

© 2023 Omid V. Heravi
ALL RIGHTS RESERVED

Quantum Machine Learning on Classical Data

Omid Vakili Heravi

Stony Brook University

03/30/2023

TABLE OF CONTENTS

Table of Contents	3
1 Introduction	1
1.1 Background on machine learning and quantum computing	1
1.2 Emergence and motivation for quantum machine learning	2
1.3 Scope of the paper: comparison of classical data and quantum-enhanced algorithm approaches	3
2 Quantum Machine Learning Fundamentals	5
2.1 Quantum computing concepts and terminologies	5
2.2 Classical machine learning concepts and terminologies	12
3 Quantum Machine Learning with Classical Data	13
3.1 Encoding Classical Data into Quantum States	14
3.2 Quantum Feature Maps and Kernel Functions	15
3.3 Suitability for Classical Data	15
3.4 Training Quantum Circuits	16
3.5 Optimizing Data and Quantum Circuits	17
3.6 Classical Data of Interest in QML	18
3.7 Classical and Quantum-inspired Optimizers	19
3.8 Quantum-Enhanced Machine Learning Model Examples	20
4 Conclusion	34
4.1 Challenges and Future Directions	34
Bibliography	37

CHAPTER 1

INTRODUCTION

1.1 Background on machine learning and quantum computing

Machine learning (ML) is a branch of artificial intelligence that focuses on developing algorithms that enable computers to learn from and make predictions or decisions based on data [7]. ML techniques are widely used in various applications, such as natural language processing, computer vision, and recommendation systems. Classical ML algorithms include supervised learning methods, such as support vector machines and neural networks, and unsupervised learning methods, such as clustering and dimensionality reduction.

Quantum computing is a paradigm that exploits the principles of quantum mechanics to perform computations more efficiently than classical computers in certain cases [8]. A quantum computer processes information using quantum bits or qubits, which can exist in a superposition of states, allowing simultaneous processing of multiple inputs, most importantly quantum computing takes advantage of the fact that the hilbert space is a large space which allows us to compute over an exponential number of instances in the number of bits. This property, along with entanglement and quantum gates, enables quantum algorithms to solve certain problems faster than classical algorithms. Prominent quantum algorithms include Shor's algorithm for integer factorization [11] and Grover's algorithm for unsorted database search [5].

Quantum machine learning (QML) aims to combine the advantages of quantum computing with machine learning techniques to develop more efficient and

powerful algorithms. QML can be applied to both classical data and quantum data, leveraging quantum-enhanced algorithms and quantum data processing methods, respectively [1]. In this paper, we will try to gain insights into their potential applications and limitations with emphasis on quantum-enhanced algorithms for classical data.

1.2 Emergence and motivation for quantum machine learning

The idea of quantum machine learning (QML) emerged from the desire to harness the unique properties of quantum computing to improve the efficiency and capabilities of classical machine learning algorithms [10]. Quantum computers have the potential to solve certain computational problems exponentially faster than classical computers, leading to significant speedup for specific machine learning tasks.

One of the primary motivations behind QML is to address the increasing computational complexity and resource requirements associated with processing large-scale datasets in various domains, such as genomics, finance, and social media analytics. By leveraging quantum algorithms, QML can provide faster training times, improved model accuracy, and better generalization for complex and high-dimensional data [2].

Furthermore, QML is also motivated by the need to analyze quantum data, which is generated from quantum systems and experiments, such as quantum sensors and simulators. Classical machine learning techniques may not be optimal for processing quantum data, and thus, quantum-enhanced algorithms and data processing methods are required for efficient analysis and interpretation of

such data [1].

1.3 Scope of the paper: comparison of classical data and quantum-enhanced algorithm approaches

In this paper, we aim to provide a comprehensive exposition on quantum-enhanced algorithms on classical data in quantum machine learning (QML). We will explore the underlying techniques, potential applications, and limitations of each approach, as well as the challenges and future research directions in the field [1].

First, we will delve into the fundamentals of quantum computing and quantum algorithms, providing a solid foundation for understanding the key concepts and terminologies relevant to QML. This will include an overview of qubits, quantum gates, quantum circuits, and prominent quantum algorithms such as Shor's and Grover's algorithms.

Next, we will discuss QML with classical data, examining quantum-enhanced algorithms designed to improve the efficiency and capabilities of classical machine learning methods, such as Quantum Support Vector Machines (QSVM), Quantum Principal Component Analysis (QPCA), and Quantum Neural Networks (QNNs) and etc. We will also explore the practical applications and limitations of these algorithms when applied to classical data, as well as the challenges in mapping classical data to quantum states and the trade-offs between speedup and accuracy.

Lastly, we will compare the similarities and differences between the tech-

niques, as well as their respective advantages and disadvantages. We will also discuss the performance of these approaches in terms of speedup, accuracy, and resource consumption, and outline the implications of our findings for the future development and application of QML in various domains [2].

Our goal is to provide a thorough understanding of classical data in QML and offer valuable insights for researchers, practitioners, and stakeholders interested in harnessing the potential of quantum computing for machine learning tasks.

CHAPTER 2

QUANTUM MACHINE LEARNING FUNDAMENTALS

2.1 Quantum computing concepts and terminologies

Quantum computing is an emerging field that exploits the principles of quantum mechanics to perform computations more efficiently than classical computers in certain cases [8]. In this section, we will introduce the fundamental concepts and terminologies of quantum computing, including qubits, quantum gates, quantum circuits, and prominent quantum algorithms.

