J. Math. Math. Sci., 2022 (**1**), 57–65

STATISTICAL QUANTUM NEURAL NETWORKS

Do Ngoc Diep (Hanoi, Vietnam)

(Received 20 December 2021; accepted 12 February 2022)

Abstract. We explain a new idea of how to use the high probability interval thresholds for neurons in quantum neural networks. Some basic quantum neural networks were analyzed and constructed in a recent work of the author. In particular the Least Square Error Problem (LSEP) and the Linear Regression Problem (LRP) was discussed. In this paper we analyze a new look on the threshold rules for neurons, taking the intervals of high probability in place of classical sigmoid half-line threshold and then we construct the least-square quantum neural network (LS-QNN), the polynomial interpolation quantum neural network (PI-QNN), the polynomial regression quantum neural network (PR-QNN) and chi-squared quantum neural network (χ^2-QNN). We use the corresponding solutions or statistical tests as the threshold for the corresponding training rules.

1. Introduction

The classical machine learning (ML) [9],[5] theory was created in 1950, but only 9 years later in 1959 Arthur Samuel gave a definition of ML being ".... computers learning without being explicitly programmed". It should understand that the unknown functions (inputs-outputs) are deduced from a set of training data. The classical ML is characterized by the types: 1) *supervised learning*, i.e. classes of inputs corresponds to different classes, (2) *unsupervised learning*, i.e. the large data are summarized into a few stereotypes, and (3) *reinforcement learning*, i.e. one rewards, reinforces the current strategy. Normally the classical MLs are working with *big data*, see [7],[3],[8].

Key words and phrases: Qubit, quantum gate, quantum network, statistical tests
2010 Mathematics Subject Classification:

The quantum Machine Learning (QML) are characterized by using quantum computing into the ML theory. One uses the ordinary interpretation of qubits, 1-qubit quantum gates, such as the Pauli matrices,

$$\mathbf{1} = -\boxed{\text{Id}}- \sim \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

$$X = -\boxed{\text{X}}- \sim \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

$$Y = -\boxed{\text{Y}}- \sim \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},$$

$$Z = -\boxed{\text{Z}}- \sim \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

then the 2-qubit gates like

$$\text{XOR} = -\boxed{\text{XOR}}- \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$\text{SWAP} = -\boxed{\text{SWAP}}- \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

and finally, Measurements

$$\text{M} = -\boxed{\text{M}}-,$$

etc.

One uses the quantum algorithms to solve the ML problems with use of quantum computing. The most important ingredients in QML are: - choices of *training sets*, i.e. finite sets of given vectors in order to then find some value corresponding to another input, - *pattern completion*, i.e. adding missing informations to incomplete inputs, and - *associative memory*, i.e. retrieving stored memory vectors upon an input.

This paper is the second part of the paper [2], in which we continue to treat the cases of polynomial regression with the high probability region bounds used as the corresponding thresholds. In Section 2, we analyze the comceptions of classical aritificial neural networks (ANN) and quantum neural networks (QNN). We explain also how to introduce some approxiamtion of sigmoid functions by normal probability distributions. This let us to use a lot of test and confidential intervals and criteria from statistics. The next Section 3 is devoted

to the problem of training the least square quantum neural networks (LS-QNN), like the least square interpolation, the general polynomial regression quantum neural network (PR-QNN) and the chi-squared test training (χ^2-QNN) in the next Section 4. We look at the problem of least square problem (LSP) solution of the general polynomial regression and propose to use the quantum Gauss-Jordan Elemination (GJE) Code to solve the LSP equation. This let us to make the network works outperform the classical approaches. The paper is finished with a conclusion in Section 5.

2. Quantum Neural Networks

2.1. Thresholds

Following the model of Deutsch, a quantum neural network QNN(s, d) is a set of all quantum circuits of size s and depth d with thresholds bounded by w. Quantum gates are interconnected by wires, preserve the sources and sink gates (measured the qubits and removed the entanglements with the maining qubits. Examples of QNNs are the implementation of NAND gate, dissipative $D(m, \delta)$ and sink gates [4].

A threshold circuit is a boolean function $\mathrm{Th}^{n,\Delta} : \mathbb{Z}_2^n \to \mathbb{Z}_2$ of n integral variables $x_1, \ldots, x_n$ such that $\mathrm{Th}^{n,\Delta}(x_1, \ldots, x_n) = 1$ if and only if $\sum x_i \geq \Delta$. The class $TC(s(n), d(n))$ of threshold circuits of size $s(n)$ and depth $d(n)$, weighted by weight bound w can be approximated by elementary functions.

