2020

SPARSITY AND WEAK SUPERVISION IN QUANTUM MACHINE LEARNING

Seyran Saeedi
Virginia Commonwealth University

SPARSITY AND WEAK SUPERVISION IN QUANTUM MACHINE LEARNING

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy at Virginia Commonwealth University.

by

SEYRAN SAEEDI

Master of Science in Computer Science from University of Tehran, 2011

Bachelor of Science in Applied Mathematics from Tabriz University, 2007

Director: Dr. Tomasz Arodz,

Associate Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

October, 2020

TABLE OF CONTENTS

List of Algorithms

LIST OF FIGURES

iv

Abstract

SPARSITY AND WEAK SUPERVISION IN QUANTUM MACHINE LEARNING

By Seyran Saeedi

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2020.

Director: Dr. Tomasz Arodz,
Associate Professor, Department of Computer Science

Quantum computing is an interdisciplinary field at the intersection of computer
science, mathematics, and physics that studies information processing tasks on a quan-
tum computer. A quantum computer is a device whose operations are governed by the
laws of quantum mechanics. As building quantum computers is nearing the era of com-
mercialization and quantum supremacy, it is essential to think of potential applications
that we might benefit from. Among many applications of quantum computation, one
of the emerging fields is quantum machine learning. We focus on predictive models
for binary classification and variants of Support Vector Machines that we expect to be
especially important when training data becomes so large that a quantum algorithm
with a guaranteed speedup becomes useful. We present a quantum machine learning
algorithm for training Sparse Support Vector Machine for problems with large datasets
that require a sparse solution. We also present the first quantum semi-supervised al-
gorithm, where we still have a large dataset, but only a small fraction is provided with
labels. While the availability of data for training machine learning models is steadily
increasing, oftentimes it is much easier to collect feature vectors to obtain the corre-
sponding labels. Here, we present a quantum machine learning algorithm for training

Semi-Supervised Kernel Support Vector Machines. The algorithm uses recent advances in quantum sample-based Hamiltonian simulation to extend the existing Quantum LS-SVM algorithm to handle the semi-supervised term in the loss while maintaining the same quantum speedup as the Quantum LS-SVM.

CHAPTER 1

INTRODUCTION

The theory of computation has traditionally been studied almost entirely in the abstract, as a topic in pure mathematics. This is to miss the point of it. Computers are physical objects, and computations are physical processes. What computers can or cannot compute is determined by the laws of physics alone, and not by pure mathematics.

— David Deutsch [1].

Computers are physical devices for processing information by executing algorithms. Algorithms are finite steps for performing well-defined procedures that realize an information-processing task. All information-processing tasks on computers are eventually transformed into a physical task. For example, on a classical computer, we perform information processing by operating logical gates on bits using transistors as gates (e.g. AND and NOT) and encoding bits with voltages. Quantum computers replace the bits with quantum 2-dimensional systems (qubits), the logical gates with a set of unitary operations (such as Pauli operators, Hadamard, C-NOT), and replace read-out with quantum measurement.

Quantum computing is an interdisciplinary field at the intersection of computer science, mathematics, and physics that studies information processing tasks on a quantum computer. A quantum computer is a device whose operations are governed by the laws of quantum mechanics. In the absence of reliable large-scale quantum computers, researchers use an abstract computing model that expresses our expectation of what a quantum device would be capable of doing, much like a Turing machine or a logical gates model captures what a classical processor can do. Quantum mechanics rules im-

ply a drastic departure from the classical setting that usually comes with the idea of fundamentally improving the speed and/or security of certain algorithms. For example, a classical bit can only take two discrete values, 0 and 1. A qubit not only takes $|0\rangle = (1,0)^T$ and $|1\rangle = (0,1)^T$ in 2-dimensional Euclidean complex space $\mathbb{C}^2$ (which correspond to the classical bits 0 and 1, respectively), but it also takes linear combinations $\alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$, and $|\alpha|^2 + |\beta|^2 = 1$. This is called *superposition*, a central ingredient in many quantum algorithms that plays *quantum parallelism* role. While superposition makes quantum computers sound stronger, quantum mechanics also imposes some restrictions. For example, copying and reading bits are eminently reasonable and obvious tasks on a classical computer, but on a quantum computer, qubits do not behave normally. In general, cloning an unknown quantum state is prohibited by the *No-Cloning* theorem. Reading a qubit that is initially in a superposition will change the qubit. For example, reading (in computational basis) a qubit in the state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, gives the result 0 fifty percent of the time, and the result 1 fifty percent of the time and will change the state to $|0\rangle$ and $|1\rangle$, respectively.

Although building universal large-scale quantum computers still seems a hard task to accomplish, much progress has been made from both sides of academia and industry. For the last five years, quantum computing has left the purely academic home and is on the agenda of the research labs of some of the largest companies such as IBM, Google, and Microsoft. There is also an ongoing interest on the development of quantum programming languages and software packages by tech companies. As building quantum computers is nearing the era of commercialization and reaching *quantum supremacy*[1], it is important to think of potential applications that we might benefit from. Among many applications of quantum computation, here we focus on the emerging field of *quantum machine learning.*

---

[1]Quantum supremacy is defined as the goal of demonstrating that a programmable quantum device can solve a problem that no classical computer can solve in any feasible amount of time.

Machine Learning is the science and art of making computers learn from available data to solve problems where the exact sequence of steps in an algorithm is hard to develop by humans. Both machine learning and quantum computing are expected to have a radical impact on the way society deals with processing information. Therefore it is natural to ask how these two booming disciplines impact each other, especially as machine learning and optimization (the mathematical core of machine learning) are promising candidates for finding applications that outperform classical algorithms for specific tasks. To answer this question, the field of quantum machine learning has emerged.

Roughly speaking, quantum machine learning refers to all approaches that merge machine learning with quantum information processing. Depending on whether the data are classical or quantum and also if the processing on data is achieved by classical computers or quantum devices, there are different directions based on how machine learning is combined with quantum computing. Here we restrict the scope of our study to using quantum algorithms as part of a larger implementation to process classical dataset on a quantum computer. The hope is that a quantum algorithm outperforms classical algorithms, and achieves what is known as as *quantum speedup*. However, we would need a quantum computer to determine its advantages in speed or, alternatively, the largest solvable problem size, over classical ones. Currently, in the absence of large-scale universal quantum computers, presence or absence of quantum speedup for quantum machine learning algorithms is determined using complexity theory measures, where both classical and quantum algorithms are compared in terms of asymptotic growth of their query and gate complexity with respect to problem size.

