量子并行神经网络

陈佳临 王伶俐
(复旦大学微电子学院 上海 200433)

摘 要 本文在前期量子概率神经网络(QPrNN)的基础上,提出了一种物理可实现的量子神经网络,称为量子并行神经网络(QPNN).主要特点是基于量子神经元的激活机制,利用量子并行性跟踪所有网络状态来提高分类结果.与之前的研究相比,在网络各个中间层和输入层之间添加了连接,增加了量子神经网络的非线性表达能力,所以结构上可以向深层网络发展.由于 QPNN 独特的量子门性质,该模型在很多条件下对噪声不敏感,涵盖了相位偏移和幅值翻转噪声.QPNN 的另一个优势是可以作为内存使用,不但可以像经典内存一样存取数据,还可以作为生成模型,产生新数据.在实验验证部分,本次研究选取了两个标准的例子,MNIST 手写体识别和 Cifar-10 来验证其测试误差.实验结果表明,QPNN 只需采用经典神经网络 3% 左右的神经元资源即可超过相对应的全连接前向神经网络.与 QPrNN 相比,MNIST 的分类测试准确率提高了 0.2%;Cifar-10 测试准确率提高了 3%.同时,MNIST 的正确取回概率平均提高了 2%.

关键词 量子神经元;量子并行神经网络;量子可实现;容错性;量子内存
中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2019.01205

Quantum Parallel Neural Network

CHEN Jia-Lin WANG Ling-Li
(*School of Microelectronics, Fudan University, Shanghai* 200433)

Abstract Based on the previous quantum probability neural networks (QPrNN) research, an improved quantum-implementable neural network, namely Quantum Parallel Neural Network (QPNN) model is proposed in this paper. QPNN is a kind of quantum feed-forward neural networks which is composed of a new type of quantum neurons, or qurons, and their connections. If the input quron x' satisfies $x'\cdot\omega'\geq 0$, then the output quron will be activated with a probability larger than 0.5 and rest otherwise. In this sense, qurons are similar to classical neurons based on the sigmoid function. Taking advantage of quantum parallelism, QPNN can trace all possible network states to get the final output. Moreover, the most interesting points of QPNN is that we can combine several basic networks with different parameters even different structures at the same time to improve the result. To achieve this purpose, only n qubits are needed to perform quantum multiplexer gates to create 2^n separable networks. Therefore, QPNN has unique advantages over classical feed-forward neural networks. Compare to the previous QPrNN, direct links between each layer and input layer are added to enhance the nonlinearity of QPNN, and hence can be developed into deep network structure. Due to its unique quantum nature, this model is robust to several quantum noises under certain conditions, such as phase-flip channel and bit-flip channel, which can be efficiently implemented by universal quantum computers. Another advantage is that QPNN can be used as memory to retrieve the most relevant data and even to generate new data. During the learning phase of QPNN, the most expensive part is the

收稿日期:2017-10-25;在线出版日期:2018-12-28.**陈佳临**,博士,助理研究员,主要研究方向为量子计算、量子物理架构和机器学习.E-mail:jl_chen@fudan.edu.cn.**王伶俐**(通信作者),博士,教授,主要研究领域为逻辑综合、可重构计算、量子计算和机器学习.E-mail:llwang@fudan.edu.cn.

summation over all possible states of the hidden layer qurons. Therefore, in order to focus on states with relatively large probabilities, classical sampling methods are used to sample the layer. In the experiments, this strategy is used to tradeoff between the learning speed and the accuracy, where for a hidden layer with m qurons, only 2^{m-3} sampling states are needed to calculate. Alternatively, in a real quantum computer (suppose there exit a real quantum computer), this can be done by repeatedly measuring the hidden layer several times to obtain a set of most likely layer states. Note that the classical methods sample the layer efficiently only in the second layer as they are tensor product states. As a result, for a deeper network structure, this strategy does not work well and only quantum computers perform efficiently. For verifying the performance of QPNN, we apply it to two real-life classification applications, i. e. MNIST handwritten digit database recognition and Cifar-10 classification. Here, in both experiments, Matlab simulation results show that hat only about 3% neuron resources are required in QPNN to obtain a better result than the classical feedforward neural network. Compare to the previous QPrNN, the test accuracies of MNIST and Cifar-10 are improved by 0.2% and 3% respectively. In addition to the resources saving, QPNN can also be used as memory to retrieve the most relevant data where the successful retrieve probability of MNIST is improved by 2% than QPrNN.

Keywords Quantum neuron (Quron); Quantum Parallel Neural Network (QPNN); quantum-implementable; fault tolerant; quantum memory

1 引 言

近年来,量子计算已经成为了信息学、计算机科学和物理学中最前沿的交叉研究领域之一.量子计算机的概念最早由费曼提出[1],目的只是为了利用量子效应来帮助研究量子力学.不过,以叠加和纠缠为代表的量子效应,后来被发现在一些经典计算问题中也有广阔的应用,例如大数分解[2]、数据库搜索[3]和全局二值优化[4]等.随着量子硬件研究的不断进步,量子计算机距离实际应用也越来越近.

然而,能体现量子计算机独一无二优势的领域依然有限,最主要的原因在于大自然的物理法则限制了量子计算机提供海量信息的能力.尽管 n 个量子比特理论上可以存储 2^n 个复数的信息,但每次访问其结果的时候只能按照概率分布获取其中之一.这就要求量子算法本质上必须与经典算法有明显区别,经典算法通常是某组输入到某组输出的函数,而量子算法需要同时考虑所有可能的输入-输出组合.至今为止,量子算法的重要成就是相比经典算法,在寻找隐含子群的问题上提供了指数级加速的能力,以及在数据库搜索中提供几何级加速.因此一直以来,量子算法研究的重心就是如何在更广的应用范围内拓展这种指数级或者几何级加速的能力.然而,在一些其他问题,例如机器学习中,初步研究表明量子算法具有其他优势,包括更高的分类精度、更深层的知识挖掘[5]以及记忆容量的增加[6]等.本质上来说,与经典的 Kolmogorov 概率论相比,量子概率理论更加适合用来对人类的认知和决策进行建模[7].

机器学习专门研究计算机怎样模拟或实现人类大脑的学习行为,涵盖了图像识别、语音识别、语言翻译和模式识别等[8].在众多机器学习算法和模型中,人工神经网络(Artificial Neural Network, ANN)占据了重要的地位.ANN 是一种简化的大脑模型,通过模拟神经元和突触的结构来处理各种复杂问题.一般来说,只要改变神经元的个数、突触的连接方式以及网络的深度,ANN 就能以较高的精度逼近任意函数.自从 20 世纪中叶发明至今,ANN 的学习性能一直在逐步提高,尤其是深度人工神经网络已经在图像和语音等某些领域达到甚至超过了人类水平[9].

然而,随着大数据时代的到来以及摩尔定律的逐步失效,寻找更好的数据处理方式已经迫在眉睫,很多学者已经开始探索量子计算和机器学习结合的方法[10-11].其中,一种简单可行的途径是利用量子效应来加速经典算法的效率,比如 K-means、K-近邻和支持向量机的量子化已经取得了可观的加速[12-14].然而,占据当今机器学习主导地位的神经网络却一

直难以量子化.关键原因在于神经元激活函数是非线性和耗散的,这和量子力学线性可逆的动力学特征相矛盾.而这种性质的神经元是前馈神经网络(Feedforward Neural Network,FFNN)、循环神经网络(Recruit Neural Network,RNN)等大部分神经网络的基础.因此早期阶段,量子神经网络的研究集中于 Hopfield 等一类基于关联内存(associative memory)而不是非线性激活函数的神经网络中[15-16].关联内存可以存储 n 个网络状态,然后对每个输入数据,输出一个与输入有最短汉明距离(Hamming distance)的网络状态.事实上,很少有实际的分类问题依赖于汉明距离,因此量子 Hopfield 网络的应用价值有限.随着深度学习的兴起,基于玻尔兹曼机(Boltzmann machine)的神经网络开始得到关注.由于玻尔兹曼机的基本原理基于量子统计力学,因此对其进行量子化是最简单的选择[17].量子玻尔兹曼机的成功很大程度上取决于量子退火器或者可编程量子光学开关阵列等特殊硬件[18-20],因为这类量子硬件可提供比经典计算机更快的采样速度来调节模型参数,加速训练过程的收敛.尽管以量子玻尔兹曼机为代表的量子神经网络被普遍认为将超越传统神经网络,但由于特殊硬件的缺失,使得这类基于量子退火等特殊算法的网络很难被模拟,至今也没有发表过在标准测试集,例如 MNIST 等上的实验数据.

