一种改进的量子神经网络训练算法

张翼鹏* 陈 亮 郝 欢
(解放军理工大学通信工程学院 南京 210007)

摘 要：针对训练多层激励函数量子神经网络(MAF-QNN)时权值与量子间隔的目标函数存在冲突，导致收敛速度和网络性能下降的问题，该文提出一种改进的量子神经网络的训练算法。通过设计输出均方误差和这一目标函数对权值和量子间隔进行统一训练，同时引入Levenberg-Marquardt(LM)算法降低目标函数陷入局部极小值的概率，实现了对量子神经网络的高效训练。实验结果表明，该文提出的训练算法有效减少了迭代次数，显著提高了网络收敛精度，可应用于数据分类、函数逼近等场合，扩展了多层激励函数量子神经网络的应用领域。

关键词：量子神经网络；多层激励函数；Levenberg-Marquardt(LM)算法；最速下降
中图分类号：TP183 **文献标识码**： A **文章编号**：1009-5896(2013)07-1630-06
DOI: 10.3724/SP.J.1146.2012.01417

An Improved Training Algorithm for Quantum Neural Networks

Zhang Yi-peng Chen Liang Hao Huan
(*College of Communications Engineering*, *PLA University of Science and Technology*, *Nanjing* 210007, *China*)

Abstract: An improved training algorithm is proposed to solve the conflict of objective functions between weights and quantum interval during the training process of Multilevel Activation Functions-Quantum Neural Network (MAF-QNN), which will result in the degradation of training speed and network performance. By the criterion of least mean square error, the objective functions of the weights and quantum interval are unified and trained simultaneously. And then, by introducing the Levenberg-Marquardt (LM) algorithm, the probability of which the training results fall into local minimum is reduced. Therefore, the MAF-QNN can be trained effectively and efficiently. Simulation results show that, the proposed algorithm can decrease the iteration times efficiently and improve convergence precision significantly. In this way, it can be applied to data classification, function approximation and so on, expanding the application fields of MAF-QNN.

Key words: Quantum Neural Networks (QNN); Multilevel Activation Function (MAF); Levenberg-Marquardt (LM) algorithm; Steepest descent

1 引言

人工神经网络(Artificial Neural Network, ANN)，简称神经网络，是一种由大量人工神经元按照一定的拓扑结构相互联接来模拟人脑神经系统的数学模型。神经网络具有容错性、自组织、自学习等优良特性。目前，神经网络已广泛应用于智能控制、传感技术、信号处理、模式识别等诸多领域[1]。

经过60多年的发展，目前已经衍生出各式各样的神经网络模型，如BP神经网络、Hopfield神经网络、概率神经网络等[2]。多层激励函数量子神经网络是学者Purushothaman和Karayiannis[3]提出的一种主要用于数据分类的神经网络，以下简称量子神经网络。该神经网络由输入层、隐层和输出层构成。隐层中的神经元借鉴了量子态叠加的思想，将多个sigmoid函数相叠加，构建成具有多层结构的激励函数。这样，一个隐层神经元就可以表示多个量级，而一个传统的sigmoid函数只能表示两个量级。通过调整量子间隔的位置，可以使不同类的数据映射到不同的量级上，从而获得更多的自由度。

在训练中，网络权值采用传统的基于输出均方误差最小的梯度下降法，而量子间隔采用了不同的目标函数：同类输入数据的隐层神经元输出方差的总和最小，其训练算法依然是基于目标函数的梯度下降法。训练中，首先对权值进行调整，使输入数据映射到不同的类空间中，其次调整量子间隔，从而体现数据的不确定性[4]。

尽管量子神经网络具有优良的性能、可以感知分类数据间模糊性。但其训练算法与量子间隔的目

2012-11-08 收到，2013-03-21 改回
国家自然科学基金(61072042)资助课题
*通信作者：张翼鹏 plakkksam@yahoo.cn

标函数却存在明显的缺陷。在网络训练中，权值和量子间隔采用了不同的目标函数，且两种参数的训练交替独立进行，因此，在训练过程中两者的训练结果会相互冲突，从而导致训练时间的延长与网络性能的下降[5]。孙健等人[5]在两个目标函数的梯度优化过程中引入惩罚函数，在对一个目标函数的优化中考虑对另外一个目标函数的影响，一定程度上解决了训练过程中两目标函数的冲突问题。王金明等人[6]利用人工免疫算法训练量子间隔，在一定程度上解决梯度下降法易使训练结果陷入局部极小值的问题。但是，在这些算法中，量子间隔依旧采用针对数据分类的目标函数，这就将多层激励函数量子神经网络的应用局限在了数据分类上。