Qubits

The term “qubit” was coined by Benjamin Schumacher, who created the term during a conversation with William Wootters in 1995. A quantum bit, or qubit, is the basic unit of quantum information. Unlike classical bits, which can only take the values of 0 or 1, qubits can exist in a superposition of both states, allowing simultaneous processing of multiple inputs. The standard representation of a qubit in quantum mechanics involves the use of orthonormal basis states or basis vectors. These basis states are typically denoted as $|0\rangle$ and $|1\rangle$ and written using Dirac notation or “bra-ket” notation. Thus a qubit is represented as a linear combination of the basis states $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.1}$$

When a qubit is measured in the standard basis, according to the Born rule,

the probabilities of obtaining outcome $|0\rangle$ with value "0" and outcome $|1\rangle$ with value "1" are given by $|\alpha|^2$ and $|\beta|^2$, respectively. The sum of the absolute squares of the probability amplitudes must equal 1 according to the second axiom of probability theory, expressed as $|\alpha|^2 + |\beta|^2 = 1$. It is the quantum counterpart of the classical binary bit and is realized physically with a two-state device. A qubit is a two-state or two-level quantum-mechanical system, representing one of the simplest quantum systems that exhibit the peculiarities of quantum mechanics. Examples of qubits include the spin of an electron, where the two levels are spin up and spin down, or the polarization of a single photon, where the two states are vertical polarization and horizontal polarization. Unlike a classical bit, which can only be in one state or the other, a qubit can exist in a coherent superposition of both states simultaneously. This property of superposition is fundamental to quantum mechanics and quantum computing. Unlike a classical bit, measuring a qubit would destroy its coherence and disturb the superposition state. A qubit can encode more information than a classical bit and has the potential to hold multiple bits of information using techniques such as superdense coding. In classical physics, a system of n components can be described using n bits, while in quantum physics, it requires 2^n complex numbers or a point in a 2^n -dimensional vector space. When measured, a qubit collapses to either $|0\rangle$ or $|1\rangle$ with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively.

A note for the reader, "qubits" are often realized physically with a two-state device, namely what it means is that a physical system is used to represent and manipulate the qubit's states. The types of these physical devices can vary significantly and there are many proposed solutions for this, but for purposes of brevity, we'll only mention a few; such as trapped ions, superconducting circuits, or quantum dots, different physical properties are leveraged to represent

the two states of a qubit. For example, in the case of a superconducting qubit, the two states can be encoded in the energy levels of a superconducting circuit. The lowest energy state corresponds to one qubit state (e.g., $|0\rangle$), and an excited energy state corresponds to the other qubit state (e.g., $|1\rangle$). The manipulation and control of the superconducting circuit allow for the coherent superposition and entanglement of qubits.

Quantum gates

Quantum gates are the fundamental operations that manipulate qubits in a quantum computer. which are square matrices that satisfy the condition $U^\dagger U = U U^\dagger = I$, where U^\dagger is the conjugate transpose of U and I is the identity matrix. This condition ensures that the matrix is reversible and preserves the norm of the quantum state.

The reversibility condition is a key advantage of quantum computing over classical computing, where computations are irreversible and information is lost as it is processed. In reversible computation, the inputs and outputs have a one-to-one correspondence, which means that the input bits can be restored from the output bits. This property is useful for certain types of computations, such as cryptography and error correction.

It is also important to note that this condition is one that is pre-mandated by nature itself, i.e. it is the second postulate of quantum mechanics which requires that the transformations between quantum states must be unitary, must be described by a unitary operator, which is a reversible operation or a matrix in the formal sense, thus why reversibility is forced onto this format of computation.

Moreover, in practice however, it is almost considered to be trivially true because any operation could be modified very simply to be reversible, so it is not much of a huge requirement when it comes to designing quantum algorithms to be very explicitly reversible on paper.

The unitary matrix U can be decomposed into a product of elementary gates, which are single-qubit gates and two-qubit gates. This decomposition is useful for implementing quantum algorithms on existing quantum computers, as it allows for the translation of bigger unitary gates into elementary quantum operations.

Some common quantum gates include:

- **Pauli gates:** The X , Y , and Z gates, which perform rotations around the corresponding axes of the Bloch sphere:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

are a set of three single-qubit gates that correspond to rotations around the x , y , and z axes of the Bloch sphere by π radians. The X gate is equivalent to a classical NOT gate, while the Y and Z gates are equivalent to rotations around the y and z axes, respectively. The three operators mentioned above can be seen in the figure 2.1.

- **Hadamard gate (H):** This gate creates a superposition of basis states and performs a rotation around the axis located halfway between the X and Z axes:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

is a single-qubit gate that creates an equal superposition of the $|0\rangle$ and $|1\rangle$ states when applied to a $|0\rangle$ state. It corresponds to a rotation around the x-axis of the Bloch sphere by $\pi/2$ radians, followed by a rotation around the y-axis by π radians. The Hadamard gate is used in many quantum algorithms, including the famous quantum Fourier transform.

- **Controlled-NOT gate (CNOT):** A two-qubit gate that applies an X gate to the second qubit if the first qubit is in the state $|1\rangle$, and does nothing otherwise. The matrix representation is:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

is a two-qubit gate that flips the target qubit if the control qubit is in the $|1\rangle$ state. It is a fundamental gate in quantum computing and is used to entangle and disentangle qubits. The CNOT gate, together with single-qubit rotations, can approximate any other unitary operation on a finite number of qubits.






				
$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
(a)	(b)	(c)	(d)	(e)

Figure 2.1: Corresponding circuit symbols of the operations

Its important to note that these three gates (Pauli gates, Hadamard, CNOT) form a universal set of gates, which is a theorem which states that universality of

these three sets of gates is enough to approximate any other quantum operation to arbitrary accuracy using only these 5 gates.

Quantum circuits

Quantum circuits are a sequence of quantum gates applied to a set of qubits, representing a quantum computation. Quantum circuits can be represented using diagrams, where each horizontal line represents a qubit, and boxes or symbols along the lines represent the gates acting on the qubits. Time flows from left to right in these diagrams. The following is an example of a Quantum Circuit:

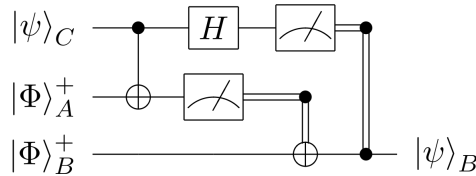


Figure 2.2: Sketch of the Quantum Teleportation Circuit, From Wikipedia

Quantum entanglement

Quantum entanglement is a phenomenon in which two or more qubits become correlated in such a way that the state of one qubit cannot be described independently of the state of the other qubits.