事实上,构建占据当今深度学习主流的 CNN 和 RNN 等网络结构的基础仍然是 FFNN.因此很多研究[21-25]尝试将 sigmoid 等非线性激活函数量子化,构建更一般的量子神经网络(Quantum Neural Network,QNN).与经典的 ANN 相比,这些 QNN 在某些应用上展现出了优势,例如鸢尾花识别[24]和非线性函数拟合[25].但问题在于,这些量子化方案都引入了非线性算符,尽管非线性量子力学早有研究[26],但非线性算符是否可以量子实现仍存在很大争议.文献[21]提出了用量子点间相互作用的时间演化来模拟单个神经元,但要模拟整个网络却很困难.此外,这种量子点方案是一种特殊的依赖硬件模型,而不是一个通用的模型.最近,文献[27]提出了一种基于量子门的量子感知器模型,然而,这种感知器却不带激活函数,作者也没有给出实验数据验证其性能.只有文献[28]给出了一种利用量子相位估计算法(quantum phase estimatiaon)实现感知器神经元的方法.显然,将经典神经元直接量子化代价很大,而且只是用量子器件实现经典功能意义不大.由上述文献发表情况可以看出,通用的量子神经网络(QNN)的研究仍然处于探索阶段,甚至连量子神经网络的标准还是一个重要的问题[23].

本文中,我们将严格遵守量子计算的基本原则,即量子酉变换和测量操作,在前期工作[29]上做出改进.在前期工作中,我们提出一种可在通用量子计算机上实现的量子概率神经网络(Quantum Probability Neural Network),为了和本文命名区别,简称 QPrNN.类似于 ANN,QPrNN 由量子神经元,简称 Quron,以及神经元之间的互连构成.由于量子计算的并行性,QPrNN 可以追踪所有可能的网络状态,甚至是不同权重和结构的网络输出来提高性能.因此,相比于经典 FFNN,QPrNN 拥有独特的优势.但该模型的不足之处在于第三层或更深层网络的激活状态和输入无关,对问题的建模主要由第二层隐层和控制层来决定,因此很难向深度发展.为此,本文在此基础上进行改进,将输入层直接作用到所有隐层,称为量子并行神经网络(Quantum Parallel Neural Network, QPaNN,或直接简称为 QPNN),为量子神经网络向深度学习发展提供可能.

在实验部分,本文采用了两个标准的分类实验,鸢尾花识别 MNIST 手写体识别①和 Cifar-10 分类实验②,来进行和经典 FFNN 和 QPrNN 的对比,展示了 QPNN 的性能和特点.这两个实验的 Matlab 仿真显示,相比于经典 FFNN,QPNN 可以达到和超过经典 FFNN 的测试精度,并且所需的神经元只有原来的 3%左右,因此在量子计算机上,这将大大减少前向的操作数.而相比于 QPrNN,改进后的模型在性能上有了小幅提高.

本文主要贡献如下:

(1)在先前的量子神经网络工作上做出了改进,提出了一种可完全物理实现可向深度发展的量子神经网络结构模型.

(2)利用量子并行性提高神经网络的预测性能,并且采用标准测试集进行数值仿真,验证了量子效应在机器学习中的作用.

(3)提出了量子记忆体的概念,初步讨论了其在 MNIST 手写体识别中的应用.

本文第 2 节给出文中符号的规定和人工神经网络的基本介绍;第 3 节给出 QPNN 的网络结构和量子门线路图;第 4 节给出 QPNN 的学习算法;第 5

① The MNIST DATABASE of handwritten digits website. http://yann.lecun.com/exdb/mnist/

② The Cifar-10 dataset. http://www.cs.toronto.edu/~kriz/cifar.html

节通过两个分类实验对比 QPNN 和经典 FFNN 以及 QPrNN;第 6 节是结论.

2 预备知识

2.1 符号约定

狄拉克符号,例如 $|\boldsymbol{x}^n\rangle$ 表示将样本 n 制备成相应的量子态.

本文采用张量分析中常用的符号约定来保持公式的简洁和可读性.约定如下:x_i^n 表示样本 n 的第 i 个分量;ω_j^i 表示神经元 i 到神经元 j 的权重;输出 y_k^n 表示样本 n 的输出条件概率值 $p(|y^n\rangle=|k\rangle|\boldsymbol{x}^n)$,或者简写为 $p(|y^n\rangle=|k\rangle)$.张量的缩并操作采用爱因斯坦求和约定①的一个变种,即一个表达式中凡是在上标和下标都出现过至少一次的记号,将对该指标在其值域内自动求和.例如:

$$y=\sum_{i=1}^{3}c_i x^i d_i=c_1x^1d_1+c_2x^2d_2+c_3x^3d_3$$

将简写为

$$y=c_i x^i d_i.$$

此外,根据上下文,f' 表示函数 f 对特定变量的偏导;$x=(x_1\cdots x_m)_2$,表示 x 的 m 位二进制展开,即每个分量 $x_i\in\{0,1\}$.

2.2 前向神经网络介绍

在 FFNN 中,前一层神经元的输出经过线性变换输入到下一层,然后对该值做非线性变换得到对应神经元的输出.事实上,如果没有非线性函数(又称为激活函数),即使 FFNN 层数再多,也只是一个线性划分分类器,用途有限.由此可知,激活函数是 FFNN,实际上也是大部分 ANN 的关键组成部分.一种常用的激活函数是感知函数(perceptron function)

$$y_j=\begin{cases}1, & \text{如果 } x_i\omega_j^i\geq 0\\ -1, & \text{其他}\end{cases},$$

其中,y_j 是第 j 个神经元的输出,x_i 表示第 i 个输入神经元的值.上述函数的意义就是第 j 个输出神经元的值 y 是上一层所有神经元输出的线性组合,然后与特定的阈值 0 进行比较.

感知函数神经元的输出只有离散的两种值,1 和 −1,尽管计算比较简单,但难以进行求导学习.因此,为了改进这一缺陷,发展了连续型的 sigmoid 激活函数

$$y_j=sgm(x_i\omega_j^i)=\frac{1}{1+e^{-x_i\omega_j^i}}.$$

Sigmoid 函数如图 1 所示,其值域 $y\in(0,1)$.

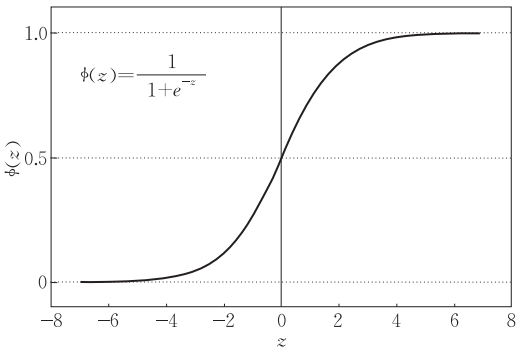


图 1 Sigmoid 函数示意图

基于 sigmoid 等非线性函数的 FFNN 被广泛运用在各种实际问题之中,一个关键原因就是该网络具有很强的学习能力.在机器学习里,学习的定义是:对一个特定的任务(例如图片分类),在一个假设集 H(例如,线性组合类)中寻找最优的元素 $h\in H$,使得该 h 的损失函数值在 H 的所有元素中最小.因此,学习的第一步就是先假设 H,第二步是选择合适的损失函数(loss function),然后求解最优化问题得到 h.

对于很多实际应用来说,回归任务的损失函数选择平方差函数(squared error),而分类会选择交叉熵(cross entropy),当然这并不绝对.FFNN 的突出优势就是模型复杂度(也就是假设集 H)足够高,并且通过反向传播(back-propagation)算法能找到比较优的 h.换句话说,FFNN 一般总能找到合适的权重使得输出和实际值吻合地非常好.

