、p、q，那么隐含

层处理机进行乘法运算的数量为 n×p；输出层处理机进行乘法运算的数量为 p×q。两处理机之间的运算量比值为(n×p)/(p×q)=n/q。即输入层神经元个数比上输出层神经元个数。而在大多数情况下，输入层神经元个数是要多于输出层神经元个数的。因此隐含层处理机的运算量就要比输出层处理机的运算量大，隐含层处理机就成为了整个网络计算的瓶颈。而在反向传播的过程中，输出层只需要通过减法运算计算出输出层的误差，而隐含层需要将输出层的误差与权值相乘求得加权和，因此其运算量又进一步加大。负载平衡难以实现。

(2) 通信开销也限制了并行效率的提高。在这种并行模式下，隐含层处理机和输出层处理机之间在处理一条样本时需要交换各自神经元的信息，因此它属于“细粒度”的并行方式。如果训练样本集非常大，那用于通信的时间将会占去网络训练的大部分时间，所带来的后果将是并行效率低下。

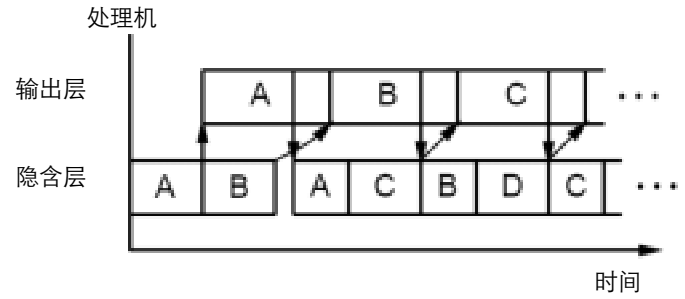


图 3.3 流水线方式

3.3.2 数据并行

通过前面的分析可以看到，结构并行是将神经网络的结构进行分割的一种并行方式，它属于“细粒度”的并行。虽然在大型的并行计算机上，通信量的增加不会有非常严重的影响，但在机群系统中，通信是通过网络来实现的，网络传输的速率要大大低于总线的信息传播速率。尤其是当训练样本非常大，网络结构非常复杂的情况下，采用结构并行的方式就难以实现。因此，又有了基于数据并行的划分方式。这种方式中，每个处理机节点都存储有一个完整的网络结构，通过将训练数据进行划分，各处理机针对不同的数据进行训练，然后将训练的结果相互交换，获得最终的网络。

1．批处理和在线处理训练模式

批处理和在线处理是神经网络权值更新的两种不同方式。而数据并行的神经网络是基于批处理的训练模式的。

(1) 在线处理模式

在这种方式下，网络的权值是在每条样本被处理完之后进行更新的。通过将样本连续地输入给网络，每次经过网络，网络的权值都会被更新。这种方式的缺陷是网络有可能对后输入的样本训练效果比较好，而对于最开始输入的样本因为训练时间长久而“遗忘”。而且采用这种训练模式，网络的权值是串行更新的，无法将其用于数据并行的神经网络之中。

(2) 批处理模式

在这种方式下，网络的权值是在所有的样本被处理完之后进行更新的，这样的一次更新叫做一个 epoch。在在线训练模式中，网络的误差是单条样本所产生的误差，而批处理模式下，网络的误差则是所有样本的误差总和。所求得的各个权值的梯度值也是所有样本作用于网络所产生出的梯度和，通过将各个样本集的梯度综合到一起，以获得更准确的梯度。而且只要样本集是完备的，最后对梯度的估计也是精确的。同时，网络是在所有的样本被处理之后再更新的，那么样本输入的顺序不会对网络的训练造成影响，这也为数据并行提供了条件。

事实上，在 BP 算法的数学推导过程中，网络的误差即所有样本均方误差的期望值被第 k 次均方误差所代替[33]。在 2.2.2 节中公式 2.1 实际上是第 k 条样本的均方误差，可以写成

$$E(k)=\frac{1}{2}\sum_{j}(y_j^t(k)-y_j(k))^2 \tag{3.1}$$

又因为所有样本所产生的总误差为 $E=\sum_{k}E(k)$，于是真正的网络误差为

$$E=\frac{1}{2}\sum_{k}\sum_{j}(y_j^t(k)-y_j(k))^2 \tag{3.2}$$

所以权值变量在误差函数上的梯度为

$$\frac{\partial E}{\partial w_{ij}}=\frac{\partial\sum_{k}E(k)}{\partial w_{ij}}=\sum_{k}\frac{\partial E(k)}{\partial w_{ij}} \tag{3.3}$$

所以梯度是所有样本的梯度累加和。这里用全局的梯度取代了局部的梯度，因此对网络权值的整体改变方向估计更加准确，也有利于收敛速度的加快。

2．样本数据并行

样本数据并行是基于批处理模式的。由于最后所获得的梯度是所有样本产生的梯度和，那么权值的改变量 Δw 也要通过每个样本对网络的改变量 $\Delta w(k)$ 累加获得，即

$$\Delta w_{ij} = a\frac{\partial E}{\partial w_{ij}} = a\sum_{k}\frac{\partial E(k)}{\partial w_{ij}} = \sum_{k}\Delta w_{ij} \tag{3.4}$$

由于样本的输入顺序对最后的结果没有影响，那么可以将 $\Delta w(k)$ 的计算分配给不同的处理机，最后将这些处理结果累加起来。其示意图如图 3.4 所示。

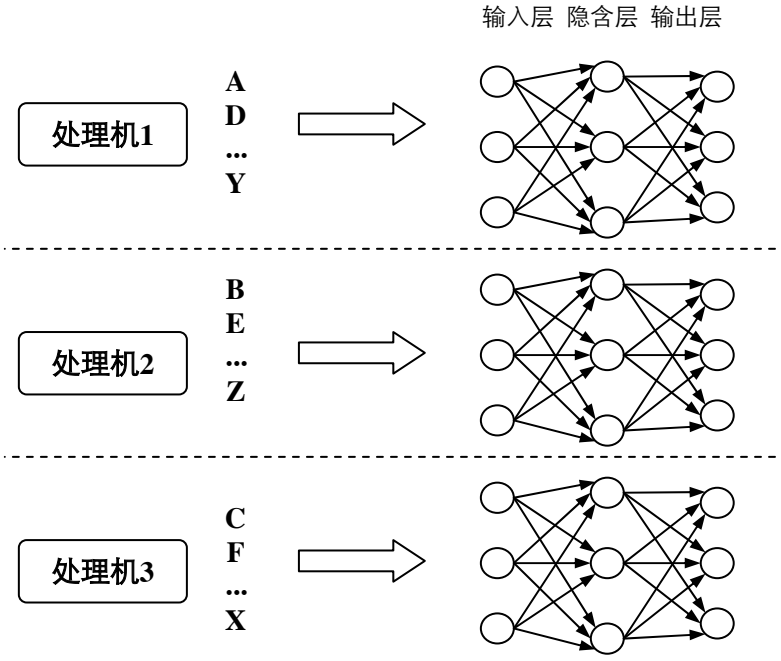


图 3.4 训练样本数据并行示意图

因此样本数据并行的方式是这样的：将训练样本平均分配到各个处理机，在每个处理机上分别对这些样本进行训练。对于每一对训练集{P(k), T(k)}，每执行一次反向传播计算，神经元都有权值改变 $\Delta w(k)$，在施加了 N 对训练集后可以认为神经元权值的改变累计为 $\Delta w = \sum_{k=1}^{N}\Delta w(k)$。这样，就可以在每个处理机内都存储有所有神经元的权值，而把训练集平均分配到各个处理机，各自进行计算，最后把各神经元权值改变的累计 $\Delta w(k)$ 相互交换，统计出总的权值改变量 Δw 并更改各自的权值。