Quantum algorithms for machine learning problems started to emerge in recent years, with quantum principal component analysis and quantum support vector machine (SVM) being two of the first. Our focus is on predictive models for binary classification, and we focus on variants of SVM that we expect to be especially important when

training data becomes so large that a quantum algorithm with a guaranteed speedup becomes useful. We present a quantum machine learning algorithm for training Sparse Support Vector Machine, where we deal with a large dataset having a sparse solution. We also present the first quantum semi-supervised algorithm, where we have still large dataset but only a small fraction is provided with labels.

This dissertation is organized as follows. In Chapter 2, we provide background on the basics of linear algebra and quantum computation. In Chapter 3 we elaborate the quantum algorithm for sparse support vector machines. In Chapter 4 we proposed a quantum algorithm for semi-supervised kernel learning.

BACKGROUND

...the "paradox" is only a conflict between reality and your feeling of what reality "ought to be."

— Richard Feynman.

Quantum phenomena do not occur in a Hilbert space. They occur in a laboratory.

—Asher Peres

2.1 Linear Algebra

In quantum computation and quantum information, we usually deal with finite dimensional quantum mechanics, which for our purpose is essentially more linear algebra rather than a physical theory. This is why we need a solid grasp of linear algebra for understanding of both quantum mechanics and quantum computation. In this section, we give a brief review of linear algebra crucial to the content of this dissertation.

The basic object of linear algebra is a vector space, which is a collection of objects called vectors. We are particularly interested in the complex d-dimensional vector space $\mathbb{C}^d$. We denote a complex vector $|v\rangle \in \mathbb{C}^d$ (read: ket v) as

$$|v\rangle = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix}. \tag{2.1}$$

Here we used Dirac notation $|.\rangle$, the standard quantum mechanical notation for a vector in a vector space. We denote the *zero vector* of a vector space by 0. Note that $0 \neq |0\rangle$, the latter is a possibly non-zero vector that we labelled 0 instead of v. The purpose

of doing that will become apparent once we start using vectors to represent binary numbers.

The conjugate transpose of $|v\rangle$ is denoted $\langle v|$ (read: *bra*) and is a row vector

$$\langle v| = (v_1^*, v_2^*, \ldots, v_d^*),\tag{2.2}$$

where v_i^* is the complex conjugate of the complex number v_i.

Inner products. The *inner product* of two vectors $|v\rangle, |w\rangle \in \mathbb{C}^d$ is defined as

$$\langle v|w\rangle := \langle v, w\rangle = \sum_{i=1}^{d} v_i^* w_i.\tag{2.3}$$

The (Euclidean) norm of a vector $|v\rangle$ is defined by $\| |v\rangle \| = \sqrt{\langle v|v\rangle}$. We say a vector $|v\rangle$ is a *unit vector* (or *normalized*) if $\| |v\rangle \| = 1$. A vector space equipped with an inner product is called an *inner product space*.

Orthonormal bases. A set of vectors $|v_1\rangle, \ldots, |v_d\rangle \in \mathbb{C}^d$ is called a basis if they are linearly independent and all $|v\rangle \in \mathbb{C}^d$ can be written as a linear combination $|v\rangle = \sum_{i=1}^{d} \alpha_i |v_i\rangle$ for some $\{\alpha_i\} \in \mathbb{C}$. A set of vectors $\{|e_i\rangle\} \subseteq \mathbb{C}^d$ is orthonormal if $\langle e_i|e_j\rangle = \delta_{ij}$, where δ_{ij} is the Kroenecker delta, whose value is 1 if $i = j$ and 0 otherwise. A natural way for representing a vector $|v\rangle$ is in terms of *orthonormal bases* $|v\rangle = \sum_{i=1}^{d} \alpha_i |e_i\rangle$ for an orthonormal basis $\{e_i\}$ of the vector space.

A common basis in quantum computation is *computational* basis (also called *standard* basis). It is composed of vectors e_i such that the vector e_i has 1 at i-th position and null values elsewhere; for example, $e_0 \in \mathbb{C}^d$ is a vector represented in the computational basis by 1 followed by $d - 1$ zeros. In quantum computing, we often represent the index i in binary form; for example, e_{1101} instead of e_{13}. We frequently denote computational basis in Dirac notation as $\{|i\rangle\}_{i=0}^{d-1}$ instead of $\{e_i\}_{i=0}^{d-1}$, which leads to a more convenient representation of basis vectors; for example e_{1101} will be represented

by $|1101\rangle$. The computational basis for $\mathbb{C}^2$ is

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.4}$$

Thus for example, any vector in $\mathbb{C}^2$ can be written as $|v\rangle = \alpha|0\rangle + \beta|1\rangle$ for some $\alpha, \beta \in \mathbb{C}$, and $|v\rangle$.

Linear operators. Almost all operators encountered in quantum mechanics are *linear operators*. Applying a linear operator $A : \mathbb{C}^d \mapsto \mathbb{C}^d$ on a vector $|v\rangle = \sum_{i=1}^{d} \alpha_i |v_i\rangle$ satisfies:

$$A\left(\sum_i \alpha_i |v_i\rangle\right) = \sum_i \alpha_i A |v_i\rangle. \tag{2.5}$$

The set of all linear maps from vector space V to W is denoted by $\mathcal{L}(V, W)$, and $\mathcal{L}(V)$ is a shorthand for $\mathcal{L}(V, V)$.

Matrix representation of linear operators. The most convenient way of representing a linear operator in finite dimensional spaces is in terms of matrix representation. These two concepts are intimately related and we use them interchangeably. The matrix representation of a linear map $A : \mathbb{C}^d \mapsto \mathbb{C}^d$ in the computational basis $\{|i\rangle\}_{i=0}^{d-1}$ is a $d \times d$ matrix

$$A = \begin{bmatrix} A(|0\rangle), A(|1\rangle), \ldots, A(|d-1\rangle) \end{bmatrix}, \tag{2.6}$$

where $A(|i\rangle)$ is a column vector indicating applying the operator A on the computational basis for $\mathbb{C}^d$.

Outer product representation of linear operators. Using inner product, we can make another useful way of representing linear operators called *outer product* representation. Suppose $|v\rangle, |w\rangle \in \mathbb{C}^d$, the outer product of these two vectors gives a $d \times d$ matrix, $|v\rangle\langle w|$. If we choose $|v\rangle$, and $|w\rangle$ from computational basis $\{|i\rangle\}_{i=0}^{d-1}$ of the inner vector space, then any $d \times d$ matrix A can be expressed as $\sum_{ij} A_{i,j} |i\rangle\langle j|$. This is called

the outer product representation of A. Note that $A_{i,j} = \langle i|A|j \rangle$.