Mathematically, quantum entanglement can be represented by a state vector in the tensor product space of two qubits. If we have two qubits, A and B, the state of the system is represented by a vector in the tensor product space of A and B. The most general state of the system can be written as:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \kappa|10\rangle + \gamma|11\rangle$$

where $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ are the basis states of the two-qubit system, and α , β , γ , and δ are complex numbers. Thus system is said to be entangled if it cannot be factored into a product of two single-qubit states. A common example of an entangled state is the Bell state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \quad (2.2)$$

Physically, entanglement can be created using quantum gates. For example, we can create the Bell state $|\Phi^+\rangle$ from the initial state $|00\rangle$ by first applying a Hadamard gate to the first qubit, which puts it into a superposition of states, and after which applying the controlled-not (CNOT) gate to the new state, which flips the second qubit if the first qubit is in state $|1\rangle$.

The advantage of entanglement in quantum circuits is that it allows for quantum correlations that cannot be achieved with classical bits. This is what enables the power of quantum algorithms, such as Shor's algorithm for factoring large numbers and Grover's algorithm for searching unsorted databases. Additionally, entanglement is the key resource for quantum teleportation and quantum key distribution, which are important for quantum communication.

However, creating and maintaining entanglement in a real physical system is a major challenge due to decoherence, which is the loss of quantum coherence due to interactions with the environment. Overcoming this challenge is one of the main goals of current research in quantum computing and quantum information science.

Quantum algorithms

Quantum algorithms exploit the unique properties of quantum computing, such as superposition and entanglement, to solve problems more efficiently than classical algorithms. Some prominent quantum algorithms include:

- **Shor's algorithm:** An algorithm for integer factorization that can break cryptographic schemes based on the hardness of factoring large numbers, such as RSA [11].
- **Grover's algorithm:** A quantum search algorithm that provides a quadratic speedup over classical unsorted database search algorithms, reducing the number of required queries from $O(N)$ to $O(\sqrt{N})$ [5].
- **Quantum phase estimation:** An algorithm used to estimate the eigenvalues of a unitary operator, which is a key subroutine in many quantum algorithms, such as Shor's algorithm and quantum simulation algorithms [8].

2.2 Classical machine learning concepts and terminologies

CHAPTER 3

QUANTUM MACHINE LEARNING WITH CLASSICAL DATA

We preface this chapter with an important note; namely that we should be aware that there are no comprehensive theory of quantum learning yet. Moreover, Classical Machine Learning is a subject in it's own that requires much mathematical expertise and the discussions of how quantum learning theory can help achieve certain advantages is theoretically and practically speaking, an "unprovable" task for most claims of "quantum advantage" in QML. Thus the results and ideas that are discussed in the following chapter are for most part either experimental observations and or theoretical claims at best, but for the most part, the research in QML consists of results that are not exact mathematical proofs that claim anything with certainty.

Following the remarks above, a theory of quantum learning would refer to methods of quantum information processing that learn input-output relations from training input, either for the optimisation of system parameters (i.e. unitary operators) or etc. There are many open questions of how an efficient quantum learning procedure would look like. That is; how can we efficiently implement an optimisation problem (that is usually solved by iterative and dissipative methods such as gradient descent) on a coherent and thus reversible quantum computer? How can we translate and process important structural information, such as distance metrics, using quantum states without losing the data from the quantum state due to coherence and etc? How do we formulate the data into a quantum state as an input? And thus the overall question, is there a general way how quantum physics can in principle speed up Quantum-enhanced algorithms for classical data so that we can exploit quan-

tum computing techniques to accelerate the processing and analysis of classical data. In essence, these algorithms boil down to deciding how to encode the input which is classical data into quantum states, which prediction models to choose, calculate the error from our prediction and update the appropriate parameters, and lastly how to utilize quantum computing and its peculiarities to perform “better” machine learning as opposed to classical machine learning, hence what, if any, “quantum advantage” might be achieved. Some of the notable quantum-enhanced algorithms for classical data include quantum support vector machines, quantum principal component analysis, quantum kernel methods, and quantum neural networks.

3.1 Encoding Classical Data into Quantum States

A crucial step in quantum-enhanced algorithms for classical data is the encoding of classical data into quantum states. This process, known as quantum state preparation or data encoding, transforms classical data points \vec{x} into quantum states $|\phi(\vec{x})\rangle$ using a suitable encoding scheme or feature map. The choice of the encoding scheme is essential, as it determines the efficiency of the quantum algorithm and the accuracy of the resulting quantum model.

There are several approaches to encoding classical data into quantum states, including amplitude encoding, angle encoding, and qubit-wise encoding. Amplitude encoding encodes classical data points as amplitudes of a quantum state, while angle encoding maps classical data to rotation angles of quantum gates. Qubit-wise encoding represents classical data as binary strings and maps each bit to a separate qubit. The choice of the encoding scheme depends on the problem at hand and the properties of the quantum algorithm being used.

3.2 Quantum Feature Maps and Kernel Functions

Quantum feature maps are used to transform classical data points into high-dimensional quantum states, which can then be used to compute kernel functions or other similarity measures. A quantum feature map $\phi(\vec{x})$ is a function that maps classical data points \vec{x} to quantum states $|\phi(\vec{x})\rangle$. The choice of the feature map is essential, as it determines the expressiveness of the quantum model and the computational complexity of the quantum algorithm.