在基于数据并行的方法中，只有在所有的样本经过网络之后(一个 epoch)再进行一次通信，在每次迭代中所有的样本都参与了训练。因此这是一种“粗粒度”的并行方式，它可以很有效的减少网络通信开销，有利于在分布式的网络环境下实现。

而且在这种方式下，由于每个处理机所存储的网络结构是一样的，而且数据也是平均分配到各个处理机之中，只要处理机的运算能力一样，则其所需的运算量是相同的，这样就很容易做到负载平衡。

3.4 本章小结

本章介绍 BP 网络的并行化方法。首先介绍了并行计算，并行计算的方式有按数据并行和按任务并行两种。并行计算机按照体系结构分为多核处理机、对称多处理机和分布式并行处理机。介绍了 PC 机群系统，这是当今较为流行的网络并行环境。接着介绍了并行算法，探讨了在机群系统中并行算法的设计原则。然后介绍了并行编程环境 MPI 及其主要实现。在分析 BP 算法的两种并行策略的基础上，得出数据并行的方式在降低通信、实现负载平衡上有很大的优势。因此本文采用数据并行的方式，在机群的环境下，通过 MPI 编程环境实现 BP 网络的并行化。

4 基于数据并行 BP 神经网络的设计与实现

在第三章中通过对 BP 网络并行策略的分析，得出采用基于数据并行的 BP 算法有利于增加并行粒度，降低通信量，提高并行效率，尤其在机群系统中这种策略远远优于基于结构并行的 BP 算法。因此本章主要从样本数据处理、网络结构确定和并行训练等方面来说明基于数据并行的 BP 神经网络的设计与实现。

4.1 样本数据的预处理

4.1.1 处理缺失数据

在获取样本数据时由于部分数据无法获取或获取数据不完整时，就存在着缺失的数据，而大量缺失的数据对神经网络的训练结果有着非常严重的影响，甚至造成网络无法收敛。因此处理缺失数据成为了数据预处理中重要的一环。

处理的方法是给每一项缺失数据一个替代值，用来补全数据集。这种方法叫插补。插补的方式有多种。

(1) 固定值插补

就是将所有的缺失值用一个固定的数值代替，这种方法处理非常简单，但插补的数据集仍有可能不准确，不完备。

(2) 均值插补

这种方法是用所有记录中某一项的均值代替缺失数据。这种方法较固定值插补更合理，因为综合考虑了样本数据中的其他数据，所以获得的数据集更加准确。本文采用这种方法来填补缺失值。

4.1.2 数据标准化

一般的 BP 算法由于使用 sigmoid 函数作为传递函数，这种函数将大范围的输入压缩为小范围的输出。当输入很大时，sigmoid 函数的变化趋近于 0。也就是说输入的变化对于函数值的影响非常小。这将直接影响网络的训练情况，由于网络对输入值的反映减弱，导致网络陷入停滞状态最终无法收敛到理想的结果。因此需要采取某种方法将输入数据的范围压缩到某个区间，通常是(-1, 1)或(0, 1)范围之内。这种方

法叫数据标准化。

(1) 线性标准化

假定有输入样本集 P

$$P=\begin{bmatrix} p1 \\ p2 \\ \vdots \\ pj \\ \vdots \\ pm \end{bmatrix}=\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1i} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2i} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p_{j1} & p_{j2} & \cdots & p_{ji} & \cdots & p_{jn} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mi} & \cdots & p_{mn} \end{bmatrix} \tag{4.1}$$

其中 $i=1,2,3...n$； $j=1,2,3...m$。P 由 m 条样本组成，每条样本有 n 维。线性标准化首先要求得各个列的最大值和最小值，比如第 i 列最大值和最小值分别为 $Max(i)$、$Min(i)$，则 $Min(i)\le p_{ji}\le Max(i)$，即 $0\le\dfrac{p_{ji}-Min(i)}{Max(i)-Min(i)}\le 1$。令

$$p_{ji}=\frac{p_{ji}-Min(i)}{Max(i)-Min(i)} \tag{4.2}$$

于是 p_{ji} 被规一化到[0，1]范围内。这种方法的缺点是：如果遇到超过最大最小值范围的样本，将会发生错误。尤其是在进行预测时，由于预测数据的范围未知，这种方法有可能影响预测结果。

(2) 零均值标准差标准化

其标准化公式为

$$p_{ji}=\frac{p_{ji}-\mu_i}{\sigma_i} \tag{4.3}$$

μ_i 为第 i 列属性的均值，σ_i 为第 i 列属性的标准差。通过这种方法使各属性项的均值为 0，标准差为 1。这种方法不需要知道属性的最大和最小值，不用担心预测过程中输入超出范围的情况。本文采用这种方法对数据进行标准化。

4.2 网络结构的确定

4.2.1 输入、输出层节点

由于输入和输出层节点数通常根据所研究的问题来确定。输入节点数与问题的影响因素或特征量的个数一致，输出节点数则等于系统待分析的因变量个数。

而在某些情况下，因所研究问题较复杂，影响因素多造成了输入层节点数过多。而在这些影响因素中，有很多都是相关的因素，因此这些因素是可以去除的，从而也可以减少输入层的节点数，降低网络训练的计算量。通常采用主成分分析法来确定主要因素。

主成分分析法是利用降维的思想，从多因素中删去一些重复的因素而转化为少数几个不相关的综合因素的一种多元统计分析方法，通过这种方法提高整个网络的训练效率。在进行主成分分析之前，先要对数据进行标准化，再建立各变量因素的协方差矩阵，并求解矩阵的特征值和特征向量，利用标准化值计算变量之间的相关系数，由k个变量建立k阶相关矩阵，然后选取主成分，计算第i个主成分对总方差的贡献率，即方差贡献率为

$$\rho_i = \lambda_i / \sum_{i=1}^{k} \lambda_i \tag{4.4}$$

其中 λ_i 为第 i 个特征向量，并按贡献率由大到小进行排序，贡献率最大的主成分称为第一主成分，其次称为第二主成分，以此类推，再建立主成分方程，计算主成分值，形成新的训练样本集和预测样本集[39]。