3 网络结构和量子门实现

由于量子力学中的状态转移算符都是线性算符,所以量子力学是一种线性理论.并且为了保持概率和守恒,该线性算符必须是酉算符,即 $UU^\dagger=I$.由此可见,量子神经网络的首要难题就是如何用线性的酉算符实现神经网络的非线性,因为非线性是 ANN 成功的关键之一.

显然,线性的量子门电路无法构造非线性算符,除非加入非线性门.在严格的量子计算与量子信息领域,非线性量子门只有测量算符.因此,为了部分

① https://en.wikipedia.org/wiki/Einstein_notation

地解决这一难题，我们基于测量与编码的方式引进一类新型的量子神经元，称为 Quron，并在此基础上构建完整的 QNN.

Quron 的功能如图 2 所示. 假设第一层中每个神经元都处于某个计算基态，那么下一层与之连接的神经元 y 将处于叠加态：

$$|y\rangle=\frac{1}{2}(1+ie^{i(x\cdot\omega+b)})|0\rangle+\frac{1}{2}(1-ie^{i(x\cdot\omega+b)})|1\rangle \tag{1}$$

从上式可知，神经元 y 处于 $|0\rangle$ 状态的几率为 $\left|\frac{1}{2}(1+ie^{i(x\cdot\omega+b)})\right|^2=\frac{1}{2}-\frac{1}{2}\sin(x\cdot\omega+b)$；处于激活 $|1\rangle$ 状态的几率为 $\left|\frac{1}{2}(1-ie^{i(x\cdot\omega+b)})\right|^2=\frac{1}{2}+\frac{1}{2}\sin(x\cdot\omega+b)$. 如果引进向量 $\boldsymbol{\omega}'=(b,\omega_1,\cdots,\omega_d)$ 和 $\boldsymbol{x}'=(1,x_1,\cdots,x_d)$，那么就有 $x\cdot\omega+b=\boldsymbol{x}'\cdot\boldsymbol{\omega}'$. 因此，若 $\boldsymbol{x}'\cdot\boldsymbol{\omega}'(\mathrm{mod}\,2\pi)>0$，则 y 激活的概率大于 0.5，反之小于 0.5.

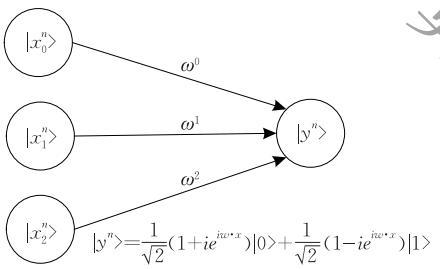


图 2　Quron 示意图

在输入为计算基态的情况下，Quron 可以看作是 sigmoid 神经元的量子化版本. 假设输入并非计算基态，即神经元 $|x\rangle$ 处于叠加态，那么该层的状态是所有 n 个神经元的张量积，写为 $\sum_{i=0}^{2^n-1}c_i|i\rangle=|x_0\rangle\otimes\cdots\otimes|x_n\rangle$，其中 $|i\rangle$ 表示 2^n 维 Hilbert 空间的第 i 个计算基态. 因此，图 2 输入-输出的联合状态将是

$$|\psi\rangle=\sum_i c_i|i\rangle\otimes(f_1(d_j^i\omega^j)|1\rangle+f_0(d_j^i\omega^j)|0\rangle) \tag{2}$$

其中，$f_0(x)=\frac{1}{2}(1+ie^{ix})$，$f_1(x)=\frac{1}{2}(1-ie^{ix})$，$\boldsymbol{d}^i$ 是行向量，其分量 d_j^i 表示数字 i 二进制展开的第 j 位数字. 例如，因为 2 的二进制表示是 $(10)_2$，所以 $d_1^2=0$ 和 $d_2^2=1$.

上述 Quron 可以通过量子相位门来实现：假设量子比特 $|y\rangle$ 和 $|x\rangle$ 分别被初始化为 $\cos\frac{\theta}{2}|0\rangle+e^{i\varphi}\sin\frac{\theta}{2}|1\rangle(\theta,\varphi\in\mathbb{R})$，以及计算基态 $|0\rangle$ 或者 $|1\rangle$. 以 $|y\rangle$ 为控制位，$|x\rangle$ 为目标位，作用下列控制相位门

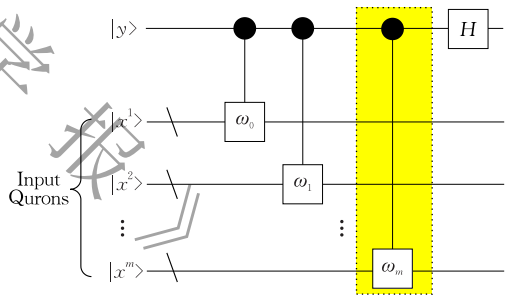
$$CP(\omega)=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{i\omega}\end{pmatrix},$$

接着再对 $|y\rangle$ 施加一个 H 门，记为 H_y，那么最终这个系统的状态变为

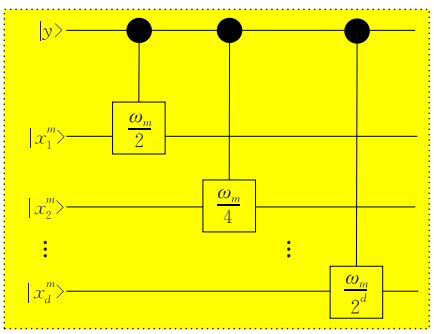
$$H_yCP_{\omega_m}(|y\rangle|x\rangle)=\frac{1}{2}\left(\cos\frac{\theta}{2}+\sin\frac{\theta}{2}e^{i(\varphi+\omega\cdot x)}\right)|0\rangle|x\rangle+\frac{1}{2}\left(\cos\frac{\theta}{2}-\sin\frac{\theta}{2}e^{i(\varphi+\omega\cdot x)}\right)|1\rangle|x\rangle.$$

可以看出，为了实现量子神经元方程(1)，只需设定 $\theta=\pi$ 和 $\varphi=\frac{\pi}{2}+b$ 即可.

基于上述方程，对于一个样本 $x\in\mathbb{R}^m$，如果每一个分量都用 d 个二进制来表示，那么 x 可以用 m 个量子神经元来表示，每个神经元包含 d 个量子比特. 如图 3(a)所示，输入量子神经元 j 与输出量子神经元 y 的权重 ω_j 可以用一个控制相位门 $CP(\omega_j)$ 作用到 $|y\rangle=1/\sqrt{2}(|0\rangle+ie^{ib}|1\rangle)$ 和 $|x^j\rangle$ 上来实现，然后再施加一个 H 门.



(a) 实现量子神经元的量子线路图



(b) 在 $|x^m\rangle$ 上实现 $CP(\omega_m)$ 的量子线路图

图　3

若 x^m 满足 $0<x^m<1$，则用 d 个二进制数字来表示它，即数值精度为 2^{-d}. 图 3(b)显示了作用到输入量子神经元 m 的相位门细节，即对于量子神经元

m 的第 k 个 qubit，需作用相位门 $CP(2^{-k}\omega_m)$.

我们将基于上述 Quron 构建的神经网络，称为量子并行神经网络(简称 QPNN). 类似于经典的马尔科夫过程，QPNN 也有概率转移矩阵，称为网络层转移矩阵(Layer Transition Matrix，LTM)，定义如下：

LTM：若第 $n-1$ 和第 n 层分别含有 k_{n-1} 和 k_n 个 Quron，定义第 n 层的 LTM 是一个 $2^{k_{n-1}}\times 2^{k_n}$ 矩阵 $\boldsymbol{\beta}_n$，其分量 $(\boldsymbol{\beta}_n)_i^j=\langle i|\boldsymbol{\beta}_n|j\rangle$ 表示第 $n-1$ 层的状态为 $|j\rangle$，第 n 层的状态为 $|i\rangle$ 的转移概率. 并假定当 $n=1$ 时，$\boldsymbol{\beta}_1$ 是输入概率分布组成的向量.