In quantum kernel methods, the goal is to compute the inner product of the quantum states corresponding to two classical data points, which serves as the kernel function:

$$K(\vec{x}, \vec{y}) = \langle \phi(\vec{x}) | \phi(\vec{y}) \rangle \quad (3.1)$$

It is important for the feature map to be hard to approximate classically to ensure that the quantum algorithm provides a genuine advantage over classical methods. If a feature map can be efficiently approximated using classical techniques, the quantum algorithm would not offer significant speedup or improvements in accuracy. We will be diving into this topic more later in the paper.

3.3 Suitability for Classical Data

Quantum-enhanced algorithms for classical data are well-suited for problems where the computational complexity of classical methods becomes prohibitive, such as large-scale datasets or complex models. By leveraging the unique

properties of quantum computing, such as superposition and entanglement, quantum-enhanced algorithms can potentially provide significant speedups and improvements in accuracy over their classical counterparts. That is to say; Since the volume of globally stored data is supposed to be growing by around 20 percent year over year, the pressure to find innovative approaches to machine learning is rising.

However, the suitability of quantum-enhanced algorithms for classical data also depends on the specific problem at hand and the availability of efficient quantum algorithms and hardware. In some cases, classical methods may still be more practical or efficient, particularly when dealing with small-scale problems or problems with simple structures. It is essential to carefully consider the trade-offs between quantum and classical methods when designing quantum-enhanced algorithms for classical data.

3.4 Training Quantum Circuits

The training of quantum circuits in quantum-enhanced algorithms for classical data is typically performed using a combination of quantum and classical optimization techniques. The goal is to find a set of optimal parameters for the quantum circuit that minimizes a cost function, which is often a measure of the error between the predictions of the quantum model and the true target values.

In the case of variational quantum algorithms, the quantum circuit is parameterized by a set of adjustable parameters $\vec{\theta}$. During the training process, the quantum circuit is executed on a quantum computer or a quantum simulator to compute the cost function for a given set of parameters. The gradients

of the cost function with respect to the parameters are then estimated using either classical or quantum techniques, such as the parameter shift rule or the Hadamard test. The gradients are used to update the parameters using a classical or quantum-inspired optimization algorithm, and the process is repeated until convergence.

3.5 Optimizing Data and Quantum Circuits

Optimizing data and quantum circuits involves finding the best encoding scheme, feature map, and quantum circuit architecture for a given problem. This can be achieved by systematically exploring different combinations of encoding schemes, feature maps, and quantum circuit architectures, and evaluating their performance on a validation set.

One approach to optimizing data and quantum circuits is to use techniques from classical machine learning, such as hyperparameter optimization, model selection, and feature selection. These techniques can be adapted to the quantum setting by treating the encoding scheme, feature map, and quantum circuit architecture as hyperparameters or features to be optimized.

Another approach is to use quantum-inspired optimization algorithms, such as the Quantum Approximate Optimization Algorithm (QAOA) or the Variational Quantum Eigensolver (VQE), to search the space of possible encoding schemes, feature maps, and quantum circuit architectures more efficiently than classical methods. These algorithms can potentially provide significant speedups over classical optimization techniques, particularly when dealing with large-scale problems or complex quantum circuit architectures.

3.6 Classical Data of Interest in QML

In quantum machine learning (QML), one may be interested in working with classical data that exhibits one or more of the following characteristics:

1. High-dimensional data: Quantum computing can potentially provide exponential speedups when working with high-dimensional data, as quantum states can efficiently represent high-dimensional vectors or tensors or with use of novel techniques like QRAM.
2. Complex data structures: Quantum computing can be advantageous when working with complex data structures, such as graphs or trees, as quantum algorithms can efficiently perform operations on these structures, with use of proposed state of art models such as Hidden quantum Markov models and or those from Geometric Quantum Machine Learning which take advantage of the inherent symmetry in the data.
3. Computationally intensive tasks: Quantum computing can potentially provide significant speedups for computationally intensive tasks, such as optimization, sampling, or solving linear systems of equations, which are common in machine learning.
4. Data with intricate patterns or correlations: Quantum computing can potentially capture and exploit intricate patterns or correlations in classical data more effectively than classical methods, thanks to the unique properties of quantum states and operations.

3.7 Classical and Quantum-inspired Optimizers

When training quantum circuits, one can use classical or quantum-inspired optimizers to update the parameters of the circuit. Some popular classical optimizers include gradient descent, stochastic gradient descent (SGD), and their variants, such as Adam, RMSprop, and Adagrad. These optimizers are widely used in classical machine learning and can be easily adapted to the quantum setting.

In addition to classical optimizers, there are also quantum-inspired optimizers that leverage the properties of quantum computing to explore the parameter space more efficiently. Some examples of quantum-inspired optimizers include the Quantum Natural Gradient Descent (QNGD) and the Rotoselect algorithm. Quantum Natural Gradient Descent (QNGD) is an optimization technique that incorporates the geometry of the quantum state space into the optimization process. By taking into account the non-Euclidean geometry of the quantum state space, QNGD can potentially lead to more efficient and robust optimization of quantum circuits [12].

The Rotoselect algorithm, on the other hand, is a gradient-free optimization technique designed specifically for parameterized quantum circuits [9]. The algorithm iteratively searches for the optimal rotation gates in the quantum circuit by exploring different gate angles and evaluating their performance on the training data. The Rotoselect algorithm is particularly well-suited for problems where the gradient of the cost function is difficult to compute or noisy.

3.8 Quantum-Enhanced Machine Learning Model Examples

Quantum-enhanced algorithms for classical data have been applied to various machine learning tasks, such as classification, regression, clustering, dimensionality reduction, and feature selection. Some notable examples include:

1. Quantum Kernel Maps:

Classical Kernel Methods

In classical kernel methods, we map input data to a high-dimensional feature space using a kernel function. Given two data points $x, y \in \mathbb{R}^d$, a kernel function $k(x, y)$ computes the inner product in the feature space between the images of x and y , in essence it is computing the similarity between two data points, similar to a distance function in the new feature space.