4.2.2 隐含层节点

隐含层神经元数的选择是一个十分复杂的问题。因为没有很好的公式表示，可以说隐层神经元数与所求问题、输入层与输出层神经元的数量、训练样本的数量等都有直接关系。事实上隐层神经元太少不可能将网络训练出来，但太多又使学习时间过长，泛化能力下降，即不能识别以前没有直接接收到的样本，容错性差。

目前多数文献中提出的确定隐层节点数的计算公式都是针对训练样本任意多的情况，而且多数是针对最不利的情况，一般工程实践中很难满足，不宜采用。事实上，各种计算公式得到的隐层节点数有时相差几倍甚至上百倍。为尽可能避免训练时出现“过拟合”现象，保证足够高的网络性能和泛化能力，确定隐层节点数的最基本原则是：在满足精度要求的前提下取尽可能紧凑的结构，即取尽可能少的隐层节点数。研究表明，隐层节点数不仅与输入/输出层的节点数有关，更与需解决的问题的复杂程度和转换函数的形式以及样本数据的特性等因素有关。

同时在确定隐层节点数时必须满足条件：隐层节点数必须小于 N-1（其中 N 为训练样本数），否则，网络模型的系统误差与训练样本的特性无关而趋于零，即建立的

网络模型没有泛化能力，也没有任何实用价值。同理可推得：输入层的节点数（变量数）必须小于 N-1。

总之，若隐层节点数太少，网络可能根本不能训练或网络性能很差；若隐层节点数太多，虽然可使网络的系统误差减小，但一方面使网络训练时间延长，另一方面，训练容易陷入局部极小点而得不到最优点，也是训练时出现“过拟合”的内在原因。因此，合理隐层节点数应在综合考虑网络结构复杂程度和误差大小的情况通过多次实验来确定。

4.2.3 初始权值和阈值

在网络结构和训练样本确定的情况下，初始权值和阈值是影响网络训练的唯一因素。权值的初始化通常是采用随机的方法，即将(-1, 1)区间上的随机值赋给各个权值和阈值，这种初始化方法随机性太大，造成网络的训练结果不稳定，要获得一个较好的随机初始值需要经过多次实验才能得到。这里选用 Nguyen-Widrow 方法[40]来初始化权值和阈值。

Nguyen-Widrow 法保证每个神经元的权值都能在它们的 sigmoid 函数变化最大之处进行调节。该方法仅使用在隐含层的初始值的选取上，输出层的初始值仍然采用随机赋值的方式。由 sigmoid 函数图形可以得知，其在(-0.7, 0.7)范围内函数变化最大。通过设置权值 W 和阈值 b 的初始值，使隐含层神经元的输入落在这个区间。其计算公式[41]为

$$W = 0.7 \times S^{\frac{1}{N}} \times normr(2rand(S,N) - I(S,N)) \tag{4.5}$$

$$\theta = 0.7 \times S^{\frac{1}{N}} \times (2rand(S,1) - I(S,1)) \tag{4.6}$$

其中，W 为权重矩阵，θ 为阈值矩阵，S和 N 分别为网络隐含层结点数和输入层节点数。$rand(S,N)$ 为 S行 N 列的均匀分布的随机数矩阵，$I(S,N)$ 为 S行 N 列的全1矩阵，$normr(M)$ 为 M 矩阵的标准化规一矩阵。

4.3 并行训练的实现

4.3.1 MPI 基本模式

本文选用 MPICH 建立 MPI 并行环境，MPICH 是 MPI 的主要实现，可以作为库

被 C 或 FORTRAN 语言所调用。

MPI 程序由 4 个部分组成。分别是头文件，程序初始化，程序体计算与通信和程序结束。对于 C 程序而言，其中头文件要包含 mpi.h，相关变量声明中要声明当前处理器编号的变量 myid，处理器总数的变量 numprocs，处理器名字长度的变量 namelen。

MPI 环境的初始化和结束流程如下：在进入 MPI 程序体之前，各个进程都执行 MPI_INIT，初始化 MPI 环境。接着调用 MPI_COMM_SIZE 获得总进程的数量，调用 MPI_ COMM_RANK 获取当前进程的编号。然后，进程可以根据需要，向其它节点发送消息或接收其它节点的消息。发送和接收消息的函数为 MPI_SEND 和 MPI_RECV。最后，当 MPI 程序执行完后，调用 MPI_FINALIZE 退出 MPI 环境。这就是一般的 MPI 并行计算的流程。

在设计 MPI 并行程序时经常采用对等模式和主从模式两种最基本的设计模式。

(1) 对等模式

所谓对等模式，就是说 MPI 程序的各个进程的功能、地位相同或相近，MPI 程序的代码也应该是相近的，所不同的只是处理的对象和操作的数据，比如 MPI 程序让各个进程同时对一个数组的不同部分并行赋初值，各个进程间的关系就是典型的对等关系。

(2) 主从模式

所谓主从模式，就是 MPI 程序的各个进程所起的作用和地位并不相同，一个或者一些进程完成一类任务，而另外的进程完成其它的任务，这些功能或者地位不同的进程所对应的代码也有较大的差别。主从模式下，由一组进程共同完成同一个计算任务，其中一个为主进程，负责其它从进程的生成、初始化，并分配负载给从进程。从进程完成实际计算后，将计算结果传递给主进程，最后由主进程收集计算结果并输出。

4.3.2 进程之间的消息通信

1.消息通信的过程

由于进程之间不共享存储空间，进程之间变量的交换就要通过消息传递来实现。消息传递就是将一个进程发送缓冲区中的消息发送到另一个进程的接收缓冲区中。其过程如图 4.1 所示。

MPI 的消息传递过程可以分为三个阶段。

(1) 消息装配，将发送数据从发送缓冲区中取出，加上消息信封等形成一个完整的消息。

(2) 消息传递，将装配好的消息从发送端传递到接收端。

(3) 消息拆卸，从接收到的消息中取出数据送入接收缓冲区。

图 4.1 消息传递的过程

2.消息通信函数

(1) 在消息传递过程中，MPI_Send 和 MPI_Recv 是主要的消息通信函数。

MPI_Send 将发送缓冲区 buf 中的 count 个 datatype 数据类型的数据发送到目的进程，目的进程在通信域中的标识号是 dest，本次发送的消息标志是 tag，使用这一标志，就可以把本次发送的消息和本进程向同一目的进程发送的其它消息区别开来。

int MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

MPI_Recv 从指定的进程 source 接收消息，并且该消息的数据类型和消息标识和本接收进程指定的 datatype 和 tag 相一致，接收到的消息所包含的数据元素的个数最多不能超过 count。然后将接收到的数据放入到接收缓冲区 buf 中。

int MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)

(2) 在一对多和多对一的组通信情况下，MPI_Bcast 和 MPI_Reduce 是两个常用的函数。MPI_Bcast 的作用是从一个标识为 root 的进程将一条消息广播发送到组内的所有其它的进程，同时也包括它本身在内。在执行该调用时组内所有进程（不管是 root 进程本身还是其它的进程）都使用同一个通信域 comm 和根标识 root，其执行结果是将根进程通信消息缓冲区中的消息拷贝到其他所有进程中去。

int MPI_Bcast(void* buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)