Levenberg-Marquardt(LM)算法是介于牛顿法与梯度下降法之间的一种非线性优化方法，可以大大降低目标函数陷入局部极小值的概率，且训练时迭代次数少，收敛速度快，非常适合用于前向型神经网络的训练[7]。张鸿燕等人[8]从理论上通过分解矩阵结构揭示了LM算法优良特性的根源。Indrajit等人[9]通过大量实验验证了LM算法的优良特性。Wilamowski等人[10]改进了Hessen矩阵的计算过程，在一定程度上解决了LM算法需要较大存储量的问题，并且有效提高了运算速度。

本文算法的训练中，量子间隔与网络权值采用同一个目标函数：输出均方误差和。从源头上避免了两个目标函数的冲突，同时突破了多层激励函数量子神经网络只能应用于数据分类的限制，使其可以应用在函数逼近、分析预测等更多领域。同时，用LM算法替换原最速下降法来训练权值与量子间隔，降低训练结果陷入局部最极小值的概率。本文接下来将介绍量子神经网络基于梯度下降的原始训练算法，然后给出本文设计的基于LM算法的量子神经网络训练策略，再通过仿真实验验证本文算法性能，最后给出结论。

2　网络结构与原训练算法

设量子神经网络有 n_i 个输入神经元，n_h 个隐层神经元，n_o 个输出神经元。其网络结构如图1所示。

隐层神经元选用的multi-sigmoid函数：

$$\tilde{h}_{j,k}=\frac{1}{n_s}\sum_{r=1}^{n_s}h_{j,k}^{r}=\frac{1}{n_s}\sum_{r=1}^{n_s}\mathrm{sgm}\left(\beta_h\left(\sum_{i=1}^{n_i}\upsilon_{ji}x_{i,k}+b_j^1-\theta_j^r\right)\right) \tag{1}$$

其中 $\tilde{h}_{j,k}$ 表示输入为 $\boldsymbol{x}_k$ 时，第 j 个隐层神经元的输出，$h_{j,k}^r$ 为相应的第 j 个隐层神经元中第 r 个sigmoid函数的输出。n_s 为激励函数中sigmoid函数的个数，即多层激励函数中的层数。β_h 为隐层中各个sigmoid函数中的斜率参数。θ_j^r 为第 j 个隐层神经元中第 r 个sigmoid函数的平移距离，也就是需要训练的量子间隔。υ_{ji} 为第 j 个隐层神经元到第 i 个输入神经元的权值，b_j^1 为第 j 个隐层神经元的偏置。$\mathrm{sgm}(\tau)$ 表示sigmoid函数，如式(2)所示：

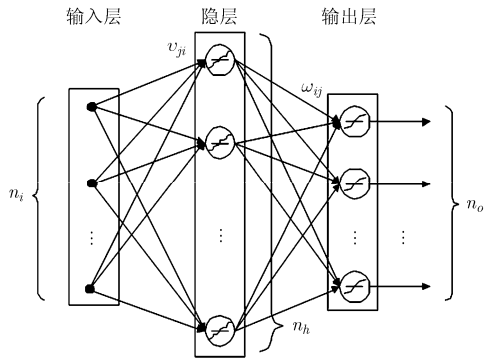

图1 量子神经网络结构图

$$\mathrm{sgm}(\tau)=\frac{1}{1+\exp(-\tau)} \tag{2}$$

训练过程分为两步，先训练网络的权值 $\boldsymbol{\upsilon}$ 和 $\boldsymbol{\omega}$，其训练目标是使网络输出值与目标值的误差平方和最小，如式(3)所示：

$$\{\boldsymbol{\upsilon},\boldsymbol{\omega}\}=\arg\min\left\{E=\sum_{k=1}^{K}\sum_{i=1}^{n_o}\left(y_{i,k}-\hat{y}_{i,k}\right)^2\right\} \tag{3}$$