Kernel functions have the property that $k(x, y) = \langle \phi(x), \phi(y) \rangle$, where $\phi : \mathbb{R}^d \rightarrow \mathbb{H}$ is a mapping from the input space to the feature space \mathbb{H} , and $\langle \cdot, \cdot \rangle$ denotes the inner product in \mathbb{H} . The kernel trick then involves performing computations in the feature space implicitly, without ever having to compute the mapping $\phi(x)$ explicitly.

Quantum Kernel Methods

In Quantum Kernel Methods, we extend the concept of kernel methods to quantum data. We consider a set of quantum states $|\psi_i\rangle$, where each $|\psi_i\rangle$ is a state in a quantum Hilbert space \mathcal{H} . A quantum kernel function $k(|\psi_x\rangle, |\psi_y\rangle)$ then computes the inner product in a quantum feature space between the images of $|\psi_x\rangle$ and $|\psi_y\rangle$.

The quantum kernel function has the property that $k(|\psi_x\rangle, |\psi_y\rangle) = |\langle \phi(\psi_x) | \phi(\psi_y) \rangle|^2$, where $\phi : \mathcal{H} \rightarrow \mathcal{H}'$ is a quantum feature map that maps quantum states in \mathcal{H} to quantum states in another quantum Hilbert space \mathcal{H}' . The quantum kernel trick involves performing computations in the quantum feature space implicitly, without ever having to compute the quantum feature map $\phi(\psi)$ explicitly.

The computation of the quantum kernel function can be performed on a quantum computer by preparing the states $|\phi(\psi_x)\rangle$ and $|\phi(\psi_y)\rangle$, and then measuring their overlap. This process can potentially provide exponential speedups over classical methods, as the dimension of the quantum feature space is exponential in the number of qubits.

However, the implementation of Quantum Kernel Methods on current quantum computers faces several challenges, such as the preparation of the quantum states $|\phi(\psi_x)\rangle$ and $|\phi(\psi_y)\rangle$, and the measurement of their overlap. In addition, the design of effective quantum feature maps is a non-trivial task and is an active area of research.

2. Quantum Support Vector Machines (QSVM)

The Quantum Support Vector Machine (QSVM) is a quantum variant of the classical Support Vector Machine (SVM), which is a popular supervised learning algorithm for classification and regression. The QSVM extends the SVM to quantum data by using quantum feature maps and quantum algorithms.

Classical SVM

A classical SVM finds the hyperplane that maximizes the margin between

two classes in a feature space. Given a training set of instance-label pairs (x_i, y_i) , where $x_i \in \mathbb{R}^n$ and $y_i \in \{1, -1\}$, the SVM solves the following optimization problem:

$$\begin{aligned}
& \underset{w, b, \xi}{\text{minimize}} && \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \\
& \text{s.t.} && y_i(w \cdot x_i - b) \geq 1 - \xi_i, \quad i = 1, \dots, N \\
& && \xi_i \geq 0, \quad i = 1, \dots, N
\end{aligned} \tag{3.2}$$

Here, w is the normal vector to the hyperplane, b is the bias term, ξ_i are the slack variables, and C is a regularization parameter.

Quantum SVM

The Quantum SVM performs a similar operation but in a high-dimensional Hilbert space defined by a quantum feature map. Given a quantum feature map $\Phi : x \mapsto |\phi(x)\rangle$, where $|\phi(x)\rangle$ is a state in a quantum Hilbert space \mathcal{H} , note that the initial state can be constructed by using a QRAM, thus then the QSVM finds the quantum state $|w\rangle \in \mathcal{H}$ that maximizes the margin between the two classes.

The quantum kernel is defined as $K(x, z) = |\langle \phi(x) | \phi(z) \rangle|^2$, which is the squared distance between the quantum states corresponding to x and z . This kernel can be estimated on a quantum computer using the Swap Test.

The QSVM then solves a similar optimization problem as the classical SVM, but in the quantum feature space:

$$\begin{aligned}
& \underset{\alpha}{\text{minimize}} && \frac{1}{2} \alpha^T K \alpha - 1^T \alpha \\
& \text{s.t.} && 0 \leq \alpha_i \leq C, \ i = 1, \dots, N \\
& && y^T \alpha = 0
\end{aligned} \tag{3.3}$$

Here, α are the Lagrange multipliers, K is the kernel matrix, and 1 is a vector of ones. The decision function is given by $f(x) = \sum_{i=1}^N \alpha_i y_i K(x_i, x) + b$.

In this way, the QSVM extends the SVM to quantum data by leveraging the computational power of quantum computers to process high-dimensional feature spaces.

In practice, however, there are challenges in implementing QSVMs due to the noise and limited qubit connectivity in current quantum devices. One common approach to mitigate these issues is to use variational quantum algorithms, which combine classical and quantum computation and are more robust to noise.

A *Variational Quantum Support Vector Machine* (VQSVM) is one such algorithm. It uses a variational circuit to prepare the quantum state $|\phi(x)\rangle$, and a classical optimizer to train the parameters of the circuit. The VQSVM solves the following optimization problem:

$$\begin{aligned}
& \underset{\theta}{\text{minimize}} && \frac{1}{2} \theta^T K(\theta) \theta - 1^T \theta \\
& \text{s.t.} && 0 \leq \theta_i \leq C, \ i = 1, \dots, N \\
& && y^T \theta = 0
\end{aligned} \tag{3.4}$$

Here, θ are the parameters of the variational circuit, and $K(\theta)$ is the kernel matrix computed using the variational circuit. The decision function is given by

$$f(x) = \sum_{i=1}^N \theta_i y_i K(x_i, x; \theta) + b.$$

The VQSVM has the advantage of being more flexible than the QSVM, as the variational circuit can be tailored to the specific data and task. However, it also has the disadvantage of requiring more classical computation and more quantum-classical feedback, which can be slow and error-prone.

In conclusion, Quantum Support Vector Machines extend the powerful SVM framework to quantum data, opening up new possibilities for machine learning on quantum computers. However, they also bring new challenges and trade-offs, and their practical implementation and performance are active areas of research.