LTM 能方便的描述 Quron 的动力学过程. 例如，图 2 的输出可以写为

$$p(y=1)=(\boldsymbol{\beta}_1)_i(\boldsymbol{\beta}_2)_1^i \tag{3}$$

其中，$(\boldsymbol{\beta}_1)_i=|c_i|^2$ 和 $(\boldsymbol{\beta}_2)_1^i=|f_1(d_j^i\omega^j)|^2$ 分别是第一层和第二层的 LTM.

文献[29]描述了一个三层全连接 QNN 的前向状态传播过程，如图 4 所示. 该结构是经典神经网络的直接量子化，但通过式(3)可以发现，输出对于输入的非线性只存在于 $\boldsymbol{\beta}_1$，即输入层向隐层的转移矩阵之中，而 $\boldsymbol{\beta}_2$ 系数则只与权重 ω 有关. 这一不足限制了 QPNN 向深层网络发展的潜力，因为本质上更多层的网络等价于三层结构. 为了弥补这一缺陷，本文在此基础上，提出一种改进的结构，如图 5 所示.

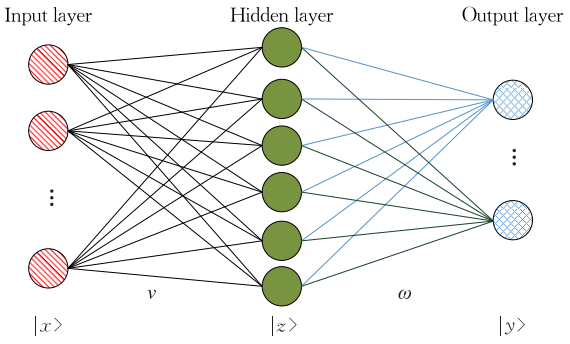


图 4 三层结构 QPrNN 示意图

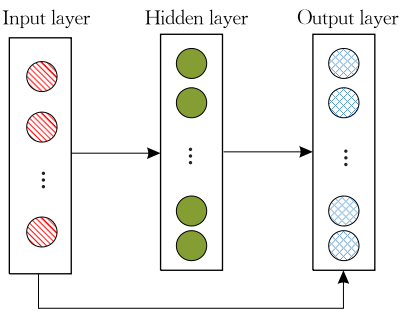


图 5 QPNN 结构意图

图 5 中，输入层不仅与隐层连接，与输出层也直接连接. 这一改进的目的在于使 $\boldsymbol{\beta}_2$ 的矩阵元素同样成为输入的非线性函数，同时这也使 QPNN 与经典神经网络在本质思想上有了不同.

正如引言中所述，人工神经网络来源于对大脑复杂化学过程的简化. 主要过程是输入对象作用到部分神经元，然后经过突触传播到更深层神经元，产生输出. 然而，若大脑神经元处于量子纠缠态，那么对一部分神经元的作用，将通过纠缠作用到其他神经元上. 某种程度上，这一过程近似为输入直接作用到所有神经元上. 本文中，QPNN 就是对这一过程的建模. 输入首先作用到所有层的量子神经元；其次，不同层的量子神经元状态通过转移矩阵 LTM，向下一层进行转移. QPNN 的具体动力学过程如下：

首先，设第 n 个训练样本为向量 $\boldsymbol{x}^n\in\mathbb{R}^d$，那么输入层第 j 个神经元的状态为 $|x_j^n\rangle$，$j=1,2,\cdots,d$. 根据式(1)，可得到隐层中第 i 个神经元的状态为

$$|z_i^n\rangle=f_0(x_j^n v_i^j)|0\rangle+f_1(x_j^n v_i^j)|1\rangle,$$

v 代表隐层的权重. 由于隐层含有 m 个 Quron，那么该层的状态为 $\sum_{i=0}^{2^m-1}c_i^n|i\rangle=|z_1^n\rangle\otimes\cdots\otimes|z_m^n\rangle$. 下一步，输出层第 l 个输出 Quron 的状态为

$$|y_l^n\rangle=c_i^n f_1(m_l^{n,i})|1\rangle+c_i^n f_0(m_l^{n,i})|0\rangle,$$

其中

$$m_l^{n,i}=d_j^i\omega_l^j+x_j^n u_l^j,$$

ω 和 u 分别代表隐层和输入到输出层的权重. 对于一个多分类问题，需要测量多个 Quron 的量子态来表示分类结果. 对于第 n 个样本 $\boldsymbol{x}^n$，测量输出层得到结果为 $k=(k_1\cdots k_p)_2$ 的条件概率为

$$\begin{aligned}p(y^n=k|\boldsymbol{x}^n)&=\sum_{i=0}^{2^m-1}(\boldsymbol{\beta}_2)_i^n(\boldsymbol{\beta}_3)_{k=(k_1\cdots k_p)_2}^{n,i}\\&=\sum_{i=0}^{2^m-1}\left|c_i^n\prod_{l=1}^{p}f_{k_l}(m_l^{n,i})\right|^2\end{aligned} \tag{4}$$

其中，$\boldsymbol{\beta}_2$ 和 $\boldsymbol{\beta}_3$ 是第二层(隐层)和第三层(输出层)的 LTM. 为了清晰起见，式(3)把求和变量 i 的范围写了出来，因为这表达了 QPNN 的基本思想：输出是隐层所有 2^m 个可能状态的期望值. 由此看出，QPNN 不能被经典计算机有效模拟. 某种程度上，经典神经网络的输出只是考虑了隐层的一种状态，即所有有神经元都被激活的网络状态，而量子神经网络则追踪了所有可能状态，并计算了它们输出的期望值. 值得注意的是，式(4)中 $f_{k_l}(m_l^{n,i})$ 仍然是输

入的非线性项，而在式(2)中这一项与输入无关，因此相对于图 4 中的结构，理论上提高了建模能力，并且因为每一层的输入都是非线性项，为深层网络发展提供了可能性.

从上述分析可知，QPNN 建模的本质是将输出的条件概率分布 $p(y|x)$分解为输出 Quron 张量积的线性组合. 类似于傅里叶级数展开，函数在越多的基函数上的展开越逼近真实解. 式(4)中每一个张量积项 $\prod_{l=1}^{p} f_{k_l}(m_l^{n,i})$都可以认为是真实概率分布的一个基函数. 为了产生更多基函数，我们进一步利用量子并行性，同时计算多个不同权重或结构的神经网络. 为了实现这个目的，需要利用量子多选门(quantum MUX)来选择不同的权重或者结构，如图 6 所示.

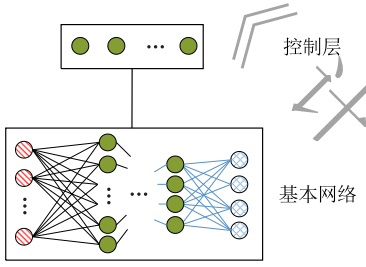


图 6　具有控制层的 QPNN 结构示意图

图 6 将整个结构划分为控制层和基本网络，两者之间施加下列形式的量子多选门(Quantum Multiplexer，QMUX)

$$\begin{pmatrix} U_1 & & \\ & \ddots & \\ & & U_{2^n} \end{pmatrix} \tag{5}$$

其中，U_i表示实现第 i 个网络的算符. 因此控制层中的 n 个 Quron 就可以实现 2^n个不同权重或结构的基本网络. 此外，为了提高分类性能，在每个基本网络的输出端可以施加 2^n个不同的量子门$\{\boldsymbol{P}^{(i)}, i=1,\cdots,2^n\}$.

定义.　P 称为输出变换门，Output Transformation Gate，简称 OTG.

一个完整结构的 QPNN 的输出可以表示如下：

$$Y_k^n = \gamma_i^n y^{(i)n}_j P^{(i)j}_k,\ i=1,2,\cdots,2^n \tag{6}$$

i 表示第 i 个基本网络，γ_i^n表示控制层处于基态$|i\rangle$的概率，即 $\gamma_i^n = p(|\gamma^n\rangle = |i\rangle)$，$|y^{(i)}\rangle$和 $P^{(i)}$分别是第 i 个基本网络的输出和 OTG.

