A Quantum Implementation Model for Artificial Neural Networks

Ammar Daskin

Computer Engineering Department, Istanbul Medeniyet University, Istanbul, Turkey. E-mail: adaskin25@gmail.com

Editors: ***George Svetlichny, Danko Georgiev & Chris C. King***

Article history: Submitted on October 12, 2017; Accepted on January 30, 2018; Published on February 20, 2018.

The learning process for multilayered neural networks with many nodes makes heavy demands on computational resources. In some neural network models, the learning formulas, such as the Widrow–Hoff formula, do not change the eigenvectors of the weight matrix while flatting the eigenvalues. In infinity, these iterative formulas result in terms formed by the principal components of the weight matrix, namely, the eigenvectors corresponding to the non-zero eigenvalues. In quantum computing, the phase estimation algorithm is known to provide speedups over the conventional algorithms for the eigenvalue-related problems. Combining the quantum amplitude amplification with the phase estimation algorithm, a quantum implementation model for artificial neural networks using the Widrow–Hoff learning rule is presented. The complexity of the model is found to be linear in the size of the weight matrix. This provides a quadratic improvement over the classical algorithms.

Quanta 2018; 7: 7–18.

1 Introduction

Artificial neural networks [1–3] are adaptive statistical models which mimic the neural structure of the human brain to find optimal solutions for multivariate problems. In the design of artificial neural networks are determined the following: the structure of the network, input-output variables, local activation rules, and a learning algorithm. Learning algorithms are generally linked to the activities of neurons and describe a mathematical cost function. Often, a minimization of this cost function composed of the weights and biases describes the learning process in artificial neural networks. Moreover, the learning rule in this process specifies how the synaptic weights should be updated at each iteration. In general, learning rules can be categorized as supervised and unsupervised learning: In supervised learning rules, the distance between the response of the neuron and a specified response, called target t, is considered. However, it is not required in unsupervised learning rules. Hebbian learning rule [4] is a typical example of the unsupervised learning, in which the weight vector at the $(j+1)$th iteration is updated by the following formula (we will mainly follow [2] to describe learning rules):

$$\mathbf{w}_{[\mathbf{j+1}]} = \mathbf{w}_{[\mathbf{j}]} - \eta t \mathbf{x}. \tag{1}$$

Here, $\mathbf{x}$ is the input vector, η is a positive learning constant and $\mathbf{w}_{[\mathbf{j}]}$ represents the weights at the jth iteration. And t is the target response. Learning is defined by getting an output closer to the target response.

On the other hand, Widrow–Hoff learning rule [5], which is the main interest of this paper, illustrates a typical supervised learning rule [2, 3, 6]

$$\mathbf{w}_{[\mathbf{j+1}]} = \mathbf{w}_{[\mathbf{j}]} - \eta\sigma'(v)(t-y)\mathbf{x}, \quad (2)$$

where $v = \mathbf{x}^T\mathbf{w}$ is the activation of the output cell and $\sigma'(v)$ is the derivative of the activation function which specifies the output of a cell in the considered network, $y = \sigma(v)$: e.g., the sigmoid function, $\sigma(v) = 1/(1+\exp(-v))$. While in the Hebbian iteration the weight vector is moved in the direction of the input vector by an amount proportional to the target, in the Widrow–Hoff iteration, the change is proportional to the error $(t-y)$. If we consider multi-neurons; the activation, the output, and the target values becomes vectors: viz., $\mathbf{v}, \mathbf{y}$ and $\mathbf{t}$, respectively. When there are several input and target associations, the set of inputs, targets, activations, and outputs can be represented by the matrices X, T, V, and Y, respectively. Then, the above equations come in matrix forms as follows

$$W_{[j+1]} = W_{[j]} - \eta XT^T, \quad (3)$$

$$W_{[j+1]} = W_{[j]} - \eta(\sigma'(V) \circledast X)(T-Y)^T, \quad (4)$$

where W represents the matrix of synaptic weights.

It is known that the learning task for multilayered neural networks with many nodes makes heavy demands on computational resources. Algorithms in quantum computational model provide computational speedup over their classical counterparts for some particular problems: e.g., Shor's factoring algorithm [7] and Grover's search algorithm [8]. Using adiabatic quantum computation [9, 10] or mapping data set to quantum random access memory [11,12] speedups in big data analysis have been shown to be possible [13–15]. Furthermore, Lloyd and collaborators [16] have described a quantum version for principal component analysis.

In the recent decades, relating the neurons in the networks with qubits [17], a few different quantum networks analogous of the artificial neural networks have also been developed: e.g. [18–23] (For a complete review and list of references, see [24]). These models should not be confused with the classical algorithms (cf. [25, 26]) inspired by the quantum computing. Furthermore, using the Grover search algorithm [8], a quantum associative memory is introduced [27]. Despite some promising results, there is still need for further research on new models [24].

The quantum phase estimation algorithm (PEA) [28] provides computational speedups over the known classical algorithms in eigenvalue related problems. The algorithm mainly finds the phase value of the eigenvalue of a unitary matrix (considered as the time evolution operator of a quantum Hamiltonian) for a given approximate eigenvector. Because of this property, PEA is ubiquitously used as a subcomponent of other algorithms. While in the general case, PEA requires a good initial estimate of an eigenvector to produce the phase; in some cases, it is able to find the phase by using an initial equal superposition state: e.g., Shor's factoring algorithm [7]. In [29], it is shown that a flag register can be used in the phase estimation algorithm to eliminate the ill-conditioned part of a matrix by processing the eigenvalues greater than some threshold value. Amplitude amplification algorithm [8, 30–32] is used to amplify amplitudes of certain chosen quantum states considered. In the definition of quantum reinforcement learning [33, 34], states and actions are represented as quantum states. And based on the observation of states a reward is applied to the register representing actions. Later, the quantum amplitude amplification is applied to amplify the amplitudes of rewarded states. In addition, in a prior work [35] combining the amplitude amplification with the phase estimation algorithm, we have showed a framework to obtain the eigenvalues in a given interval and their corresponding eigenvectors from an initial equal superposition state. This framework can be used as a way of doing quantum principal component analysis (QPCA).

For a given weight matrix W; in linear auto-associators using the Widrow–Hoff learning rule; during the learning process, the eigenvectors do not change while the eigenvalues go to one [2, 6]: i.e., $lim_{j\to\infty}W_{[j]}$ converges to QQ^T, where Q represents the eigenvectors of W. Therefore, for a given input $\mathbf{x}$, the considered network produces the output $QQ^T\mathbf{x}$. In this paper, we present a quantum implementation model for the artificial neural networks by employing the algorithm in [35]. In particular, we show how to construct $QQ^T\mathbf{x}$ on quantum computers in linear time. In the following section, we give the necessary description of Widrow–Hoff learning rule and QPCA described in [35]. In Section 3, we shall show how to apply QPCA to the neural networks given by the Widrow–Hoff learning rule and discuss the possible implementation issues such as the circuit implementation of W, the preparation of the input $\mathbf{x}$ as a quantum circuit, and determining the number of iterations in the algorithm. In Section 4, we analyze the complexity of the whole application. Finally, in Section 5, an illustrative example is presented.

2 Methods

In this section, we shall describe the Widrow–Hoff learning rule and the quantum algorithms used in the paper.