MPI_Reduce 将组内每个进程输入缓冲区中的数据按给定的操作 op 进行运算，并将其结果返回到序列号为 root 的进程的输出缓冲区中。输入缓冲区由参数 sendbuf、count 和 datatype 定义，输出缓冲区由参数 recvbuf、count 和 datatype 定义，要求两者的元素数目和类型都必须相同。因为所有组成员都用同样的参数 count、datatype、op、root 和 comm 来调用此进程，故而所有进程都提供长度相同、元素类型相同的输入和输出缓冲区。

int MPI_Reduce(void* sendbuf, void* recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)

4.3.3 主/从训练模式

1．主/从模式

数据并行的训练采取 master/slave(主/从)模式，如图 4.2 所示，这种模型由一个主进程和若干个从进程组成。主进程运行于一台 master 机上，负责生成从进程，初始化网络，计算任务分配，收集显示结果。而从进程运行于其他的 slave 机上，执行相同程序，处理不同数据，计算这些数据的总体误差和各权值对此误差的偏导信息，传递给主进程汇总及相关处理。各从进程之间互不通信，完全独立，以减少通信开销。

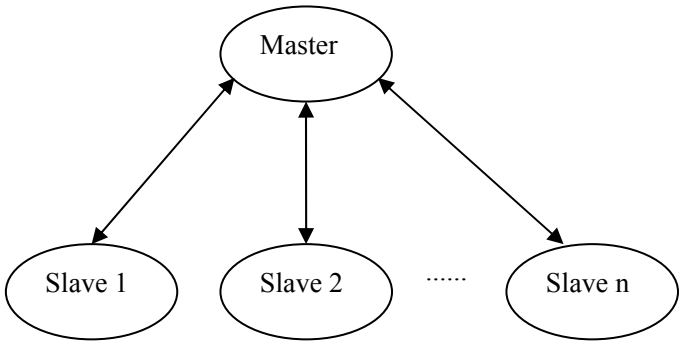


图 4.2 主/从训练模式

2．样本数据的分配

样本数据的分配要充分考虑到各个处理机之间的负载情况。由于本文所基于的机群环境是同构的网络环境，因此各个处理机的计算能力均相同。只要将样本数据平均分配给各个节点就很容易做到负载平衡。样本数据划分的方式有随机划分方式、分段划分方式和间隔划分方式等。由于输入样本的顺序对结果没有影响，采取任何

一种划分方式均不会影响神经网络的训练结果。但为了便于实现，本文采用间隔划分方式。在这种间隔划分方式中，将总体样本按顺序进行编号，如果处理机的个数为 n，则每台处理机从总编号中间隔 n 个抽取出各自的样本数据。比如样本号为{1,2,3,...10}，处理机个数为 3，则第 1 台处理机的样本号为{1,4,7,10}；第 2 台的样本号为{2,5,8}；第 3 台的样本号为{3,6,9}。

3．权值更新的过程

由于主节点只负责参数的传递和权值矩阵的累加工作，未参与网络训练过程，因此其运算量较小，为了节省资源提高效率，可以将主节点和从节点进行合并，使主节点在管理各从节点的同时也进行网络的训练。具体算法如下。

(1) 初始化 MPI 运行环境；

(2) 如果当前进程为主进程，则获取网络结构、样本个数、期望误差、迭代次数以及算法的参数；

(3) 主进程向各进程广播网络结构、样本个数、期望误差、迭代次数等参数；

(4) 各进程建立神经网络结构并初始化；

(5) 各进程初始化一维数组 Grad[]用于传递各进程的梯度累加值；

(6) 各进程根据进程 ID 从本地硬盘中以文本形式读入样本数据{P, T}；

(7) 如果当前进程为主进程，则主进程将其网络权值、阈值矩阵的初始值转换成一维形式填入数组 Grad[]中；

(8) 各进程进行同步；

(9) 各进程权值、阈值初始化为 0；

(10) 主进程将 Grad[]广播到各进程中；

(11) 各进程将一维数组 Grad[]转换成权值、阈值的增量；

(12) 各进程根据权值、阈值的增量改变权值、阈值矩阵；

(13) 各进程训练一个 epoch 求得各个权值增量；

(14) 各进程进行同步；

(15) 各进程将 Grad[]规约到主进程中，并求得梯度总和存放到主进程 Grad[]中；

(16) 如果当前进程为主进程，则输出误差值。如果误差<期望误差，则结束训练；根据 RP 算法求出权值和阈值的改变量$\square W$，并将$\square W$放入一维数组 Grad[]中；如果迭代次数小于最大迭代次数则转到第(9)步；否则结束训练。

4．算法流程图

基于数据并行 BP 算法流程图如图 4.3 所示。

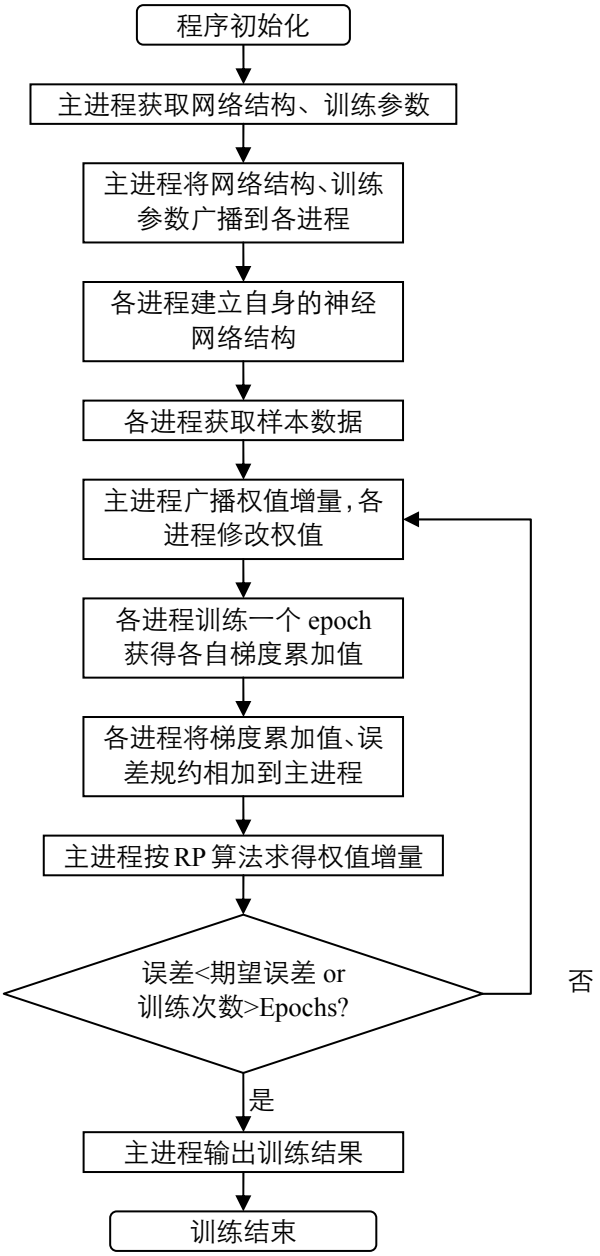


图 4.3 数据并行 BP 算法流程图

4.4 并行训练的改进

4.4.1 样本数据存储

对于样本数据的存储，有两种方式。

(1) 样本数据存放在主进程的存储器中。先由主进程获取样本数据，再根据不同的进程编号发送给不同的从进程。这种方式操作简单，主节点可以自由选择样本数据，从节点不许做任何修改。但代价是数据传输造成的性能的损失，特别是样本数据非常大的情况下，主节点将数据分发到各从节点，网络传输会消耗大量时间。

