Quantum Computing Methods for Supervised Learning

Viraj Kulkarni[1], Milind Kulkarni[1], Aniruddha Pant[2]

[1] Vishwakarma University
[2] DeepTek Inc

June 23, 2020

Abstract

The last two decades have seen an explosive growth in the theory and practice of both quantum computing and machine learning. Modern machine learning systems process huge volumes of data and demand massive computational power. As silicon semiconductor miniaturization approaches its physics limits, quantum computing is increasingly being considered to cater to these computational needs in the future. Small-scale quantum computers and quantum annealers have been built and are already being sold commercially. Quantum computers can benefit machine learning research and application across all science and engineering domains. However, owing to its roots in quantum mechanics, research in this field has so far been confined within the purview of the physics community, and most work is not easily accessible to researchers from other disciplines. In this paper, we provide a background and summarize key results of quantum computing before exploring its application to supervised machine learning problems. By eschewing results from physics that have little bearing on quantum computation, we hope to make this introduction accessible to data scientists, machine learning practitioners, and researchers from across disciplines.

1 Introduction

Supervised learning is the most commonly applied form of machine learning. It works in two stages. During the training stage, the algorithm extracts patterns from the training dataset that contains pairs of samples and labels and converts these patterns into a mathematical representation called a model. During the inference stage, this model is used to make predictions about unseen samples. Machine learning algorithms, in general, are data hungry; their performance depends heavily on the size of the datasets used for training. However, this training process is computationally expensive, and working with large datasets requires huge amounts of computational horsepower. As the number and volume of datasets available for research and commercial purposes continues to grow exponentially, new technologies such as NVIDIA CUDA [1] and Google TPUs [2] have emerged to enable faster processing of data. Computational speeds have been increasing rapidly over the last several decades in

accordance with the observation that the number of transistors in a dense integrated circuit (IC) doubles about every two years - this is informally known as Moores Law [3]. The semiconductor manufacturing process has shrunk from 10 micrometers in 1970 to about 5 nanometers in 2020. It is believed, however, that we are nearing the limits of the processing power that classical computers can offer us [4]. At scales smaller than this, quantum mechanical effects come into play [5], and they impose physical limitations on how small electronic components can get.

Quantum computing, on the other hand, proposes to leverage these quantum mechanical effects to carry out computation. In contrast to classical computers that operate on bits that can exist in only one out of two states at a time, quantum computers exploit the fact that quantum bits (qubits) can exist in any one of the infinite possible linear superpositions of these two states. This allows quantum computers to execute multiple paths of computation simultaneously. Quantum computers can efficiently solve computational problems which are believed to be intractable for classical computers [6]. Owing to its roots in theoretical physics, most research articles on the topic are written for physicists; this makes them difficult to access for researchers from other fields. The purpose of this paper is to give a broad overview of the synergies between quantum computing and machine learning. We briefly outline the history of quantum physics, describe the preliminaries of quantum computation, and then review the latest research on applying the principles of quantum computation to supervised machine learning. A large-scale general-purpose quantum computer does not yet exist, but restricted quantum machines capable of solving optimization problems are already being sold commercially [7][8]. By eschewing results from physics that have little bearing on quantum computation and by providing additional background that may benefit the unfamiliar reader, we hope to make this introduction accessible to data scientists, machine learning practitioners, and researchers from other fields.

Quantum mechanics arose through a number of discoveries in the early twentieth century. Einstein [9] explained the photoelectric effect in 1905 by postulating that light and all electromagnetic radiation is made up of discrete particles that later came to be called photons. Previously in 1865, Maxwell [10] had demonstrated that electric and magnetic fields travel through space as waves. De Broglie [11] proposed in 1923 that particles can exhibit wave-like properties and waves can behave like particles. Building on his approach, Heisenberg [12] developed matrix mechanics, and Schrdinger [13] developed wave mechanics, both of which were later found to be equivalent. These developments laid the foundation of quantum mechanics. The equations of quantum mechanics have since been extensively tested in innumerable experiments, but even after almost a century of debate, physicists strongly disagree over how those equations should be interpreted and mapped to reality [14].

Benioff [15] in 1980 and Feynman [16] in 1982 observed that simulating the evolution of certain quantum systems may be an intractable problem that cannot be solved efficiently by computers, and yet, these quantum systems solved the problem by merely evolving thus suggesting that the evolution of quantum systems could be used as a method of computation. In 1985, Deutsch [17] designed a universal quantum computer as a quantum-counterpart to the Universal Turing Machine. Deutsch and Jozsa [18] proposed the Deutsch-Jozsa problem for which the deterministic quantum algorithm is exponentially faster than any deterministic classical solution. Shor's algorithm [19] for factoring large integers was the first quantum algorithm that could solve a problem of practical significance faster than any

classical algorithm. Grover's algorithm [20] showed quadratic improvement in unordered search problems. These results laid the foundation of quantum computation. Since then, quantum algorithms have been proposed for numerous areas including cryptography, search and optimisation, simulation of quantum systems, solving large systems of linear equations, and machine learning [21].

The benefits quantum computing can bring to machine learning go beyond speed-up in execution. Many tasks in machine learning such as maximum likelihood estimation using hidden variables, principal component analysis, training of neural networks etc. require optimization of a non-convex objective function. Optimizing non-convex functions is an NP-hard problem. Classical optimization methods such as gradient descent can get stuck at local minimum or saddle points and may never find the global minimum. Adiabatic quantum computers use quantum annealing to solve non-convex optimization problems by finding low-energy configurations of an appropriate energy function by exploiting tunneling effects to escape local minima [22]. Methods based on Grover's search can find the global minimum in a discrete unordered search space. Many machine learning algorithms involve repeated execution of linear algebra routines on large matrices. Quantum solutions can offer exponential speed-up for these routines [23][24]. Besides optimizing subroutines used in classical machine learning algorithms, many fully quantum versions of these algorithms have also been developed.

The term *quantum machine learning* is generally used to denote analysis of classical data on quantum computers. This is known as *quantum-enhanced* machine learning. There are however other ways in which the fields of quantum computing and machine learning overlap. Classical machine learning can be applied to data emanating from quantum systems to solve problems in physics. Another stream of research deals with generalizing classical machine learning to work with quantum data where the input and output are quantum states. Recently, Tang [25] developed a classical algorithm for recommendation systems that was inspired by quantum computing creating a new category referred to as *quantum-inspired* algorithms. These are classical algorithms that can be run on conventional computers which borrow ideas from quantum computing to achieve significant theoretical speed-ups over the best prevailing classical algorithms. This paper limits itself to quantum-enhanced machine learning and presents a selection of quantum approaches for implementing supervised machine learning algorithms on quantum computers. Far from being a comprehensive review of the field, it aims to offer the reader a background on the multitude of approaches proposed over the years with enough detail to set the stage for a more detailed exploration. For additional information, we recommend the following excellent surveys and reviews: [26][27][28][29][30][31][32][33][21].

