第28卷 第5期
2002年9月

自 动 化 学 报
ACTA AUTOMATICA SINICA

Vol. 28, No. 5
Sept., 2002

# 两种典型神经网络容错方法的比较[1)]

许荔秦 胡东成

(清华大学自动化系 北京 100084)

(E-mail: xuliqin@163.net)

**摘 要** Behnam提出的SC算法和文中提出的rehidden算法是两种典型的前向神经网络容错算法,前者改进BP算法进行学习,后者对已学习的网络进行隐层节点冗余.这两种算法各有优缺点.文中对这两种算法进行了仿真实验分析,最终得到了每种算法适用的网络规模和硬件条件,在不同环境下应采用不同的方法才能得到可行的容错网络.最后还对SC算法的一些改进进行了讨论.

**关键词** 神经网络,容错,冗余

**中图分类号** TP202.1

# COMPARISON OF TWO TYPICAL FAULT-TOLERANCE ALGORITHMS OF NEURAL NETWORKS

XU Li-Qin HU Dong-Cheng

(*Department of Automation, Tsinghua University, Beijing* 100084)

(E-mail: xuliqin@163.net)

**Abstract** There are two typical fault-tolerance algorithms of feed-forward neural networks. One is SC algorithm presented by Behnam and the other is rehidden algorithm presented by this paper. The former modifies the BP algorithm in the training phase to gain fault-tolerant network, and the latter gives some redundant nodes to the hidden-layer of the trained neural network. There are advantage and shortage in both algorithms. We do simulations on these two algorithms. Analysis of the simulation results show that either algorithm has its own applicable network scale and hardware condition. In different condition, different algorithm should be used to gain suitable fault-tolerant neural network. Finally, we also give some analysis to some improvement of SC algorithm.

**Key words** Neural networks, fault-tolerance, redundancy

1) 清华大学博士学位论文基金资助

收稿日期 2000-07-21 收修改稿日期 2001-12-05

# 1　引言

多层感知器(MLP)的容错性在近几年来受到了很大的重视,已有不少文献报导过这方面的研究.从研究方法上看,提高 MLP 的容错性主要有两种方法:改进 MLP 的学习算法[1~3];在网络学习后对 MLP 进行结构冗余[4~6].两种方法各有特点:前者对 MLP 最常见的 BP 算法进行改进,从而使得网络有较好的容错性,但需要牺牲大量的学习时间;后者利用已训练好的网络,对其中的一部分节点进行冗余来达到期望的容错效果,但冗余后的网络更为庞大,规模无法很好地控制.我们在前面的工作中提出了新的节点冗余方法,并在理论上证明了这种方法无论对单故障还是全局随机故障都是有效的[6],它能利用相对较少的节点达到较好的容错效果.

在实际设计容错网络时,采用哪类方法成为一个主要问题.事实上,不同的网络规模和学习样本数会导致学习时间上差别很大.对于一些问题,增加大量的冗余节点也许会有较大的困难,因此需对这两种方法进行比较.在以前的研究中,多是针对网络中的单故障,因为这是神经网络中最常见的占绝大多数的故障.而实际上网络是一个各种随机故障全局存在的系统,也需要讨论这两种方法在全局随机故障情况下的容错性.

# 2　研究对象与模型

本文研究对象为单隐层 MLP,隐层神经元作用函数单调可微.用 $W_{ij}^{(n)}$ 表示连接第 $n-1$ 层第 $j$ 个神经元与第 $n$ 层第 $i$ 个神经元的权值,$bias_i^{(n)}$ 表示第 $n$ 层第 $i$ 个神经元的偏置,$O_i^{(n)}$ 表示第 $n$ 层第 $i$ 个神经元的输出,$I_i$ 表示第 $i$ 个输出层神经元的净输入,则有

$$I_i = \sum_{j=1}^{N_1} (W_{ij}^{(2)} O_j^{(1)} - bias_i^{(2)}) \tag{1}$$

$$O_i^{(2)} = f(I_i) \tag{2}$$

其中 $N_1$ 为隐层神经元数目,$f(I_i)$ 为输出层作用函数.