Eigenvectors and eigenvalues. Let $A \in \mathcal{L}(\mathbb{C}^d)$. A non-zero vector $|\lambda\rangle \in \mathbb{C}^d$ is an *eigenvector* of A with *eigenvalue* $\lambda \in \mathbb{C}$ if $A|\lambda\rangle = \lambda|\lambda\rangle$. To be consistent with quantum computing literature, here we used the same notation λ for both the eigenvector and for the label of its corresponding eigenvalue.

Transpose, complex conjugate, adjoint, and trace. Here, we review a number of matrix operations appearing frequently in quantum computation. The *transpose, complex conjugate*, and *adjoint* operation are defined, respectively as follows:

$$A^T(i,j) = A(j,i) \qquad A^*(i,j) = (A(i,j))^* \qquad A^\dagger = (A^*)^T. \qquad (2.7)$$

The *trace* of a matrix is a linear map $\mathrm{Tr} : \mathcal{L}(\mathbb{C}^d) \mapsto \mathbb{C}$ defined as $\mathrm{Tr}(A) = \sum_{i=1}^d A(i,i)$. A very useful property of the trace is that it is *cyclic*, i.e. $\mathrm{Tr}(ABC) = \mathrm{Tr}(CAB)$. This implies $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$, even if $AB \neq BA$.

Hermitian and unitary operators. The operator $A \in \mathcal{L}(\mathbb{C}^d)$ is *Hermitian* if $A^\dagger = A$, where $A^\dagger = (A^T)^*$ is the complex conjugate of the transpose of the matrix A. All eigenvalues of a Hermitian operator are real. Another more restricted class of operators are called *positive semidefinite* (denoted by $A \succeq 0$) whose all eigenvalues are non-negative. If A is positive semidefinite then for every $|v\rangle \in \mathbb{C}^d$, $\langle v|A|v\rangle \geq 0$. A linear operator $U \in \mathcal{L}(\mathbb{C}^d)$ is called *unitary* if satisfies $UU^\dagger = U^\dagger U = I$. All eigenvalues of U are complex numbers of modulus 1, therefore they can be written, using Euler's formula, as $e^{i\theta}$ for some $\theta \in \mathbb{R}$. An important property of unitary operators is that they preserve the inner product between two vectors, i.e. $\langle U|v\rangle, U|w\rangle\rangle = \langle v|U^\dagger U|w\rangle = \langle u|w\rangle$.

Hermitian, positive semidefinite, and unitary operators are subclasses of a more generalized class of operations called *normal operators*. A linear operator $N \in \mathcal{L}(\mathbb{C}^d)$ is normal if it satisfies $N^\dagger N = NN^\dagger$.

Spectral decompositions. A matrix A is said to be *diagonalizable* if it can be written as $A = \sum_{i=1} \lambda_i |\lambda_i\rangle\langle\lambda_i|$, where $\{|\lambda_i\rangle\}_{i=1}$ is orthonormal set of eigenvectors corresponding to eigenvalues λ_i of A. This *spectral decomposition* is an extremely useful tool in our study, providing a sufficient and necessary condition for diagonalizable operators: an operator A is normal if and only if its matrix representation is diagonalizable. Spectral decomposition of a normal matrix can also be written as $A = UDU^\dagger$, where U is the unitary matrix whose i-th column is $|\lambda_i\rangle$ and D is a diagonal matrix with $D(i, i) = \lambda_i$.

Since eigenvectors of a matrix A form an orthonormal basis for the vector space, the spectral decomposition of A immediately tells us how it acts on the vectors of the vector space represented in that basis. Another immediate application of the spectral decomposition is that $\text{Tr}(A) = \sum_i^d \lambda_i$.

Operator functions. Given the spectral decomposition of a normal operator $A = \sum_{i=1}^d \lambda_i |\lambda_i\rangle\langle\lambda_i|$, and a function $f : \mathbb{C} \mapsto \mathbb{C}$, applying f on A is defined as

$$f(A) := \sum_i^d f(\lambda_i)|\lambda_i\rangle\langle\lambda_i|.$$

The commutator and anti-commutator. Let $A, B \in \mathcal{L}(\mathbb{C}^d)$, the *commutator* between A and B is defined as

$$[A, B] := AB - BA.$$

If $[A, B] = 0$, i.e. $AB = BA$, then we say A and B *commute*. The *anti-commutator* is defined as

$$\{A, B\} = AB + BA,$$

we say A *anti-commute* with B if $\{A, B\} = 0$.

Many important properties related to a pair of operators can be derived from their commutator and anti-commutator. For example, if two operator A and B are

Hermitian, then $[A, B] = 0$ if and only if there exist an orthonormal basis such that both A and B are diagonal with respect to this basis, i.e. A and B are simultaneously diagonalizable.

Tensor products of Hilbert spaces. Suppose $V = \mathbb{C}^m$ and $W = \mathbb{C}^n$, then the *tensor product space* $V \otimes W$ is a complex vector space of dimensionality mn, $\mathbb{C}^m \otimes \mathbb{C}^n = \mathbb{C}^{m \times n}$. Elements of $V \otimes W$ result from linear combinations of $|i\rangle \otimes |j\rangle$, where $|i\rangle \in V$ and $|j\rangle \in W$ are computational bases of V and W, respectively and where $|i\rangle \otimes |j\rangle$ is used to denote an ordered pair of vectors. This definition provides tools needed for describing joint quantum systems. If $u \in \mathbb{C}^m$, $w \in \mathbb{C}^n$, then $(v \otimes w)_{ij} = v_i w_j$, for all $i \in [m]$ and $j \in [n]$. In the tensor product space, the addition and multiplication in $\mathbb{C}^n$ have the following properties

$$c\{|v\rangle \otimes |w\rangle\} = \{c|v\rangle\} \otimes |w\rangle = |v\rangle \otimes \{c|w\rangle\}, \tag{2.8}$$

$$|v\rangle \otimes |w\rangle + |v'\rangle \otimes |w\rangle = \{|v\rangle + |v'\rangle\} \otimes |w\rangle,$$

$$|v\rangle \otimes |w\rangle + |v\rangle \otimes |w'\rangle = |v\rangle \otimes \{|w\rangle + |w'\rangle\}.$$

The inner product between $|v\rangle \otimes |w\rangle$ and $|v'\rangle \otimes |w'\rangle$ is defined as a product of individual inner products

$$\langle |v\rangle \otimes |w\rangle, |v'\rangle \otimes |w'\rangle \rangle = \langle v|v'\rangle \langle w|w'\rangle. \tag{2.9}$$

It immediately follows that $\| |v\rangle \otimes |w\rangle \| = \| |v\rangle \| \; \| |u\rangle \|$; in particular, a tensor product of two unit-norm vectors, from V and W, respectively, is a unit norm vector in $V \otimes W$. Formally, the Hilbert space $V \otimes W$ is a space of equivalence classes of pairs $|v\rangle \otimes |w\rangle$; for example $\{c|v\rangle\} \otimes |w\rangle$ and $|v\rangle \otimes \{c|w\rangle\}$ are equivalent ways to write the same vector. A vector in a tensor product space is often simply called a *tensor*.