2   Background of Quantum Computation

Quantum mechanics is based on four fundamental postulates [32][6]: (1) the pure state of a quantum system is given by a unit vector $|\psi\rangle$ in a complex Hilbert space; (2) the evolution of a pure state in a closed system is governed by a Hamiltonian H as specified by Schrödinger's equation $H|\psi\rangle = i\hbar\frac{\partial}{\partial t}|\psi\rangle$; (3) the quantum state of a composite system is given by the tensor product of the individual systems; (4) projective measurements (observables) are specified by

3

Hermitian operators, and the process of measurement changes the observed system from $|\psi\rangle$ to an eigenstate $|\phi\rangle$ with probability given by the Born rule $p(\phi) = |\langle\psi|\phi\rangle|^2$. In this section, we briefly set up the background of quantum computation based on the above postulates.

2.1 Single Qubit

A classical bit can exist in one of two states denoted as 0 and 1. A quantum bit or qubit can exist not only in these two discrete states but in all possible linear superpositions of them. Mathematically, the quantum state of a qubit is represented as a state vector in a two-dimensional Hilbert space. In the Dirac notation, the state vector of a qubit ψ is called a *ket* and is written as:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \tag{1}$$

where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. The Born's rule tells us that if this qubit is measured, we will get $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. Quantum measurements are non-deterministic, and the act of measurement changes the quantum state irreversibly. Before measurement, the qubit exists in a quantum superposition of the states $|0\rangle$ and $|1\rangle$. The outcome of the measurement, however, is not quantum but classical i.e. you get either a $|0\rangle$ or a $|1\rangle$ but not a superposition of the two. During the measurement, the quantum state collapses[1] to the classical state it gets observed in, and all subsequent measurements deterministically result in this same outcome with a probability equal to 1.

The choice of basis vectors $|0\rangle$ and $|1\rangle$ is arbitrary. We can represent the system using a different set of orthogonal basis vectors such as $|+\rangle$ and $|-\rangle$ (called the Hadamard or sign basis). Once the computational basis is decided, kets can be represented as column vectors:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2}$$

$$|\psi\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{3}$$

The two representations for $|\psi\rangle$ given in equations (1) and (3) are equivalent.

2.2 Multiple Qubits

The quantum state of a system consisting of more than one *unentangled* qubits can be represented as the tensor product of the quantum states of the individual qubits. The state of a two-qubit system comprising of qubits represented by $|\psi_1\rangle$ and $|\psi_2\rangle$ can be written as

---

[1]Different interpretations exist regarding the collapse of the quantum state [14]. The popular Copenhagen Interpretation suggests that the wave function of a quantum system collapses on observation. The alternative Many Worlds Interpretation suggests that there is no collapse of the wave function; instead, the act of observation results in the observer getting entangled with the observed system.

$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$. In general, the state of n qubits $\{|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_n\rangle\}$ can be represented as:

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_k\rangle = |\psi_1\psi_2 \ldots \psi_k\rangle . \tag{4}$$

However, not all multi-qubit states can be represented as a tensor product of individual states. Consider the state below, one of the Bell states:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{5}$$

Suppose it could be decomposed into the tensor product of two states as below:

$$\begin{aligned} |\Phi^+\rangle &= (a_1 |0\rangle + b_1 |1\rangle) \otimes (a_2 |0\rangle + b_2 |1\rangle) \\ &= a_1 a_2 |00\rangle + a_1 b_2 |01\rangle + a_2 b_1 |10\rangle + b_1 b_2 |11\rangle \end{aligned} \tag{6}$$

From equations (5) and (6), we know that $a_1 b_2 = a_2 b_1 = 0$. Therefore, either $a_1 a_2 = 0$ or $b_1 b_2 = 0$. But, from equation (5), both $a_1 a_2 \neq 0$ and $b_1 b_2 \neq 0$. This proves that the Bell state $|\Phi^+\rangle$ cannot be decomposed into the tensor product of two single-qubit states. In such cases, we say that the two qubits are *entangled*. Given an entangled pair of qubits, measurement on one qubit instantaneously affects the other qubit. Entanglement plays a central role in many quantum algorithms especially in the field of quantum cryptography. There is no counterpart to quantum entanglement in classical physics.

2.3 Quantum Gates

Classical computers manipulate information stored in bits using logic gates such as AND, OR, NOT, NAND, XOR etc. Likewise, quantum computers manipulate qubits using quantum gates. Transformations on quantum states are represented as rotation of the Hilbert space. Rotation is linear and reversible. Consequently, all transformations on quantum states must be linear and reversible. Quantum gates essentially transform the system from one state to another state. These transformations can be represented as matrices. The simplest quantum gate is the NOT gate. The NOT gate transforms $|\psi_1\rangle = \alpha |0\rangle + \beta |1\rangle$ to $|\psi_2\rangle = \alpha |1\rangle + \beta |0\rangle$ and can be represented as:

$$NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{7}$$

The Hadamard gate acts on a single qubit. It is often used to map a qubit from one of its basis states into an equal superposition of all basis states. It transforms $|0\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ and is given by:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{8}$$

In general, an n-qubit Hadamard gate is used to initialize an n-qubit system into an equal superposition of all basis states:

$$|0^n\rangle \xrightarrow{H^{\otimes n}} \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle \tag{9}$$

5

where $x \in \{0,1\}^n$ denotes all strings of length n consisting of 0 and 1.

The CNOT (controlled-NOT) gate acts on two qubits where the first qubit acts as a control signal that decides whether the NOT operation should be performed on the second qubit. If the control qubit is $|1\rangle$, the NOT operation is applied; if it is $|0\rangle$, it is not applied. The CNOT gate leaves the states $|00\rangle$ and $|01\rangle$ unchanged, while it maps $|10\rangle$ to $|11\rangle$ and $|11\rangle$ to $|10\rangle$. It is represented as:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{10}$$

The SWAP gate swaps the states of two qubits transforming $|\psi, \phi\rangle$ to $|\phi, \psi\rangle$. The CSWAP (controlled-SWAP) gate acts on three qubits and swaps the state of the second and third qubit if the first qubit is $|1\rangle$. The Toffoli gate (CCNOT) acts on three qubits and performs the computation:

$$|a, b, c\rangle \rightarrow |a, b, c \oplus ab\rangle \tag{11}$$

2.4 Quantum Parallelism

While classical computers can execute only one computational path at a time, quantum computers can leverage the ability of quantum states to exist in superpositions to simultaneously execute multiple computational paths. For example, consider the classical function $f(x) : \{0,1\}^2 \rightarrow \{0,1\}$. The function takes two bits as input and outputs a single bit. To evaluate f on all two-bit permutations using classical computation, we need to call f four times: $f(0,0)$, $f(0,1)$, $f(1,0)$, and $f(1,1)$. Quantum superposition allows us to evaluate all four inputs in a single call to f.