(2) 样本数据分布式存放在各个从节点的存储器中。程序运行时，各从节点根据各自的进程编号从其存储器上读入对应的样本数据，这种方式就省去了数据从主节点传输给从节点的过程，提高了性能。但是这样做缺点也是存在的，因为数据的分布式存储就给数据的管理带来了困难，当训练样本发生改变时，各个从节点的数据都要做相应的修改。

采用分布式存储的方式并行效率更高，因此本文使用第二种方式存放样本数据。

4.4.2 初始权值筛选

神经网络的初始权值对训练的结果影响非常大，合适的权值会加快网络的收敛。而神经网络的权值是随机初始化的，这样就给神经网络的训练带来了不确定的因素，即同一个网络每次训练的结果都可能不一样。要想获得较佳的实验结果，往往需要通过多次实验对比来确定。本文采取的改进方法就是利用了初始权值的随机性，通过多机同时训练来筛选出较好的初始权值。

通过实验证明，在训练最初的迭代过程中，如果误差减少的比较快，那么这个网络也相应的收敛得要快。因此在训练的初期就可以根据网络误差的情况大体上判定网络的收敛情况。

在一般并行 BP 算法中，各个节点的初始权值都是相同的，如果将不同的初始权值分别赋给不同的节点让其同时开始对所有的样本进行训练，则在进行一定训练次数之后可以得到误差最小的节点，然后将这个节点的权值广播到各个节点再开始进行数据并行训练，那么有利于提升并行算法的速度，加快网络的收敛。

基于上述改进策略，在下一章中将对这些改进进行实验，并分析实验结果。

4.5 本章小结

本章介绍了基于数据并行的 BP 神经网络的实现方法。首先说明了数据预处理的方法，然后探讨了如何通过所研究的问题对神经网络结构进行确定。重点讲述了 MPI 并行计算环境的建立，介绍了消息通信的主要实现方法。本文采用主从结构的并行

模式，通过将数据平均分配给各个节点分别进行训练，然后由主节点收集训练结果，再将权值的更新值广播到各个节点，用来同步改变网络的权值。最后针对并行 BP 网络的样本数据的存储和权值的划分进行了一些改进。

5 实验结果及分析

5.1 实验数据的选取

本实验的数据来源于华中科技大学同济医学院于 2000 年对宜昌三峡坝区 35 岁常住人口的高血压问卷报告。报告统计了 3054 位居民的人口学资料，生活饮食情况，健康状况以及患高血压疾病等信息。报告总共由 150 项组成。图 5.1 为数据样本的部分截图。

年龄	年龄分组	性别	民族	婚姻状况	文化程度	职业	家庭毛收	BMI	是否高血压
39	1	2	1	1	3	4	1	28.93407	0
37	1	2	1	1	2	4	2	20.95717	0
39	1	1	1	1	3	4	2	26.89232	0
36	1	1	1	1	3	6	5	29.05329	0
39	1	2	1	1	3	4	5	25.96953	0
40	1	1	1	1	3	4	5	28.25699	0
36	1	1	1	1	3	4	5	22.30815	0
35	1	1	1	1	3	4	5	22.75831	0
39	1	1	1	1	3	4	5	21.70792	0
38	1	2	1	1	3	4	3	24.43519	0
38	1	2	1	1	4	4	3	28.39872	0
37	1	2	1	1	3	4	4	23.45856	0
38	1	2	1	1	3	4	5	22.21368	0
38	1	2	1	1	4	4	5	19.83471	0
36	1	1	1	1	3	4	5	17.71542	0
39	1	2	1	1	2	4	4	22.82688	0
40	1	1	1	1	2	4	4	22.03857	0
40	1	1	1	1	4	3	5	22.09317	0
38	1	1	1	1	3	4	4	24.16716	0
35	1	1	1	1	3	4	4	26.67276	0
39	1	2	1	1	4	4	5	20.83	0
39	1	1	1	1	2	4	4	22.22906	0
38	1	2	1	1	3	4	4	23.8054	0
40	1	1	1	1	3	4	5	19.05197	0

图 5.1 样本数据截图

在数据预处理阶段，首先将各项的缺失数据用该项的均值替代。然后采用 0 均值标准差法对数据进行标准化处理。通过主成分分析方法筛选出了“年龄”、“性别”、“婚姻状况”、“BMI”、“吸烟”、“饮酒”、“家族史”、“WHR”、“文化程度”等 17 项属性作为神经网络的输入属性，将“是否患有高血压”作为输出属性。通过用神经网络对样本数据进行训练，挖掘各种危险因素与高血压患病之间的联系，辅助医生

或病人对高血压疾病进行预测和预警。图 5.2 为神经网络结构图。

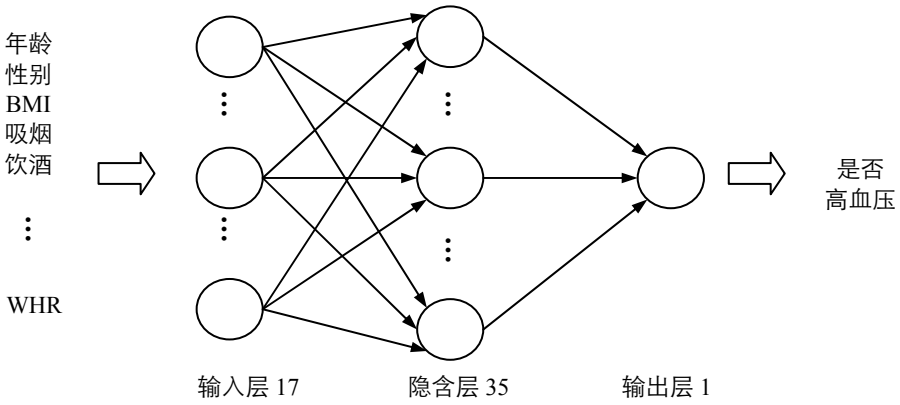


图 5.2 神经网络结构图

该实验在基于机群的网络环境下进行，在网络环境下，对系统的通信开销有一定的要求。由于该高血压样本由 3000 多份数据组成，大量的数据有利于提高计算所占比重，降低通信比重。可以说样本数据的复杂性和数量是满足并行算法要求的。