For operators $A \in \mathcal{L}(\mathbb{C}^m)$ and $B \in \mathcal{L}(\mathbb{C}^n)$, the tensor product $A \otimes B$ is an operator in

$\mathcal{L}(\mathbb{C}^{mn})$ represented by a matrix of dimension $mn \times mn$. Its elements are defined as

$$(A \otimes B)_{(i_1,i_2),(j_1,j_2)} := A_{(i_1,i_2)}B_{(j_1,j_2)}, \tag{2.10}$$

for $i_1, i_2 \in [m]$ and $j_1, j_2 \in [n]$. The tensor product has the following properties for any $A, B \in \mathcal{L}(\mathbb{C}^m), C, D \in \mathcal{L}(\mathbb{C}^n), c \in \mathbb{C}$:

$$\begin{aligned}
(A + B) \otimes C &= A \otimes C + B \otimes C &\tag{2.11}\\
A \otimes (C + D) &= A \otimes C + A \otimes D &\tag{2.12}\\
c(A \otimes C) &= (cA) \otimes C = A \otimes (cC) &\tag{2.13}\\
(A \otimes C)^\dagger &= A^\dagger \otimes C^\dagger &\tag{2.14}\\
(A \otimes C)(B \otimes D) &= AB \otimes CD &\tag{2.15}\\
\mathrm{Tr}(A \otimes C) &= \mathrm{Tr}(A)\mathrm{Tr}(C) &\tag{2.16}
\end{aligned}$$

(Hilbert-Schmidt) inner product on Operators. The (*Hilbert-Schmidt* or *trace*) inner product of two operators $A, B \in \mathcal{L}(\mathbb{C}^d)$ is defined as

$$\langle A, B \rangle := \mathrm{Tr}(A^\dagger B) \tag{2.17}$$

2.2 Quantum Computation

Quantum computers are devices which perform computing according to the laws of quantum mechanics. Therefore learning how to work with these devices requires understanding the basics of quantum mechanics. Quantum mechanics, despite its reputation as a difficult field, has just a few simple rules, all based on linear algebra. These rules intuitively describe the following ideas: how to describe quantum systems (a single quantum system such as a qubit, or composite quantum systems), how quantum systems evolve over time, what operations can be applied on a quantum system, and how to extract (or measure) classical information from a quantum system. In this section

we discuss the basic knowledge of quantum mechanics we need to work with quantum computers.

2.2.1 Describing quantum states

Single quantum systems. In classical computation, the basic building block of information is a *bit*. A bit can only take one of two values, usually represented as 0 and 1. In quantum computation, a *quantum bit* (or *qubit*, for short) is the basic unit of quantum information, encoding 0 and 1 as $|0\rangle$ and $|1\rangle$, respectively. One of the key differences between classical and quantum computation is that a qubit can be in a linear combination of $|0\rangle$ and $|1\rangle$, often called *superposition*:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.18}$$

where $\alpha, \beta \in \mathbb{C}$ are called the *probability amplitudes* of $|0\rangle$ and $|1\rangle$ respectively, and satisfy $|\alpha|^2 + |\beta|^2 = 1$. Therefore, any unit vector in $\mathbb{C}^2$ can be the state of a qubit. More generally, any unit vector in $\mathbb{C}^d$ is the state of a d-dimensional quantum system, called a *qudit*. A qudit $|\psi\rangle \in \mathbb{C}^d$ is described as

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \cdots + \alpha_{d-1}|d-1\rangle = \sum_{i=0}^{d-1} \alpha_i|i\rangle, \tag{2.19}$$

where $\alpha_i \in \mathbb{C}$, $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$, and $\{|i\rangle\}_{i=0}^{d-1} \in \mathbb{C}^d$ denotes the computational basis for $\mathbb{C}^d$.

Composite quantum systems. To build an interesting quantum computer we certainly need multiple quantum bits, that is, multiple quantum systems. Therefore we need to mathematically describe composite quantum systems as well. The mathematical tool that we use for describing composite quantum systems is the tensor product. Suppose quantum systems A and B are described by the state vectors $|\psi_A\rangle \in \mathbb{C}_A^d$ and $|\psi_B\rangle \in \mathbb{C}_B^d$, respectively. Quantum mechanics describes the state of the joint system AB as a vector in the tensor product space of the systems A and B,

i.e. $|\psi_A\rangle \otimes |\psi_B\rangle \in \mathbb{C}^{d_A} \otimes \mathbb{C}^{d_B} = \mathbb{C}^{d_A \times d_B}$. As an example the state of two single-qubits $|\psi\rangle \otimes |\phi\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$ can be described as

$$|\psi\rangle \otimes |\phi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$

where each amplitude is a complex number, and $\sum_{i,i\in\{0,1\}} |\alpha_{ij}|^2 = 1$. Note $\{|ij\rangle\}_{i,j\in\{0,1\}}$ is the computation basis for $\mathbb{C}^4$

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix},$$

$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Density Operators *Density operator formalism* is an alternative formulation for quantum mechanics that allows probabilistic mixtures of pure states, more generally referred to as *mixed states*. All states, including superposition states, that we descibed

above are pure states – they describe a single, specific state of the quantum system. On the other had, a mixed state describes an ensemble $\{p_i, |\psi\rangle_i\}$ of states and is written as

$$\rho = \sum_{i=1}^{k} p_i |\psi_i\rangle\langle\psi_i|, \tag{2.20}$$

where $\sum_{i=1}^{k} p_i = 1$ forms a probability distribution and ρ is called *density operator*. The outer product formalism is used here to differentiate mixed states from superpositions $\sum_i \alpha_i |\psi_i\rangle$, which are pure states. If a quantum system is in a pure state $|\psi\rangle$, the density operator of is a rank-1 matrix $\rho = |\psi\rangle\langle\psi|$. From the definition we can imply that a density operator ρ is positive-semidefinite, that is $\rho \succeq 0$ [1], and $\mathrm{Tr}(\rho) = 1$ [2].