2.1 Widrow–Hoff Learning

For a linear autoassociator, i.e. $Y = V$, $\sigma'(V) = I$, and $T = X$; Widrow–Hoff learning rule given in Eq. (4), also known as LMS algorithm, in matrix form can be described as follows [2,3]

$$W_{[j]} = W_{[j-1]} + \eta(X - W_{[j-1]}X)X^T. \tag{5}$$

This can be also expressed by using the eigendecomposition of $W = Q\Lambda Q^T$: i.e., $W_{[j]} = Q\Phi_{[j]}Q^T$, where $\Phi_{[j]} = [I - (I - \eta\Lambda)^j]$. $\Phi_{[j]}$ is called the eigenvalue matrix at the epoch j. Based on this formulation, Widrow–Hoff error correction rule only affects the eigenvalues and flattens them when $\eta \leq 2\lambda_{max}^{-1}$ (λ_{max} is the largest eigenvalue of W): i.e., $\lim_{j\to\infty} \Phi_{[j]} = I$. Thus, in infinity, the learning process W ends up as: $W_{[\infty]} = QQ^T$.

2.2 Quantum Algorithms Used in the Model

In the following, we shall first explain two well-known quantum algorithms and then describe how they are used in [35] to obtain the linear combination of the eigenvectors.

2.2.1 Quantum Phase Estimation Algorithm

The phase estimation algorithm (PEA) finds an estimation for the phase of an eigenvalue of a given operator [28,36]. In mathematical terms, the algorithm seen in Figure 1 as a circuit works as follows:

- An estimated eigenvector $\left|\varphi_j\right\rangle$ associated to the jth eigenvalue $e^{\iota\phi_j}$ of a unitary matrix, U of order N is assumed given. U is considered as a time evolution operator of the Hamiltonian (H) representing the dynamics of the quantum system

$$U = e^{\iota tH/\hbar}, \tag{6}$$

  where t represents the time and $\hbar$ is the Planck constant. As a result, the eigenvalues of U and H are related: while $e^{\iota\phi_j}$ is the eigenvalue of U, its phase ϕ_j is the eigenvalue of H.

- The algorithm uses two quantum registers dedicated to the eigenvalue and the eigenvector, respectively, $|reg_1\rangle$ and $|reg_2\rangle$ with m and ($n = log_2N$) number of qubits. The initial state of the system is set to $|reg_1\rangle|reg_2\rangle = |\mathbf{0}\rangle\left|\varphi_j\right\rangle$, where $|\mathbf{0}\rangle$ is the first standard basis vector.

- Then, the quantum Fourier transform is applied to $|reg_1\rangle$, which produces the following equal superposition state

$$U_{QFT}\,|reg_1\rangle\,|reg_2\rangle = \frac{1}{\sqrt{M}}\sum_{k=0}^{M-1}|\mathbf{k}\rangle\left|\varphi_j\right\rangle, \tag{7}$$

  where $M = 2^m$ and $|\mathbf{k}\rangle$ is the kth standard basis vector.

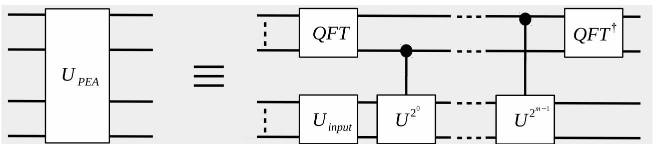


Figure 1: *The phase estimation part of the algorithm.*

- For each kth qubit in the first register, a quantum operator, $U^{2^{k-1}}$, controlled by this qubit is applied to the second register. This operation leads the first register to hold the discrete Fourier transform of the phase, ϕ_j.

- The inverse quantum Fourier transform on the first register produces the binary digits of ϕ_j.

- Finally, the phase is obtained by measuring the first register.

2.2.2 Quantum Amplitude Amplification Algorithm

If a given quantum state $|\psi\rangle$ in N-dimensional Hilbert space can be rewritten in terms of some orthonormal states considered as the *good* and the *bad* parts of $|\psi\rangle$ as

$$|\psi\rangle = \sin(\theta)\left|\psi_{good}\right\rangle + \cos(\theta)\left|\psi_{bad}\right\rangle, \tag{8}$$

then amplitude amplification technique [8,32,37] can be used to increase the amplitude of $\left|\psi_{good}\right\rangle$ in magnitude while decreasing the amplitude of $|\psi_{bad}\rangle$. The technique mainly consists of two parts: the marking and the amplifying implemented by two operators, respectively U_f and U_ψ. Here, U_f marks-flips the sign of-the amplitudes of $\left|\psi_{good}\right\rangle$ and does nothing to $|\psi_{bad}\rangle$. U_f can be implemented as a reflection operator when $\left|\psi_{good}\right\rangle$ and $|\psi_{bad}\rangle$ are known

$$U_f = I - 2\left|\psi_{good}\right\rangle\left\langle\psi_{good}\right|, \tag{9}$$

where I is an identity matrix. In the amplification part, the marked amplitudes are amplified by the application of the operator U_ψ

$$U_\psi = I - 2\,|\psi\rangle\,\langle\psi| \tag{10}$$

To maximize the probability of $\left|\psi_{good}\right\rangle$, the iteration operator $G = U_\psi U_f$ is applied iteratively $O(\sqrt{N})$ times to the resulting state.

2.3 Quantum principal component analysis

In [35], we have shown that combining PEA with the amplitude amplification, one can obtain eigenvalues in certain intervals.

In the phase estimation part, the initial state of the registers is set to $|\mathbf{0}\rangle\,|\mathbf{0}\rangle$. Then, the second register is put into the equal superposition state $1/\sqrt{N}(1,\dots,1)^T$. The phase estimation process in this input generates the superposition of the eigenvalues on the first and the eigenvectors on the second register. In this final superposition state, the amplitudes for the eigenpairs are proportional to the norm of the projection of the input vector onto the eigenvector: i.e., the normalized sum of the eigenvector elements. This part is represented by U_{PEA} and also involves the input preparation circuit, U_{input}, on the second register.

In the amplification part, first, U_f is applied to the first register to mark the eigenvalues determined by the binary values of the eigenvalues: For instance, if we want to mark an eigenvalue equal to 0.25 in $|reg_1\rangle$ with 3 qubits, we use $U_f = I - 2\,|010\rangle\,\langle 010|$ since the binary form of 0.25 is (010) (the left most bit represents the most significant bit). The amplitudes of the marked eigenvalues are then amplified by the application of U_ψ with $|\psi\rangle$ representing the output of the phase estimation

$$|\psi\rangle = U_{PEA}\,|reg_1\rangle\,|reg_2\rangle = U_{PEA}\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle\,. \qquad (11)$$

Using the above equation, U_ψ can be implemented as

$$U_\psi = I - 2\,|\psi\rangle\,\langle\psi| = U_{PEA}U_0U_{PEA}^\dagger, \qquad (12)$$

where $U_0 = I - 2\,|\mathbf{0}\rangle\,\langle\mathbf{0}|$. The amplitudes of the eigenvalues in the desired region are further amplified by the iterative application of the operator $G = U_\psi U_f$. At the end of this process, a linear combination of the eigenvectors with the coefficients determined by the normalized sum of the vector elements of the eigenvectors are produced. In the following section, we shall show how to apply this process to model the implementation of the neural networks based on the Widrow–Hoff learning rule.