Since quantum transformations must be reversible and f is not reversible, we define a reversible quantum function:

$$U_f |x\rangle |y\rangle \rightarrow |x\rangle |y \oplus f(x)\rangle \tag{12}$$

The input $|\phi\rangle$ is set up in a superposition of states by initializing two qubits to $|0\rangle$ and applying the Hadamard transform:

$$|\phi\rangle = (H \otimes H) |00\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{13}$$

Setting $|y\rangle = |0\rangle$, we apply U_f as follows:

$$\begin{aligned} U_f |\phi\rangle |0\rangle &= \frac{1}{2} U_f(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes |0\rangle \\ &= \frac{1}{2} U_f(|00, 0\rangle + |01, 0\rangle + |10, 0\rangle + |11, 0\rangle) \\ &= \frac{1}{2}(|00, f(00)\rangle + |01, f(01)\rangle + |10, f(10)\rangle + |11, f(11)\rangle) \end{aligned} \tag{14}$$

Thus, with a single application of f, we simultaneously evaluate four inputs. Using the Hadamard transform the set up in the input in an equal superposition of all basis vectors is a useful starting point for many quantum algorithms.

2.5 No Cloning Theorem

An important result that has profound implications is the *no cloning theorem* [34] which states that it is not possible to create a copy of an unknown quantum state. Since measurement irreversibly changes the quantum state, given a single copy of the state $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, the values of the amplitudes α and β cannot be exactly determined. Although quantum parallelism can be leveraged to simultaneously execute multiple computational paths, the no cloning theorem places restrictions on the amount of information one can extract from the final quantum state[2].

2.6 Adiabatic Quantum Computation

Numerous models have been proposed for quantum computing such as the quantum Turing machine, quantum circuit model, adiabatic quantum computing, measurement-based quantum computing, blind quantum computing, topological quantum computing etc. [32]. All these models are computationally equivalent, but they are implemented very differently. An approach that has shown promise in solving optimization problems is adiabatic[3] quantum computing [36] and is of particular interest because building restricted quantum computers to perform *quantum annealing* (section 3.2) based on adiabatic quantum computing is simpler than building universal quantum computers. In adiabatic quantum computing, the optimization problem to be solved is encoded as a boolean satisfiability problem such that the ground state of its Hamiltonian[4] represents the desired solution. The quantum system is initially set up with a simple Hamiltonian that is easy to construct. The system is then evolved from the initial state to a final state. The adiabatic theorem states that if the system is evolved slowly enough, it will remain in the ground state of the instantaneous Hamiltonian throughout the evolution. The final system configuration then represents the solution to the optimization problem.

3 Quantum Machine Learning

Quantum computing methods for machine learning can be divided into two broad classes: (1) methods designed to run on a universal quantum computer that involve preparation, storage, and processing of quantum states and the retrieval of the classical solution from these states; (2) methods designed to run on *quantum annealers* that solve optimization problems through the physical evolution of quantum systems according to the principles of adiabatic quantum computing. In section 3.1, we describe common subroutines designed for circuit quantum computers that can be applied to machine learning problems. In section 3.2, we present quantum annealing that can solve quadratic unconstrained binary optimization (QUBO) problems. Finally, in section 3.3, we present quantum versions of selected classical machine learning algorithms.

---

[2]Although perfect cloning is impossible, Buzek and Hillery [35] proposed a universal cloning machine that can make imperfect copies of unknown quantum states with high fidelity.

[3]In the context of quantum computing, an adiabatic process is a process which changes the state of a system so gradually that the state can adapt its configuration at each point.

[4]The Hamiltonian operator represents the total energy of the system.

3.1 Important Subroutines of Quantum Algorithms

A straightforward approach to achieving speed-ups over classical machine learning algorithms is to identify their computationally expensive and frequently executed subroutines and develop quantum alternatives for them. In this section, we describe some common subroutines that form a part of many quantum learning algorithms.

3.1.1 Quantum Encoding

Qubits are a scarce and expensive resource. The restricted physical implementations of quantum computers available today have very few qubits[5]. An important question therefore that has implications for performance and feasibility is how to represent classical data in quantum states. Suppose we have a dataset D of N instances:

$$D = \{x_1, ..., x_i, ..., x_N\} \tag{15}$$

where each x_i is a real number. In *basis encoding*, each instance x_i is encoded as $|b_i\rangle$ where b_i is the binary representation of x_i. The dataset D can then be represented as a superposition of all computational basis states:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |b_i\rangle \tag{16}$$

In *amplitude encoding*, data is encoded in the amplitudes of quantum states. The above dataset D can be encoded as:

$$|\psi\rangle = \frac{\sum_{i=1}^{N} x_i |i\rangle}{||D||} \tag{17}$$

Besides basis and amplitude encoding, many other methods of encoding exist such as *Qsample encoding, dynamic encoding, squeezing embedding, displacement embedding, Hamiltonian encoding* etc. [38][39].

3.1.2 Grover's Algorithm and Amplitude Amplification

Grover's algorithm [20] is a quantum search algorithm that offers a quadratic speed-up over classical algorithms when performing a search over an unstructured search space. Suppose we are given a set of N elements $X = \{x_1, ..., x_i, ..., x_N\}$ where $x_i \in \{0,1\}^m$ and a boolean function $f : \{0,1\}^m \rightarrow \{0,1\}$ such that:

$$f(x) = \begin{cases} 1 & x = x^* \\ 0 & x \neq x^* \end{cases} \tag{18}$$

Any classical algorithm that performs a search for x^* in X is $O(N)$ in time. Grover's algorithm can perform such a search in $O(\sqrt{N})$. The algorithm has three steps to it.

---

[5]The recent *quantum supremacy* experiment conducted by Google used only 54 qubits [37].