传统的学习算法为 BP 算法,为了提高收敛速度以及收敛到全局最优点,可以采用带冲量项的自适应学习算法如下:

$$\Delta W_{ij}^{(l)}(t+1) = mc \cdot \Delta W_{ij}^{(l)}(t) + lr(t) \cdot mc \cdot \frac{\partial J}{\partial W_{ij}^{(l)}(t)} \tag{3}$$

其中冲量项 $mc$ 为一常量,可以取 0.9;$J$ 为期望输出与实际输出之间的方差;学习速率 $lr(t)$ 为一自适应量,当 $J$ 减少时,$lr(t+1)=lr(t)\times 1.05$,当 $J$ 增加到上一步的 1.05 倍时,$lr(t+1)=lr(t)\times 0.7$.在以下的分析中,将这种改进的 BP 算法称为 gdx 算法.

神经网络中最常见的故障为权值连接断路与神经元损坏.对于权值连接断路故障,可以用权值 $W_{ij}^{(n)}$ $s$-$a$-0 来表示;对于神经元损坏,主要表现为神经元输出为正电源或负电源电位,因此可以用 $O_i^{(n)}$ $s$-$a$-1 或 $s$-$a$-$(-1)$来表示.而一个比较可靠的网络中单故障占故障中的绝大多数,因此在通常情况下可以只考虑这几种单故障情形以做简化处理.

但是,理论上更普遍的情况是各种故障同时以一定概率发生,称之为全局随机故障.在硬件实现中权值往往会发生偏移,称之为软故障,用相对误差 $\Delta$ 来表示.分别用随机变量 $\Delta$,$X$

来表示权值偏移以及权值 $s$-$a$-0 故障的分布，则权值实际的分布可以用随机变量 $\boldsymbol{w}_{ij}^{(n)}=W_{ij}^{(n)}(1+\Delta)X$ 来表示(其中 $W_{ij}^{(n)}$ 为标量常量)；对于神经元损坏，用随机变量 $\boldsymbol{o}_i^n=Z(O_i^{(n)})=\{-1, O_i^{(n)},1\}$ 表示其分布. 同时，由于输出层单元故障为严重故障，会严重影响网络输出，因此不在本文研究范围. 各 $\Delta,X,Z$ 独立分布，则显然各 $\boldsymbol{w}_{ij}^{(2)},\boldsymbol{o}_i^{(1)}$ 独立分布.

## 3 SC 算法

由于传统的 BP 学习算法的目标函数未考虑网络在各种扰动或故障作用情况下的输出误差，因此训练所得各权值重要度不均，使得 MLP 的容错性受到很大限制.

Behnam S Arad 提出了一种提高网络容错性的 SC 算法[2]. 这种算法采用训练中对网络注入故障，具体是定义故障集 $K$(包括隐层神经元 $s$-$a$-1 及 $s$-$a$-(−1)故障)，在 BP 学习的每一步中，改变优化函数为 $E=\sum_{k\in K}E_k$，其中 $E_k$ 为注入故障集 $K$ 中的一个故障后的优化函数，再采用 BP 或其改进算法进行学习. 采用这种算法可以大幅度提高网络的容错性，但是由于在每一步中都须注入所有类型的故障，因此要耗用大量的学习时间. 对于上面的故障集，一个隐层神经元数为 $N_h$ 的网络所需的每一步学习时间大致为 BP 算法的 $2\times N_h$ 倍，而由于 SC 算法所需收敛的步数一般更多，因此实际的学习时间往往大于 gdx 算法的 $2\times N_h$ 倍. 同时随着网络隐层节点的增多，学习时间也将增加.