其中 E 表示网络权值 $\boldsymbol{\upsilon}$ 和 $\boldsymbol{\omega}$ 的目标函数，K为训练的样本个数。$y_{i,k}$ 表示当输入为 $\boldsymbol{x}_k$ 时，输出神经元 i 的目标值，$\hat{y}_{i,k}$ 表示实际值。再训练隐层神经元的量子间隔 $\boldsymbol{\theta}$，其训练目标是使同类输入数据的隐层神经元输出方差的总和最小，如式(4)所示：

$$\begin{aligned}\{\boldsymbol{\theta}\}&=\arg\min\left\{G=\sum_{j=1}^{n_h}\sum_{m=1}^{n_o}\sigma_{j,m}^2\right\}\\&=\arg\min\left\{\sum_{j=1}^{n_h}\sum_{m=1}^{n_o}\sum_{\boldsymbol{x}_k:\boldsymbol{x}_k\in C_m}\left(\left\langle\tilde{h}_{j,C_m}\right\rangle-\tilde{h}_{j,k}\right)^2\right\}\end{aligned} \tag{4}$$

其中，G 表示量子间隔 $\boldsymbol{\theta}$ 的目标函数，C_m 表示由第 m 类输入数据构成的集合。$\left\langle\tilde{h}_{j,C_m}\right\rangle$ 表示由第 m 类输入数据所产生的，第 j 个隐层神经元输出结果的均值，如式(5)所示：

$$\left\langle\tilde{h}_{j,C_m}\right\rangle=\frac{1}{|C_m|}\sum_{\boldsymbol{x}_k:\boldsymbol{x}_k\in C_m}\tilde{h}_{j,k} \tag{5}$$

其中 $|C_m|$ 表示集合 C_m 的势，即集合 C_m 中的数据个数。

对网络权值与量子间隔的迭代均采用针对目标函数的梯度下降法。在每次迭代的过程中依次调整这两类参数。其具体推导与迭代求解过程可以参见文献[3]。

由上述训练算法可知，网络权值和量子间隔采用了不同的目标函数 E 和 G，且调整过程没有考虑

两组参数的相互影响，而是分别独立进行。当迭代次数逐渐增加时，可能出现调整其中一组参数使其目标函数值减小，但同时会导致另外一个目标函数值的增加[5]。由式(4)可知，量子间隔的目标函数是针对数据分类操作所设定的，以同类数据为处理单位进行计算。所以此目标函数限制了多层激励函数量子神经网络的应用范围，通过重新设定量子间隔的目标函数就可以扩展其应用领域。

3 基于 LM 的量子神经网络训练算法

将量子间隔与网络权值的目标函数同时设定为 E，训练目标均为使网络输出值与目标值的误差平方和最小。这样，统一训练网络权与值量子间隔，避免不同目标函数的相互冲突。

$$\{\boldsymbol{\upsilon},\boldsymbol{\omega},\boldsymbol{\theta}\}=\arg\min\left\{E=\sum_{k=1}^{K}\sum_{i=1}^{n_o}\left(y_{i,k}-\hat{y}_{i,k}\right)^2\right\} \tag{6}$$

目标函数 E 可以表示为 K 个样本集合的平方误差之和：

$$\begin{aligned}E&=\sum_{k=1}^{K}\sum_{i=1}^{n_o}\left(y_{i,k}-\hat{y}_{i,k}\right)^2=\sum_{k=1}^{K}\sum_{i=1}^{n_o}(e_{i,k})^2\\&=\sum_{i=1}^{N}(v_i)^2=\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}\end{aligned} \tag{7}$$

其中

$$\begin{aligned}\boldsymbol{v}^{\mathrm{T}}&=[v_1\ v_2\cdots v_N]=[e_{1,1}\ e_{2,1}\cdots e_{n_o,1}\ e_{1,2}\cdots e_{n_o,K}],\\N&=K\times n_o\end{aligned} \tag{8}$$

将量子神经网络的权值与偏置以及量子间隔合并成一个参数向量：

$$\begin{aligned}\boldsymbol{x}^{\mathrm{T}}=[x_1\ x_2\cdots\ x_n]=\Big[&\upsilon_{1,1}\ \upsilon_{1,2}\cdots\upsilon_{n_h,n_i}\ b_1^1\cdots b_{n_h}^1\\&\omega_{1,1}\cdots\ \omega_{n_o,n_h}\ b_1^2\cdots\ b_{n_o}^2\ \theta_1^1\cdots\theta_{n_h}^{n_s}\Big]\end{aligned} \tag{9}$$

其中 $n=n_h(n_i+1)+n_o(n_h+1)+n_hn_s$。若利用牛顿法更新 $\boldsymbol{x}$，则

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k-\boldsymbol{A}_k^{-1}\boldsymbol{g}_k \tag{10}$$