QSVMs can provide significant speedup over classical SVMs for certain problems, particularly when the dataset is large and the kernel function is computationally expensive. More specifically; support vector machines are in fact part of a larger class of so called kernel methods that suffer from the fact that calculating kernels can get very expensive in terms of computational resources. More precisely, quadratic programming problems of this form have a complexity of $\Omega((n)^3)$, and computational resources therefore grow significantly with the size of the training data. It is thus crucial for support vector machines to find a method of evaluating an inner product efficiently. This is where quantum computing comes into play. Rebentrost, Mohseni and Lloyd claim that in general, the evaluation of an inner product can be done faster on a quantum computer.

3. Quantum Principal Component Analysis (QPCA)

Principal Component Analysis (PCA) is a statistical technique widely used in the fields of machine learning and data science. It applies an orthogonal transformation to transform a set of possibly correlated variables into a set of linearly uncorrelated variables, which are the principal components. Quantum Principal Component Analysis (QPCA) is the quantum analogue of this technique, designed to operate on quantum data.

Classical PCA

In classical PCA, we start with a data set $X = \{x_1, x_2, \dots, x_N\}$, where each $x_i \in \mathbb{R}^d$ is a d -dimensional vector. The objective of PCA is to find a new set of basis vectors for this data set that are uncorrelated and that capture the most variance in the data.

The first step in PCA is to center the data by subtracting the mean vector $\mu = \frac{1}{N} \sum_{i=1}^N x_i$ from each data point. Then, we compute the covariance matrix of the centered data, which is given by $C = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)(x_i - \mu)^T$.

The principal components of X are the eigenvectors of the covariance matrix C , and they are typically ordered by their corresponding eigenvalues in decreasing order, as the eigenvalues represent the variance of the data in the direction of the corresponding eigenvector. Thus, the k -th principal component is the eigenvector corresponding to the k -th largest eigenvalue.

Quantum PCA

In Quantum PCA, we consider a set of quantum states $|\psi_i\rangle$, where each $|\psi_i\rangle$ is a state in a quantum Hilbert space \mathcal{H} . Instead of classical vectors, we now deal with quantum states, and we want to find a new set of basis states that are

uncorrelated and capture the most variance in the quantum data.

The first step in QPCA is similar to classical PCA: we compute the quantum mean state $|\mu\rangle = \frac{1}{N} \sum_{i=1}^N |\psi_i\rangle$, and then we compute the quantum covariance matrix, which is given by $\rho = \frac{1}{N} \sum_{i=1}^N (|\psi_i\rangle - |\mu\rangle)(\langle\psi_i| - \langle\mu|)$.

The quantum principal components are then the eigenvectors of the quantum covariance matrix ρ , ordered by their corresponding eigenvalues in decreasing order. However, due to the exponential size of the quantum covariance matrix in the number of qubits, computing its eigenvectors directly on a quantum computer is infeasible.

To overcome this, QPCA typically uses a quantum algorithm to estimate the eigenvectors and eigenvalues of ρ . One such algorithm is the Quantum Phase Estimation (QPE) algorithm. In the QPE algorithm, we prepare a superposition of the states $|\psi_i\rangle$, apply the phase estimation on the quantum covariance matrix ρ , and then measure the system. The outcome of the measurement gives an estimate of the eigenvectors and eigenvalues of ρ .

It is important to note that QPCA, as a quantum algorithm, can in theory provide exponential speedup over classical PCA due to the inherent parallelism and high-dimensionality of quantum systems. However, the actual implementation of QPCA on current quantum computers faces several challenges.

However there are some technical challenges and Limitations to bear in mind. That is; the first challenge is the preparation of the initial quantum state. In QPCA, we need to prepare a superposition of the states $|\psi_i\rangle$, which can be difficult to do in practice, especially for large and complex data sets.

The second challenge is the execution of the Quantum Phase Estimation (QPE) algorithm. QPE requires controlled-U gates, where U is the unitary evolution operator associated with the quantum covariance matrix ρ . However, implementing controlled-U gates for arbitrary U is a non-trivial task on current quantum computers, which typically have limited gate sets and connectivity.

The third challenge is the measurement process. In QPE, the output is a quantum state that encodes the eigenvectors and eigenvalues of ρ , and we need to perform a measurement to extract this information. However, quantum measurements are probabilistic and can only provide an estimate of the eigenvectors and eigenvalues. Furthermore, due to the no-cloning theorem of quantum mechanics, we cannot copy the output state to perform multiple measurements, so we need to repeat the whole QPCA algorithm multiple times to get reliable estimates.

Despite these challenges, QPCA represents a promising direction for quantum machine learning. With the rapid development of quantum technologies, it is expected that these challenges will be gradually overcome, and that QPCA will become a powerful tool for analyzing quantum data. In particular, QPCA has the potential to enable new applications in quantum chemistry and quantum information theory, where we often deal with high-dimensional quantum states.

In conclusion, Quantum Principal Component Analysis extends the powerful PCA framework to quantum data, opening up new possibilities for data analysis on quantum computers. However, it also brings new challenges and trade-offs, and its practical implementation and performance are active areas of research. As we continue to develop and refine our quantum technologies, we

can expect to see more advances and

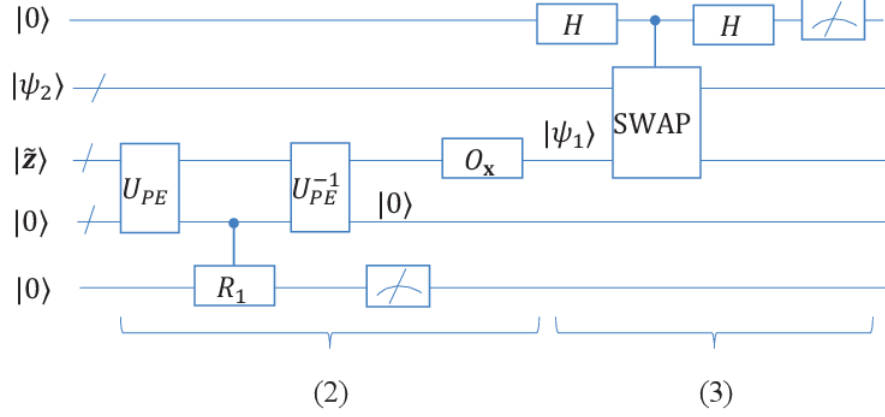


Figure 3.1: Quantum Circuit to perform Principal Component Analysis [13]