3 Application to the Neural Networks

Since the weight matrix in Widrow–Hoff learning rule converges to the principal components in infinity [6]: i.e., $W_{[\infty]} = QQ^T$, the behavior of the trained network on some input $|\mathbf{x}\rangle$ can be concluded as

$$W_{[\infty]}\,|\mathbf{x}\rangle = QQ^T\,|\mathbf{x}\rangle\,. \qquad (13)$$

Our main purpose is to find an efficient way to implement this behavior on quantum computers by using the quantum principal component analysis. For this purpose, we form U_f in a way that marks only the non-zero eigenvalues and their corresponding eigenvectors: For zero eigenvalues (in binary form $(0\dots0)$), the first register

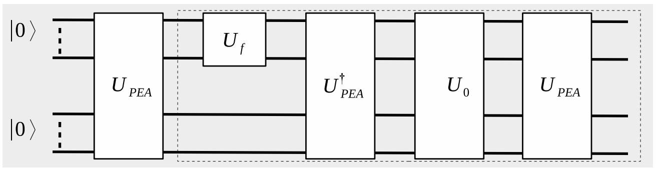


Figure 2: *The general quantum algorithm to find the principal components of W. The dashed box indicates an iteration of the amplitude amplification.*

is in $|\mathbf{0}\rangle = (1,0,0\dots,0,0)^T$ state. Therefore, we need to construct a U_f which *marks* the nonzero eigenvalues and does nothing to $|\mathbf{0}\rangle$. This can be done by using a vector $|\mathbf{f}\rangle$ in the standard basis which has the same non-zero coefficients for the all basis states except the first one

$$U_f = I - 2\,|\mathbf{f}\rangle\,\langle\mathbf{f}|\,, \text{ with } |\mathbf{f}\rangle = \frac{1}{\mu}\begin{pmatrix}0\\1\\1\\\vdots\\1\end{pmatrix}. \qquad (14)$$

Here, μ is a normalization constant equal to $\frac{1}{\sqrt{M-1}}$. U_f does nothing when the first register in $|\mathbf{0}\rangle$ state; however, it does not only flip the signs but also changes the amplitudes of the other states. Then, U_ψ is applied for the amplification of the marked amplitudes. The iterative application of $U_\psi U_f$ results in a quantum state where the amplitude of $|\mathbf{0}\rangle$ becomes almost zero and the amplitudes of the other states become almost equal. At this point, the second register holds $QQ^T\,|\mathbf{x}\rangle$ which is the expected output from the neural network. This is explained in more mathematical terms below.

3.1 Details of the Algorithm

Here, we assume that $U = e^{\iota Wt}$ is given: Later, in Section 3.4, we shall also discuss how U may be obtained as a quantum circuit from a given W matrix.

Figure 2 shows the algorithm as a quantum circuit where the dashed lines indicate an iteration in the amplitude amplification. At the beginning, U_{PEA} is applied to the initial state $|\mathbf{0}\rangle|\mathbf{0}\rangle$. Note that U_{PEA} includes also an input preparation circuit, U_{input}, bringing the second register from $|\mathbf{0}\rangle$ state to the input $|\mathbf{x}\rangle$. U_{PEA} generates a superposition of the eigenvalues and associated eigenvectors, respectively, on the first and the second registers with the amplitudes defined by the overlap of the eigenvector and the input $|\mathbf{x}\rangle$

$$|\psi\rangle = U_{PEA}\,|\mathbf{0}\rangle\,|\mathbf{0}\rangle = \sum_{j=0}^{N-1}\alpha_j\left|\lambda_j\right\rangle\left|\varphi_j\right\rangle, \qquad (15)$$

where $\alpha_j = \left\langle\varphi_j\right|\,|\mathbf{x}\rangle$.

In the second part, the operator $G = U_\psi U_f$ is applied to $|\psi\rangle$ iteratively until $QQ^T |\mathbf{x}\rangle$ can be obtained on the second register. The action of U_f applied to $|\psi\rangle$ is as follows

$$\begin{aligned} |\psi_1\rangle = U_f |\psi\rangle = (I - 2|\mathbf{f}\rangle\langle\mathbf{f}|) \sum_{j=0}^{N-1} \alpha_j \left|\lambda_j\right\rangle \left|\varphi_j\right\rangle \\ = |\psi\rangle - 2\mu |\mathbf{f}\rangle |\bar{\varphi}\rangle . \end{aligned} \tag{16}$$

Here, assuming the first k number of eigenvalues are zero, the *unnormalized* state $|\bar{\varphi}\rangle$ is defined as

$$|\bar{\varphi}\rangle = \sum_{j=k}^{N-1} \alpha_j \left|\varphi_j\right\rangle . \tag{17}$$

It is easy to see that $|\bar{\varphi}\rangle = QQ^T |\mathbf{x}\rangle$, which is our target output. When U_ψ is applied to the output in Eq. (16), we simply change the amplitudes of $|\psi\rangle$

$$\begin{aligned} U_\psi |\psi_1\rangle &= U_\psi (|\psi\rangle - 2\mu |\mathbf{f}\rangle |\bar{\varphi}\rangle) \\ &= (I - 2|\psi\rangle\langle\psi|)|\psi\rangle - 2\mu (I - 2|\psi\rangle\langle\psi|)|\mathbf{f}\rangle|\bar{\varphi}\rangle \\ &= -|\psi\rangle - 2\mu \left(I - 2|\psi\rangle \sum_{j=0}^{N-1} \alpha_j \left\langle\varphi_j\right| \left\langle\lambda_j\right| \right) |\mathbf{f}\rangle |\bar{\varphi}\rangle \\ &= -|\psi\rangle - 2\mu |\mathbf{f}\rangle |\bar{\varphi}\rangle + 4\mu^2 P_f |\psi\rangle \\ &= (4\mu^2 P_f - 1)|\psi\rangle - 2\mu |\mathbf{f}\rangle |\bar{\varphi}\rangle . \end{aligned} \tag{18}$$

Here, P_f is the initial success probability and equal to $\sum_{j=k}^{N-1} \alpha_j^2$. The repetitive applications of G only changes the amplitudes of $|\psi\rangle$ and $|\mathbf{f}\rangle |\bar{\varphi}\rangle$: e.g.,

$$\begin{aligned} G^2 |\psi\rangle = (c^2 - 3c + 1)|\psi\rangle - (c-2)2\mu |\mathbf{f}\rangle |\bar{\varphi}\rangle \\ G^3 |\psi\rangle = (c^3 - 5c^2 + 6c - 1)|\psi\rangle - (c^2 - 4c + 3)2\mu |\mathbf{f}\rangle |\bar{\varphi}\rangle \end{aligned} \tag{19}$$

where $c = (4\mu^2 P_f - 1)$. The normalized probability of $(2\mu |\mathbf{f}\rangle |\varphi\rangle)$ is presented in Figure 3 by using different values for c (The amplitudes of $|\psi\rangle$ and $(2\mu |\mathbf{f}\rangle)$ are normalized.). The amplitude of $|\psi\rangle$ through the iterations of the amplitude amplification oscillates with a frequency depending on the overlaps of the input with the eigenvectors. When the amplitude of $|\psi\rangle$ becomes close to zero, the second register in the remaining part $|\mathbf{f}\rangle |\bar{\varphi}\rangle$ is exactly $QQ^T |\mathbf{x}\rangle$ and the first register is equal to $|\mathbf{f}\rangle$.