图 7 实现了一个完整的三层 QPNN，其中$|\varphi\rangle$表示控制层的 Quron；$|y\rangle$，$|z\rangle$和$|x\rangle$分别是图 5 中输出层、隐层和输入层的 Quron. 其中，隐层和输出层的每个 Quron 都由一个量子比特构成，初始化为 $1/\sqrt{2}(|0\rangle + ie^{ib}|1\rangle)$的形式，$b$ 是待定参数由训练得到. 网络中的其他参数也都由训练来确定. 此外，控制层量子神经元状态$|\varphi\rangle$可以固定或者由输入决定，图中显示的是固定状态的控制层. 为了实现式(4)中的 QMUX，需要将不同的相位门作用到对应量子神经元上. 具体来说，就是要实现下列两种形式的 QMUX

$$\begin{pmatrix} CP(\omega_i^{(1)}) & & \\ & \ddots & \\ & & CP(\omega_i^{(2^n)}) \end{pmatrix} \tag{7}$$

$$\begin{pmatrix} P^{(1)} & & \\ & \ddots & \\ & & P^{(2^n)} \end{pmatrix} \tag{8}$$

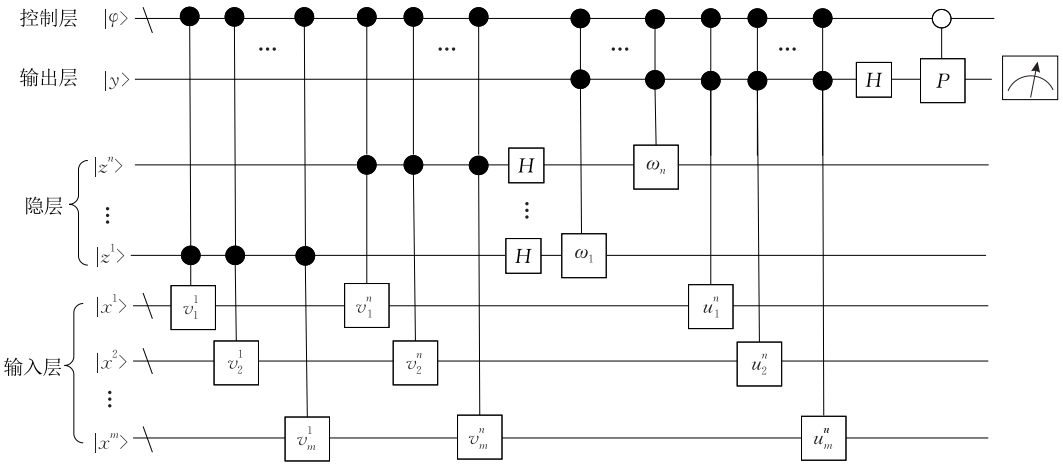


图 7　三层 QPNN 的量子线路图

在上述两种以$|\varphi\rangle$为控制比特的 QMUX 中,第一种作用在隐层和输入层上,实现不同权重连接的功能,$\omega^{(j)}$表示第 j 个子网络的权重;第二种作用在输出层上,进行输出变换,$\boldsymbol{P}^{(j)}$表示第 j 个子网络的输出变换矩阵.最后,在计算基态下测量输出层得到分类结果.由于量子测量的随机性,上图所示的过程应该重复多次,然后选择最有可能的测量值作为分类结果.对于 QPNN 中所需量子门的物理实现细节可参考文献[29]第 4 节.

现实中,任何量子门电路的物理实现都不可避免的会引入环境噪声,这里简单分析下 QPNN 对于噪声影响的敏感度.从方程(3)可以看出,如果上一层的状态经受了相位错误 $R_z(\theta)$,那么 $\boldsymbol{\beta}_2$ 和 $\boldsymbol{\beta}_3$ 都将保持不变,因此对相位错误免疫.其次,在与输出神经元作用后,任何输入神经元上的 X 翻转都不会影响到测量结果,因为 $\boldsymbol{\beta}_2$ 和 $\boldsymbol{\beta}_6$ 的上下标 i 同时进行了改变,所以不影响最后的求和结果.实际上,只有去极化通道(depolarizing channel)和幅值衰减(amplitude damping)才会在任何时间影响测量.因此,可以认为 QPNN 是一种容错的量子网络结构,对于很多噪声都有鲁棒性.

QPNN 的另一大优势是可以用作记忆体.记忆是指信息被编码、存储和取回的过程.正如前文所述,QPNN 为了输入样本数据,需要把经典数据储存为二进制的基态形式.这个策略的代价是需要运用和经典比特一样多的量子比特数目,但是,利用量子比特的叠加态性质,我们可以同时将所有的样本存为一个叠加态.具体来说,对 n 个样本 $x^1,x^2,\cdots,x^n$,输入层可以初始化为下述状态

$$|x\rangle=\sum_{i=1}^{n}\frac{1}{\sqrt{n}}|x^i\rangle,$$

对所有隐层 Quron 求偏迹,那么系统的约化密度算符为$|\phi\rangle\langle\phi|$,其中$|\phi\rangle$的形式如下:

$$\rho=\frac{1}{\sqrt{n}}\sum_{i,j}p_i^j|x^iy_j\rangle \tag{9}$$

p_i^j表示第 i 个输入样本输出是 j 的概率.从上式可知,为了取回特定类型的数据,我们只需对输出层进行测量,例如测量结果为 j,那么再对输入层进行测量,其结果将大概率的坍缩到标签为 j 的样本上.此外,利用量子幅值放大算法,为了得到概率为 $1/n$ 的标签,只需试验 $O(\sqrt{n})$次即可.

更一般的,若输入层所有 qubit 都初始化为均衡叠加态$|+\rangle=1/\sqrt{2}(|0\rangle+|1\rangle)$.在这种情况下,输入层的状态将是所有可能样本的均衡叠加态,这一数目是如此之大以至于任何超级计算机都不可能进行模拟.但是 QPNN 仍然可以对这样一个状态进行处理,其输出就是所有可能样本的分类结果.这样,按照式(8),对一个特定的标签,QPNN 很可能产生一个新的输入值,即新的样本.换句话说,QPNN 可以产生新的数据,是一种生成模型.

4 学习算法

在神经网络领域,学习算法的目的是在参数空间内寻找到一组合适的参数(权重)使得输出的结果最接近正确值.那么在 QPNN 中,学习过程就是寻找合适的权重使得正确结果的概率最大化.为了实现这个目的,通常是先选定一个损失函数 $L^n(\omega)$,然后运用梯度下降方法来最小化该函数.

梯度下降是一个经典的迭代算法.主要步骤如下:

1. 设定 ω 的初始值,例如 $\omega^{(1)}=0$;
2. 计算梯度 $\nabla_\omega L^n$;
3. 更新 $\omega^{(t+1)}=\omega^{(t)}-\lambda\nabla_\omega L^n$,$\lambda$ 是学习速率.

这样在迭代 n 次以后,得到了最终的权值向量 $\boldsymbol{\omega}^{(n)}$.由此可见,学习算法中计算量最大的就是第二步计算梯度 $\nabla_\omega L^n$.本节将叙述如何用反向传播的方法计算每一层的梯度.为了描述方便,符号 y_k^n 表示输出层状态为$|k\rangle$的概率,即 $P(|y^n\rangle=|k\rangle)$.

首先,计算输出层权重 ω.将式(4)重写为如下形式:

$$y_k^n=(\boldsymbol{\beta}_2)_i^n(\boldsymbol{\beta}_3)_k^{n,i}.$$

一般来说,对一个具有 p 个 Quron 的层来说,该层 LTM 的分量 $\beta_k^{n,i}=\prod_{l=1}^{p}f_{k_l}(m_l^{n,i})$,满足 $k=(k_p\cdots k_1)_2$.若定义 $m_l^{n,i}$ 为 Quron l 的预激活值(pre-activation value),则 $\beta_k^{n,i}$ 的偏导为:

$$\frac{\partial\beta_k^{n,i}}{\partial(m_l^{n,i})}=\frac{(-1)^{k_l+1}}{2}\cos(m_l^{n,i})\prod_{l'\neq l}f_{k_{l'}}(m_{l'}^{n,i}).$$

因此,若设 L^n 是 $\boldsymbol{x}^n$ 的损失函数,则 ω 的梯度为:

$$\nabla_{\omega_j}L^n=\sum_k\frac{\partial L^n}{\partial y_k^n}\frac{\partial y_k^n}{\partial\omega_j}=\sum_k e_k^n(\beta_3')_{k,j}^{n,i}(\boldsymbol{\beta}_2)_i^n\boldsymbol{d}_i^{\mathrm{T}},$$

其中,$\boldsymbol{d}_i^{\mathrm{T}}$是 d_i 的转置,$e_k^n=\dfrac{\partial L^n}{\partial y_k^n}$以及$(\beta_3')_{k,j}^{n,i}=\partial(\boldsymbol{\beta}_3)_k^{n,i}/\partial(m_l^{n,i})$.