In the first step, a quantum state is set up in an equal superposition of basis states using the Hadamard transform. As an example, consider $N = 8$. We set up the state using 3 qubits as:

$$
\begin{aligned}
|\psi_1\rangle &= (H \otimes H \otimes H)\,|000\rangle \\
&= \frac{1}{2\sqrt{2}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle) \\
&= \begin{bmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \end{bmatrix}^T
\end{aligned}
\tag{19}
$$

The second step referred to as *phase inversion* deals with flipping the amplitude of each $|x\rangle$ if $f(x) = 1$ and leaving it unchanged if $f(x) = 0$. To do this, we define a unitary quantum oracle $O\,|x\rangle = (-1)^{f(x)}\,|x\rangle$. Suppose, in our example, x^* is present at the fourth position. Applying gate O on $|\psi_1\rangle$ gives us:

$$
|\psi_2\rangle = \begin{bmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{-1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \end{bmatrix}^T
\tag{20}
$$

The third step referred to as *inversion around the mean* involves flipping all amplitudes around their collective mean $\mu = \frac{1}{N}\sum_x \alpha_x$. This is performed by the *Grover diffusion operator* G:

$$
\sum_x \alpha_x\,|x\rangle \xrightarrow{G} \sum_x (2\mu - \alpha_x)\,|x\rangle
\tag{21}
$$

Applying G to $|\psi_2\rangle$ gives us:

$$
|\psi_3\rangle = \begin{bmatrix} \frac{1}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{5}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} \end{bmatrix}^T
\tag{22}
$$

Thus, after one iteration, the amplitude of the target element x^* is higher than the amplitudes of other elements. If we were to measure the system at this point, we would get the target element as outcome with a probability of $(\frac{5}{4\sqrt{2}})^2 = 78\%$. The second and third steps are repeated $\sqrt{N}$ times to maximize this probability. After the second iteration, we get the below state which will find the target element with a probability of 95%:

$$
|\psi_4\rangle = \begin{bmatrix} \frac{-1}{8\sqrt{2}} & \frac{-1}{8\sqrt{2}} & \frac{-1}{8\sqrt{2}} & \frac{11}{8\sqrt{2}} & \frac{-1}{8\sqrt{2}} & \frac{-1}{8\sqrt{2}} & \frac{-1}{8\sqrt{2}} & \frac{-1}{8\sqrt{2}} \end{bmatrix}^T
\tag{23}
$$

The same algorithm can also find k matching entries instead of a single entry. Several modifications have been proposed that extend this work. Durr and Hoyer [40] propose a quantum algorithm to find the index of the minimum value from a list of N values in time $O(c\sqrt{N})$ with a probability of at least $(1 - \frac{1}{2^c})$. These methods generalize Grover's search and are collectively referred to as *amplitude amplification* techniques [41].

3.1.3 Calculating Inner Products using Swap Test

The swap test [42] is a simple subroutine used to compute the overlap between two quantum states $|\phi\rangle$ and $|\psi\rangle$. Quantum procedures can be easily described using circuit diagrams. The
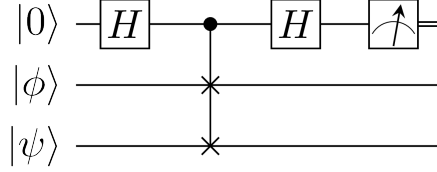
Figure 1: Circuit diagram describing swap test. Computation proceeds from left to right.

circuit diagram of the swap test is shown in figure 1. The system is initially prepared in the state $|0, \phi, \psi\rangle$. The Hadamard gate applied on the first *ancilla qubit*[6] transforms the state to $\frac{1}{\sqrt{2}}(|0, \phi, \psi\rangle + |1, \phi, \psi\rangle)$. The CSWAP further transforms it to $\frac{1}{\sqrt{2}}(|0, \phi, \psi\rangle + |1, \psi, \phi\rangle)$. After the application of the second Hadamard gate to the first qubit, the state can be written as $\frac{1}{2}|0\rangle(|\phi, \psi\rangle + |\psi, \phi\rangle) + \frac{1}{2}|1\rangle(|\phi, \psi\rangle - |\psi, \phi\rangle)$. The probability of measuring the first qubit as $|0\rangle$ is given by $p = \frac{1}{2} + \frac{1}{2}|\langle\psi|\phi\rangle|^2$. If $|\phi\rangle$ and $|\psi\rangle$ are equal, $|\langle\psi|\phi\rangle|^2 = 1$, and the observed value of p is 1. If $|\phi\rangle$ and $|\psi\rangle$ are orthogonal, $|\langle\psi|\phi\rangle|^2 = 0$, and the observed value of p is $\frac{1}{2}$. The degree of overlap given by the inner product of the two states can be estimated with this method to precision ϵ using $O(1/\epsilon)$ copies [32].

3.1.4   Solving Systems of Linear Equations (HHL)

Solving a system of linear equations is an important problem ubiquitous throughout science and engineering. A seminal result in quantum computation is the HHL algorithm [23] that solves the following problem: given a Hermitian matrix $A \in \mathbb{R}^{N \times N}$ and a unit vector $\overrightarrow{b} \in \mathbb{R}^N$, find a solution vector $\overrightarrow{x} \in \mathbb{R}^N$ that satisfies the equation $A\overrightarrow{x} = \overrightarrow{b}$.

We present here a condensed outline of the algorithm. The solution we are interested in is $|x\rangle = A^{-1}|b\rangle$. Let $\{v_1, ..., v_N\}$ be the eigenvectors of A with corresponding eigenvalues $\{\lambda_1, ..., \lambda_N\}$. The vector $\overrightarrow{b}$ is encoded using amplitude encoding (section 3.1.1) as $|b\rangle = \sum_{i=1}^{N} \beta_i |v_i\rangle$. *Hamiltonian simulation* is used to transform the matrix A into a unitary operator, and *quantum phase estimation*[7] is used to carry out eigendecomposition to get the state $\sum_{i=1}^{N} \beta_i |v_i\rangle |\lambda_i\rangle$. An ancilla qubit is added, rotation conditioned on $|\lambda_i\rangle$ is carried out, and the eigenvalue register $|\lambda_i\rangle$ is uncomputed to yield a state proportional to $\sum_{i=1}^{N} \beta_i \lambda_i^{-1} |v_i\rangle = A^{-1}|b\rangle = |x\rangle$.

It is important to note that while a classical algorithm finds all coefficients x_i for x, the HHL algorithm finds the quantum state $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$. Obtaining values for all x_i takes $O(N)$ repetitions; this observation nullifies the speed-up the quantum algorithm has over the classical counterparts. Hence, the HHL algorithm is most useful when used as a subroutine carrying out an intermediate step in a larger process where the quantum state $|x\rangle$ is consumed by the next subroutine in the process.

---

[6]Input qubits that do not hold any input data but are added to satisfy other conditions (most often reversibility of the transformation) are called auxiliary or ancillary qubits.

[7]The quantum phase estimation algorithm can estimate the phase (or eigenvalue) of an eigenvector of a unitary operator.

3.1.5 Quantum Random Access Memory

Most quantum algorithms assume parallel access to amplitude encoded states; this is performed using a quantum random access memory or QRAM [43]. The classical RAM takes a memory address as input and returns the data stored at that address. A QRAM performs a similar operation using qubit registers. The input register contains a superposition of addresses $\sum_a c_a |a\rangle$ and the output register contains a superposition of the data at those addresses $\sum_a c_a |a\rangle |D_a\rangle$. While a classical RAM queries N addresses in $O(N)$, QRAM performs the operation in $O(\log N)$.