Figure 4 represents the iterations of the algorithm for a random $2^7 \times 2^7$ matrix with $2^7/2$ number of zero eigenvalues and a random input $|\mathbf{x}\rangle$. In each subfigure, we have used different numbers of qubits for the first register to see the effect on the results. The bar graphs in the subfigures shows the probability change for each state $|\mathbf{j}\rangle$, $j = 0 \dots 1$, of the first qubit (A different color tone indicates a different state.). When the probability for $|\mathbf{0}\rangle$ becomes close

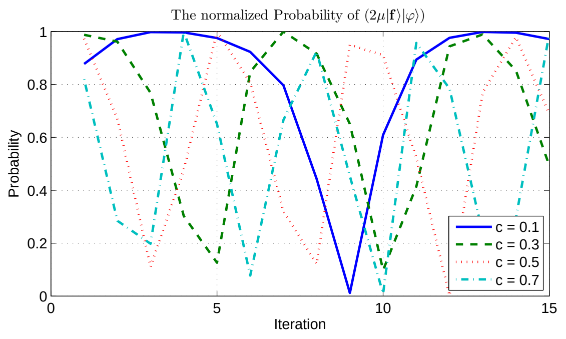


Figure 3: *The normalized probability of* $(2\mu |\mathbf{f}\rangle |\varphi\rangle)$ *through the iterations for different values of c.*

to zero, the probabilities for the rest of the states become equal and so the total sum of these probabilities as shown in the bottom figure of each subfigure becomes almost one. At that point, the fidelity found by $\left|\langle reg_2| QQ^T |\mathbf{x}\rangle\right|$ also comes closer to one.

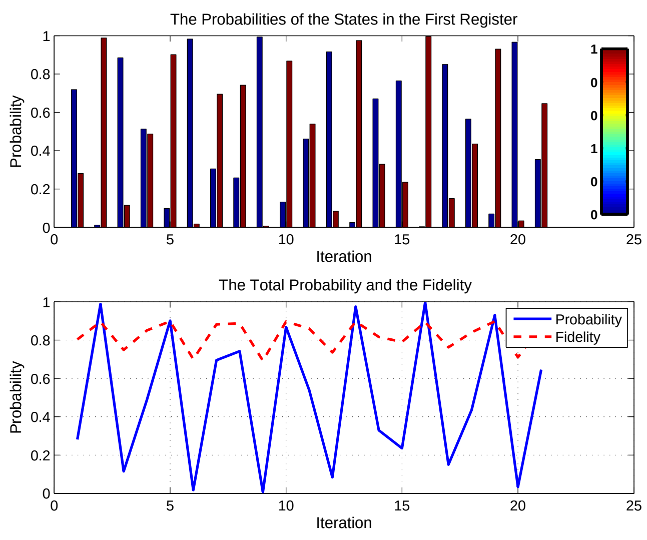
3.2 Number of Iterations

Through the iterations, while the probability for $|\mathbf{0}\rangle$ state goes to zero, the probabilities for the rest of the states become almost equal. This indicates that the individual states of each qubit turn into the equal superposition state. Therefore, if the state of a qubit in the first register is in the almost equal superposition state, then the success probability is very likely to be in its maximum level. In the Hadamard basis, $|0\rangle$ and $|1\rangle$ are represented in the equal superposition states as follows

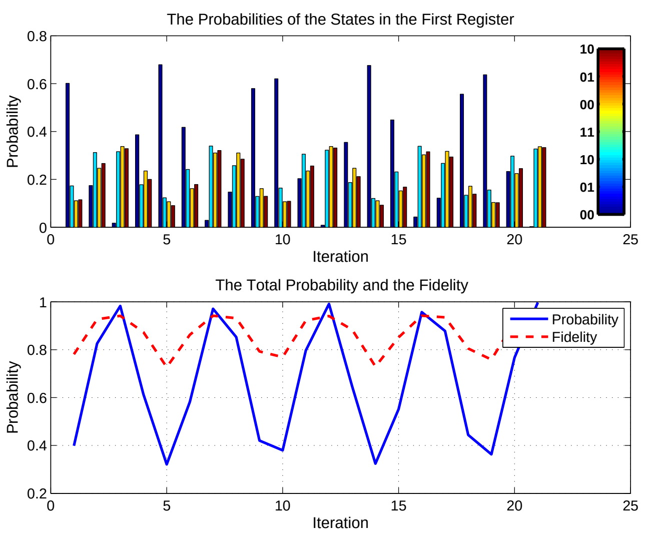
$$|0\rangle = \frac{|\mathbf{0}\rangle + |\mathbf{1}\rangle}{\sqrt{2}} \text{ and } |1\rangle = \frac{|\mathbf{0}\rangle - |\mathbf{1}\rangle}{\sqrt{2}}. \tag{20}$$

Therefore, using the Hadamard basis, if the probability of measuring $|0\rangle$ is close to one, in other words, if $|1\rangle$ is not seen in the measurement, then the second register likely holds $QQ^T |\mathbf{x}\rangle$ with a maximum possible fidelity. Figure 5 shows the comparisons of the individual qubit probabilities (i.e., the probability to see a qubit in the first register in $|0\rangle$ in the Hadamard basis.) with the total probability observed in Figure 4f for the random case: As seen in the figure, the individual probabilities exhibit the same behavior as the total probability.

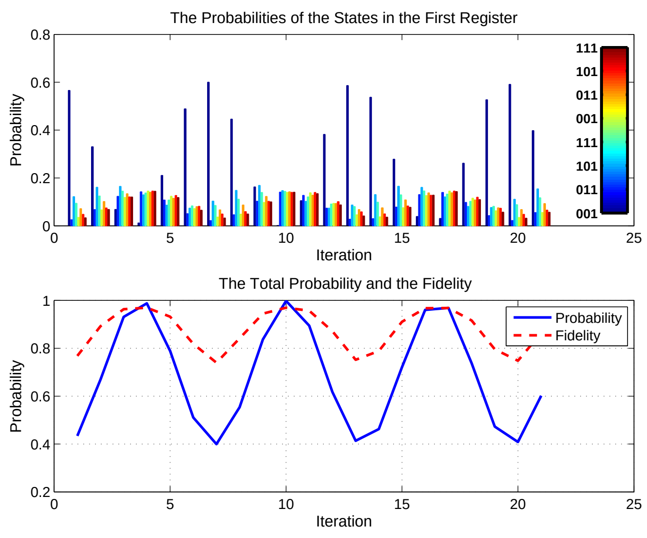
Generally, obtaining a possible probability density of an unknown quantum state is a difficult task. However, since we are dealing with only a single qubit and do not require the exact density, this can be done efficiently. For instance, if $|0\rangle$ is seen n number of times in ten measurements, then the success probability is expected to be $n/10$. Here, the number of measurements obviously determines the precision in the obtained probability which may also affect the fidelity.