其中 $\boldsymbol{A}_k\equiv\nabla^2E\,|_{\boldsymbol{x}=\boldsymbol{x}_k}$，$\boldsymbol{g}_k\equiv\nabla E\,|_{\boldsymbol{x}=\boldsymbol{x}_k}$。目标函数 E 对 $\boldsymbol{x}$ 的梯度为

$$\nabla E=2\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{v}(\boldsymbol{x}) \tag{11}$$

其中

$$\begin{aligned}\boldsymbol{J}(\boldsymbol{x})&=\begin{bmatrix}\dfrac{\partial v_1}{\partial x_1}&\dfrac{\partial v_1}{\partial x_2}\cdots\dfrac{\partial v_1}{\partial x_n}\\\dfrac{\partial v_2}{\partial x_1}&\dfrac{\partial v_2}{\partial x_2}\cdots\dfrac{\partial v_2}{\partial x_n}\\\vdots&\vdots\qquad\vdots\\\dfrac{\partial v_N}{\partial x_1}&\dfrac{\partial v_N}{\partial x_2}\cdots\dfrac{\partial v_N}{\partial x_n}\end{bmatrix}\\&=\begin{bmatrix}\dfrac{\partial e_{1,1}}{\partial \upsilon_{1,1}}\cdots\dfrac{\partial e_{1,1}}{\partial \upsilon_{n_h,n_i}}&\dfrac{\partial e_{1,1}}{\partial b_1^1}\ \cdots\ \dfrac{\partial e_{1,1}}{\partial \theta_{n_h}^{n_s}}\\\vdots\qquad\vdots&\vdots\qquad\vdots\\\dfrac{\partial e_{n_o,1}}{\partial \upsilon_{1,1}}\cdots\dfrac{\partial e_{n_o,1}}{\partial \upsilon_{n_h,n_i}}&\dfrac{\partial e_{n_o,1}}{\partial b_1^1}\cdots\dfrac{\partial e_{n_o,1}}{\partial \theta_{n_h}^{n_s}}\\\vdots\qquad\vdots&\vdots\qquad\vdots\\\dfrac{\partial e_{n_o,K}}{\partial \upsilon_{1,1}}\cdots\dfrac{\partial e_{n_o,K}}{\partial \upsilon_{n_h,n_i}}&\dfrac{\partial e_{n_o,K}}{\partial b_1^1}\cdots\dfrac{\partial e_{n_o,K}}{\partial \theta_{n_h}^{n_s}}\end{bmatrix}\end{aligned} \tag{12}$$

$\boldsymbol{J}(\boldsymbol{x})$ 为目标函数 E 的雅克比矩阵。E 的 Hessen 矩阵为

$$\nabla^2E=2\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{J}(\boldsymbol{x})+2\boldsymbol{S}(\boldsymbol{x}) \tag{13}$$

其中

$$\boldsymbol{S}(\boldsymbol{x})=\sum_{i=1}^{N}v_i(\boldsymbol{x})\nabla^2v_i(\boldsymbol{x}) \tag{14}$$

由于 $\boldsymbol{S}(\boldsymbol{x})$ 一般较小，可以忽略不计。则将式(11)和式(13)代入式(10)可得高斯-牛顿法的更新算式：

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k-[\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{x}_k)\boldsymbol{J}(\boldsymbol{x}_k)]^{-1}\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{x}_k)\boldsymbol{v}(\boldsymbol{x}_k) \tag{15}$$

这样就避免计算二阶导数，但 Hessen 矩阵 $\boldsymbol{H}=\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$ 可能不可逆，LM 算法在 $\boldsymbol{H}$ 中引入了修正因子 $\mu\boldsymbol{I}$ 以保证构造的矩阵 $\boldsymbol{G}$ 可逆，且 $\boldsymbol{G}$ 与 $\boldsymbol{H}$ 的特征向量保持不变。其中 $\boldsymbol{I}$ 为单位矩阵。

$$\boldsymbol{G}=\boldsymbol{H}+\mu\boldsymbol{I}=\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}+\mu\boldsymbol{I} \tag{16}$$

将 $\boldsymbol{G}$ 代入式(15)，得到 LM 算法的更新公式：

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k-[\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{x}_k)\boldsymbol{J}(\boldsymbol{x}_k)+\mu_k\boldsymbol{I}]^{-1}\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{x}_k)\boldsymbol{v}(\boldsymbol{x}_k) \tag{17}$$