文[1]将这种算法与部分注入随机选取的故障的方法进行了比较，结果发现，这种方法可以显著地提高学习时间，有较好的容错效果，并且收敛性也得到增强，与掺杂噪声的方法相比则容错能力有大幅度的提高. 研究还表明，这种方法还可以显著改善网络的推广能力. 因此 SC 算法应当是一种目前来说比较好的实用的容错算法，在对 BP 算法进行改进的这类算法中是很典型的. 因此我们用这种算法作为分析对象.

在对 SC 算法的实际应用中需要做一点改进. 由于学习目标函数变为在故障下的均方差优化函数，因此在学习过程中可能无法收敛到期望的目标函数值. 因此在学习过程中当目标函数小于期望值或收敛时(在一定步数内变化小于某一微小量)都认为学习结束.

## 4 隐层节点冗余容错方法

同样，隐层神经元冗余也是一个很好的神经网络容错的解决办法. 文[4,5]中提出了采用对整组隐层单元冗余以取得对 MLP 识别器中单故障完全容错的方法. 这种结构的缺点是冗余数太大，因此并不实用. 我们曾提出了一种新的冗余方法[6]，对单个隐层神经元进行冗余以减小冗余单元的成本，并对此方法的容错能力做了理论上的分析.

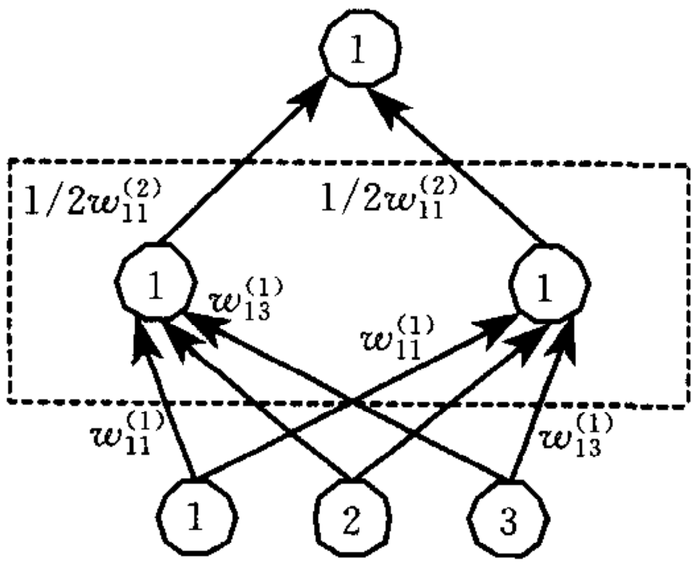


图 1 隐层神经元冗余结构

这种冗余结构如图 1 所示. 对原网络各隐层单元按重要度(重要度的概念在后面有定义)不同做不同的冗余，第 $j$ 个隐层单元冗余 $m_j$ 组(由于偏置 *bias* 也算作一个输出恒为 1 的隐层神经元，因此偏置神经元不做冗余)，即加入 $m_j$ 个隐层单元对第 $j$ 个隐层单元构成冗余结构. 隐层单元组

包括与某个隐层神经元相连的所有权值(其中包括与输出层相连的权值以及与输入层相连的权值).这样,在冗余之后此隐层单元组中即包含 $m_j+1$ 个隐层神经元.该隐层单元组内与输出层相连的各权值连接为原值 $W_{ij}^{(2)}$ 的 $\frac{1}{m_j+1}$,与输入层相连的各权值连接不变.图 1 给出了对第 1 个隐层节点冗余 1 组的结构.这样的冗余结构能够使网络在无故障时保持性能不变,即输出误差、推广能力等指标完全不变.