3.2 Quantum Annealing

Quantum annealing is a metaheuristic optimization algorithm that leverages quantum effects to solve quadratic unconstrained binary optimization (QUBO) problems that deal with optimizing functions of the form:

$$C(x_1, x_2, ..., x_n) = \sum_i a_i x_i + \sum_{i,j} b_{i,j} x_i x_j \tag{24}$$

where $a_i \in \mathbb{R}$, $b_{i,j} \in \mathbb{R}$, $x_i \in \{0,1\}$. A wide range of problems can be mapped to QUBO and then solved by quantum annealers which are special-purpose quantum computers specifically built to perform quantum annealing.

The Ising model is used in physics to represent a large variety of systems. It was originally proposed to model magnetic materials where every molecule has a spin that can align or anti-align with an applied magnetic field [44]. The Hamiltonian of the system representing its energy is given by:

$$H = \sum_i h_i s_i + \sum_{i,j} J_{i,j} s_i s_j \tag{25}$$

where $s_i \in \{-1, +1\}$ is the spin of the ith molecule, h_i is the strength of the magnetic field at the ith molecule, and $J_{i,j}$ is the strength of the interaction between neighboring spins i and j. From equations (24) and (25), it can be seen that the QUBO problem convenient maps onto the Ising Hamiltonian with the mapping $s_i = 2x_i - 1$.

Quantum annealing works as follows. An initial Hamiltonian H_0 that is easy to construct is chosen. The system is evolved under a time-dependent Hamiltonian given by:

$$H(t) = (1-t)H_0 + tH_f \tag{26}$$

where t is gradually changed from 0 to 1 and the final Hamiltonian H_f is the same as in equation (25). At $t = 0$, the system starts in the ground state of H_0. According to the quantum adiabatic theorem, the system remains in the ground state of the instantaneous Hamiltonian $H(t)$ throughout its evolution provided it is changed sufficiently slowly [45]. At $t = 1$, the final Hamiltonian of the system will encode the solution to the problem.

Quantum annealing should not be conflated with the more general adiabatic quantum computing. Quantum annealing specifically solves optimization problems; adiabatic quantum computing is a model of quantum computing that is equivalent to a universal quantum

computer. The Hamiltonians used in quantum annealing are classical Hamiltonians, while adiabatic quantum computing uses quantum Hamiltonians that have no classical counterparts [46]. For a more comprehensive treatment of adiabatic quantum computing, we refer the reader to [47].

3.3   Quantum Algorithms for Machine Learning

In this section, we explore how the background and subroutines described in the previous sections can be applied to solve machine learning problems. The common supervised machine learning setting is as follows. The training set consists of M instances $\{X_1, ..., X_m, ..., X_M\}$ with corresponding labels $\{Y_1, ..., Y_m, ..., Y_M\}$. Each X_m is represented by an N-dimensional feature vector $\{x_1^{(m)}, x_2^{(m)}, ..., x_N^{(m)}\}$. Each label Y_m can be either a real value (for regression problems) or a discrete class label (for classification problems). In the *training* phase, the algorithm extracts patterns from the dataset and *learns* a model. In the *inference* phase, this model is used to process unseen instances and predict their corresponding labels.

3.3.1   k-Nearest Neighbors

The k-nearest neighbors (KNN) is one of the simplest supervised learning algorithms. To predict the label of a new unseen instance x_{test}, the algorithm looks at k instances in the training set that are closest to x_{test} and chooses the class that appears most often in the labels of these *k-nearest neighbors* as the predicted label (for regression, the algorithm assigns the mean value of the k-nearest neighbors as the label). An advantage of KNN is that, unlike many other supervised algorithms, it is non-parametric and makes no assumptions about the data distribution. However, since all computation is deferred, inference can become prohibitively expensive for large training sets. During inference, the distance of the test instance from all other training instances is calculated; this is the most computationally intensive step in the process. Hence, quantum versions of KNN focus on faster evaluation of the distance between two instances.

Aïmeur et al. [48] propose using the overlap $|\langle a|b\rangle|$ as computed by the swap test (section 3.1.3) as a measure of similarity between $|a\rangle$ and $|b\rangle$. Llyod et al. [49] develop a technique based on the swap test to compute the distance between $\overrightarrow{a}$ and $\overrightarrow{b}$. A state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0, a\rangle + |1, b\rangle)$ is constructed by setting up an ancilla. A second state given by $|\phi\rangle = \frac{1}{\sqrt{Z}}(|\overrightarrow{a}| |0\rangle - |\overrightarrow{b}| |1\rangle)$ is constructed where $Z = |\overrightarrow{a}|^2 + |\overrightarrow{b}|^2$. Using $\langle x|x\rangle = |\overrightarrow{x}|^{-1}|\overrightarrow{x}|$, the authors make the observation that $|\overrightarrow{a} - \overrightarrow{b}|^2 = Z|\langle\phi|\psi\rangle|^2$. With this, the distance between $\overrightarrow{a}$ and $\overrightarrow{b}$ can be retrieved using a swap test [26]. The authors use the above technique to implement the nearest-centroid classification algorithm in which the centroids of all training instances belonging to each class are precomputed; during inference, a class label is assigned to the test instance by calculating its distance from all centroids. Given a training set of M instances, this procedure solves the problem of classifying an N-dimensional test vector into one of several classes in $O(log(MN))$ compared to $O(MN)$ required by classical algorithms.

Wiebe et al. [50] argue that the nearest-centroid classification presented above can perform poorly in practice since training instances are often embedded in complicated

manifolds, and the centroids may lie outside these manifolds. They propose two fast methods for computing the distance between vectors based on an alternative representation of classical information in quantum states, amplitude amplification, and Durr-Hoyer minimum finding [40].

3.3.2 Support Vector Machines

Support vector machines (SVM) [51] is a popular classification algorithm that determines the optimal hyperplane that separates instances of two classes in the training data and classifies test instances based on which side of the separating hyperplane they lie on. Given training instances $\{(x_1, y_1), ..., (x_i, y_i), ..., (x_M, y_M)\}$ where $x_i \in \mathbb{R}^N$ and $y_i \in \{-1, +1\}$, the algorithm learns an N-dimensional hyperplane given by $w = [\beta_1, \beta_2, ..., \beta_N]^T$ that separates the instances of the two classes with maximum margin. Classification of a test instance x_t is performed as:

$$y_t = sign(w^T x_t + b) \tag{27}$$

Since the instances may not be strictly separable, slack variables are introduced that provide a *soft* margin by allowing some data points to violate the margin criterion. This formulation is known as the maximum margin classifier or support vector classifier; this however still requires the data points to be *linearly* separable. SVMs overcome this limitation of linear separability by what is known as the *kernel trick* which transforms the feature space into a new, higher-dimensional feature space. Instances that were not linearly separable in the original feature space may be linearly separable in the new feature space. Mathematically, the kernel trick generalizes the dot product between two feature vectors $\langle x_i, x_j \rangle$ by a kernel function $k(x_i, x_j)$. Different kernels can be chosen depending on the data distribution. Training an SVM involves solving the quadratic programming problem [52]:

$$
\begin{aligned}
minimize \quad & E = \frac{1}{2} \sum_{nm} \alpha_n \alpha_m t_n t_m k(x_n, x_m) - \sum_n \alpha_n, \\
subject\,to \quad & 0 \leq \alpha_n \leq C, \\
and \quad & \sum_n \alpha_n t_n = 0
\end{aligned}
\tag{28}
$$

where $\alpha_n \in \mathbb{R}$, C is the regularization parameter, and $k(\cdot, \cdot)$ is the kernel function.