4. **Quantum Neural Networks (QNNs):** Quantum Neural Networks (QNNs) are an extension of classical neural networks to the quantum domain, aiming to harness the power of quantum computing to perform more efficient and accurate learning tasks [4]. QNNs consist of layers of quantum gates that manipulate qubits to perform complex operations, analogous to the layers of neurons in classical neural networks.

Training a QNN typically involves optimizing the parameters of the quantum gates to minimize a cost function, which can be done using classical optimization algorithms, such as gradient descent, or quantum optimization algorithms, such as the Quantum Approximate Optimization Algorithm (QAOA) [3].

QNNs have the potential to provide speedup and improved generalization

capabilities compared to classical neural networks, especially for tasks that involve high-dimensional and complex data structures.

5. Quantum Variational Algorithms: Quantum Variational Algorithms are a class of hybrid quantum-classical algorithms that can be used for optimization and machine learning tasks. They typically involve a parametrized quantum circuit, also known as an ansatz, and a classical optimization routine. The Variational Quantum Eigensolver (VQE) [?] and the Quantum Approximate Optimization Algorithm (QAOA) [3] are two notable examples of quantum variational algorithms.

The general structure of a quantum variational algorithm can be described as follows:

1. Prepare the initial state of the quantum system, typically denoted as $|\psi_0\rangle$.
2. Apply the parametrized quantum circuit (ansatz) $U(\vec{\theta})$ to the initial state: $|\psi(\vec{\theta})\rangle = U(\vec{\theta})|\psi_0\rangle$, where $\vec{\theta}$ represents the parameters of the ansatz.
3. Measure the expectation value of an observable O (usually related to the cost function) on the final state: $\langle O(\vec{\theta}) \rangle = \langle \psi(\vec{\theta}) | O | \psi(\vec{\theta}) \rangle$.
4. Update the parameters of the ansatz using a classical optimization algorithm to minimize the expectation value: $\vec{\theta}_{\text{opt}} = \underset{\vec{\theta}}{\text{argmin}} \langle O(\vec{\theta}) \rangle$.
5. Repeat steps 2-4 until the desired level of convergence is achieved.

Quantum variational algorithms are particularly well-suited for near-term

quantum devices due to their relatively low requirements on quantum coherence time and error rates. They have been applied to various machine learning tasks, including supervised learning, unsupervised learning, and reinforcement learning [?].

6. Variational Quantum Eigensolver VQE: The Variational Quantum Eigensolver (VQE) is a quantum variational algorithm designed to find the ground state energy of a given Hamiltonian H [?]. The VQE algorithm aims to minimize the expectation value of the Hamiltonian subject to the constraint that the quantum state $|\psi(\vec{\theta})\rangle$ has a fixed number of qubits:

$$E_{\min} = \min_{\vec{\theta}} \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle. \quad (3.5)$$

VQE has been applied to a variety of quantum chemistry problems and can be adapted for use in machine learning tasks, such as solving linear systems of equations and optimizing high-dimensional cost functions.

7. Quantum Approximate Optimization Algorithm: The Quantum Approximate Optimization Algorithm (QAOA) is another quantum variational algorithm designed for combinatorial optimization problems [3]. Given a cost function $C(z)$ that depends on a set of binary variables z , the QAOA algorithm

aims to find the optimal assignment of the variables that minimizes the cost function:

$$z_{\text{opt}} = \underset{z}{\operatorname{argmin}} C(z). \quad (3.6)$$

The QAOA consists of two main steps:

1. Construct a quantum state $|\gamma, \beta\rangle$ by applying a sequence of parametrized unitary operators to an initial state $|s\rangle$:

$$|\gamma, \beta\rangle = e^{-i\beta_p B} e^{-i\gamma_p C} \dots e^{-i\beta_1 B} e^{-i\gamma_1 C} |s\rangle, \quad (3.7)$$

where B and C are Hermitian operators derived from the cost function, and γ_i and β_i are variational parameters.

2. Measure the expectation value of the cost function with respect to the quantum state:

$$\langle C(\gamma, \beta) \rangle = \langle \gamma, \beta | C | \gamma, \beta \rangle. \quad (3.8)$$

The goal is to optimize the variational parameters γ and β to minimize the expectation value of the cost function:

$$(\gamma_{\text{opt}}, \beta_{\text{opt}}) = \underset{\gamma, \beta}{\operatorname{argmin}} \langle C(\gamma, \beta) \rangle. \quad (3.9)$$

Once the optimal parameters are found, the solution to the combinatorial optimization problem can be extracted from the optimized quantum state.

QAOA has been applied to various combinatorial optimization problems, such as the Max-Cut problem, the Traveling Salesman problem, and the Quadratic Unconstrained Binary Optimization (QUBO) problem. It can also be adapted for machine learning tasks, such as clustering and classification.

8. HHL Algorithm : The HHL algorithm is a quantum algorithm for solving linear systems of equations of the form $Ax = b$, where A is a sparse $N \times N$ matrix and b is a known vector [6]. The algorithm has a runtime of $O(\kappa^2 \log N)$, where κ is the condition number of the matrix A . This represents an exponential speedup compared to the best classical algorithms, which have a runtime of $O(N\kappa)$.