5.2 实验环境的建立

本实验是在局域网内 4 台 PC 组成的机群系统上进行的。每台 PC 硬件配置都为：Intel P4 3.06GHz，512MB 内存。操作系统为 Windows Server 2003，每台 PC 上安装 MPICH2-1.05p2 用来运行 MPI 并行程序。程序采用 Microsoft Visual Studio 2005 编写，所用的语言是 C++和 C#。图 5.3 为程序界面。

图 5.3 并行 BP 演示程序界面

5.3 实验结果评价

5.3.1 并行计算算法性能评价标准

对于并行算法，我们最关心的是算法运行的时间，除此之外还要考虑并行算法相对于串行算法对计算速度提升的程度。同时参加运算的节点个数也对算法的性能有重要的影响，因此我们又需要研究不同节点个数下算法的并行效率。

(1) 并行计算时间

并行程序的执行时间包括了计算时间和通信时间，尤其在机群系统中，网络的通信时间是不能被忽略的。因此$T_p = T_{comp} + T_{comm}$，即总时间 = 计算时间 + 通信时间。在同构的系统中，保证负载平衡的情况下，如果各个节点所分配到的计算相同，则总计算时间就等于每个节点的计算时间。通信时间与消息的大小，底层的连接方式，信息传递的方式都有关，对于工作站机群来说，通信时间可以近似表示为：$T_{comm} = T_{start} + nT_{data}$。其中$T_{start}$为启动时间，也叫消息时延，主要包括消息传递过程中进行消息打包和解包的时间。当节点数很多时，将导致启动时间很长。nT_{data}是传送 n 个字节数据所花费的时间。因此在并行计算的过程中，节点的个数以及传送的字节个数将直接影响通信时间。

(2) 加速比

假设T_s为串行程序执行时间，T_p为对应并行程序在 p 台处理器上并行执行的时间。则加速比$S_p = T_s / T_p = T_s /(T_{comp} + T_{comm})$，如果没有通信时间，则$S_p = T_s / T_{comp}$。假定各个处理机的处理能力相同，计算任务也相同，则$S_p = p$，当处理机个数 p 增加时，加速比也会线性地增加。但是在机群系统中，由于通信占开销大，加速比往往不会线性增加。如图 5.4 所示。

(3) 并行算法效率

算法效率也是衡量并行算法的指标之一。并行算法的效率定义为：$E_p = S_p / p$。其中S_p是加速比，p为处理机个数。由第(2)部分关于加速比的讨论可知，当通信时间为0时，并行算法的加速比等于处理机个数，即$S_p = p$，并行效率为1，当然这是理想情况。而在实际情况下$S_p < p$，并行效率小于1。同时随着处理机个数p的增加，加速比往往不会以相同的速度增大，那么对应的并行效率反而会降低。

短时间和高效率是设计并行算法所追求的目标。对于多数问题，二者之间往往存在着矛盾，即增加处理机数量，可以减少算法执行时间，但降低了算法的效率；相反，减少处理机数量，提高了算法效率，但可能导致计算时间增加。二者之间要仔细权衡。

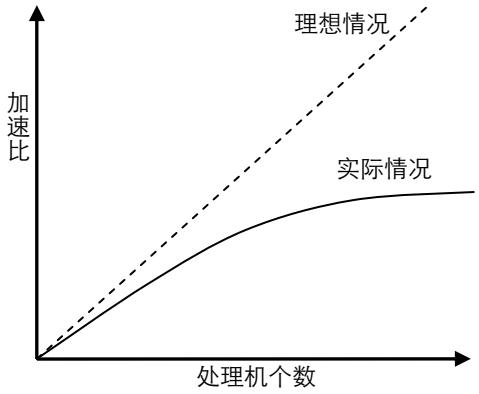


图 5.4 加速比和处理机个数的关系

5.3.2 并行 BP 算法实验结果及分析

1．实验说明

(1) 神经网络结构

神经网络采用三层结构。根据高血压样本中提取的危险因素将输入层节点个数设为 17 个，输出层节点 1 个。通过多次实验隐含层节点个数选为 35 个实验效果较好。

(2) 算法参数

设置弹性 BP 算法中初始 delta=0.1，最小 delta=0.000001，最大 delta=50.0，delta_inc=1.05，delta_dec=0.7。最小误差为 0.01，最大迭代次数 30000。

2．实验结果

(1) 串并行对比实验结果

分别用一台处理机和多台处理机对 BP 网络进行训练，统计训练的时间，迭代次数以及并行加速比等结果，如表 5.1 所示。

为了使实验数据更具说服力，在每次实验中，设置神经网络的初始权值矩阵相等，因此最后得到的误差以及样本的正确率是一样的，而且迭代的次数都相同。这一点从表 5.1 中可以看到。但是总运行时间随着处理机个数的增加减少了，这是因为

一次迭代就是将所有样本进行处理的一次过程，而参加计算的处理机个数越多，每台处理机处理的样本数就越少，于是每次迭代的时间就会减少，由于迭代次数不变，则总时间会减少。由此说明了并行算法的有效性。

在处理机个数增加时，虽然加速比会相应增大，但并不是成线性增加的趋势。这是因为随着处理机个数的增加，每台处理机所分得的样本个数减少，所以计算量下降；而处理机的增多，又造成了通信量的增加，通信增加的时间有可能抵消掉计算所减少的时间，甚至随着节点个数增加到一定限度，运行时间反而会增长，加速比减少。

并行效率也是并行算法的重要评估标准，并行效率反映了处理机的利用率。随着处理机个数的增加，并行效率降低，这也是因为通信的时间的增加降低了处理机的利用率。由此可以得知在兼顾加速比和并行效率的情况下，找到合适的处理机个数是并行算法好坏的关键。

表 5.1 串行与并行的实验结果对比

	1 台处理机	2 台处理机	3 台处理机	4 台处理机
运行时间(s)	771.28	419.19	328.24	283.82
迭代次数	14101	14101	14101	14101
加速比	1	1.84	2.35	2.72
并行效率	1	0.92	0.78	0.68
样本集正确率	0.991159	0.991159	0.991159	0.991159

(2) 筛选初始权值

在改进初始权值筛选法中，首先让各个节点对其网络权值进行随机初始化。再对整个样本集进行训练，在进行完 50 次迭代后，统计出误差最小的节点，将该节点的权值矩阵广播到各个节点，再由各个节点进行并行 BP 网络训练。其实验结果如表 5.2 所示。

表 5.2 和图 5.5 中显示，采用了改进的初始权值筛选法后，在多处理机并行训练的情况下，因为使用不同的初始权值进行训练，所以迭代次数都不相同，而且迭代次数都有相应的减少。加速比在处理机个数相同时相对于原始并行算法有所增加，并且更加接近于理想值。在处理机个数为 4 时，加速比甚至达到了 3 以上。同时并行效率也获得了提升，从而充分利用了多处理机的计算能力。由此证明，改进的权值筛选法是有效的。