下一步,在输出层的基础上计算隐层权重:

$$\begin{aligned}\nabla_{v_j}L^n&=\sum_{k,i}\frac{\partial L^n}{\partial y_k^n}\frac{\partial y_k^n}{\partial(\boldsymbol{\beta}_2)_i^n}\frac{\partial(\boldsymbol{\beta}_2)_i^n}{\partial v_j}\\&=\sum_k e_k^n(\boldsymbol{\beta}_3)_k^{n,i}(\beta_2')_{i,j}^n\boldsymbol{x}_n^{\mathrm{T}}.\end{aligned}$$

可以看出，下一层输出 $|k\rangle$ 的误差 e_k^n 通过乘以 $(\boldsymbol{\beta}_3)_k^{n,i}$ 反向传播回了上一层，作为了当前层的误差项，这是因为 $\boldsymbol{\beta}_3$ 是上一层到输出的概率传输矩阵. 因此，为了写出梯度的一般式，这里我们定义对于第一层，即输入层，$d_i^{(1)}=x_i$ 以及 $\boldsymbol{\beta}_1=1$. 这样，第 m 层的梯度可以写成如下形式：

$$\nabla_{\omega_j^{(m)}} L^n=\sum_k e^{(m)}{}_k^n(\beta'_m)_{k,j}^{n,i}(\hat{\beta}_{m-1})_i^n d^{(m-1)\mathrm{T}}{}_i \quad (10)$$

其中，$e^{(m)}{}_k^n$ 表示第 m 层的误差中由下一层误差 e_k^n 反向传播过来的部分以及

$$\hat{\beta}_{m-1}=\prod_{m'=1}^{m-1}\beta_{m'}$$

表示从输入到 $m-1$ 层的 LTM.

从式(10)可以看出，梯度的计算涉及到对当前层所有可能的层状态 i 求和，是该层神经元个数的指数级规模. 也就是说，若该层有 n 个 Quron，则需累加 2^n 项. 因此，加速 QPNN 学习过程的一个关键是减少层状态的有效个数. 式(4)显示，若一个状态的转移概率，例如 $(\boldsymbol{\beta}_3)_k^{n,i}$ 很小，那么其对最后结果的贡献可以忽略不计，所以，原则上我们可以只计算那些相对概率较大的状态即可. 在经典计算机的模拟上，这可以通过数值采样来做到. 在下一节的两个实验中，这一策略被用来降低运行复杂度. 具体来说，对一个含有 m 个 Quron 的层来说，我们采样 2^{m-3} 次，然后计算其和作为式(10)的梯度的近似.

而在量子计算机上(假设我们有真正的量子计算机)，这一加速过程就是要对当前层进行反复的测量，因为其测量结果就是相对概率较大的状态. 实际上，经典计算机上的采样方法只适合于很少网络层的结构，因为只有第二层的状态是每个 Quron 的张量积，所以可以分别对 Quron 进行数值采样. 而更深网络层的状态并不满足这一条件，所以无法通过只对每个 Quron 进行 0-1 采样得到最后结果，只有量子计算机才能克服这一困难.

5　数值实验

在机器学习中，衡量一个模型的真实误差可以分为两部分：训练误差和测试误差. 一个好的模型往往就是这两种误差间的一个权衡. 在文献[29]中，我们用 Matlab 仿真的方式验证了 QPrNN 和经典 FFNN 的性能对比. 从实验结果可以得知，由于 QPNN 的本质是通过三角函数来逼近目标函数，而三角函数存在周期性，所以误差会表现出一定的波动性. 但随着梯度的不断下降以及学习速率的下降，这一波动性也将越来越弱，最终趋于平稳.

本文中，由于 QPNN 改进了原有的网络结构，使其理论上可以往深度学习方向发展. 但正如上一节结尾所述，由于还没有量子计算机，所以训练一个深度的量子网络很困难. 例如，训练含有两个隐层，每个隐层含有 10 个 Quron 的 QPNN，则需计算 2^{20} 个状态的梯度下降，这对经典计算架构提出了很大的挑战. 因此，为了更简单的反映 QPNN 和 QPrNN 的性能差别，本文仍将采用文献[29]相同的网络结构，通过 MNIST 和 Cifar-10 两个标准测试集来对比 QPNN、QPrNN 以及经典 FFNN 的测试误差.

(1) MNIST 手写体识别

MNIST 数据集包含了 0～9 这十种类别的例子，事实上，这个简单的图像分类例子可能是模式识别领域最著名的实验用例，类似于果蝇在生物实验中的地位. 无论多么复杂的神经网络，MNIST 都能体现出其特别的一面. 所以我们通过这个实验来验证 QPNN 的测试误差. 标准的 MNIST 数据集包含 60 000 个训练样本和 10 000 个测试样本，每个样本都是一个 28×28 的归一化灰度图.

为 MNIST 所设计的 QPNN 整体结构如图 8 所示.

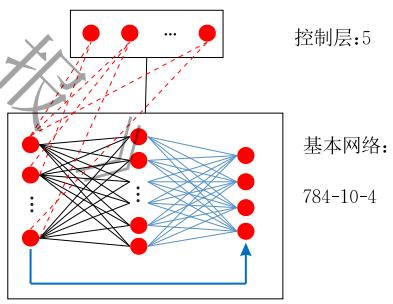


图 8　QPNN 结构图

其结构主要包含以下特点：

① 基本网络. 三层结构，输入层 784 个 Quron，隐层 10 个 Quron，输出层 4 个 Quron. 每个输入 Quron 包含 10 个 qubit，因此每个样本的数值精度为 2^{-10}. 输出最多可以有 $2^4=16$ 类，取其中的 10 类作为结果.

② 控制层. 包含 5 个 Quron，每个 Quron 都是 1 位 qubit，与输入层进行全连接. 因此可以生成 32 个不同权重的基本网络.

③ 输出层. 输出转移矩阵是 32 个随机选定的 16×16 置换矩阵，用符号 $\boldsymbol{P}^{(i)}$，$i=1,\cdots,32$ 来表示.

图 9 显示了不同个数基本网络在训练 50 遍以后的测试误差,可以看出在只有一个基本网络的时候,误差约为 11%,而随着网络数的递增,32 个基本网络同时存在的情况下可以将误差缩小到 2%左右.不过随着网络数目的递增,误差减小幅度将会越来越少,因此在权衡训练时间后,我们的基本网络数目最终选择了 32.

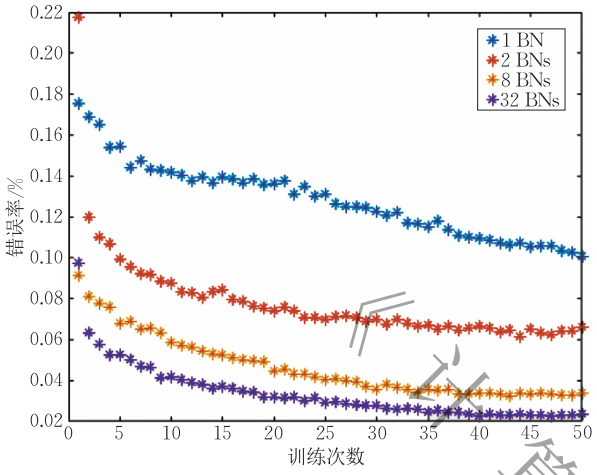


图 9 不同数量基本网络的训练误差对比图(BN 是基本网络的缩写)

此外,QPNN 输出的概率值可以作为下一个分类器的输入.例如,我们可以将 3 个最高概率值作为可能的类别,然后进行进一步处理.数值实验显示,如果采用这样的策略,在 QPNN 输出这一层的测试正确率将达到 99.7%.当然,如何进一步处理已经超出了本次研究的范围.表 1 列出了实验结果对比.其中,QPNN 的错误概率为 2.11%,相比 QPrNN 有了小幅的提高,同时也高于很多全连接的 FFNN,但是落后于一些运用了额外策略的 FFNN(例如 elastic distortions 等数据处理方法).