定义全局故障率为发生全局随机故障下输出层单元的净输入误差均方差

$$P_e=\frac{1}{N_p}\sum_{p=1}^{N_p}E\left(\frac{1}{N_o}\sum_{i=1}^{N_o}\left(I_i^{(p)}-I_{i,d}^{(p)}\right)^2\right) \tag{4}$$

其中 $I_i^{(p)}$ 为样本 $p$ 对于输出层单元 $i$ 的净输入分布的随机变量;$I_{i,d}^{(p)}$ 为无故障情况下样本 $p$ 对于输出层单元 $i$ 的标准净输入;$E(X)$ 为随机变量 $X$ 的期望值;$N_o$,$N_p$ 分别为输出节点和样本的个数,以此作为 MLP 容错性的评估标准.

加入冗余节点后对于网络的容错性有如下定理.

**定理 1**. 对一个隐层神经元 $k$ 按上面的冗余结构做 $m(m\geqslant 1)$ 个冗余,将使网络全局故障率减小

$$\Delta P_e(m)=\frac{1}{N_p\times N_o}\sum_{i,p}\frac{m}{m+1}\left[E(\boldsymbol{w}_{ik}^{(2)^2}\boldsymbol{o}_k^{(1)(p)^2})-(E\boldsymbol{w}_{ik}^{(2)}E\boldsymbol{o}_k^{(1)(p)})^2\right] \tag{5}$$

且全局故障率 $P_e$ 是 $m$ 的单调递减函数.

**证明**. 首先,由于各 $\Delta_{ij}$,$B_{ij}$,$Z_j$ 独立分布,且有 $\boldsymbol{w}_{ij}^n=W_{ij}^{(n)}(1+\Delta_{ij})B_{ij}$ 及 $\boldsymbol{o}_j^{(1)}=Z_j\left\{f\left[\sum_{j=1}^{N_1}(\boldsymbol{w}_{ij}^{(1)}x_j^{(p)})\right]\right\}$,因此可以得出各 $\boldsymbol{w}_{ij}^{(2)}$,$\boldsymbol{o}_j^{(1)}$ 独立分布的结论.

为简单起见,在证明过程中忽略隐层神经元 $\boldsymbol{o}_k^{(1)(p)}$ 及输出层权值 $\boldsymbol{w}_{ik}^{(2)}$ 的上标,即用 $\boldsymbol{o}_k$,$\boldsymbol{w}_{ik}$ 来表示隐层神经元和输出层权值(由于推导过程中没有用到输入层权值,因此用 $\boldsymbol{w}_{ik}$ 不会造成混淆).

对第 $k$ 个隐层神经元做 $m$ 个冗余后共有 $m+1$ 个同组的隐层神经元.记组中第 1 个隐层神经元输出为 $\boldsymbol{o}_{k(1)}$,连接这个神经元与第 $i$ 个输出神经元的权值为 $\boldsymbol{w}_{ik(1)}$.易知冗余后 $\boldsymbol{o}_{k(1)}$ 的均值与方差不变,即 $E\boldsymbol{o}_{k(1)}=E\boldsymbol{o}_k$,$D\boldsymbol{o}_{k(1)}=D\boldsymbol{o}_k$.冗余前有 $\boldsymbol{w}_{ik}=W_{ik}(1+\Delta)X$,冗余后有 $\boldsymbol{w}_{ik(l)}=\frac{1}{m+1}W_{ik}(1+\Delta)X$,因此有 $E\boldsymbol{w}_{ik(l)}=\frac{1}{m+1}E\boldsymbol{w}_{ik}$,$D\boldsymbol{w}_{ik(l)}=\frac{1}{(m+1)^2}D\boldsymbol{w}_{ik}$,且各 $\boldsymbol{w}_{ik(1)}$,$o_{k(1)}$ 之间相互独立.