表 5.2 采用改进的初始权值筛选法的实验结果

		1 台处理机	2 台处理机	3 台处理机	4 台处理机
初始权值未经筛选	运行时间(s)	771.28	419.19	328.24	283.82
	迭代次数	14101	14101	14101	14101
	加速比	1	1.84	2.35	2.72
	并行效率	1	0.92	0.78	0.68
筛选初始权值	运行时间(s)	771.28	413.96	305.64	243.59
	迭代次数	14101	13925	13130	12102
	加速比	1	1.86	2.52	3.17
	并行效率	1	0.93	0.84	0.79

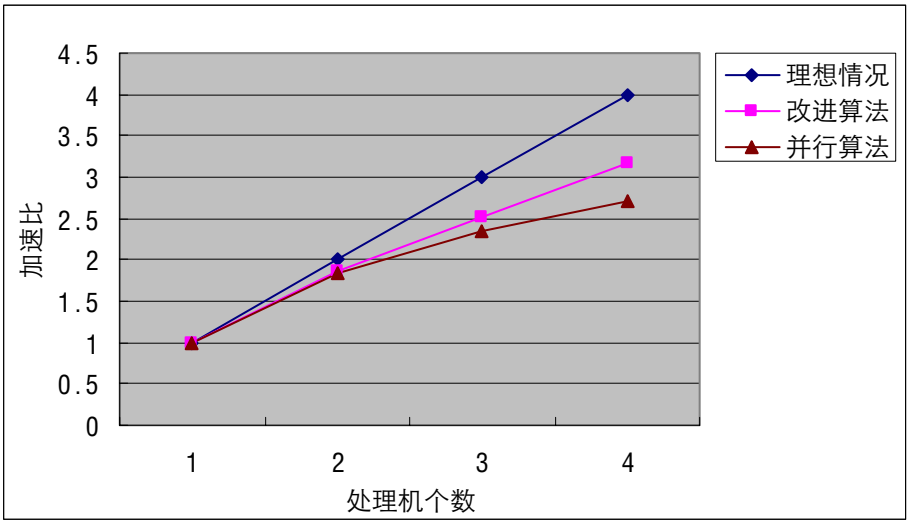


图 5.5 加速比与处理机个数的实验结果

5.4 本章小结

本章主要介绍了在局域网内多台计算机所组成机群环境下，建立 MPI 并行环境，采用基于数据并行的 BP 算法，使用宜昌地区的 3054 份高血压数据对神经网络进行训练。通过与串行 BP 算法的对比，并行 BP 算法减少了训练时间，加快神经网络收敛速度。同时通过对不同处理机数量的训练结果进行对比，发现增加处理机数量能进一步提高训练速度，但同时也会降低处理机的利用率。然后，采用改进的初始权值筛选法进行实验，实验结果显示改进的算法有利于增加并行计算的加速比，提高并行效率。

6 结束语

6.1 全文总结

人工神经网络能对信息进行处理，具有很强鲁棒性和容错性，善于联想、概括、类比与推理，具有很强的自学习等特性。BP 网络以其良好的非线性逼近能力和易于实现性在大量的实际工程中得到了应用。但是传统的 BP 算法存在学习速度慢，网络容易陷入局部极小值等缺点。本文在深入分析 BP 算法局限性的基础上，详细介绍了 BP 算法的相关改进算法，并将这些改进算法进行了对比，分析了各自的优点及不足。

针对 BP 算法训练速度慢的特点，借助并行计算技术在解决大规模、复杂问题上的优势，本文分析了各种 BP 网络的并行化方法，针对机群环境的特点，找到了一种基于数据并行的 BP 网络算法，借助 MPI 并行环境实现了 BP 网络并行化。并通过一个大的高血压样本数据库对其进行训练，证明了 BP 并行算法的正确性与可行性。

本文的研究内容和成果，概括起来主要包括以下四个方面。

(1) 本文介绍了 BP 神经网络的组成结构、算法原理以及数学推导过程。分析了传统 BP 算法存在的缺陷，以及针对这些缺陷进行的改进。比较了各种改进方法的优缺点，选择出了一种既适合实际应用又易于并行实现的改进 BP 算法。

(2) 本文对各种 BP 神经网络并行化的方法进行了研究，找到一种适合于机群环境下的 BP 并行算法。并在在机群环境下，基于 MPI 并行环境，结合弹性 BP 算法实现了基于数据并行的 BP 算法。

(3) 针对神经初始网络权值随机性的特点，结合并行计算的优点，提出了筛选初始权值的改进 BP 并行算法。通过多机同时训练筛选出较优权值以用于数据并行的训练之中。提高了训练速度，加快了网络误差的收敛。

(4) 通过并行 BP 算法与串行 BP 算法的实验对比，证明了并行算法的可行性与高效性。在并行 BP 实验中，使用不同个数的处理机分别进行实验，分析了处理机个数与并行算法加速比以及并行效率的关系。

虽然在研究的内容上取得了一点成果，但是仍然存在以下不足。

(1) 本文的实验是在同构的网络环境下进行的。也就是网络中各台处理机的计算能力相同。但是在异构的网络下，处理机数据样本的划分，网络的负载平衡都是

非常关键的问题。需要更全面的考虑和完善。

(2) 未考虑节点数增加或网络规模增大时主节点存在的计算瓶颈。此时主节点用于同步或处理各节点的消息所消耗的时间将使算法性能大大降低。因此，需要针对主节点进行一些优化和改进。

6.2 进一步的研究方向

BP 神经网络并行化是一个比较新的研究课题，有很多方面值得研究，本文的进一步研究和改进的重点如下。

(1) 进一步增加各节点的并行粒度。随着节点个数的增加，各节点所分得的样本个数减少，因此并行的粒度减小，算法的性能下降，如何在这种情况下增加并行的粒度，提高并行效率是一个值得研究的方向。

(2) 对 MPI 消息通信机制的研究与改进。由于在网络环境下，MPI 消息通信是基于 TCP/IP 协议的，需要对消息进行打包与解包，因此会增加一定的通信量。所以如何改进消息通信机制，减少通信时间需要做进一步的研究。

由于笔者水平有限，论文中存在错误之处在所难免，敬请各位老师和同学批评指正。

致 谢

首先我要感谢我的导师马光志副教授对我的关心和指导。在攻读硕士学位期间，马老师渊博的知识、严谨治学的态度、平易近人的学者风范和诲人不倦的精神深深的影响着我，并将使我终身受益。在论文的选题、开题和撰写的过程中，无不得到马老师的悉心指导和帮助。在论文即将完成之际，请允许我向马老师表达个人最诚挚的感谢。

在华中科技大学两年的学习、生活期间，得到了计算机学院众多老师的关心和指导，在此，向他们表示我深深的谢意和良好祝愿。特别要感谢 CBIB 实验室宋恩民老师、许向阳老师、李国宽老师、金人超老师，感谢他们为我们的学习营造了良好的环境，他们严谨的作风，严格的要求令我受益匪浅。

感谢华中科技大学同济医学院的聂绍发老师和周水宏同学为实验提供数据。

感谢实验室里一起工作和学习蔡秋明、戴科峰、靖伟峰、马明、程罗锋同学以及其师弟李祖定、张廆、汤海鲜同学等。平时大家相互学习，互相帮助，是一个团结合作的集体，谢谢他们在各方面给与我的帮助。

感谢室友何川、郑刚、高纯玲，谢谢他们平时对我学习和生活上的帮助和照顾。

感谢我的父母和女朋友，他们对我精神上的支持和鼓励永远是我不断进取的动力。

最后，感谢论文评审委员会的老师们百忙之中对我论文的悉心指正。

参考文献

[1] Jiawei Han, Micheline Kambr. 数据挖掘概念与技术. 第二版. 范明译. 北京: 高等教育出版社, 2001.