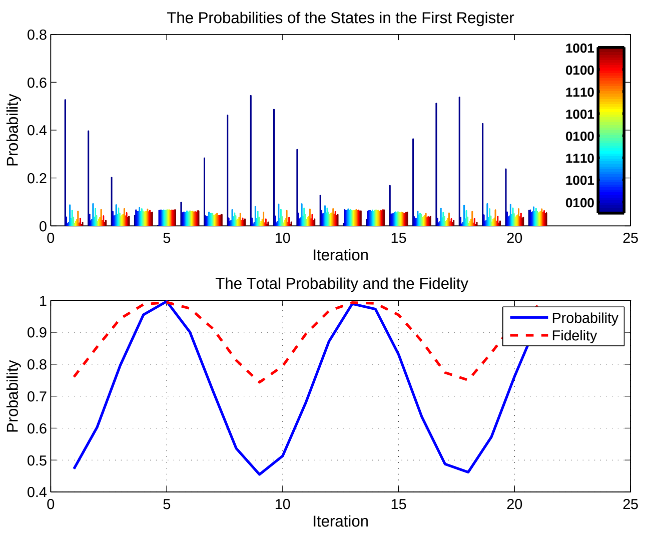
***(a)** m is 1.*



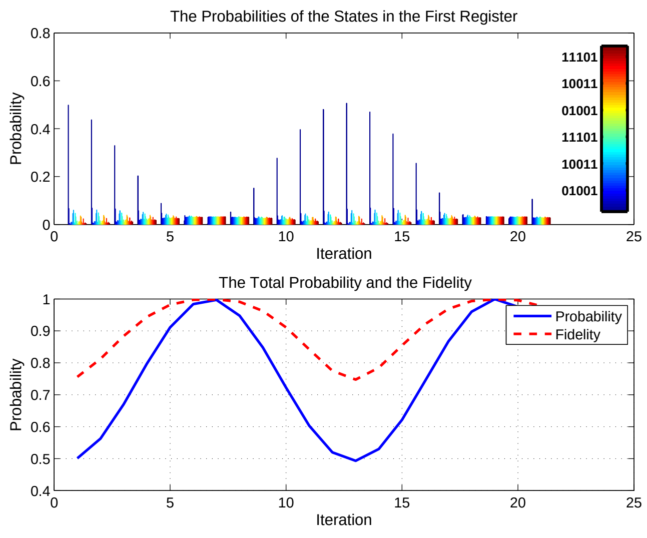
***(b)** m is 2.*



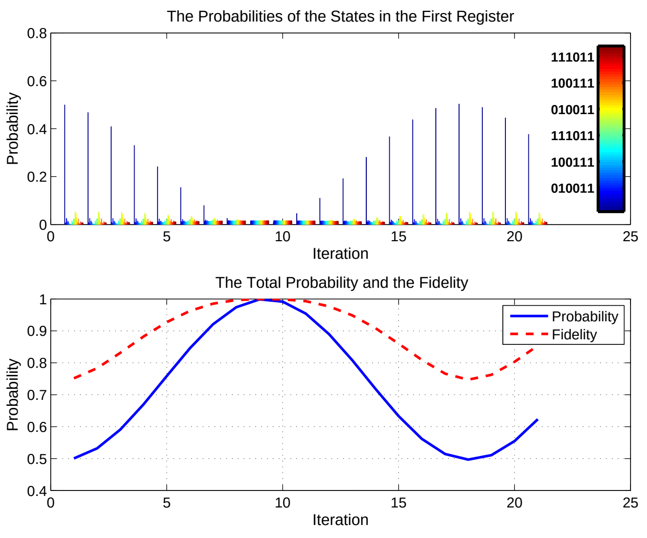
***(c)** m is 3.*



***(d)** m is 4.*



***(e)** m is 5.*



***(f)** m is 6.*

***Figure 4:** The probability changes in the iteration of the amplitude amplification for a random $2^7 \times 2^7$ matrix with $2^7/2$ number of zero eigenvalues and a random input $|\mathbf{x}\rangle$ (MATLAB code for the random generation is given in Appendix). In each subfigure, we have used different numbers of qubits, m, for the first register to see the effect on the results. The bar graphs in the subfigures shows the probability change for each state $|\mathbf{j}\rangle$, $j = 0 \dots 1$, of the first qubit. For each state, a different color tone is used.*

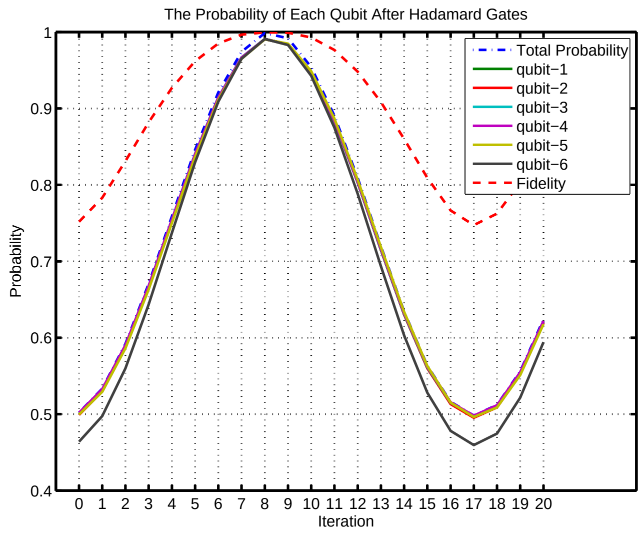


Figure 5: *The probability to see a qubit in the first register in $|0\rangle$ state after applying a Hadamard gate to the qubit and its comparison with the total probability and the fidelity given in Figure 4f. Note that the above separate curve is the fidelity. Since there are only small differences between the probabilities on the individual qubits and the total probability, the curves for the probabilities mostly overlap.*

3.3 Error-Precision (Number of Qubits in $|reg_1\rangle$)

The number of qubits, m, in the first register should be sufficient to distinguish very small nonzero eigenvalues from the ones which are zero. In our numerical random experiments, we have observed that choosing only six or five qubits are enough to get very high fidelity while not requiring a high number of iterations. The impact of the number of qubits on the fidelity and the probability is shown in Figure 4 in which each sub-figure is drawn by using different register sizes for the same random case. As seen in the figure, the number of qubits also affects the required number of iterations: e.g., while for $m = 3$, the highest fidelity and probability are seen at the fourth iteration; for $m = 6$, it happens around the ninth iteration.

3.4 Circuit Implementation of W

The circuit implementation of W requires forming a quantum circuit representing the time evolution of W: i.e., $U = e^{\iota 2\pi W t}$. When W is a sparse matrix, the circuit can be formed by following the method in [38]. However, when it is not sparse but in the following form $W = \sum_j \mathbf{x_j}\mathbf{x_j}^T$, then the exponential becomes equal to

$$U = e^{\iota 2\pi W t} = e^{\iota 2\pi t \sum_j \mathbf{x_j}\mathbf{x_j}^T}. \tag{21}$$

To approximate the above exponential, we apply the Trotter–Suzuki formula [39–42] to decompose Eq. (21) into the terms $U_j = e^{\iota 2\pi t \mathbf{x_j}\mathbf{x_j}^T} = U_{\mathbf{x_j}} \bar{I} U_{\mathbf{x_j}}^\dagger$, where $\bar{I}$ is a kind of identity matrix with the first element set to $e^{\iota 2\pi t}$, and $U_{\mathbf{x_j}}$ is a unitary matrix with the first row and column equal to $\mathbf{x_j}$. For instance, if the second order Trotter–Suzuki decomposition is applied to Eq. (21) (note that the order of the decomposition impacts the accuracy of the approximation), the following is obtained

$$e^{\iota 2\pi t \sum_{j=1}^{\kappa} \mathbf{x_j}\mathbf{x_j}^T} \approx U_j \left(e^{\iota 2\pi \frac{t}{2} \sum_{j=2}^{\kappa} \mathbf{x_j}\mathbf{x_j}^T} \right) U_j. \tag{22}$$

Then, the same decomposition is applied to the term $e^{\iota 2\pi t/2 \sum_{j=2}^{\kappa} \mathbf{x_j}\mathbf{x_j}^T}$ in the above equation. This recursive decomposition yields an approximation composed of (4κ) number of $U_{\mathbf{x_j}}$ matrices. Any $U_{\mathbf{x_j}}$ can be implemented as a Householder matrix by using $O(2^n)$ quantum operations which is linear in the size of $\mathbf{x_j}$ [43–46].