Anguita et al. [53] observe that training support vector machines may be a hard problem and propose a quantum variant that uses Durr and Hoyer's minimum finding [40] based on Grover's algorithm to solve the optimization problem. Rebentrost et al. [54] suggest computing inner products using a quantum method based on an approach similar to the one discussed in section 3.3.1 which leads to an exponential speed-up with respect to the dimension of the feature vector N. They also describe a least-squares reformulation of the SVM algorithm with slack variables e_j that converts the quadratic optimization problem into a problem of solving a system of linear equations which leads to an additional exponential speed-up in terms of the number of training instances M:

$$y_j(\overrightarrow{w} \cdot \overrightarrow{x_j} + b) \geq 1 \rightarrow (\overrightarrow{w} \cdot \overrightarrow{x_j} + b) = y_j - y_j e_j \tag{29}$$

As shown in section 3.2, quantum annealing is particularly well-suited for solving optimization problems. Willsch et al. [55] demonstrate a practical implementation of training SVMs on the commercially available D-Wave DW2000Q quantum annealer by formulating it as a QUBO problem.

3.3.3 Neural Networks

Artificial neural networks or simply neural networks were originally inspired from biological neural networks that model the activity of neurons in human brains. The basic building block of a neural network is the neuron, also called node or perceptron, that maps the input $x \in \mathbb{R}^N$ to the output $y \in \mathbb{R}$ as follows[8]:

$$y = g(\sum_{i=1}^{N} w_i x_i + b) \tag{30}$$

where $w_i \in \mathbb{R}$, $b \in \mathbb{R}$, and $g : \mathbb{R} \to \mathbb{R}$ is the activation function. The outputs of some neurons can be fed as inputs to other neurons thus creating layers within the network.

Even though neural networks were amongst the first machine learning algorithms to be proposed, their research stagnated for several decades from 1940s to 1980s due to the inherent difficulty and large computational power required to train them. They returned to popularity after the introduction of backpropagation [56] which eased these problems by offering a faster method for training. In the last decade, with GPUs and cloud computing providing cheaper access to massive computational power, neural networks have dwarfed other learning algorithms[9] to become one of the biggest success stories of modern computers finding applications in various industries including healthcare, manufacturing, finance, analytics etc. to solve problems in image processing, computer vision, natural language processing, predictive modeling, and many other areas. However, even today, significant resources are required to train neural networks, and training times for research and industrial problems can run into weeks or even months.

An obvious difficulty arises in considering quantum computation as a means for implementing neural networks. Quantum computation (and indeed quantum mechanics itself) is a theory fundamentally based on linear transformations, while an important practical advantage neural networks enjoy over many other learning algorithms is that they can model non-linear data distributions. Bringing non-linearity into quantum algorithms is a non-trivial task [57]. However, classical neural networks do make heavy use of linear algebra, and the inherent randomness of quantum mechanical effects can be leveraged to automatically introduce noise in the training process to improve model robustness - something that needs to be done purposefully in classical training [58]. Numerous neural network architectures have emerged in recent times to tackle problems belonging to a wide range of supervised, unsupervised, and reinforcement learning tasks [59]. Research in this field has been scattered with different proposals addressing narrow problems in piecemeal style. We present here a select subset of these proposals.

---

[8]This is the general form used in modern feedforward neural networks. The original perceptron used a step activation function and produced only binary outputs 0 and 1.

[9]Many informal texts now relegate all other learning algorithms to the category *conventional machine learning*.

Most early work on *quantizing*[10] neural networks focussed on Hopfield networks [60] which differ from the neural networks presently used in practice. In a Hopfield network, all neurons have undirected connections with all other neurons as opposed to feed-forward networks that are organized as layers; also, each neuron outputs a binary 0 or 1 instead of a real number. Hopfield networks are used to model associative memories which allow retrieval of data based on content rather than addresses. Kak [61] introduces the idea of *quantum neural computation* and Perus [62] describes a *quantum associative network* by drawing analogies between Hopfield networks and quantum information processing. Behrman et al. [63] present a quantum realization of neural networks by showing that a single quantum dot[11] molecule can act as a recurrent quantum neural network. More recently, Rebentrost et al. [64] present a technique based on quantum Hebbian learning and fast quantum matrix inversion to train Hopfield networks.

Boltzmann machines [65], closely related to Hopfield networks, are stochastic generative networks that can learn a probability distribution over a set of inputs. They are trained by adjusting the interconnection weights between the neurons so that the thermal statistics of the system as described by the Boltzmann-Gibbs distribution reproduces the statistics of the data [28]. Boltzmann machines can be conveniently represented by an Ising model whose spins encode features and interactions encode statistical dependencies between the features. In a restricted Boltzmann machine, connections exist only between neurons belonging to different layers; this makes them easier to train than fully-connected Boltzmann machines. Restricted Boltzmann machines can be stacked together to form *deep belief networks* that can be used to learn internal representations or can be trained under supervision to perform classification. Training Boltzmann machines is exponentially hard and is performed using approximation techniques like contrastive divergence [66][67] that rely on Gibbs sampling. Wiebe et al. [50] propose quantum methods to efficiently train full Boltzmann machines by preparing a coherent analog of the Gibbs state from which samples can be drawn. Adachi et al. [68] investigate an alternative approach of performing the sampling on a D-Wave quantum annealer instead of classical Gibbs sampling.

A different line of research involves developing quantum analogs for classical perceptrons. Schuld et al. [69] introduce a quantum perceptron model with a step activation function that can be used to develop superposition-based learning schemes in which a superposition of training vectors can be processed in parallel. Kapoor et al. [70] develop two quantum techniques for modeling perceptrons; the first provides quadratic speed-up with respect to the number of training instances, and the second provides a quadratic reduction in the scaling of the training time with the margin between the two classes.