[2] 崔荣晓. 关联规则归约问题的研究: [硕士学位论文]. 武汉: 华中科技大学图书馆, 2005

[3] 黄江华. 人工神经网络在数据挖掘中的应用: [硕士学位论文]. 长沙: 湖南师范大学图书馆, 2006

[4] Magoulas G D, Vrahatis M N, Anderoulakis G S. Effective Back-Propagation Training with Variable Stepsize. Neural Networks, 1997, 10(1): 69~82

[5] Yu Xiaohu, Chen Guoan. Efficient Back-Propagation Learning Using Optimal Learning Rate and Momentum. Neural Networks, 1997, 10(3): 517~527

[6] 吴凌云. BP 神经网络学习算法的改进及其应用. 信息技术, 2003, 27(7): 42~44

[7] Riedmiller M, Braun H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks. San Francisco: IEEE, 1993. 586~599

[8] 高雪鹏, 从爽. BP 网络改进算法的性能对比研究. 控制与决策, 2001, 16(3): 167~171

[9] 向国全, 董道珍. BP 模型中的激励函数和改进的网络训练法. 计算机研究与发展. 1997, 34(2): 113~117

[10] F.M. Silva, L.B. Almeida. Speeding up backpropagation. In: R. Eckmiller Ed. Advanced Neural Computers. North-Holland: Amsterdam, 1990. 151~158.

[11] Christian Igel, Michael Husken. Empirical Evaluation of the Improved Rprop Learning Algorithm. Neurocomputing, 2003, 50(1): 105~123

[12] Dennis, J.E., and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations. 1st edition. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[13] J.A. Leonard and M.A. Kramer, Improvement of the back-propagation algorithm for training neural networks. Computers & Chemical Engineering, 1990, 14(1): 337

[14] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial and Applied Mathematics, 1963, 11(2): 431~441

[15] Zickenheiner S, Wendt M, Klauer B et al. Pipelining and Parallel Training of Neural

Networks on Distributed-Memory Multiprocessors. In: IEEE International Conference on Neural Networks. 1994. 2052

[16] Petrowski A. Choosing among several parallel implementations of the back-propagation algorithm. In: IEEE International Conference on Neural Networks. 1994. 1981~1986

[17] Shadafan R S, Niranjan M. A Systolic Array Implementation of a Dynamic Sequential Neural Network for Pattern Recognition. In: IEEE International Conference on Neural Networks. 1994. 2034.

[18] G. Chinn. Systolic array implementations of neural nets on the MasPar MP-1 massively parallel processor. In: Proceeding of the International Joint Conference on Neural Networks. San Diego, USA. 1990. 169~173

[19] Grajski, Kamil A. Parallel Digital Implementations of Neural Networks. 1st edition. USA: IEEE Computer Society Press,1993, 51~76

[20] Jackson D, Hammerstrom D. Distributing backpropagation networks over the Intel iPSC/ 860 hypercube. In: Proceedings of International Joint Conference on Neural Networks. Seattle, USA. 1991. 569~574.

[21] Wayne Allen, Avijit Saha. Parallel neural-network simulation using back-propagation for the ES-kit environment. In: Proc. of 1989 Conf. Hypercubes. 1989. 1097~1102.

[22] Michael Witbrock, Marco Zagha. An implementation of backpropagation learning on GF11. Parallel computing, 1990(14): 329~346

[23] 刘皓, 魏平, 肖先赐. 面向特定结构的几种 BP 并行算法及比较. 系统工程与电子技术, 2000, 22(1): 70~76

[24] 高曙. 基于机群的并行 BP 算法的设计与实现. 武汉理工大学学报, 2002, 26(5): 589~591

[25] 李旗堂, 李娜, 宋国杰. 一个面向大规模 BP 神经网络并行算法. 河南广播电视大学学报, 2004, 17(1): 33~36

[26] 胡月, 熊忠阳. 一种新的 BP 算法并行策略. 计算机工程, 2005, 31(8): 148~151

[27] 刘冠蓉, 何华. 基于并行神经网络的汉字识别系统. 基础自动化, 2001, 8(1): 8~10

[28] 胡月. BP 算法并行化在数据挖掘中的研究: [硕士学位论文]. 重庆: 重庆大学图书馆, 2003

[29] 赵莉, 程荣. 一种并行 BP 神经网络的动态负载平衡方案. 计算机技术与发展, 2006, 16(7): 67~70

[30]郑庆伟, 罗艳明. 一种大规模神经网络并行技术. 大众科学(科学研究与实践), 2007: 65~66

[31]胡浩民, 马德云. 基于数据并行的神经网络预测模型. 计算机工程, 2005, 31(11): 162~164

[32]刘君. 动态隧道技术在 BP 网络中的应用与研究: [硕士学位论文]. 重庆: 重庆大学图书馆, 2005

[33]Martin T. H, Howard B. D, Mark H. B. 神经网络设计. 第一版. 戴葵译. 北京：机械工业出版社, 2002

[34]张治国, 杨毅恒, 夏立显. RPROP 算法在测井岩性识别中的应用. 吉林大学学报(地球科学版), 2005, 35(3): 390~393

[35]维基百科英文版 并行计算 http://en.wikipedia.org/wiki/Parallel_computing

[36]陈国良, 安虹, 陈崚等. 并行算法实践. 第二版. 北京:高等教育出版社, 2004

[37]张治宏. 基于 MPI 的并行计算研究: [硕士学位论文]. 北京: 中国地质大学图书馆, 2006

[38]都志辉. 高性能计算之并行编程技术. 第一版. 北京:清华大学出版社, 2001

[39]谭显胜, 周铁军. BP 算法改进方法的研究进展. 怀化学院学报, 2006, 25(2):126~127

[40]Derrik Nguyen, Bernard Widrow. Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. In: Proceeding of the International Joint Conference on Neural Networks. San Diego, USA.1990. 21~26

[41]刘刚.一种综合改进的 BP 神经网络及其实现. 武汉理工大学学报, 2002, 24(10): 57~60