Quantum entanglement. An important property between multiple qubits that deserves to be mentioned separately is *entanglement*. Entanglement refers to quantum correlations between two qubits (or more), that has no classical correspondence and is a key ingredient in many surprises in quantum computation and quantum information such as quantum teleportation and super-dense coding. Consider a bipartite (i.e., two-qubit) quantum state, known as the *Bell state* or *EPR pair*

$$|\phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.21}$$

It cannot be written as tensor product two single qubit states, i.e. there are no single qubit states $|a\rangle$ and $|b\rangle$ such that $|\phi^+\rangle = |a\rangle \otimes |b\rangle$. If someone measures the first qubit of the Bell state (we will learn how to measure quantum systems shortly), there are two possible results: 0 with probability $\frac{1}{2}$, leaving the post-measurement state $|\phi^{+\prime}\rangle = |00\rangle$, and 1 with probability $\frac{1}{2}$, leaving the post-measurement state $|\phi^{+\prime}\rangle = |11\rangle$. Now if someone else measures the second qubit, the result always gives the same result as the first qubit; the measurement results are correlated.

---

[1] A short proof follows the fact that ρ is the sum of nonnegative positive-semidefinite operators $|\psi\rangle\langle\psi|$)

[2] We can prove this property using cyclic property of the trace

Partial trace. As said earlier for expressing the composition of two vector spaces, we use the tensor product. The natural question arises here is how to reverse this action. The *partial trace* operation is defined to answer this question. The partial trace is a tool for describing the state of a subsystem of a composite system, that usually uses the density operator formalism. Consider a joint density matrix $\rho \otimes \sigma \in \mathcal{L}(V \otimes W)$, the partial trace of $\rho \otimes \sigma$ over W is defined as the linear operator $Tr_2(\rho \otimes \sigma) : \mathcal{L}(V \otimes W) \mapsto \mathcal{L}(V)$ given by

$$Tr_2(\rho \otimes \sigma) = \rho Tr(\sigma), \tag{2.22}$$

$$Tr_2 \left(\sum_{j,k} r_j s_k(\rho_j \otimes \sigma_k) \right) = \sum_{j,k} r_j s_k Tr_2(\rho_j \otimes \sigma_k). \tag{2.23}$$

2.2.2 Evolution of quantum states

Every physical system evolves in time and so the state vector $|\psi\rangle$ of a quantum system is no exception; in fact it is a function of time $|\psi(t)\rangle$, sometimes called wave function in quantum physics. Quantum mechanics postulates that the evolution of closed quantum systems acts linearly and is described using unitary transformations. A *closed* quantum system is an ideal system which does not have any unwanted interaction with other systems. That is, for any evolution from time t_1 to t_2 that occurs in a closed quantum system with the initial state $|\psi(t_1)\rangle$, there exists a unitary operator U such that the state of the system in time t_2 is described as

$$|\psi(t_2)\rangle = U|\psi(t_1)\rangle. \tag{2.24}$$

A unitary operator U maps a quantum state expressed as a density operator ρ to $U\rho U^\dagger$, where $U^\dagger$ is the complex conjugate of the operator U.

One might ask why the dynamics of physical systems acts linearly and particularly in unitary fashion? While the answer is not at all obvious, motivated empirically, we can simply answer that this is how Nature behaves, and mathematically we know that

the only linear operators that preserve norms and, more generally, magnitudes of inner products, of vectors are the unitary operators[3].

It is worth mentioning that unitary transformations are invertible; thus, all closed system dynamics are reversible in time – as long as we do not measure (observe) them.

Hamiltonian dynamics. The equation 2.24 describes how the quantum states of a closed quantum systems change over discrete time. In quantum mechanics the evolution of a closed quantum system in *continuous time* is described by the *Shrödinger equation,*

$$i\hbar\frac{d|\psi\rangle}{dt} = H(t)|\psi\rangle, \tag{2.25}$$

where $\hbar$ is a a physical constant known as *Planks's constant* and $H(t)$ is a Hermitian operator known as *Hamiltonian* of the system. Therefore, in principal the Shrödinger equation states if we know the Hamiltonian of a system, we are able to completely describe the dynamics of the system.

For our convenience we consider Hamiltonians H, and the solution of the Schrödinger equation in this case is

$$|\psi(t_2)\rangle = e^{-i\hbar H(t_2-t_1)}|\psi(t_1)\rangle. \tag{2.26}$$

Comparing the equations 2.24 and 2.25 suggests there should be a connection between unitary operators and Hamiltonians. In fact any unitary operator $U \in \mathcal{L}(\mathbb{C}^d)$ can be written as $U = e^{iH}$ for some Hermitian operator $H \in \mathcal{L}(\mathbb{C}^d)$. As we mentioned earlier, since all eigenvalues of a unitary operator can be written as $\lambda_j = e^{i\theta_j}$ (all of its eigenvalues are modulo 1), by taking the spectral decomposition we can write U as

$$U = \sum_j e^{i\theta_j}|\lambda_j\rangle\langle\lambda_j|. \tag{2.27}$$

If we define $H = \sum_j \theta_j|\lambda_j\rangle\langle\lambda_j|$ (which obviously is a Hermitian matrix), we can

---

[3]Per Wigner's theorem, anti-unitary operators also possess this property, but are not suitable for describing quantum evolution forward in time.

write $U = e^{iH}$. The eigenstates $\{|\lambda_j\rangle\}$ of a Hamiltonian are referred to as its *energy eigenstates* (or *stationary states*), and the eigenvalue λ_j corresponding to $|\lambda_j\rangle$ is the *energy* of the state $|\lambda_j\rangle$. The smallest eigenvalue λ_{min} of H is called the *ground state energy*, and $|\lambda_{min}\rangle$ is the *ground state* of H.

2.2.3 Quantum operations

Thus far we discussed the dynamics of quantum systems and how they evolve over time. As quantum computation is actually exploiting quantum mechanics for the purpose of the computing, therefore it is obvious that we are interested to manipulate qubits for our own desires. For example in classical computation we manipulate bits using classical logic gates such as NOT, AND, or XOR. Therefore it is natural to ask what type of operations we can perform on qubits. Again, the answer is unitary operations and in fact unitary constraint is the only physics-based constraint on quantum gates [4]. This immediately yields that all quantum gates are *reversible*.

Unitary operators $U \in \mathcal{L}(\mathbb{C}^\in)$ acting on a single-qubit are called *single qubit gates*, and we commonly represent them as 2×2 matrices in the computational basis.