从式(17)可以看出，LM 算法的关键在于雅克比矩阵 $\boldsymbol{J}(\boldsymbol{x})$ 的计算。文献[1]中给出了雅克比矩阵 $\boldsymbol{J}(\boldsymbol{x})$ 前 $n_h(n_i+1)+n_o(n_h+1)$ 列针对权值与偏置的求解算法。下面推导 $\boldsymbol{J}(\boldsymbol{x})$ 后 n_hn_s 列 $\boldsymbol{v}$ 对量子间隔 θ_j^r 的导数。

$$\frac{\partial\boldsymbol{v}}{\partial\theta_j^r}=\left[\frac{\partial v_1}{\partial\theta_j^r}\ \frac{\partial v_2}{\partial\theta_j^r}\cdots\frac{\partial v_N}{\partial\theta_j^r}\right]^{\mathrm{T}}=\left[\frac{\partial e_{1,1}}{\partial\theta_j^r}\cdots\frac{\partial e_{n_o,1}}{\partial\theta_j^r}\cdots\frac{\partial e_{n_o,K}}{\partial\theta_j^r}\right]^{\mathrm{T}} \tag{18}$$

其中任意的 $e_{i,k}$ 对特定 θ_j^r 的导数为

$$\frac{\partial e_{i,k}}{\partial\theta_j^r}=\frac{\partial e_{i,k}}{\partial\tilde{h}_{j,k}}\cdot\frac{\partial\tilde{h}_{j,k}}{\partial\theta_j^r} \tag{19}$$

而 $\dfrac{\partial e_{i,k}}{\partial\tilde{h}_{j,k}}$ 在求解 $\boldsymbol{J}(\boldsymbol{x})$ 的前 $n_h(n_i+1)+n_o(n_h+1)$ 列时，作为中间变量，已经求解得到，这样，求解 $\dfrac{\partial e_{i,k}}{\partial\theta_j^r}$ 只需要求解 $\dfrac{\partial\tilde{h}_{j,k}}{\partial\theta_j^r}$ 即可。

$$\frac{\partial\tilde{h}_{j,k}}{\partial\theta_j^r}=\frac{1}{n_s}\sum_{p=1}^{n_s}\frac{\partial h_{j,k}^p}{\partial\theta_j^r}=\frac{1}{n_s}\frac{\partial h_{j,k}^r}{\partial\theta_j^r}=-\frac{\beta_h}{n_s}a(1-a) \tag{20}$$

$$a=\mathrm{sgm}\left(\beta_h\left(\sum_{i=1}^{n_i}\upsilon_{ji}x_{i,k}-\theta_j^r\right)\right) \tag{21}$$

这样，依次求解出后 $n_h n_s$ 列，组合便得到完整的雅克比矩阵 $\boldsymbol{J}(\boldsymbol{x})$。

改进后量子神经网络训练算法的迭代步骤为：

(1)将样本集输入网络，得到网络输出 $\hat{\boldsymbol{y}}$ 和误差 $\boldsymbol{e}=\boldsymbol{y}-\hat{\boldsymbol{y}}$，计算所有样本的平方误差和 E，并计算出雅克比矩阵 $\boldsymbol{J}$；

(2)由式(17)计算 $\Delta\boldsymbol{x}_k$；

(3)用 $\boldsymbol{x}_k+\Delta\boldsymbol{x}_k$ 代入网络计算新的平方误差和 E'。若 $E'<E$，则将 μ 除以 $\boldsymbol{\theta}$，并设 $\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\Delta\boldsymbol{x}_k$，转到步骤(1)；否则将 μ 乘以 $\boldsymbol{\theta}$，转到步骤(2)。