冗余前有

$$E(I_i-I_{i,d})^2=E\left(\sum_{j\neq k}\boldsymbol{w}_{ij}\boldsymbol{o}_j+\boldsymbol{w}_{ik}\boldsymbol{o}_k-I_{i,d}\right)^2,\ \text{记}\ T=\sum_{j\neq k}\boldsymbol{w}_{ij}\boldsymbol{o}_j-I_{i,d},\text{则有}$$

$$E(I_i-I_{i,d})^2=ET^2+2ETE\boldsymbol{w}_{ik}E\boldsymbol{o}_k+E(\boldsymbol{w}_{ik}^2\boldsymbol{o}_k^2) \tag{6}$$

冗余后有

$$E(I_i-I_{i,d})^2=E(T+\sum_{l=1}^{m+1}\boldsymbol{w}_{ik(l)}\boldsymbol{o}_{k(l)})^2=$$

$$ET^2+2ETE\boldsymbol{w}_{ik}E\boldsymbol{o}_k+E\left(\sum_{l=1}^{m+1}\boldsymbol{w}_{ik(l)}\boldsymbol{o}_{k(l)}\right)^2=$$

$$ET^2+2ETE\boldsymbol{w}_{ik}E\boldsymbol{o}_k+\frac{1}{m+1}E(\boldsymbol{w}_{ik}^2\boldsymbol{o}_k^2)+\frac{m}{m+1}(E\boldsymbol{w}_{ik}E\boldsymbol{o}_k)^2 \tag{7}$$

比较式(6),(7)后易知冗余后 $E(I_i-I_{i,d})^2$ 减小,减小量为

$$E(I_i-I_{i,d})^2-E(I_i'-I_{i,d})^2=ET^2+2ETEw_{ik}Eo_k+E(w_{ik}^2o_k^2)-$$

$$\left[ET^2+2ETEw_{ik}Eo_k+\frac{1}{m+1}E(w_{ik}^2o_k^2)+\frac{m}{m+1}(Ew_{ik}Eo_k)^2\right]=$$

$$\frac{m}{m+1}\left[E(w_{ik}^2o_k^2)-(Ew_{ik}Eo_k)^2\right] \tag{8}$$

对所有 $i=1,\cdots,N_o$ 有同样的结论,对所有样本 $p=1,\cdots,N_p$ 也有同样结论,因此可以得出最终的全局故障率减小量

$$\Delta P_e(m)=\frac{1}{N_p\times N_o}\sum_{i,p}\frac{m}{m+1}\left[E(w_{ik}^{(2)^2}o_k^{(1)(p)^2})-(Ew_{ik}^{(2)}Eo_k^{(1)(p)})^2\right].$$

显然,$\Delta P_e(m)$ 是 $m$ 的单调递增函数,因此全局故障率 $P_e(m)$ 是 $m$ 的单调递减函数. 证毕.

在实际的算法中,不同节点的冗余对网络容错性能的提高是不一样的,而在实际应用中,所能采用的冗余数是有限的,因此必须在冗余时挑选最合适的节点进行冗余.

要挑选一个网络中最重要的节点进行冗余必须要有一个评估隐层节点重要度的标准. 由定理 1 得出了对隐层节点 $k$ 增加一个冗余节点对容错性的影响. 对 MLP 全局故障率 $P_e$ 减小量为

$$\frac{1}{N_p\times N_o}\sum_{i,p}\frac{1}{2}\left[E(w_{ik}^{(2)^2}o_k^{(1)(p)^2})-(Ew_{ik}^{(2)}Eo_k^{(1)(p)})^2\right] \tag{9}$$

因此,可以用这个式子作为节点重要度的评估标准.

但是可以看到,由于一般问题中样本数众多而且考虑到网络对未知样本的推广能力,要具体估算上式的值很困难. 对上式作简化,假设各 $o_k^{(1)(p)}$ 分布相同,其分布为 $O$,各 $\Delta_{ik}$ 同分布,记为 $\Delta$,各 $B_{ik}$ 分布相同,记为 $B$. 于是有

$$\frac{1}{N_p\times N_o}\sum_{i,p}\frac{1}{2}\left(E(w_{ik}^{(2)^2}o_k^{(1)(p)^2})-(Ew_{ik}^{(2)}Eo_k^{(1)(p)})^2\right]=$$

$$\frac{1}{2N_o}\sum_i\left\{W_{ik}^{(2)^2}E[(1+\Delta_{ik})B_{ik}O]^2-W_{ik}^2[E((1+\Delta_{ik})B_{ik}O)]^2\right\}=$$

$$\frac{1}{2N_o}D[(1+\Delta)BO]\sum_i W_{ik}^{(2)^2} \tag{10}$$

由此可以得到下面的隐层节点重要度选取标准.