表 1 MNIST 的实验结果对比

神经网络	测试错误率/%
3-layer FFNN, 1000 hidden neurons	4.5
4-layer FFNN, 500+150 hidden neurons	2.95
4-layer FFNN, 500+150 hidden layer neurons [distortions]	2.45
FFNN, 784-500-500-2000-30+ nearest neighbor, RBM+NCA training [no distortions]	1.0
7-layer FFNN 784-2500-2000-1500-1000-500-10 (on GPU) [elastic distortions]	0.35
QPrNN 10 hidden, 5 select qurons	2.38
3-layer QPNN(本文), 10 hidden, 5 select qurons	2.11

由于这些策略不属于全连接神经网络结构的研究范畴,所以结果不具有可比性.从表格中也可以看出 QPNN 的一大优势就是神经元个数的大量减少,即只需 10 个隐层神经元和 5 个控制层神经元.

值得注意的是,训练很多相同结构子网络的策略在经典 FFNN 中并不可行,因为即使运用 OTG 置换了样本的输出顺序,训练后的权重也只是空间上的变换.那为什么在 QPNN 中,OTG 能提高准确率?一般说来,分类的目的就是找到样本之间的差异.在经典的 FFNN 中,样本的标记并不重要.换句话说,如果将原来标记为 1 的样本 A 现在标记为 4,样本 B 仍然为 3,那么在 FFNN 中的改变只是将输出神经元 1 的权重和神经元 4 的权重相互交换,识别率不会有变化.但是在 QPNN 中,假设样本 A 和 B 的标记分别为其二进制形式 $|0001\rangle$ 和 $|0011\rangle$,即第三个输出神经元不同.但是当我们把 A 标记为 $|0100\rangle$ 后,A 和 B 将相差三个输出神经元,所以神经网络中的权重将会发生很大变化.因此,通过在 $[1,2,\cdots,16]$ 中随机选择 10 个数字赋予训练样本,我们将训练出同一种网络结构下的不同权重.一般来说,不同权重代表了训练样本的不同的特征,因此对子网络输出概率的线性叠加(凸组合)会给出比单个网络更好的结果,这就解释了图 9 的结果.

因此在训练过程中,我们随机选择了 32 个 16×16 的置换矩阵 $\boldsymbol{p}^{(1)},\cdots,\boldsymbol{p}^{(32)}$ 对训练样本的输出进行置换.具体置换规则是在第 i 个子网络中,样本 x^j 的输出变为 $y^j(\boldsymbol{p}^{(i)})^{\mathrm{T}}$.而在输出端,QPNN 的输出为

$$Y_k^n=\gamma_i^n y^{(i)n}{}_j \boldsymbol{p}^{(i)}{}_k^j,\quad i=1,2,\cdots,32,$$

其中,γ_i^n 表示控制层处于 $|i\rangle$ 的概率,$\boldsymbol{y}^i$ 是第 i 个子网络的输出矩阵.所以最终,各个子网络中被打乱的标签又重新被置换回到了正确的顺序.

根据式(4),QPNN 中每个基本网络的输出 $y_k^n=(\boldsymbol{\beta}_2)_i^n(\boldsymbol{\beta}_3)_k^{n,i}$,由于三维向量 $\boldsymbol{\beta}_3$ 的存在,使整个训练集上的网络输出无法写成单一的矩阵乘形式.与 QPrNN 相比,这一变化使其在经典计算机上的训练时间成倍增加,表 2 显示了 QPNN、QPrNN 以及经典 FFNN 的训练时间对比.

表 2 MNIST 的训练时间对比

神经网络	每个 epoch 耗时/s
3-layer FFNN, 500 hidden neurons	1
3-layer QPrNN, 15 hidden qurons	110
3-layer QPNN, 15 hidden qurons	910

其中，每个 epoch 为 600 个样本，硬件环境为 Intel E5-2620＋Titian XP，软件为 Matlab 2016b.

需要说明的是，尽管结果不是最优，但为了大幅减少训练时间，QPNN 的训练可以在 QPrNN 的基础上进行，即先充分训练 QPrNN，然后固定已有参数，最后只对输入和输出间直接连接的权重进行训练. 本文的两个实验都采用了这种训练策略，因为除了训练时间上的考虑，我们还发现 QPNN 对超参数比较敏感，因此这样能降低训练结果的不确定性.

(2) Cifar-10 测试

Cifar-10 由 60000 张 32×32 的 RGB 彩色图片构成，共 10 个分类，包含 50000 张训练图片和 10000 张测试图片. 从图 10 可以看出这个数据集中含有大量特征和噪声，且识别物体的比例不一. 因而，Cifar-10 相对于 MNIST 等传统图像识别数据集，具有相当的挑战性.

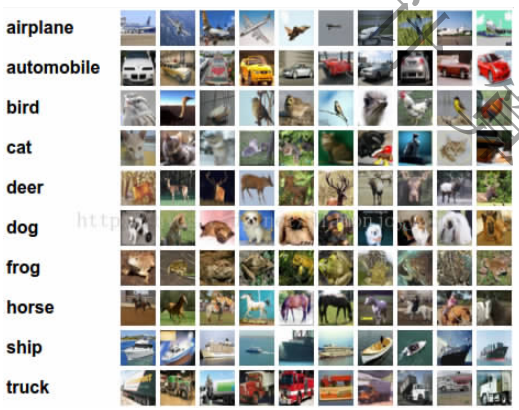


图 10　Cifar-10 数据集示意图

本实验中，输入样本为原始的 32×32×3 图片并且没有经过任何预处理，即 3072 维的输入向量. 实验采用的 QPNN 结构和上一节 MNIST 完全相同，即 5 个控制量子神经元，10 个隐藏层量子神经元，以及 4 个输出量子神经元. 从表 3 可以看出，经典的两层 FFNN 结构，错误率在 55.6%～58%，而 QPNN 错误为 49.3%. 当然，受限于现有的硬件条件和软件算法(训练 1 个 epoch 耗时 20 min)，本文没有探索更多量子神经元的全连接结构. 但由于 Cifar 相对复杂的特征，使得无论是经典还是量子全连接神经网络都很难对其进行有效分类. 一个通用的解决方法是参考 CNN，将全连接变为部分共享连接，对原始图片的每个子块进行运算. 事实上，全连接神经网络是很多其他变种，如 CNN 或 ResNet 等的基础，而本文的目的是量子化全连接神经网络，因此上述解决方法并不在考察之列.

表 3　Cifar-10 的实验结果对比

神经网络	测试错误率/%
3-layer FFNN, 300 hidden neurons	58
3-layer FFNN, 500 hidden neurons	55.6
3-layer QPrNN, 10 hidden Qurons, 5 select Qurons	52.8
3-layer QPNN(本文), 10 hidden Qurons, 5 select Qurons	49.3

从上述两个实验可以看出，QPNN 作为 FFNN 的量子化版本，相比于 FFNN 首先性能相等甚至更优；其次，神经元个数大幅减少，导致的结果就是在量子计算机上前向所需的计算操作数也大幅减少. 具体来说在 MNIST 和 Cifar-10 中，每个基本网络含有 10 个隐藏量子神经元和 4 个输出量子神经元，那么其参数个数为 $784\times10+784\times4+10\times4=11016$，即需要 11016 个 QMUX-5 相位门，因为一共有 $2^5=32$ 个基本网络. 而含有 500 个隐藏神经元的经典 FFNN 需要 $784\times500+500\times10=397000$ 个参数，约需要 397000 个乘加计算和 510 个激活函数计算. 当然，QPNN 的实际运行时间，即产生 QMUX 门的代价，取决于未来量子计算机的具体结构.