当式(11)计算的梯度的模小于给定值，或输出均方误差和 E 减小到某个目标误差，或迭代次数达到给定上限时，算法停止迭代，网络被认为收敛。

4　实验结果与分析

选择配置为Intel E8400, 2 G内存的PC机一台，以MATLAB2009a为实验平台对本文算法进行仿真实验。为验证分类效果，在TIMIT语音库中随机选取20名说话人的语音作为实验对象。提取其中稳定的浊音段为训练数据。每名说话人的训练语音长4 s，测试语音长2 s，每人有8段不同的测试语音。参考文献[6]的预处理方法，将所选语音数据以16 ms为一帧分帧处理，分别计算16维梅尔倒谱系数(Mel Frequency Cepstrum Coefficient, MFCC)和12维线性预测系数(Linear Prediction Coefficient, LPC)，合成28维特征向量作为量子神经网络的输入进行仿真实验。这样混合两种特征参数有利于提升识别结果的稳定性[6]。目标向量为一个20维向量，用于指向20名说话人。当输入数据属于第m个说话人时，此目标向量中第m个元素为1，其余元素均为0。作说话人识别操作时，将2 s测试语音分帧并通过训练好的神经网络，然后将得到的20×125维向量逐行相加，得到一个20维的向量，此时数值最大的一行对应的说话人即为算法识别的说话人。

根据输入输出数据的维数，构建具有28个输入神经元，20个输出神经元的量子神经网络，隐层神经元选用层数为3的multi-sigmoid函数。多次实验后，将量子间隔和斜率参数分别设置为[−1.5 0 1.5]和1.5的经验值。输出神经元选用purelin线性函数。目标误差为10^{-4}，迭代上限设为1000。

文献[3,5]使用最速下降法训练神经网络，其权值最大的稳定学习速率与 E 的最大曲率成反比，即：学习速率 α 小于$2/\lambda_{\max}$，其中 λ 为式(13)Hessen矩阵的特征值[1]。实验后发现，特征值约在0到2000之间。这样学习速率 α 的稳定值应小于0.0001。但是，目标函数 E 的性能曲面较为复杂，学习速率 α 设置过小，训练结果易陷入局部极小值。综合考虑将网络权值的学习速率 α 设置为经验值0.01。而 $\boldsymbol{G}$ 不能严格的表示成 $\boldsymbol{\theta}$ 的二次函数形式，故多次试验后，将量子间隔的学习速率 α_θ 设置为经验值0.05。文献[6]使用可变速率的最速下降法训练网络权值，初始学习速率 α 设置成经验值1，其学习速率在每次更新后自适应调整。

图2给出4种不同训练算法的训练收敛值。图3给出了4种不同训练算法得到了说话人识别正确率。为了克服网络初始权值与偏置的随机性，图中数据值为独立进行10次实验结果的平均值。

从图2可以看出，相比于另外3种训练算法，本文算法训练得到收敛值更小，训练效果更优。随着隐层神经元数量的增加，网络性能的提升更为明显，这说明本文训练算法具有更强的训练能力，适合训练规模较大的网络。从图3可以看出，改进后的量子神经网络的说话人识别正确率明显高于另外3种算法，表明了改进后量子神经网络优良的分类效果。

表1给出了各训练算法一次迭代的计算复杂度以及隐层神经元为25时，用tic-toc函数统计的平均一次迭代所需的运行时间。

表1 计算复杂度与平均一次迭代时间对比

训练算法	计算复杂度	平均一次迭代时间(s)
文献[3]算法	$O(Nn_in_h)+O(Nn_hn_s)$	0.465
文献[5]算法	$O(Nn_in_h)+O(Nn_hn_s)$	0.911
文献[6]算法	–	3.654
本文算法	$O(Nn^2)+O(n^3)$	4.126

其中 $n=n_h(n_i+1)+n_o(n_h+1)+n_hn_s$，$N=K\times n_o$，$K$ 为样本个数。文献[6]算法在一次迭代中先更新网络权值，再利用人工免疫算法训练量子间隔，故在一次迭代中仍有循环嵌套，计算复杂度不能确定，只有通过平均迭代一次的时间表征。由于本文训练算法在式(17)中包含矩阵乘法和求逆运算，故在计算量复杂度上会明显大于文献[3]、文献[5]的训练算法。本文算法迭代一次所需的时间约是文献[3]算法的10倍，但并不能说明本文算法的收敛速度慢于文献[3]、文献[5]和文献[6]。