Pauli gates. In the study of quantum computation and quantum information there are four ubiquitous matrices called the *Pauli matrices*, as follows:

$$
\begin{aligned}
I := \sigma_0 := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad & X := \sigma_1 := \sigma_x := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\
Y := \sigma_2 := \sigma_y := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad & Z := \sigma_3 := \sigma_z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.
\end{aligned}
\tag{2.28}
$$

There are three other quantum gates that play an important role in quantum

---

[4]Some more specific engineering constraints may appear in practical attempts to construct a workable quantum computer.

computation, as follows

$$H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad S := \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad T := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}. \tag{2.29}$$

Phase shift gates. Note that the gate S and T are special cases of a more general class of gates called *phase shift gates*

$$R_\phi := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}, \tag{2.30}$$

which maps state $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ to $R_\phi = \alpha|0\rangle + \beta e^{i\phi}|1\rangle$, that is, inject the *relative phase $e^{i\phi}$* into the second component of $|\psi\rangle$. One can easily verify that $S = R_{\frac{\pi}{4}}$, $S = R_{\frac{\pi}{2}}$, and $Z = R_\pi$.

2.2.4 Extracting information from quantum systems

We have seen that closed quantum systems evolve according to unitary evolution and also how to alter quantum states in our favor. Ultimately, it is time to ask how to observe what is going on inside a closed system and how to extract the information from the system. In quantum mechanics such an act is called a *measurement*, which is performed in a certain basis, commonly in the computational basis. Note that for measuring a closed quantum system, one (a measuring device for example) has to interact with the system, therefore the system is no longer closed and the evolution of the state of the system during a measurement process is not unitary, and thus may be non-reversible. To go beyond unitary evolution, we need to add an additional quantum mechanics postulate for describing the effects of measurements on quantum systems.

Quantum measurement. Suppose $|\psi\rangle$ is the state of a quantum a system, immediately before the measurement. A quantum measurement is described by a set of operators $\Pi := \{M_i\}$ on the state space. These operators are called *measurement*

18

operators and satisfy the following condition know as the *completeness relation*

$$\sum_i M_i^\dagger M_i = I. \tag{2.31}$$

The index i refers to the measurement outcome that may occur in the measurement process with the probability

$$p(i) = \langle\psi|M_i^\dagger M_i|\psi\rangle = \mathrm{Tr}(M_i|\psi\rangle\langle\psi|M_i^\dagger), \tag{2.32}$$

and the state of the system after the measurement is

$$|\psi'\rangle = \frac{M_i|\psi\rangle}{\sqrt{\langle\psi|M_i^\dagger M_i|\psi\rangle}} = \frac{M_i|\psi\rangle}{\sqrt{p(i)}}. \tag{2.33}$$

The above statement means that the act of measuring $|\psi\rangle$ with Π is in general a probabilistic process, and reading (or observing) a quantum system irreversibly alters the state of the system.

Measurement in the computational basis. One of the most common measurement in quantum computation is *the measurement of a qubit (or qudit) in the computational basis.* Suppose we measure a quantum system in the state $|\psi\rangle = \sum_{i=0}^{d-1}\alpha_i|i\rangle$. Note that we, as a classical observer, are not able to see a superposition itself; we can only see the classical state $|i\rangle$, which is not determined in advance. However, we can say if we measure $|\psi\rangle$ using $\Pi = \{M_i = |i\rangle\langle i|\}_{i=0}^{d-1}$, then the probability of obtaining measurement outcome i is

$$p(i) = \langle\psi|i\rangle\langle i|\psi\rangle = |\alpha_i|^2, \tag{2.34}$$

and the state after measurement is

$$|\psi'\rangle = \frac{(|i\rangle\langle i|)|\psi\rangle}{\sqrt{p(i)}} = \frac{\alpha_i}{|\alpha_i|}|i\rangle. \tag{2.35}$$

Note each M_i is Hermitian and $M_i^2 = M_i$, therefore $\sum_i M_i^\dagger M_i = \sum_i M_i^2 = \sum_i M = I$.

2.2.5 Quantum Circuit Model

In section 2.2.3, we learned that admissible operations on quantum states are unitary; they map normalized states to normalized states. In the context of quantum algorithms and quantum computational complexity, having a model for defining the efficiency of quantum algorithms is a necessity; for which we utilize *quantum circuit model*. For an efficient quantum computation, we require that the corresponding unitary operators are realized by *quantum circuits*. To begin, suppose we are given a quantum system with n qubits and a set of gates, each of which acts on one or two qubits at a time (this means it is tensor product of a single or two-qubit operator with the identity operator on the remaining qubits). A quantum computation on this system begins in the initial state $|0\rangle^{\otimes n}$ and applies a sequence of single or two-qubit gates chosen from a set of predefined gates. The last step is quantum measurement (usually in computational basis), which produces the desired output. For example, here we have a very simple quantum circuit on a single qubit input $|\psi\rangle$, and the corresponding evolution is $HX|\psi\rangle$:

$$|\psi\rangle \;-\!\boxed{X}\!-\!\boxed{H}\!-$$
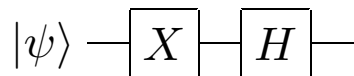
In quantum circuits, each wire denotes a quantum system that evolves from the left to the right. We depict quantum gates with a box labeled by the corresponding unitary operator (for example, Pauli gates X,Y,Z). In the above circuit diagram, if $|\psi\rangle = |0\rangle$, first Pauli gate X and then Hadamard gate should be applied: $HX|0\rangle = H(X|0\rangle) = H|1\rangle = |-\rangle$.

As an example for two-qubit gates, we mention *controlled-NOT*, denoted as CNOT. In the CNOT gate, one qubit is considered as the *control* qubit and the other as the *target* qubit. If the control qubit is $|0\rangle$, the target qubit remains intact and if the control qubit is $|1\rangle$, quantum NOT gate (i.e. Pauli X) is applied on the target qubit.