3.5 Obtaining a solution from the output

Generally, the amplitudes of the output vector (the final state of the second register) encodes the information needed for the solution of the considered problem. Since obtaining the full density of a quantum state is known to be very inefficient for larger systems, one needs to drive efficient measurement schemes specific to the problem. For instance, for some problems, comparisons of the peak values instead of the whole vectors may be enough to gauge a conclusion: In this case, since a possible outcome in a measurement would be the one with an amplitude likely to be greater than most of the states in magnitude, the peak values can be obtained efficiently. However, this alone may not be enough for some applications.

Moreover, in some applications such as the spectral clustering problem, a superposition of vectors that are forming a solution space for the problem can be used as an input state. In that case, the measurement of the output in the solution space yields the solution for the problem. This method can be used efficiently (polynomial time complexity in the number of qubits) when the vectors describing the solution space are tensor product of Pauli matrices.

4 Complexity Analysis

The computational complexity of a quantum algorithm is assessed by the total number of single gates and two qubit controlled NOT (CNOT) gates in the quantum circuit implementing the algorithm. We derive the computational complexity of the whole method by finding the complexities of U_f on the first register with m number of qubits and U_ψ on the second register with n number of qubits. We shall use $M = 2^m$ and $N = 2^n$ to describe the sizes of the operators on the registers.

4.1 The complexity of U_f

It is known that the number of quantum gates to implement a Householder matrix is bounded by the size of the matrix [43–46]. Therefore, the circuit for U_f requires $O(M)$ CNOT gates since it is a Householder transformation formed by the vector $|\mathbf{f}\rangle$ of size M.

4.2 The complexity of U_ψ

The operator U_ψ is equal to $U_{PEA}U_0U_{PEA}^\dagger$ in which the total complexity will be typically governed by the complexity of U_{PEA}. U_{PEA} involves the Fourier transforms, input preparation circuit, and the controlled $U = e^{itW}$ with different t values:

- The circuits for the quantum Fourier transform and its inverse are well known [36] and can be implemented on the first register in $O(m^2)$.
- The input preparation circuit on the second register, U_{input}, can be implemented again as a Householder transformation by using $O(N)$ number of quantum gates. It can be also designed by following Section III.B. of [47]: In that case, for every two vector elements, a controlled rotation gate is used to construct U_{input} with the initial row equal to $\mathbf{x}$; thus, $U_{input}\,|\mathbf{0}\rangle = |\mathbf{x}\rangle$.
- The circuit complexity of $U = e^{itW}$ is highly related to the structure of W. When W of order N is sparse enough: i.e., the number of nonzero entries is bounded by some polynomial of the number of qubits, $poly(n)$; then W can be simulated by using only $O(poly(n))$ number of quantum gates [38, 48, 49]. However, when W is not sparse but equal to $\sum \mathbf{x_i}\mathbf{x_i}^T$, then as shown in Section 3.4, we use Trotter–Suzuki formula which yields a product of (4κ) number of $U_{\mathbf{x_j}}$ matrices with $1 \le j \le \kappa$. Since $U_{\mathbf{x_j}}$ can be implemented as a Householder transformation by using $O(N)$ quantum gates, U requires $O(\kappa N)$ quantum gates.

If we combine all the above terms, the total complexity can be concluded as

$$O(\kappa N + M). \tag{23}$$

This is linear in system-size, however, exponential in the number of qubits involved in either one of the registers. In comparison, any classical method applied to obtain $QQ^T\mathbf{x}$ at least requires $O(N^2)$ time complexity because of the matrix vector multiplication. Therefore, the quantum model presented here may provide a quadratic speedup over the classical methods for some applications. When the weight matrix is sparse or the data is given as a quantum states, it can be implemented in $O(poly(n))$. Therefore, the whole complexity becomes linear in the number of qubits, which may provide an exponential speedup over the classical algorithms. However, when the weight matrix is not sparse, the complexity becomes exponential in the number of qubits. The current experimental research by big companies such as Google and IBM aims to build 50 qubit operational quantum computers [50]. Because of the limitations of the current quantum computer technology, when the required number of qubits goes beyond 50, the applications of the algorithm becomes infeasible.

5 An Illustrative Example

Here, we give a simple example to show how the algorithm works: Let us assume, we have given weights represented by the columns of the following matrix [51]

$$X = \frac{1}{10} \times \begin{pmatrix} -1 & +1 \\ -1 & -1 \\ +1 & -1 \\ -1 & +1 \end{pmatrix}, \tag{24}$$

where we scale the vectors by $\frac{1}{10}$ so as to make sure that the eigenvalues of W are less than one. To validate the simulation results, first, $W_{[\infty]}$ is classically computed by following the singular value decomposition of X

$$Q\Phi P^T = \begin{pmatrix} +.5774 & 0 \\ 0 & 1 \\ -.5774 & 0 \\ +.5774 & 0 \end{pmatrix} \begin{pmatrix} .2495 & 0 \\ 0 & .14142 \end{pmatrix} \begin{pmatrix} -.7071 & +.7071 \\ -.7071 & -.7071 \end{pmatrix}. \tag{25}$$

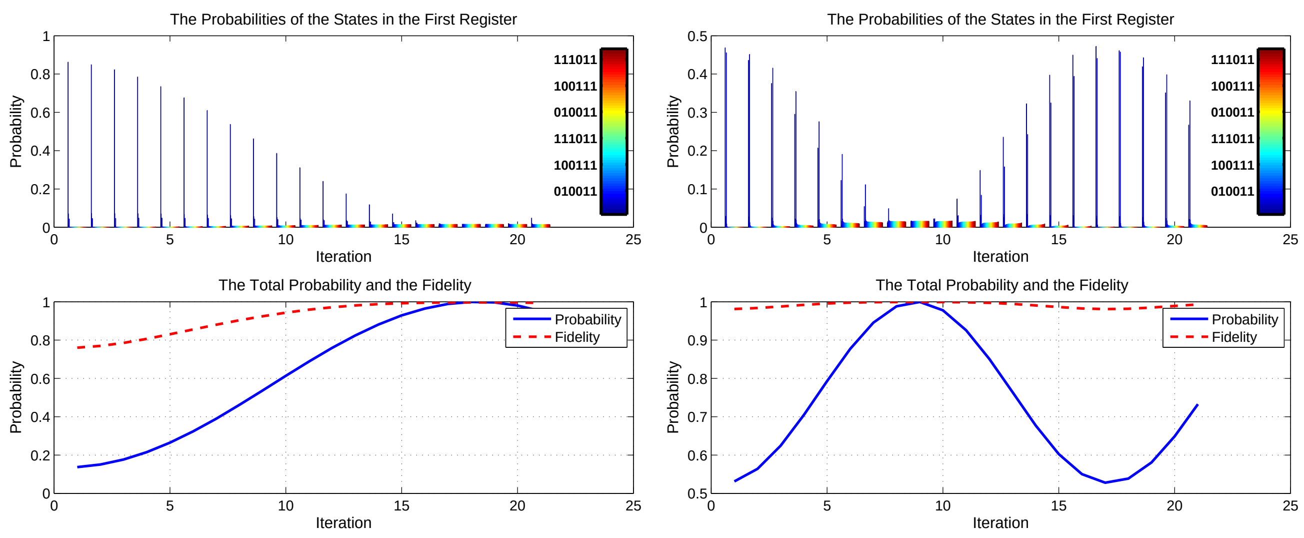
Therefore,

$$W_{[\infty]} = QQ^T = \begin{pmatrix} +.333 & 0 & -.333 & +.333 \\ 0 & 1 & 0 & 0 \\ -.333 & 0 & +.333 & -.333 \\ +.333 & 0 & -.333 & +.333 \end{pmatrix} \tag{26}$$