另一个优势是 QPNN 可以作为概率型内存使用. 在 MNIST 实验里，如果输入态制备为所有样本的叠加态，即 $|\psi\rangle=\sum_{i=1}^{n}|x^i\rangle$，那么输出将是所有样本输出概率值的叠加态. 即在输出为 j 的情况下，得到第 i 个样本的概率为 p_i^j/n. 因此，通过计算所有样本的取回概率，表 4 列出了 MNIST 在 10 个标签下的每一类样本成功取回的概率，其平均成功概率为 46.37%，超过了文献[29]中的 44.7%，基本相当. 换句话说，假设输出测量结果为 j，那么平均每 2.2 次测量输出可以取回属于第 j 类的一个样本. 此外，若输入叠加态受到某些随机噪声的影响，那么取回的将可能是一个全新的 j 类样本. 从这种意义上来讲，量子噪声对分类问题有可利用价值. 但由于还没有真正的量子计算机，所以该效应的量化还有待进一步研究.

表 4　MNIST 实验中成功取回样本的概率

标签	取回概率 (%)
0	51.6
1	53.2
2	43.7
3	43.9
4	48.1
5	42.3
6	48.0
7	44.9
8	47.2
9	40.8

6 总 结

由于机器学习,尤其是神经网络涉及到许多高维张量运算,而量子计算就是基于张量运算,因此量子神经网络很可能具有特殊优势.本文提出的量子并行神经网络 QPNN 就证实了这一观点.具体来说,QPNN 的特点有:(1)将经典神经元的激活方式与量子力学基本原理相结合,运用量子并行性原理,同时计算多个不同的神经网络;(2)改进了网络连接结构,可以向深度网络发展;(3)简单和规整的量子门电路实现,可以容错的进行物理实现.文中用了两个标准的分类实验:MNIST 手写体识别和 Cifar-10 来对其性能进行验证.实验表明,在相同甚至更好的分类精度下,量子神经网络在预测时所需的计算资源只有经典 FFNN 的 3%左右(MNIST 中隐层神经元个数比为 15∶650,Cifar-10 中为 15∶500),揭示了了量子神经网络的独特优势.

尽管 QPNN 有着上述优势,但现在通用量子计算机还没有面市,有必要对其学习算法进行更大优化.由于学习过程中,QPNN 需要考虑每一个隐层的指数多个状态,所以下一步要研究更好的采样算法对深层量子网络进行采样,加速学习过程的收敛速度.

参 考 文 献

[1] Nielsen M, Chuang I. Quantum Computation and Quantum Information. Cambridge, UK: Cambridge University Press, 2000

[2] Shor P. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 1997, 26(5): 1484-1509

[3] Childs A M, Landahl A J, Parrilo P A. Quantum algorithms for the ordered search problem via semidefinite programming. Physical Review A, 2007, 75(3): 032335

[4] Farhi E, Goldstone J, Gutmann S, et al. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. Science, 2001, 292(5516): 472-475

[5] Murphy K P. Machine Learning: A Probabilistic Perspective. Massachusetts, USA: MIT Press, 2012

[6] Diamantini M C, Trugenberger C A. High-capacity quantum associative memories. arXiv: 1506.01231v1, 2015

[7] Busemeyer J R, Bruza P D. Quantum Models of Cognition and Decision. Cambridge, UK: Cambridge University Press, 2012

[8] Schuld M, Sinayskiy I, Petruccione F. An introduction to quantum machine learning. Contemporary Physics, 2015, 56(2): 172-185

[9] Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge, USA: The MIT Press, 2016

[10] Wang Long-Hao, Long Gui-Lu. Big data and quantum computation. Chinese Science Bulletin, 2015, 60: 499-508 (in Chinese)
(王龙浩, 龙桂鲁. 大数据与量子计算. 科学通报, 2015, 60: 499-508)

[11] Huang Yi-Ming, Lei Hang, Li Xiao-Yu. A survey on quantum machine learning. Chinese Journal of Computers, 2018, 41(1): 145-163(in Chinese)
(黄一鸣, 雷航, 李晓瑜. 量子机器学习算法综述. 计算机学报, 2018, 41(1): 145-163)

[12] Wiebe N, Kapoor A, Svore K. Quantum deep learning. arXiv: 1412.3489, 2014

[13] Wiebe N, Braun D, Lloyd S. Quantum algorithm for data fitting. Physical Review Letters, 2012, 109(5): 050505

[14] Lloyd S, Mohseni M, Rebentrost P. Quantum algorithms for supervised and unsupervised machine learning. arXiv preprint arXiv:1307.0411, 2013

[15] Menneer T, Narayanan A. Quantum artificial neural network architectures and components. Information Sciences, 2000, 128(3-4): 231-255

[16] Ventura D, Martinez T. Quantum associative memory. Information Sciences, 2000, 124(1-4): 273-296

[17] Biamonte J, Wittek P, Pancotti N, et al. Quantum machine learning. Nature, 2017, 549(7671): 195-202

[18] Wiebe N, Kapoor A, Granade C, Svore KM. Quantum inspired training for Boltzmann machines. arXiv: 1507.02642, 2015

[19] Adachi S H, Henderson M P. Application of quantum annealing to training of deep neural networks. arXiv: 1510.06356, 2015

[20] Amin M H, Andriyash E, Rolfe J, et al. Quantum Boltzmann machine. arXiv:1601.02036, 2016

[21] Behrman E C, Nash L R, Steck J E, et al. Simulations of quantum neural networks. Information Sciences, 2000, 128(3): 257-269

[22] Panella M, Martinelli G. Neural networks with quantum architecture and quantum learning. International Journal of Circuit Theory and Applications, 2011, 39(1): 61-77

[23] Schuld M, Sinayskiy I, Petruccione F. The quest for a quantum neural network. Quantum Information Processing, 2014, 13(11): 2567-2586

[24] Sahni V, Patvardhan C. Iris data classification using quantum neural networks. AIP Conference Proceedings, 2006, 864: 219-227

[25] Li Fei, Zheng Baoyu. A study of quantum neural networks// Proceedings of the IEEE International Conference on Neural Networks and Signal Processing. Nanjing, China, 2003: 539-542

[26] Weinberg S. Precision tests of quantum mechanics. Physical Review Letters, 1989, 65(2): 485-488

[27] Silva A J D, Ludermir T B, Oliveira W R D. Quantum perceptron over a field and neural network architecture selection in a quantum computer. Neural Networks, 2016, 76: 55-64

[28] Schuld M, Sinayskiy I, Petruccione F. Simulating a perceptron on a quantum computer. arXiv:1412.3635, 2014

[29] Chen Jialin, Wang Lingli, Charbon E. A quantum-implementable neural network model. Quantum Information Processing, 2017, 16(10): 245

CHEN Jia-Lin, Ph. D., assistant professor. His research interests include quantum computing, quantum physical architecture and machine learning.

WANG Ling-Li, Ph. D., professor. His research interests include logic synthesis, reconfigurable computing, quantum computing and machine learning.

Background

In recent decades, quantum computing has been one of the fastest growing areas in both physics and information science. Some 'exponentially or geometrically fast' applications, such as integer factorization and database search, have been exploited. With the rise of big-data age, several recent academic contributions have explored the idea of using the advantages of quantum computing to improve machine learning algorithms. Many proposals attempt to find a quantum equivalent for the perceptron or sigmoid neurons from which artificial neural networks are constructed. However, most of them introduce nonlinear operators in their structures because of the nonlinear and dissipative dynamics of classical neurons. Although nonlinear quantum mechanics has been studied for years, the physical implementation of nonlinear quantum computing is still controversial. As a result, most previous Quantum Neural Network (QNN) proposals cannot be simulated by linear and unitary dynamics of quantum computing directly. In fact, QNN research is still in its infancy.

To partially resolve this embarrassment, we propose a quantum-implementable neural network, called QPNN, which obeys the unitary principles of quantum computing so that non unitary operators involved are measurements only. It is composed of a new type of quantum neurons, or qurons, and their connections. QPNN can utilize quantum parallelism to trace all possible network states and even create many networks with different parameters to improve the result. To verify this idea, the Matlab experimental results of MNIST handwriting recognition and Cifar-10 classification show that only about 3% neuron resources are required in QPNN to obtain a good result than the classical feedforward neural network.