**推论 1.** 对各隐层节点来讲,假设各 $o_k^{(1)(p)}$ 分布相同,各 $\Delta_{ik}$ 同分布,各 $B_{ik}$ 分布相同,则 $\sum_i W_{ik}^{(2)^2}$ 最大的隐层节点 $k$ 是最重要的节点,即对此节点冗余将使得全局故障率有最大的减小量.

这样,就有了一个很简便的节点选择原则:计算 $\sum_i W_{ik}^{(2)^2}$ 的大小来决定冗余节点. 我们可以得到一个隐层选择冗余容错方法:

1)设最大冗余数为 $R$,初始冗余数 $R(0)=0$,初始步数 $t=0$;

2)计算各隐层节点的 $\sum_i W_{ik}^{(2)^2}$,挑选其中最大的作为冗余节点,加入一个冗余节点,做冗余数为 1 的冗余,$R(t)=R(t)+1$;

3)若冗余数 $R(t)>R$,则冗余结束;否则 $t=t+1$ 转向 2).

在以下的分析中,将这种算法称之为 rehidden 算法.

## 5　算法的分析

用一个例子做仿真实验来对这两种典型算法进行分析. 采用一个做函数逼近的单输入、单输出 MLP,在[0,6]之间逼近函数 $y=\sin(x)$,采用[0,6]间均匀分布的 61 个数据点作为学习样本. 用[0,6]间随机选取的 100 个数据点作为测试样本对网络进行测试. 采用网络的输出误差均方差(MSE)作为网络性能的评价. 单故障模型下采用一类故障中所有单故障发生时网络 MSE 的平均值作为这类故障下的 MSE. 全局随机故障模型各参数如下:权值的相对偏移量 $\Delta\sim N(0,0.05)$,$s-a-0$ 的概率 $P\{X=0\}=0.05$,对隐层神经元 $s-a-1$ 与 $s-a-(-1)$概率相等,均为 0.05,即 $P\{Z(O_i)=-1\}=P\{Z(O_i)=1\}=0.05$.

首先对 gdx 算法、SC 算法和 rehidden 算法在各隐层节点数下的容错能力做分析. 采用的四种隐层节点数分别是 5,10,15,20,其中 rehidden 算法的初始学习隐层节点数为 5. 学习过程中采用的误差目标函数为 0.01. 得到如表 1 所示的结果.

表 1　gdx,SC,rehidden 三种算法的容错能力比较(MSE)

| 隐层节点 | | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|
| 无故障下 | gdx | 0.0130 | 0.0124 | 0.0074 | 0.0083 |
| | SC | 0.0512 | 0.0196 | 0.0076 | 0.0045 |
| | rehidden | 0.0130 | 0.0130 | 0.0130 | 0.0130 |
| 权值单故障下 | gdx | 0.2489 | 0.0909 | 0.0604 | 0.0553 |
| | SC | 0.0770 | 0.0375 | 0.0232 | 0.0156 |
| | rehidden | 0.2489 | 0.0685 | 0.0369 | 0.0264 |
| 节点单故障下 | gdx | 0.3525 | 0.1403 | 0.0876 | 0.0725 |
| | SC | 0.1010 | 0.0445 | 0.0283 | 0.0193 |
| | rehidden | 0.3525 | 0.0909 | 0.0475 | 0.0315 |
| 全局随机故障下 | gdx | 0.2649 | 0.2275 | 0.1928 | 0.2470 |
| | SC | 0.1095 | 0.0854 | 0.0833 | 0.0710 |
| | rehidden | 0.2670 | 0.1523 | 0.1093 | 0.0889 |