In computational basis, we have

$$\text{CNOT}\,|00\rangle = |00\rangle \quad \text{CNOT}\,|01\rangle = |01\rangle \quad \text{CNOT}\,|10\rangle = |11\rangle \quad \text{CNOT}\,|11\rangle = |10\rangle.$$

Thus, CNOT acts as a one-to-one mapping on the basis vectors and is thus an invertible, unitary transformation. The circuit diagram for CNOT is shown below.
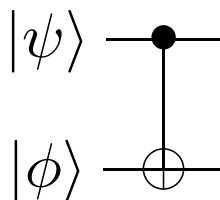
Finally, for a measurement in computation basis we use the following symbol.
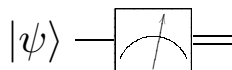
Universal Gate Sets and Quantum Gate Complexity. Any quantum computation is unitary, and thus in principle can be performed by a single unitary matrix, in unit time. Such an approach, which involves defining a potentially very large single matrix, and passing it on to the quantum computer as a program, is impractical, and the quantum circuit model is used instead in defining quantum computing algorithms and in analyzing their complexity. To be meaningful, the circuit model cannot allow arbitrary unitary transformation as units. For analyzing *cost* of a circuit, we need an priori fixed set of gates, where we can assign unit cost to each of them. It is important that we can simulate all other arbitrary gates using this set. Such a set of gates is called a *universal set*. For example, the set $\{H, T, \text{CNOT}\}$ is universal. It is also known that two-qubit gates are universal, i.e., every n-qubit gate can be implemented as composition of a sequence of two-qubit gates. *Quantum gate complexity* is defined as the total number of two-qubit gates required in implementing a quantum circuit, when

no higher-qubit gates are allowed. A quantum algorithm is said efficient if the its gate complexity is polynomial in the number of input qubits.

Oracles and Quantum Oracle Complexity. Oracles are common construct in quantum circuits. An oracle can be thought of as a black-box unitary that encodes the function $x \mapsto f(x)$; often, the oracle models some input data for the computation, for example $f(i)$ returns i-th element of the input vector. In quantum setting this is formalized as

$$O_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle. \tag{2.36}$$

Each call of the O_f is called a *query* to the oracle, which is typically assumed as a unit cost operation. *Quantum query complexity* is defined as the total number of oracle queries. If for a specific problem, we have a quantum algorithm with an efficient query complexity, then by providing an explicit circuit that realizes the oracle (the black-box) transformation, we can imply that we have an efficient quantum algorithm for that problem. For example, in quantum computation, it is common to assume there is an oracle for preparing data, an it is assumed that using quantum RAM one can do the preparation efficiently.

Quantum Input Model. As mentioned, quantum computation typically starts from all qubits in $|0\rangle$ state. To perform computation, access to input data is needed. For problems involving large amounts of input data, such as for quantum machine learning algorithms, an oracle that abstracts random access memory is often assumed. Quantum random access memory (qRAM) uses $\log N$ qubits to address any quantum superposition of N memory cell which may contains either quantum or classical information. For example, qRAM allows accessing classical data entries x_i^j in quantum superposition by a transformation

$$\frac{1}{\sqrt{mp}} \sum_{i=1}^{m} \sum_{j=1}^{p} |i,j\rangle|0...0\rangle \xrightarrow{\text{qRAM}} \frac{1}{\sqrt{mp}} \sum_{i=1}^{m} \sum_{j=1}^{p} |i,j\rangle|x_i^j\rangle,$$

where $|x_i^j\rangle$ is a binary representation up to a given precision. Several approaches for creating quantum RAM are being considered [2, 3, 4], but it is still an open challenge, and subtle differences in qRAM architecture may erase any gains in computational complexity of a quantum algorithm [5].

2.3   Supervised Machine Learning

Our work is concerned with quantum algorithms for machine learning problems. We focus our attention on supervised and weakly-supervised learning methods for binary classification problems. Binary classification involves vector-scalar pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y} = \{-1, 1\}$ and $\mathcal{X} \subset \mathbb{R}^p$ is a compact subset of p-dimensional feature space. Each pair describes an object of study, for example a brain scan or a tissue sample of a medical patient. Individual components x^j of a vector x are called features. Each feature describes some numerical property of the object represented by x, for example signal intensity in a single voxel of a brain scan, or expression level of a single gene. The value of y tells us whether the object belongs to the positive or the negative class. In many scenarios the feature vectors are easy to obtain, but the class variable is not. For example, we can measure methylation status of each CpG base-pair in patient's genome relatively easily, but deciding if the patient's prognosis is positive or negative is challenging.

In statistical learning [6], we assume that samples (x, y) come from a fixed but unknown distribution D over $\mathcal{X} \times \mathcal{Y}$. For a given feature vector x, the probabilities of either class are given by conditional distribution $D_{y|x}$ over $\mathcal{Y}$, and for a given class y, the probability density of feature vectors in that class is given by conditional distribution $D_{x|y}$ over $\mathcal{X}$. While the underlying distributions D, $D_{y|x}$, and $D_{x|y}$ are unknown, we have access to a training set Z consisting of m samples $z_i = (y_i, x_i)$ drawn independently from D. In the binary classification problem the goal is to use the training set to learn how to predict classes y for feature vectors x, even if we did not see such a feature

23

vector in the training set.

The training set can be used to construct a predictive model, in a form of a hypothesis function $h : \mathcal{X} \to \mathbb{R}$, where the sign of $h(x)$ indicates the predicted class for input feature vector x. For a given sample (x, y), the prediction is considered correct if the signs of the predicted and the true class agree, that is, if $yh(x) > 0$. The predictive model should make as few errors as possible over samples $z = (x, y)$ sampled from distribution D, that is, it should minimize $\int_{\mathcal{X} \times \mathcal{Y}} I[yh(x) \leq 0]D(z) \, dz$, where I is an indicator function over Boolean domain returning 1 for true and 0 for false.

A simple but often effective class of hypotheses is the class of linear functions $h(x; \beta, b) = \beta^T x + b = \sum_{j=1}^{p} \beta_j x^j + b$. A linear predictive model is parameterized by a vector of feature weights $\beta \in \mathbb{R}^p$ and a bias term $b \in \mathbb{R}$. To simplify the notation, we often add one more dimension to $\mathcal{X}$ with all samples having a value of one. The predictive model is then simply $h(x; \beta) = \beta^T x$, $\beta \in \mathbb{R}^{p+1}$, with β_{p+1} playing the role of bias.

Training of a linear model involves finding a suitable parameter vector β. For a single sample (x, y), the suitability of a model h with specific β will be captured by a loss function $\ell(y, h(x; \beta))$, which returns a nonnegative real number that we interpret as a measure of our dissatisfaction with the prediction $h(x; \beta)$. The natural *0/1 loss*, defined as $\ell(y, h(x; \beta)) = I[y\beta^T x \leq 0]$, is not a continuous function of the parameter vector β, and is flat almost everywhere, leading to problems with finding β that minimizes the loss. Instead of the 0/1 loss, a convex function that upper-bounds it is often used in training classification models. For example, the least-square loss $\ell(y, h) = (y - h)^2$ is used in Fisher's Linear Discriminant and in Least-Squares Support Vector Machine (LS-SVM) classifier [7].