An equality threshold circuit is a boolean function $\mathrm{Et}^n_{w_1,\ldots,w_n} : \mathbb{Z}_2^n \to \mathbb{Z}_2$ of n integral variables $x_1, \ldots, x_n$ such that $\mathrm{Et}^n_{w_1,\ldots,w_n}(x_1, \ldots, x_n) = 0$ if and only if $\sum x_i = 0$. The class $EC(s(n), d(n))$ of equality threshold circuits of size $s(n)$ and depth $d(n)$, weighted by weight bound w can be approximated by elementary functions.

It was proven that $TC(s(n), d(n)) \subseteq EC(O(s^2(n), 2d(n))$ of weight bound $O(s(n))$ and $TC(s(n), d(n)) \subseteq EC(O(s^2(n), d(n)+1)$ of weight bound $O(s^2(n))$. And finally, $EC(s(n), d(n)) \subseteq QNN(O)d(n).\log s(n)), 2d(n))$ of precision $O(\log w + d(n) \log s(n))$. (Theorem 4.6 from [4]).

The question is whether a QNN can be implemented on Quantum Turing Machine (QTM) (Church-Turing Thesis) is difficult to answer: Quantum computing showed that the answer is No, but physicists speculate that it is Yes.

2.2. High Probability Thresholds for Neurons

Let us remind that the standard neuron thresholds are defined by some sigmoid function

$$f(x) = \frac{1}{1+e^{-cx}},$$

for some constant $c > 0$. The sigmoid function has the value $1/2$ at $x = 0$, monotonicaly increasing behaviour, the horizontal asympltotes $y = 0$ for $x \to -\infty$ and $y = +1$ for $x \to +\infty$ Often using this function to define a threshold of type

$$\mathrm{Th}_f^\delta(x) : \mathbb{Z}_2 \to \mathbb{Z}_2, \quad \mathrm{Th}_f^\delta(x) = \begin{cases} 1 & \text{if } f(x) \geq \delta, \\ 0 & \text{if } f(x) < \delta \end{cases}$$

sor some $0 \leq \delta \leq 1$.

Similarly, for functions of several variables, one uses some multivariable $\mathbf{x} = (x_1, \ldots, x_n)$ sigmoid functions, namely

$$f(\mathbf{x}) = \prod_{i=1}^{n} \frac{1}{1+e^{-c_i x}},$$

for some constants $c_i > 0$ which has the value $1/2$ at $x = 0$, monotonicaly increasing behaviour, with horizontal asympltotes $y = 0$ for $x_i \to -\infty$ and $y = +1$ for $x_i \to +\infty$. Often, one uses this function to define a threshold of type

$$Th_f^{n,\delta}(x) : \mathbb{Z}_2^n \to \mathbb{Z}_2, \quad Th_f^{n,\delta}(x) = \begin{cases} 1 & \text{if } f(\mathbf{x}) \geq \delta, \\ 0 & \text{if } f(\mathbf{x}) < \delta \end{cases}$$

for some threshold $0 \leq \delta \leq 1$.

We therefore have the following observation.

Proposition 2.1. *The one variable sigmoid function $f(x)$ has the normal curve $(x, \mathcal{N}(0,1)(x))$ of the standard normal distribution*

$$\mathcal{N}(0,1)(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} dt$$

as some curved asymptote and $f(0) = \mathcal{N}(0,1)(0) = \frac{1}{2}$.

Remark that following these sigmoid functions the thresholds are defined by the accessible intervals $[\delta, +\infty)$. We can then approximate the sigmoid functions by normal distribution functions. For normal distributions we have many tests, namely z-test, t-tests, F-tests, and corresponding confidential intervals

and criteria to define the high probability region. We use the tests and confidential intervals to make thresholds in form of high probability regions in place of the half-line intervals $[\delta, +\infty)$ for sigmoid functions. The difference error between two monotonically increasing funtions: the sigmoid functions and the normal distribution functions are rather small. We have therefore the following new idea, that didn't have been introduced before.

Corollary 2.1. *We can use the normal distributions in place of sigmoid functions to define the thresholds of neurons by using the tests and confidential intervals to find the high probability region.*

3. Least Square QNN and Polynomial Regression QNN

First we remind that many problem, including the least squared problem and polynomial interpolation problems are reduced to solving systems of linear equations. In the previous work [1] we had showed that the Gauss-Jordan elimination procedure is consisting of an application of searching the pivot columns, which is reduced to use the Grover's Search Algorithm and by the way necessary arithmetic operations over rows. The following lemma [2] is fundamental in many problems of namely the least square or the polynomial interpolation quantum neural networks.