对于单故障容错,随着隐层节点数的上升,gdx 和 rehidden 算法令 MSE 均有大幅度下降;而 SC 算法 MSE 只有小幅度的下降. 对于全局随机故障容错,随着隐层节点数的上升,rehidden 算法同样使得 MSE 有大幅度下降;而 SC 只有非常小程度的缓慢下降;gdx 对于 MSE 的影响则完全没有规律了. 同时也可以看出,SC 算法在三种算法中容错能力是最强的,尤其在隐层节点较少时,SC 算法有较大的优势. 随着隐层节点的增加,rehidden 算法的容错能力渐渐逼近 SC 算法.

无故障情况下,随着隐层节点的增加,SC 算法下网络 MSE 明显下降;而 gdx 则无明显下降,rehidden 可以从理论分析知无故障情况下网络性能不变. 因此,虽然增加隐层节点不能显著增加 SC 算法的容错能力,但是可以使得 SC 算法得到的网络无故障性能得到较大提高. 另外,其学习时间与 gdx 算法相比非常冗长,且随隐层节点数增加而增加,如表 2 所示,这是它的缺点.

表 2　gdx,SC 两种算法的学习时间(单位:s)比较

| | 5 | 10 | 15 | 20 |
|---|---|---|---|---|
| gdx | 4.61 | 3.13 | 3.13 | 3.13 |
| SC | 80.90 | 150.39 | 248.70 | 260.12 |

对于 SC 算法有两种改进设想:1)由于 SC 算法中随隐层节点的增加训练时间加大,同时容错能力并得不到太大提高,因此考虑在 SC 算法结束之后再用 rehidden 算法增加隐层节点冗余来进一步提高容错能力,我们用 SC-rehidden 来表示这种设想;2)由于 SC 算法收敛速度慢,因此考虑先采用 gdx 算法得到一个网络后再用 SC 算法学习得到最终的网络,我们用 gdx-SC 来表示这种设想. 对这两种设想所做验证结果如表 3 所示.

表 3 gdx-SC,SC-rehidden 对网络容错能力与 SC 的比较

| 隐层节点 | | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|
| 无故障下 | SC | 0.0512 | 0.0196 | 0.0076 | 0.0045 |
| | gdx-SC | 0.0512 | 0.0190 | 0.0079 | 0.0042 |
| | SC-rehidden | 0.0512 | 0.0512 | 0.0512 | 0.0512 |
| 权值单故障下 | SC | 0.0770 | 0.0375 | 0.0232 | 0.0156 |
| | gdx-SC | 0.0783 | 0.0369 | 0.0234 | 0.0154 |
| | SC-rehidden | 0.0770 | 0.0493 | 0.0478 | 0.0483 |
| 节点单故障下 | SC | 0.1010 | 0.0445 | 0.0283 | 0.0193 |
| | gdx-SC | 0.1010 | 0.0439 | 0.0286 | 0.0190 |
| | SC-rehidden | 0.1010 | 0.0516 | 0.0480 | 0.0484 |
| 全局随机故障下 | SC | 0.1095 | 0.0854 | 0.0833 | 0.0710 |
| | gdx-SC | 0.1167 | 0.0820 | 0.0820 | 0.0805 |
| | SC-rehidden | 0.1095 | 0.0663 | 0.0548 | 0.0503 |

由表 3 知,采用 gdx-SC 与 SC 得到的网络容错能力基本一致,而采用 SC-rehidden 对于全局随机故障的容错能力比 SC 要强,但单故障情况下则不如 SC. 实际的网络中全局随机故障模型是更接近实际故障情况的,因此可以得出以下结论:gdx-SC 比 SC 有更高的容错能力,且学习时间短(都为最小隐层节点的 SC 学习时间),所以对 SC 进行 gdx-SC 改进是可取的.