We use the following Trotter–Suzuki formula [39–42] to compute the exponential of $W = XX'$

$$U = e^{i2\pi W} \approx e^{i\pi \mathbf{x_2}\mathbf{x_2^T}} e^{i2\pi \mathbf{x_1}\mathbf{x_1^T}} e^{i\pi \mathbf{x_2}\mathbf{x_2^T}} \tag{27}$$

In the simulation for a random input $|\mathbf{x}\rangle$, the comparison of $W_{[\infty]}\,|\mathbf{x}\rangle = QQ^T\,|\mathbf{x}\rangle$ with the output of the second register in the quantum model yields the fidelity. For two different random inputs, the simulation results in each iteration are shown in Figure 6a and Figure 6b for $|\mathbf{x}\rangle = (.3517\ .3058\ .6136\ .6374)^T$ and $|\mathbf{x}\rangle = (.7730\ .1919\ .1404\ .5881)^T$, respectively.



(a) For the generated random input $|\mathbf{x}\rangle = (.3517\ .3058\ .6136\ .6374)^T$.*(b) For the generated random input* $|\mathbf{x}\rangle = (.7730\ .1919\ .1404\ .5881)^T$.

Figure 6: *The simulation results of the quantum model for the example in Section 5 with two different input vectors.*

6 Conclusion

The weight matrix of the networks based on the Widrow–Hoff learning rule converges to QQ^T, where Q represents the eigenvectors of the matrix corresponding to the nonzero eigenvalues. Here, we applied the quantum principal component analysis method described in [35] to artificial neural networks using the Widrow–Hoff learning rule and showed that one can implement an equivalent quantum circuit which produces the output $QQ^T\mathbf{x}$ for a given input $\mathbf{x}$ in linear time. We also discussed the implementation details by using random cases, analyzed the computation complexity based on the number of quantum gates and presented a simple numerical example. The model is general and requires only linear time computational complexity in the size of the weight matrix.

Appendix: MATLAB Code for the Random Matrix

The random matrix used in the numerical example is generated by the following MATLAB code snippet:

```
%number of non-zero eigenvalues
npc = ceil(N/2);
d = rand(N,1);%random eigenvalues
d(npc+1:end) = 0;
%random eigenvectors
[Qfull,~] = qr(randn(N));
%the unitary matrix in PEA
U = Qfull*diag(exp(1i*2*pi*d))*Qfull';
%normalized input vector
x = rand(N,1); x = x/norm(x);
```

References

[1] Haykin SO. *Neural Networks and Learning Machines*, 3rd edition. Upper Saddle River, New Jersey: Pearson, 2008.

[2] Abdi H. Linear algebra for neural networks. In: *International Encyclopedia of the Social and Behavioral Sciences*. Baltes PB (editor), Oxford: Pergamon, 2001, pp. 8864–8868. `doi:10.1016/B0-08-043076-7/00609-4`

[3] Abdi H, Valentin D, Edelman B, O'Toole AJ. More about the difference between men and women: evidence from linear neural networks and the principal-component approach. *Perception* 1995; **24**(5): 539–562. `doi:10.1068/p240539`

[4] Morris RGM. D. O. Hebb: The Organization of Behavior, Wiley: New York; 1949. *Brain Research Bulletin* 1999; **50**(5): 437. `doi:10.1016/S0361-9230(99)00182-3`

[5] Widrow B, Hoff ME. Adaptive switching circuits. In: *Neurocomputing: Foundations of Research*. Anderson JA, Rosenfeld E (editors), Cambridge, Massachusetts: MIT Press, 1988, pp. 123–134.

[6] Abdi H, Valentin D, Edelman B, O'Toole AJ. A Widrow–Hoff learning rule for a generalization of the linear auto-associator. *Journal of Mathematical Psychology* 1996; **40**(2): 175–182. `doi:10.1006/jmps.1996.0017`

[7] Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. Proceedings of

the *35th Annual Symposium on Foundations of Computer Science*, November, 20–22, pp. 124–134. doi:10.1109/sfcs.1994.365700

[8] Grover LK. A fast quantum mechanical algorithm for database search. Proceedings of the *28th annual ACM symposium on Theory of Computing*, Philadelphia, Pennsylvania, May 22–24, 1996, Association for Computing Machinery, pp. 212–219. doi:10.1145/237814.237866

[9] Neven H, Denchev VS, Rose G, Macready WG. *Training a binary classifier with the quantum adiabatic algorithm.* 2008. arXiv:0811.0416

[10] Neven H, Denchev VS, Rose G, Macready WG. *Training a large scale classifier with the quantum adiabatic algorithm.* 2009. arXiv:0912.0779

[11] Aïmeur E, Brassard G, Gambs S. Quantum speed-up for unsupervised learning. *Machine Learning* 2013; **90**(2): 261–287. doi:10.1007/s10994-012-5316-5

[12] Lloyd S, Garnerone S, Zanardi P. Quantum algorithms for topological and geometric analysis of data. *Nature Communications* 2016; **7**(10138. arXiv:1408.3106, doi:10.1038/ncomms10138

[13] Rebentrost P, Mohseni M, Lloyd S. Quantum support vector machine for big data classification. *Physical Review Letters* 2014; **113**(13): 130503. arXiv:1307.0471, doi:10.1103/PhysRevLett.113.130503

[14] Wittek P. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Amsterdam: Academic Press, 2014.

[15] Schuld M, Sinayskiy I, Petruccione F. Quantum computing for pattern classification. Proceedings of the *Pacific Rim International Conference on Artificial Intelligence 2014: Trends in Artificial Intelligence*, Cham, Pham D-N, Park S-B (editors), Springer, pp. 208–220. arXiv:1412.3646, doi:10.1007/978-3-319-13560-1_17

[16] Lloyd S, Mohseni M, Rebentrost P. Quantum principal component analysis. *Nature Physics* 2014; **10**(9): 631–633. arXiv:1307.0401, doi:10.1038/nphys3029

[17] Manju A, Nigam MJ. Applications of quantum inspired computational intelligence: a survey. *Artificial Intelligence Review* 2014; **42**(1): 79–156. doi:10.1007/s10462-012-9330-6

[18] da Silva AJ, Ludermir TB, de Oliveira WR. Quantum perceptron over a field and neural network architecture selection in a quantum computer. *Neural Networks* 2016; **76**: 55–64. arXiv:1602.00709, doi:10.1016/j.neunet.2016.01.002

[19] Zhou R, Wang H, Wu Q, Shi Y. Quantum associative neural network with nonlinear search algorithm. *International Journal of Theoretical Physics* 2012; **51**(3): 705–723. doi:10.1007/s10773-011-0950-4

[20] Gupta S, Zia RKP. Quantum neural networks. *Journal of Computer and System Sciences* 2001; **63**(3): 355–383. doi:10.1006/jcss.2001.1769