Lemma 3.1. *The quantum Gauss-Jordan Elimination Code can be implemented in QNN.*

Let us consider the polynomial $f(\mathbf{x}) = \sum_{|\alpha|=0}^{N} a_\alpha \mathbf{x}^\alpha$ of degree N on n variables, with unkown coefficients a_α, those we want to inpterpolate, and let $(\{\mathbf{x}_{(j)}^\alpha\}\}_{|\alpha|=0}^{N}, y_j)$, $\alpha = (\alpha_1, \ldots, \alpha_n)$, $|\alpha| = \alpha_1 + \cdots + \alpha_n \leq N$ be the $N+1$ interpolating points of the polynomial, $x = (x_1, \ldots x_n)$ be the unkown variables, $\mathbf{x}_{(j)}^\alpha = \Pi_{i=1}^{n} x_{i,(j)}{}^{\alpha_i}, j = 0, \ldots, N$. The system of interpolating equations is a system of $N+1$ equation on $N+1$ *unknown variables* $a_\alpha, |\alpha| = 0, \ldots, N$:

$$f(\mathbf{x}_{(j)}) = \sum_{|\alpha|=0}^{N} a_\alpha \mathbf{x}_{(j)}^\alpha = y_j; j = 0, \ldots, N.$$

The determinant of the system is of the Vandermonde type and of size $(N+$

1) $\times (N+1)$

$$|A| = \begin{vmatrix} 1 & x_{(0)}^{(1,\ldots,0)} & \ldots & x_{(0)}^{(0,\ldots,N)} \\ 1 & x_{(1)}^{(1,\ldots,0)} & \ldots & x_{(1)}^{(0,\ldots,N)} \\ . & \ldots\ldots & \ldots & \ldots\ldots \\ 1 & x_{N}^{(1,\ldots,0)} & \ldots & x_{N}^{(0,\ldots,N)} \end{vmatrix},$$

then the system can be written as

$$A^{\dagger}A[a_{\alpha}] = A^{\dagger}[\mathbf{y}_j]. \tag{3.1}$$

The matrix of the system is nondegenerate if the interpolating points are in a generic position. In that case the solution of the system is $[a_{\alpha}]_{|\alpha=0}^{N} = (A^{\dagger}A)^{-1}A^{\dagger}\mathbf{b}$, where $\mathbf{b} = [\mathbf{y}_j]_{j=0}^{N}$.

In general case the matrix can not be invertible, but the system is consistent. Based on Lemma 3.1, we can use the Gauss-Jordan elimination procedure on quantum neural networks to find out a basis of the null-space of the augmented matrix of the system (3.1). If the sytem satisfies the consistency conditions, then there exists at least one solution. Let $(A^{\dagger}A)_{psi}^{-1}$ be the Moore-Penrose pseudoinverse of $A^{\dagger}A$, then the solution to the interpolation problem is $[a_{\alpha}] = (A^{\dagger}A)_{psi}^{-1}A^{\dagger}b.$, where $\mathbf{b} = \text{proj}_{col(A^{\dagger}A)}A^{\dagger}[\mathbf{y}_j]$ is the projection on the column space of the matrix $A^{\dagger}A$.

The general interpolated solution is

$$\hat{f}(\mathbf{x}_{(j)}) = \sum_{|\alpha|=0}^{N} a_{\alpha}\mathbf{x}_{(j)}^{\alpha} = \hat{y}_j; j = 0, \ldots, N. \tag{3.2}$$

Let us now dicuss about the implementation method on quantum neural networks. The XOR gate is implementedf in neural network (see [7]), and then the Fourier transform is implemented on neural networks ([2], Figs 2.3). Recently we used those to implement the quantum Gauss-Jordan elemination on neural networks. The algorithms are applied to our situation and we have an implementation of our least square quantum neural neworks.

We have therefore the following result

Theorem 3.1. *The Least Square Quantum Neural Network (LS-QNN) and Polynomial Interpolation Quantum Neural Networks(PI-QNN) are implementable on QNN, with complexity* $O(\sqrt{N})$.

We now apply the Least Square Method to the problem of (general) regression (GRP). Let us remind that the Grover's Searh Code can be implemented in QNN because the basic step is to repeatedly use the XOR quantum network

gate [2]. The method of QGJE [1], [6] is based on use of the Quantum Grover's Search to find the pivot columns in the matirx $A^\dagger A$.

We have therefore the following result

Theorem 3.2. *The Polynomial Regression Quantum Neural Network (PR-QNN) is implementable, i.e. the general regression problem GRP can be solved by a QNN, with complexity* $O(\sqrt{N})$.