图4给出了在具有25个隐层神经元的条件下，前100次迭代中，各算法网络输出均方误差和的变化关系。以输出均方误差和为0.1为收敛界限，表2给出

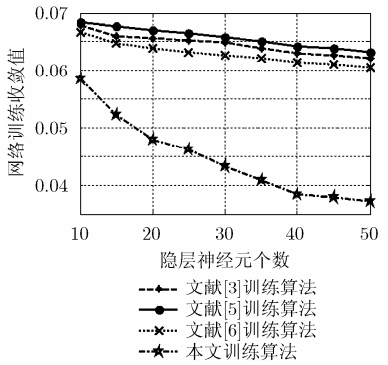

图2 网络收敛值比较图

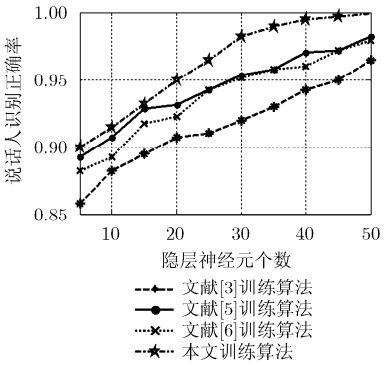

图3 说话人识别正确率比较图

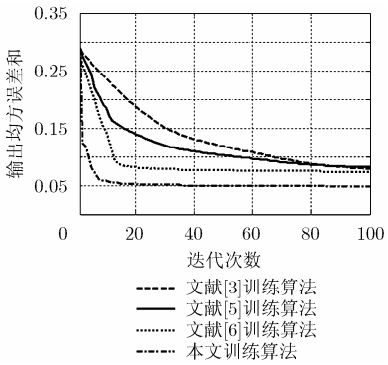

图4 输出均方误差和变化图

表2 收敛迭代次数、收敛值与收敛时间对比

训练算法	迭代次数	收敛值	收敛时间(s)
文献[3]算法	67	0.0991	31.16
文献[5]算法	54	0.0996	49.19
文献[6]算法	13	0.0924	47.50
本文算法	5	0.0794	20.63

了网络收敛时，各个算法所需的迭代次数、收敛值以及所用时间。

从表2可以看出，本文算法经过5次迭代就使网络收敛到0.0794，在迭代次数与收敛值上都明显由于其它算法。就收敛时间来看，尽管本文算法的平均一次迭代时间多于其它算法，但是收敛时间是各算法中最少的，这表明了本文算法具有更快的收敛速度。

采用函数逼近实验来验证本文训练算法在数据拟合上的效果。为增加函数逼近的复杂度，构建具有1个输入神经元，4个输出神经元的量子神经网络。每一个输出神经元逼近一个函数。输入数据为 n = [-5:0.05:4.95]，表示从-5到4.95之间每隔0.05取一点，共200个点的数组。所需逼近的4个函数分别为 $\sin(\pi n)$, $\sin(\pi n+\pi/2)$, $\sin(\pi n+\pi)$, $\sin(\pi n-\pi/2)$ 。隐层神经元选用层数为3的multi-sigmoid函数，量子间隔和斜率参数的初始值分别为[-1.5 0 1.5]和1.5的经验值。输出神经元选用purelin线性函数。目标误差为 10^{-6}，迭代上限为1000。

图5给出了在具有10个隐层神经元的条件下，网络输出均方误差和随迭代次数的变化关系。图6给出了不同隐层神经元条件下，两种算法完成收敛所需的迭代次数。此迭代次数为10次实验的平均值，这样可以在一定程度上消减了权值选取随机性带来的影响。在每一次实验中，两种训练算法选取的初始权值相同。

由于量子神经网络在隐层使用了多层激励函数和量子间隔，相比Back-Propagation(BP)神经网络就具有了更高的自由度，可以在隐层做出更多细微的调整。如果将神经网络看成一个参数黑箱，量子神经网络具备了更多的参数，需要更多的计算量，也获得更良好的性能。LM-BP神经网络的计算复杂度为 $O(Nn_{bp}^2)+O(n_{bp}^3)$，其中，$n_{bp}=n_h(n_i+1)+n_o\cdot(n_h+1)$；量子神经网络的计算复杂度为 $O(Nn^2)+O(n^3)$，其中，$n=n_h(n_i+1)+n_o(n_h+1)+n_hn_s$。LM-BP神经网络平均一次迭代时间为0.0751 s；本文算法平均一次迭代时间为0.0813 s，基本处于一个数量级。而从图5可以看出引入LM算法的量子神经网络在迭代220次左右，输出均方误差和即下降到 10^{-3}，而LM-BP算法直到迭代结束也未达到这一量级。当训练结束时，本文算法的网络误差收敛到约 10^{-4}，相比于LM-BP的 10^{-3}，性能提高了约10倍。