Feedforward networks are one of the simplest neural network architectures in which the connections between neurons do not form any loops or cycles. They are usually trained using backpropagation, and the optimization is performed using some variant of gradient descent. Most machine learning architectures used in practice today are based on feedforward networks or their derivatives such as convolutional neural networks or recurrent neural networks [59]. Allcock et al. [58] define an efficient quantum subroutine for robust inner

---

[10]Developing a quantum alternative to a classical computation technique is often referred to as *quantizing* it although we prefer this term is used sparingly.

[11]Quantum dots are nanometre-scale semiconductor particles.

15

product estimation using QRAM [43] and use it to demonstrate quadratic speed-ups in the size of the network over classical counterparts; they additionally claim that the proposed quantum method naturally imitates regularization techniques like drop-out leading to more robust networks. Farhi et al. [36] present a general framework for binary classification specifically designed for near-term quantum processors in which the input strings are mapped to computational basis states, and the predicted label is given by the outcome of a Pauli operator measured on a readout qubit. This framework extends to a full quantum neural network that can classify both classical and quantum data. Convolutional neural networks (CNNs) [71] have achieved great success [72] in image classification tasks in recent times. They, however, suffer from the fact that the operation of convolution is computationally expensive. Kerenidis et al. [73] design a quantum CNN based on quantum computation of the convolution product between two tensors. They also propose a quantum tomography[12] sampling approach to recover classical information from the network output and a quantum backpropagation algorithm for efficient training of the quantum CNN.

It is important to distinguish between *quantum-enhanced* machine learning that focuses on techniques to implement learning on classical data using quantum computers from *quantum-generalisation* of machine learning algorithms that deals with developing fully quantum algorithms that work with quantum data. This is especially true for neural networks for which active research is underway on both fronts. We restrict this paper to quantum-enhanced techniques and refer the reader to [57][74][75][76][77] for work on quantum generalisations. The principles of quantum computing have inspired development of new classical randomized algorithms that show exponential speed-ups over conventional algorithms [25]. With these *quantum-inspired* algorithms, the gap between certain classical and quantum algorithms is no longer exponential but polynomial. Arrazola et al. [78] provide a study of these algorithms and observe that they work well only under stringent conditions which occur rarely in practice. We do not cover these in this paper.

4   Discussion

Quantum computation has made great strides in the last two decades in both theory and practice. A significant corpus of research has emerged in applying the principles of quantum computation to problems across many fields of science and engineering. At the same time, several approaches for physical realization of quantum computers based on superconducting quantum bits, trapped ions, optical lattices, photonic computing, nuclear magnetic resonance etc. have shown promise. However, fundamental challenges remain unresolved on both fronts. While developing quantum algorithms, we must consider the input problem and the output problem [28]. The input problem refers to the fact that the cost of reading classical data and encoding it in quantum states can sometimes dominate performance and render the further downstream speed-up irrelevant. The output problem refers to the reverse process of decoding the full classical solution from quantum states. Some important hardware challenges to constructing, operating, and maintaining large-scale quantum computers include achieving longer coherence, greater circuit depth, higher qubit quality, and higher control

---

[12]Quantum tomography is the process by which a quantum state is reconstructed using measurements on an ensemble of identical quantum states.

over qubits. Quantum error correction plays an important role, and it will likely span across hardware and software in the future.

Owing to its roots in quantum physics, research in quantum computing has so far been confined within the purview of the physics community. Although realization of quantum computers in the form of hardware will remain a problem for physicists, we believe this need not be the case when it comes to applying quantum computing to solve machine learning problems. Classical computing and machine learning, like physics and many other fields, serve as prime examples of disciplines where theoretical results were obtained far before technological progress made possible their experimental realizations. Small-scale quantum computers with less than 100 qubits and quantum annealers with around 2000 qubits have been developed and are already being sold commercially [7][8]. We hope this article serves its purpose as introductory material for interested machine learning researchers and practitioners from various disciplines.

References

[1] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[2] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12, 2017.

[3] T. L. Friedman, "Moores law turns 50," *The New York Times*, vol. 13, 2015.

[4] C. E. Leiserson, N. C. Thompson, J. S. Emer, B. C. Kuszmaul, B. W. Lampson, D. Sanchez, and T. B. Schardl, "Theres plenty of room at the top: What will drive computer performance after moores law?," *Science*, vol. 368, no. 6495, 2020.

[5] E. Sperling, "Quantum effects at 7/5nm and beyond," *Semiconductor Egineering*, 2018.

[6] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[7] D. Castelvecchi, "Ibm's quantum cloud computer goes commercial," *Nature News*, vol. 543, no. 7644, p. 159, 2017.

[8] H. Johnston, "D-wave sells second quantum computer-this time to nasa," *Physics World*, vol. 26, no. 07, p. 9, 2013.

[9] A. Einstein, "On a heuristic viewpoint concerning the production and transformation of light," *Annalen der Physik*, 1905.

[10] J. C. Maxwell, "Viii. a dynamical theory of the electromagnetic field," *Philosophical transactions of the Royal Society of London*, no. 155, pp. 459–512, 1865.

[11] L. d. Broglie, "Xxxv. a tentative theory of light quanta," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 47, no. 278, pp. 446–458, 1924.

[12] D. A. Edwards, "The mathematical foundations of quantum mechanics," *Synthese*, vol. 42, no. 1, pp. 1–70, 1979.

[13] E. Schrödinger, "An undulatory theory of the mechanics of atoms and molecules," *Physical review*, vol. 28, no. 6, p. 1049, 1926.

[14] M. Schlosshauer, J. Kofler, and A. Zeilinger, "A snapshot of foundational attitudes toward quantum mechanics," *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics*, vol. 44, no. 3, pp. 222–230, 2013.

[15] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines," *Journal of statistical physics*, vol. 22, no. 5, pp. 563–591, 1980.

[16] R. P. Feynman, "Simulating physics with computers," *Int. J. Theor. Phys*, vol. 21, no. 6/7, 1999.

[17] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[18] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1907, pp. 553–558, 1992.

[19] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[20] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996.

[21] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, no. 1, pp. 1–8, 2016.

[22] G. E. Santoro and E. Tosatti, "Optimization using quantum mechanics: quantum annealing through adiabatic evolution," *Journal of Physics A: Mathematical and General*, vol. 39, no. 36, p. R393, 2006.

[23] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical review letters*, vol. 103, no. 15, p. 150502, 2009.

[24] N. Wiebe, D. Braun, and S. Lloyd, "Quantum algorithm for data fitting," *Physical review letters*, vol. 109, no. 5, p. 050505, 2012.

[25] E. Tang, "A quantum-inspired classical algorithm for recommendation systems," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 217–228, 2019.

[26] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.

[27] P. Wittek, *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

[28] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[29] J. Adcock, E. Allen, M. Day, S. Frick, J. Hinchliff, M. Johnson, S. Morley-Short, S. Pallister, A. Price, and S. Stanisic, "Advances in quantum machine learning," *arXiv preprint arXiv:1512.02900*, 2015.