从表 4 可以看出 gdx-SC 的学习时间与 SC 相差很小,因此这种设想是不可取的.

表 4 gdx-SC 与 SC 学习时间(单位:s)的比较

| | 5 | 10 | 15 | 20 |
|---|---|---|---|---|
| gdx-SC | 80.03 | 152.63 | 222.29 | 338.61 |
| SC | 80.90 | 150.39 | 248.70 | 260.12 |

## 6 结论

我们对比了两种很典型的神经网络容错方法,从上节的分析中可以得出以下结论.

1)在网络规模较小且对网络训练时间要求不高时,SC 算法优于 rehidden 算法. 这时 SC 算法虽然学习时间较长,但网络规模较小时还是可以容忍的,其容错能力也优于 rehidden 算法.

2)在网络规模较大时,rehidden 算法优于 SC 算法. 这时 SC 算法训练时间太长,是不可行的.

3)在网络有较大的冗余节点余量的情况下,rehidden 算法优于 SC 算法. 由于有较大的冗余余量,因此可以使得 rehidden 算法的容错能力逼近 SC 算法. 同时由于 rehidden 算法的学习时间(约等于 gdx 算法的学习时间)远小于 SC 算法,因此 rehidden 算法较为可取.

4)对于 SC 算法,采用最小的隐层节点数就可以达到较高的容错能力,加大隐层节点进行 SC 学习并不能显著提高容错能力,而只能使得网络在无故障时收敛到一个较小的误差目标函数.如果先采用较少节点进行 SC 算法学习,再用 rehidden 算法增加冗余节点,则可以显著提高全局随机故障情况下的容错能力,同时学习时间减短,但是与最初就采用最大冗余量的隐层节点进行 SC 算法学习相比,收敛的误差目标函数值较大,即无故障时网络误差较大.

5)先采用 gdx 算法再采用 SC 算法学习并不能加快学习速度.

在实际应用当中,需要根据不同的问题设计不同的网络,在规模及容错冗余容量确定后,采用不同的冗余方法才能得到较好的实用的容错网络.

## 参 考 文 献


1 Philippe K, Philippe R. Theoretical ınvestıgation of the robustness of multilayer perceptrons: Analysis of the lınear case and extension to nonlinear networks. *IEEE Trans. Neural Networks*, 1995, **6**(3):560～571

2 Behnam S A, Ahmed E A. On fault tolerant training of feedforward neural networks. *Neural Networks*, 1997, **10**(3):539～557

3 Deodhare D, Vidyasagar M, Keerthı S S. Synthesis of fault-tolerant feedforward neural networks using minimax optimization. *IEEE Trans. Neural Networks*, 1998, **9**(5):891～900

4 Phatak D S, Koren I. Complete and partial fault tolerance of feedforward neural nets. *IEEE Trans. Neural Networks*, 1995, **6**(2):446～456

5 Phatak D S, Koren I. Fault tolerance of feedforward neural nets for classification tasks. In: Proceedings of the International Joint Conference on Neural networks. Nagoya, Japan: IEEE, 1993, II. 386～391

6 Xu L, Hu D, Gao J. Hidden-layer neuron redundancy-analysis and application ın MLP's fault tolerance. In: Proceedıngs of the 3rd World Congress on Intelligent Control Automation. Hefei: Press of University of Scıence and Technology of Chına, 2000, II. 800～804



**许荔秦**　1996 年获清华大学自动化系自动控制专业学士学位,现为自动化系检测技术与装置专业直读博士研究生.现主要研究方向为人工神经网络的鲁棒性分析与容错性设计.

**胡东成**　1970 年毕业于清华大学电机工程系,留校于自动化系任教.1983 年至 1985 年由国家公派赴西德学习,后又数次出国访问或合作研究.现为清华大学副校长、教授、博士生导师.长期从事电子与自动化方面的教学与科研工作,主要研究方向为自动测试、故障诊断与可靠性.