In machine learning, solving linear systems of equations is a common task that arises in various contexts, such as linear regression, kernel methods, and support vector machines. By leveraging the HHL algorithm, it is possible to develop quantum machine learning algorithms that offer significant speedups over their classical counterparts, particularly when dealing with large-scale problems.

The main steps of the HHL algorithm are as follows:

1. Encode the vector b into a quantum state $|b\rangle$ using a quantum algorithm for state preparation.
2. Perform a quantum phase estimation to compute the eigenvalues of the matrix A .
3. Apply a controlled rotation to encode the inverse of the eigenvalues into the corresponding eigenvectors.
4. Perform an inverse quantum phase estimation to obtain the state $|x\rangle = A^{-1}|b\rangle$.
5. Measure the quantum state $|x\rangle$ and decode the solution vector x .

Note that the HHL algorithm assumes that the matrix A is sparse, Hermitian, and has a known condition number κ . In practice, these assumptions might not always hold, but various techniques can be employed to transform a given linear system into a suitable form for the HHL algorithm.

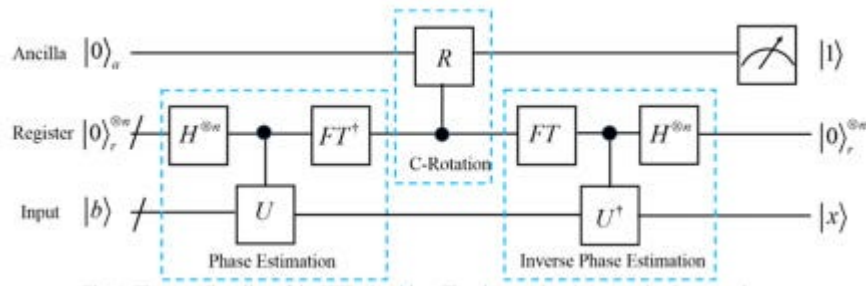


Figure 3.2: Circuit Diagram of The HHL Algorithm [14]

CHAPTER 4

CONCLUSION

4.1 Challenges and Future Directions

Despite the potential advantages of quantum machine learning algorithms, there are several limitations and challenges that need to be addressed to achieve practical quantum advantages, particularly in the context of near-term quantum computing capabilities. These limitations include:

1. **Speedup:** The speedup provided by quantum machine learning algorithms over their classical counterparts is often problem-dependent and can be influenced by factors such as the data distribution, the complexity of the algorithm, and the efficiency of the quantum state preparation and encoding schemes. In some cases, the theoretical speedup provided by quantum algorithms may not translate into a practical advantage due to the overhead of quantum state preparation, data encoding, or error correction. Furthermore, recent studies have shown that, for certain quantum machine learning algorithms, the speedup might be limited by the classical data processing capabilities or the quantum-classical communication complexity [?].

2. **Accuracy:** Quantum machine learning algorithms need to balance the trade-off between accuracy and resource consumption. Due to the limited number of qubits and gates available on near-term quantum devices, the complexity of quantum circuits and feature maps may need to be reduced, which can affect the accuracy of the resulting models. Moreover, the presence of noise in near-term quantum devices can introduce errors in the computation, leading to

a decrease in the accuracy of the quantum machine learning models [?].

3. **Noise:** Near-term quantum devices, also known as Noisy Intermediate-Scale Quantum (NISQ) devices, are inherently noisy, which can have a significant impact on the performance and reliability of quantum machine learning algorithms. Errors introduced by gate and qubit imperfections, as well as decoherence, can accumulate throughout the computation, leading to incorrect or suboptimal results. Error mitigation techniques, such as error-correcting codes, noise-aware training, or post-processing methods, can help reduce the impact of noise but often come with additional resource requirements or computational overhead [?, ?].

4. **Resource consumption:** Quantum machine learning algorithms typically require a large number of qubits and gates to achieve a practical quantum advantage. However, the resources available on near-term quantum devices are limited, and scaling up the number of qubits and gates can be challenging due to the issues of noise, connectivity, and error correction. Moreover, the resource requirements of quantum machine learning algorithms can be affected by the choice of the data encoding scheme, feature map, and quantum circuit architecture, making it essential to develop efficient and scalable quantum algorithms and techniques that can exploit the limited resources of near-term quantum devices effectively [?, 1].

5. **Quantum-classical integration:** Quantum machine learning algorithms often involve a combination of quantum and classical processing steps, requiring efficient integration of quantum and classical hardware and software. The communication and data transfer between quantum and classical devices can introduce additional overhead and latency, which can affect the overall

performance of the quantum machine learning algorithms. Developing efficient quantum-classical interfaces and co-designing quantum and classical algorithms and techniques can help mitigate these challenges and enhance the practicality and performance of quantum machine learning algorithms [?].

To achieve a practical quantum advantage in machine learning, it is crucial to address these limitations and develop quantum algorithms and techniques that can exploit the unique capabilities of near-term quantum devices while mitigating their

BIBLIOGRAPHY

- [1] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.
- [2] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, and Simone Severini. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, 2018.
- [3] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Hartmut Neven. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.
- [4] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018.
- [5] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.
- [6] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.
- [7] Tom M Mitchell. *Machine learning*. McGraw-Hill, 1997.
- [8] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information: 10th anniversary edition*. Cambridge University Press, 2010.
- [9] Mieszko Ostaszewski, Edward Grant, and Marcello Benedetti. Quantum gradient descent algorithms. *arXiv preprint arXiv:1909.07621*, 2019.
- [10] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2014.
- [11] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134, 1994.
- [12] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum natural gradient. *Quantum*, 4:269, 2020.

- [13] Chao-Hua Yu, Fei Gao, Song Lin, and Jingbo Wang. Quantum data compression by principal component analysis - quantum information processing. *SpringerLink*, Jul 2019.
- [14] Xiong Zhang, Zhenwei Yang, and Xiangdong Zhang. Simplified experimental scheme of quantum algorithm for solving linear equations with single photons. *Opt. Express*, 27(3):3369–3378, Feb 2019.