[30] S. Arunachalam and R. de Wolf, "Guest column: A survey of quantum learning theory," *ACM SIGACT News*, vol. 48, no. 2, pp. 41–67, 2017.

[31] D. Kopczyk, "Quantum machine learning for data scientists," *arXiv preprint arXiv:1804.10068*, 2018.

[32] V. Dunjko and H. J. Briegel, "Machine learning & artificial intelligence in the quantum domain: a review of recent progress," *Reports on Progress in Physics*, vol. 81, no. 7, p. 074001, 2018.

[33] V. Dunjko and P. Wittek, "A non-review of quantum machine learning: trends and explorations," *Quantum Views*, vol. 4, p. 32, 2020.

[34] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, 1982.

[35] V. Bužek and M. Hillery, "Quantum copying: Beyond the no-cloning theorem," *Physical Review A*, vol. 54, no. 3, p. 1844, 1996.

[36] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum computation by adiabatic evolution," *arXiv preprint quant-ph/0001106*, 2000.

[37] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[38] M. Schuld and N. Killoran, "Quantum machine learning in feature hilbert spaces," *Physical review letters*, vol. 122, no. 4, p. 040504, 2019.

[39] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, "Quantum embeddings for machine learning," 2020.

[40] C. Durr and P. Hoyer, "A quantum algorithm for finding the minimum," *arXiv preprint quant-ph/9607014*, 1996.

[41] G. Brassard and P. Hoyer, "An exact quantum polynomial-time algorithm for simon's problem," in *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, pp. 12–23, IEEE, 1997.

[42] H. Buhrman, R. Cleve, J. Watrous, and R. De Wolf, "Quantum fingerprinting," *Physical Review Letters*, vol. 87, no. 16, p. 167902, 2001.

[43] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Physical review letters*, vol. 100, no. 16, p. 160501, 2008.

[44] Z. Bian, F. Chudak, W. G. Macready, and G. Rose, "The ising model: teaching an old problem new tricks," *D-wave systems*, vol. 2, 2010.

[45] A. Ambainis and O. Regev, "An elementary proof of the quantum adiabatic theorem," *arXiv preprint quant-ph/0411152*, 2004.

[46] R. Biswas, Z. Jiang, K. Kechezhi, S. Knysh, S. Mandrà, B. OGorman, A. Perdomo-Ortiz, A. Petukhov, J. Realpe-Gómez, E. Rieffel, *et al.*, "A nasa perspective on quantum computing: Opportunities and challenges," *Parallel Computing*, vol. 64, pp. 81–98, 2017.

[47] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, p. 015002, 2018.

[48] E. Aïmeur, G. Brassard, and S. Gambs, "Machine learning in a quantum world," in *Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 431–442, Springer, 2006.

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

[50] N. Wiebe, A. Kapoor, and K. Svore, "Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning," *arXiv preprint arXiv:1401.2142*, 2014.

[51] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[52] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Nu@articlewiebe2014quantum, title=Quantum deep learning, author=Wiebe, Nathan and Kapoor, Ashish and Svore, Krysta M, journal=arXiv preprint arXiv:1412.3489, year=2014 merical recipes 3rd edition: The art of scientific computing.* Cambridge university press, 2007.

[53] D. Anguita, S. Ridella, F. Rivieccio, and R. Zunino, "Quantum optimization for training support vector machines," *Neural Networks*, vol. 16, no. 5-6, pp. 763–770, 2003.

[54] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big feature and big data classifica-tion. corr, vol. abs/1307.0471, 2014," 2012.

[55] D. Willsch, M. Willsch, H. De Raedt, and K. Michielsen, "Support vector machines on the d-wave quantum annealer," *Computer Physics Communications*, vol. 248, p. 107006, 2020.

[56] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[57] Y. Cao, G. G. Guerreschi, and A. Aspuru-Guzik, "Quantum neuron: an elementary building block for machine learning on quantum computers," *arXiv preprint arXiv:1711.11240*, 2017.

[58] J. Allcock, C.-Y. Hsieh, I. Kerenidis, and S. Zhang, "Quantum algorithms for feedforward neural networks," *arXiv preprint arXiv:1812.03089*, 2018.

[59] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[60] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[61] S. C. Kak, "Quantum neural computing," in *Advances in imaging and electron physics*, vol. 94, pp. 259–313, Elsevier, 1995.

[62] M. Peruš, "Neural networks as a basis for quantum associative networks," *Neural Netw. World*, vol. 10, no. 6, pp. 1001–1013, 2000.

[63] E. C. Behrman, L. Nash, J. E. Steck, V. Chandrashekar, and S. R. Skinner, "Simulations of quantum neural networks," *Information Sciences*, vol. 128, no. 3-4, pp. 257–269, 2000.

[64] P. Rebentrost, T. R. Bromley, C. Weedbrook, and S. Lloyd, "Quantum hopfield neural network," *Physical Review A*, vol. 98, no. 4, p. 042308, 2018.

[65] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[66] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[67] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Artificial intelligence and statistics*, pp. 448–455, 2009.

[68] S. H. Adachi and M. P. Henderson, "Application of quantum annealing to training of deep neural networks," *arXiv preprint arXiv:1510.06356*, 2015.

[69] M. Schuld, I. Sinayskiy, and F. Petruccione, "Simulating a perceptron on a quantum computer," *Physics Letters A*, vol. 379, no. 7, pp. 660–663, 2015.

[70] A. Kapoor, N. Wiebe, and K. Svore, "Quantum perceptron models," in *Advances in Neural Information Processing Systems*, pp. 3999–4007, 2016.

[71] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[73] I. Kerenidis, J. Landman, and A. Prakash, "Quantum algorithms for deep convolutional neural networks," *arXiv preprint arXiv:1911.01117*, 2019.

[74] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, "Training deep quantum neural networks," *Nature communications*, vol. 11, no. 1, pp. 1–6, 2020.

[75] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, "Quantum boltzmann machine," *Physical Review X*, vol. 8, no. 2, p. 021050, 2018.

[76] K. H. W@articlewiebe2014quantum, title=Quantum deep learning, author=Wiebe, Nathan and Kapoor, Ashish and Svore, Krysta M, journal=arXiv preprint arXiv:1412.3489, year=2014 an, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. Kim, "Quantum generalisation of feedforward neural networks," *npj Quantum Information*, vol. 3, no. 1, pp. 1–8, 2017.

[77] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, p. 12731278, Aug 2019.

[78] J. M. Arrazola, A. Delgado, B. R. Bardhan, and S. Lloyd, "Quantum-inspired algorithms in practice," *arXiv preprint arXiv:1905.10415*, 2019.