[21] Andrecut M, Ali MK. A quantum neural network model. *International Journal of Modern Physics C* 2002; **13**(1): 75–88. doi:10.1142/S0129183102002948

[22] Altaisky MV. *Quantum neural network.* 2001. arXiv:quant-ph/0107012

[23] Schuld M, Sinayskiy I, Petruccione F. Quantum walks on graphs representing the firing patterns of a quantum neural network. *Physical Review A* 2014; **89**(3): 032333. arXiv:1404.0159, doi:10.1103/PhysRevA.89.032333

[24] Schuld M, Sinayskiy I, Petruccione F. The quest for a quantum neural network. *Quantum Information Processing* 2014; **13**(11): 2567–2586. arXiv:1408.7005, doi:10.1007/s11128-014-0809-8

[25] Kouda N, Matsui N, Nishimura H, Peper F. Qubit neural network and its learning efficiency. *Neural Computing and Applications* 2005; **14**(2): 114–121. doi:10.1007/s00521-004-0446-8

[26] Li P, Xiao H, Shang F, Tong X, Li X, Cao M. A hybrid quantum-inspired neural networks with sequence inputs. *Neurocomputing* 2013; **117**: 81–90. doi:10.1016/j.neucom.2013.01.029

[27] Ventura D, Martinez T. A quantum associative memory based on Grover's algorithm. Proceedings of the *Artificial Neural Nets and Genetic Algorithms*, Vienna, Dobnikar A, Steele NC, Pearson DW, Albrecht RF (editors), Springer, pp. 22–27. doi:10.1007/978-3-7091-6384-9_5

[28] Kitaev AY. Quantum measurements and the Abelian Stabilizer Problem. *Electronic Colloquium on Computational Complexity* 1996; **3**: TR96-003. arXiv: quant-ph/9511026, https://eccc.weizmann.ac.il/report/1996/003

[29] Harrow AW, Hassidim A, Lloyd S. Quantum algorithm for linear systems of equations. *Physical Review Letters* 2009; **103**(15): 150502. arXiv: 0811.3171, doi:10.1103/PhysRevLett.103.150502

[30] Grover LK. Quantum computers can search rapidly by using almost any transformation. *Physical Review Letters* 1998; **80**(19): 4329–4332. arXiv:quant-ph/9712011, doi:10.1103/PhysRevLett.80.4329

[31] Mosca M. Quantum searching, counting and amplitude amplification by eigenvector analysis. In: *MFCS'98 Workshop on Randomized Algorithms*. 1998, pp. 90–100.

[32] Brassard G, Høyer P, Mosca M, Tapp A. Quantum amplitude amplification and estimation. In: *Quantum Computation and Information*. Lomonaco Jr. SJ, Brandt HE (editors), Contemporary Mathematics, vol. 305, Providence, Rhode Island: American Mathematical Society, 2002, pp. 53–74. arXiv: quant-ph/0005055, doi:10.1090/conm/305

[33] Chen CL, Dong DY, Chen ZH. Quantum computation for action selection using reinforcement learning. *International Journal of Quantum Information* 2006; **4**(6): 1071–1083. doi:10.1142/S0219749906002419

[34] Dong D, Chen C, Li H, Tarn TJ. Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 2008; **38**(5): 1207–1220. doi:10.1109/tsmcb.2008.925743

[35] Daskin A. Obtaining a linear combination of the principal components of a matrix on quantum computers. *Quantum Information Processing* 2016; **15**(10): 4013–4027. arXiv:1512.02109, doi:10.1007/s11128-016-1388-7

[36] Nielsen MA, Chuang IL. *Quantum Computation and Quantum Information*, 10th Anniversary edition. Cambridge: Cambridge University Press, 2010.

[37] Brassard G, Hoyer P. An exact quantum polynomial-time algorithm for Simon's problem. Proceedings of the *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, June 17–19, 1997, pp. 12–23. arXiv:quant-ph/9704027, doi:10.1109/istcs.1997.595153

[38] Berry DW, Ahokas G, Cleve R, Sanders BC. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics* 2007; **270**(2): 359–371. arXiv:quant-ph/0508139, doi:10.1007/s00220-006-0150-x

[39] Trotter HF. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society* 1959; **10**(4): 545–551. JSTOR:2033649, doi:10.1090/S0002-9939-1959-0108732-6

[40] Suzuki M. Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Communications in Mathematical Physics* 1976; **51**(2): 183–190. doi:10.1007/bf01609348 https://projecteuclid.org/euclid.cmp/1103900351

[41] Hatano N, Suzuki M. Finding exponential product formulas of higher orders. In: *Quantum Annealing and Other Optimization Methods*. Das A, K. Chakrabarti B (editors), Lecture Notes in Physics, vol. 679, Berlin: Springer, 2005, pp. 37–68. arXiv: math-ph/0506007, doi:10.1007/11526216_2

[42] Poulin D, Hastings MB, Wecker D, Wiebe N, Doberty AC, Troyer M. The Trotter step size required for accurate quantum simulation of quantum Chemistry. *Quantum Information and Computation* 2015; **15**(5–6): 361–384. arXiv:1406.4920

[43] Ivanov PA, Kyoseva ES, Vitanov NV. Engineering of arbitrary $U(N)$ transformations by quantum Householder reflections. *Physical Review A* 2006; **74**(2): 022323. arXiv:0708.2811, doi:10.1103/PhysRevA.74.022323

[44] Urías J, Quiñones DA. Householder methods for quantum circuit design. *Canadian Journal of Physics* 2015; **94**(2): 150–157. doi:10.1139/cjp-2015-0490

[45] Ivanov PA, Vitanov NV. Synthesis of arbitrary unitary transformations of collective states of trapped ions by quantum Householder reflections. *Physical Review A* 2008; **77**(1): 012335. doi:10.1103/PhysRevA.77.012335

[46] Bullock SS, O'Leary DP, Brennen GK. Asymptotically optimal quantum circuits for *d*-level systems. *Physical Review Letters* 2005; **94**(23): 230502. arXiv:quant-ph/0410116, doi:10.1103/PhysRevLett.94.230502

[47] Daskin A, Grama A, Kollias G, Kais S. Universal programmable quantum circuit schemes to emulate an operator. *Journal of Chemical Physics* 2012; **137**(23): 234112. arXiv:1204.3600, doi:10.1063/1.4772185

[48] Aharonov D, Ta-Shma A. Adiabatic quantum state generation and statistical zero knowledge. Proceedings of the *35th annual ACM symposium on Theory of Computing*, San Diego, California, June 9–11, 2003, Association for Computing Machinery, pp. 20–29. arXiv:quant-ph/0301023, doi:10.1145/780542.780546

[49] Childs AM, Kothari R. Simulating sparse Hamiltonians with star decompositions. In: *Theory of Quantum Computation, Communication, and Cryptography*. van Dam W, Kendon VM, Severini S (editors), Lecture Notes in Computer Science, vol. 6519, Berlin: Springer, 2011, pp. 94–103. arXiv:1003.3683, doi:10.1007/978-3-642-18073-6_8

[50] Castelvecchi D. Quantum computers ready to leap out of the lab in 2017. *Nature* 2017; **541**(7635): 9–10. doi:10.1038/541009a

[51] Abdi H, Valentin D, Edelman B. *Neural Networks*. Quantitative Applications in the Social Sciences, vol. 124, Thousand Oaks, California: SAGE Publications, 1998.