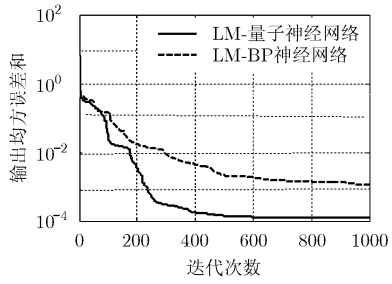

图5 输出均方误差和变化图

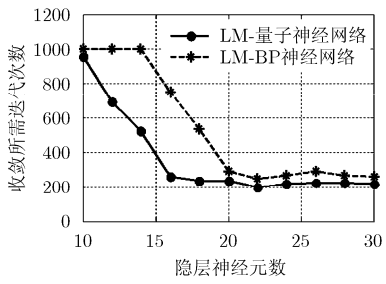

图6 收敛所需迭代次数比较图

从图6可以看出，本文算法相较于LM-BP算法，平均提前176步即可使网络达到收敛。在隐层神经元个数为12和14时，本文算法相比LM-BP算法可以分别提前462和489步使网络收敛。这表明了LM-量子神经网络具有更快的收敛速度。

5　结束语

本文修改了多层激励函数量子神经网络中量子间隔的目标函数，从而避免了原训练算法中两个目标函数的相互冲突，突破了多层激励函数量子神经网络的只能用于数据分类的应用限制，扩展了其应用领域。同时将 LM 算法引入其训练算法，使网络收敛值明显下降，并有效提高训练速度与训练效果。最后通过仿真验证了本文所述训练算法的性能。

改进之后的量子神经网络可以应用在函数逼近、分析预测等诸多领域。下一阶段将重点研究如何降低本文算法的计算复杂度，进一步提高网络收敛速度。

参 考 文 献

[1] Hagan M T, Demuth H B, and Beale M H. Neural Network Design[M]. Boston: PWS Publishing Company, 1995, Chapter 12, 19–27.

[2] Haykin S. Neural Networks: A Comprehensive Foundation [M]. 2nd Ed, Upper Saddle River, N.J, Prentice Hall, 1999: 38–44.

[3] Purushothaman G and Karayiannis N B. Quantum Neural Networks (QNNs): inherently fuzzy feedforward neural networks[J]. *IEEE Transactions on Neural Networks*, 1997, 8(3): 679–693.

[4] Karayiannis N B and Xiong Y. Training reformulated radial basis function neural networks capable of identifying uncertainty in data classification[J]. *IEEE Transactions on Neural Networks*, 2006, 17(5): 1222–1234.

[5] 孙健, 张雄伟, 孙新建. 一种新的量子神经网络训练算法[J]. 信号处理, 2011, 27(9): 1306–1312.
Sun Jian, Zhang Xiong-wei, and Sun Xin-jian. A new training algorithm for quantum neural networks[J]. *Signal Processing*, 2011, 27(9): 1306–1312.

[6] 王金明, 王耿, 郑国宏, 等. 一种量子神经网络说话人识别方法[J]. 解放军理工大学学报(自然科学版), 2012, 13(3): 242–246.
Wang Jin-ming, Wang Geng, Zheng Guo-hong, *et al.*. Speaker recognition method based on quantum neural networks[J]. *Journal of PLA University of Science and Technology* (*Natural Science Edition*), 2012, 13(3): 242–246.

[7] Hagan M T and Menhaj M B. Training feedforward networks with the marquardt algorithm[J]. *IEEE Transactions on Neural Networks*, 1994, 5(6): 989–993.

[8] 张鸿燕, 耿征. Levenberg-Marquardt 算法的一种新解释[J]. 计算机工程与应用, 2009, 45(19): 5–8.
Zhang Hong-yan and Geng Zheng. Novel interpretation for Levenberg-Marquardt algorithm[J]. *Computer Engineering and Applications*, 2009, 45(19): 5–8.

[9] Indrajit M and Srikanta R. Comparing the performance of neural networks developed by using Levenberg-Marquardt and Qusai-Newton with the gradient descent algorithm for modeling a multiple response grinding process[J]. *Expert Systems with Applications*, 2012, 39(3): 2397–2407.

[10] Wilamowski B M and Hao Yu. Improved computation for Levenberg-Marquardt training[J]. *IEEE Transactions on Neural Networks*, 2010, 21(6): 930–937.

张翼鹏：男，1988 年生，硕士生，研究方向为人工神经网络与信息隐藏技术.

陈　亮：男，1974 年生，教授，硕士生导师，研究方向为智能信号处理.

郝　欢：男，1988 年生，硕士生，研究方向为信息隐藏技术.