Let us analyze how to train the GRP code in QNN. With the above interpolating quantum code, we can divide the data y_j into to treatments: regression treatment $Y_{regr} = [\hat{y}_j]$ and residual treatment $Y_{resid} = [y_j - \hat{y}_j]$, where

$$\hat{y}_j = f_{regr}(\mathbf{x}_{(i)}) = \hat{f}(\mathbf{x}_{(j)}). \tag{3.3}$$

Let us denote by

$$F = \frac{\mathrm{MS}_{regr}}{\mathrm{MS}_{resid}} = \frac{\frac{(r^2 SS_Y}{1}}{\frac{(1-r^2)SS_{resid}}{N-2}} = \frac{(N-2)r^2}{1-r^2}, \tag{3.4}$$

where r is the Pearson correlation, $r = \mathrm{Cor}(X, Y)$. We may fix a level α of explained proportion of variance and define the F-ratio $F_{(1,N-2),\alpha}$. Therefore we define the *training threshold* as if the F-ratio is in the high probability $1-\alpha$ region

$$F < F_{(1,N-2),\alpha}. \tag{3.5}$$

□

4. Chi-Squared QNN

In the nonparametric statistics, the χ^2-test plays important roles in many problems like contingency tables, homogeneity, ... Let use conside the corresponding quantum code in QNN. Denote by $\mathbf{e} = [e_{ij}]_{n\times r}$ be a contingency matrix of expected values e_{ij}. The random distribution $X = [x_{ij}]$ is a matrix of size $n \times r$. The degree of freedom is

$$df_X = \begin{cases} (n-1)\times(r-1), & \text{if } r > 1 \\ (n-1), & \text{if } r = 1 \end{cases}$$

. The chi-squared statistic is of form

$$\chi^2_X = \sum_{i=1}^{n}\sum_{j=1}^{r} \frac{(x_{ij} - e_{ij})^2}{e_{ij}}. \tag{4.1}$$

Our aim is to implement the χ^2-test in a QNN and use the chi-sqaured test as the rule of training.

Theorem 4.1. *The Chi-Squared Quantum Neural Network (χ^2-QNN) is implementable, , i.e. the χ^2-tests can be solved by a QNN, with complexity $O(\sqrt{(n-1)\times(r-1)})$.*

Indeed, the high probability $1-\alpha$ region is

$$\chi^2_X < \chi_{df,\alpha} \tag{4.2}$$

for a fixed α-level of confidence and
the *training rule* is to sink the network if the constraint is faile to be satisfied. If the constraint holds, it passes to the next layer of QNN. □

5. Conclusion

We implemented the quantum neural networks: the least square quantum Neural Network (LS-QNN) and the polynomial interpolation quantum neural networks (PI-QNN), the Polynomial Regression Quantum Network (PR-QNN) and the Chi-Squared Quantum Neural Network (χ^2-QNN). The training rules are provided with the corresponding tests from Statistics.

References

[1] D. N. Diep, D. H. Giang, N. V. Minh, *Quantum Gauss-Jordan elimination and simulation ofnaccounting principles on quantum computers*, Inter. J. of Theor. Physics, 56(2017), No 6, 1948-1960.

[2] D. N. Diep, *Some quantum neural networks*, Intl. J. Theor. Phys. 59 (2020), No. 6, 1179-1187.

[3] A. A. Ezhov, D. Ventura, *Quantum neural networks*, in Future *Directions for Intelligent Systems and Information Science*, N. Kasabov (ed.), Physica-Verlag, pp. 213-235, 2000.

[4] S. Gupta, R.K. P. Zia, *Quantum Neural Network*, Journal of computer and system sciences, 63(2001), 355-383.

[5] G. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*, Department of Computer Science, University of Toronto.

[6] K. Nagata, S. K. Patro, H. Geurdes, S. Heidari, D. N. Diep, T. Nakamura, *Various New Forms of the Bernstein-Vazirani Algorithm Beyond Qubit Systems*, Asian J. Math. & Phys., 3, No 1(2019) 1-12.

[7] M. Schuld, I. Sinayskiy, Petruccione, *An introduction to quantum machine learning*, arXiv:1409.3097v1[quant-ph]2014.

[8] M. Schuld, I. Sinayskiy, F. Petruccione, *Prediction by linear regression on a quantum computer*, arXiv:1601.07823v2[quant-ph], 2016.

[9] N. Wiebe, A. Kapoor, K. Swore, Quantum deep learning, arXiv:1412.3489v2[quant-ph], 2015.

Do Ngoc Diep
Institute of Mathematics and Appliedf Sciences
Thang Long University
Nghiem Xuan Yem road
Hoang Mai district
Hanoi
Vietnam
e-mail adress: diepdn@thanglong.edu.vn