Once the loss function is chosen, the goal of training a model is to find the parameter vector β that minimizes the expected loss $\mathbb{E}_{z \sim D} \ell(y, h(x; \beta))$, referred to as *risk* of the model, $R(\beta) = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, h(x; \beta))D(z) \, dz$. Since D is unknown, a surrogate goal is

to search for β that leads to low loss on samples from the training set. For example, the *empirical risk minimization* strategy involves finding parameters β that minimize *empirical risk*, that is, the average loss on the training set, $\hat{R}(\beta) = \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, h(x_i; \beta))$.

The model β that minimizes the empirical risk may have high generalization risk $R(\beta)$, that is, may fare poorly on samples outside of the training set, especially if the number of training samples m is smaller, or not much larger, than the number of features p, the features are not statistically independent, or the feature values are noisy. Often, the generalization error can be reduced if a penalty on the complexity of the model is introduced into the optimization problem. Typically, this penalty term, known as regularization term, is based on $\|\beta\|$, a norm of the vector of model parameters, leading to *regularized empirical risk minimization* strategy, which finds parameters that minimize $\hat{L}(\beta) = \hat{R}(\beta) + \lambda f(\|\beta\|)$. For example, most Support Vector Machines [8] use squared L_2 norm of β, $\|\beta\|_2^2$, as the regularizer.

CHAPTER 3

QUANTUM SPARSE SUPPORT VECTOR MACHINES

3.1 Introduction

Steadily increasing ability to measure large number of features in large number of samples leads to ongoing interest in fast methods for solving large-scale classification problems. One of the approaches being explored is training the predictive model using a quantum algorithm that has access to the training set stored in quantum-accessible memory. In parallel to research on efficient architectures for quantum memory [9, 10, 11], work on quantum machine learning algorithms and on quantum learning theory is under way (for review, see [4, 12, 13, 14]), and is starting to attract interest form the machine learning community [15, 16, 17].

In order to achieve quantum speedup, the choice of the variant of the machine learning problem often matters. For example, the pioneering quantum machine learning method – quantum SVM [18] – focused on quadratic loss and quadratic regularizer, in order to be able to utilize solvers for quantum linear systems of equations, such as HHL [19]. This and other recent solvers based on quantum manipulation of eigenvalues [20, 21] can lead to exponential speedup compared to classical methods, as long as the eigenvalues of the linear kernel matrix are of similar magnitude. However, linear systems arise only from unconstrained, or equality-constrained quadratic problems, such as LS-SVM, but not from hinge-loss SVM. A more recent quantum supervised learning method [16] that can achieve sublinear training time involves efficient quantum primal-dual approach for solving minimax problems, and as a consequence focuses on minimizing the maximum – not the average – loss over the training set.

In this chapter, we focus on the possibility of achieving quantum speedup for the

Sparse SVM [22, 23, 24, 25], a linear classifier that combines hinge loss with L_1 regularizer. Sparse SVM is useful in cases where the number of features is large compared to the number of training samples, and where interpretability of the classifier matters, not just its predictive abilities. For example, in biomedicine it is important to know if, and how much, each feature contributes to the prediction, and often a small number of features is enough to tell the classes apart. Thus, linear models[1] $h(x; \beta) = \beta^T x$ with weights $\beta, x \in \mathbb{R}^p$, which learn a single multiplicative weight β_j for each feature x^j of a p-dimensional sample x, are often preferred over non-linear approaches, such as kernel methods, ensembles, or neural networks. In these settings, we often expect that highly-accurate predictions can be made using just a few discriminative features; we expect that a well-performing model should be sparse, that is, there is a vector β composed mostly of zeros that achieves near-optimal accuracy. The remaining features either carry no information about the separation of classes, or the information is redundant. To find sparse solutions to classification problems, a regularization term in the form of L_1 norm of β is often included in the objective function.

3.1.1 Our Contribution

Quantum Sparse SVM (QsSVM) results from using a quantum algorithm for solving linear programming (LP) problems [26] instead of a classical solver. Our aim is to analyze whether using a quantum solver provides benefits in terms of computational complexity of model training, that is, if sublinear training time in terms of the number of training samples, m, and the number of features, p, be achieved. This problem is challenging, since quantum LP solver [26] and similar quantum SDP/LP solvers [27, 28, 29, 30] express the complexity not only in terms of m and p, but also in terms of characteristics of the primal and dual solution to the LP/SDP instance being solved

---

[1]For brevity, we fold the bias term into the feature weight vector β, by adding a constant-one feature.

(see Fig. 1). So far, realistic application scenarios with characteristics that provably lead to quantum speedup have been scarce.

We show that for arbitrary binary classification problems no quantum speedup is achieved using quantum LP solvers. More broadly, we prove that the lower bound for solving Sparse SVM using any quantum algorithm with black-box access to training data is $\Omega(\min(m, p))$. However, we show there are realistic cases in which a sparse linear model will have high accuracy, and sublinear time in m and p can be guaranteed.
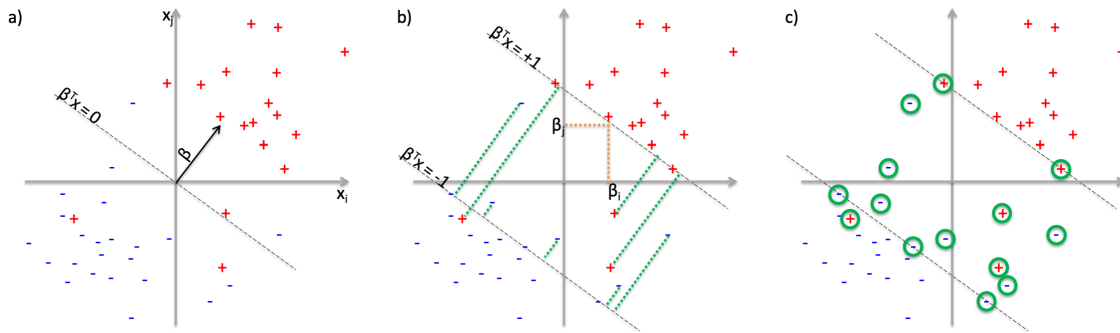
Figure 1: Graphical illustration of the factors contributing to the computational complexity of Quantum Sparse SVM for a classification problem with linear solution vector β (**a**). Complexity is proportional to the product Rr, where R is the sum of sample loss values (**b**: green lines) and feature weight magnitudes (**b**: orange lines), and r is the sum of weights of support vectors (**c**: green circles).

3.2   Quantum Sparse SVM

The training of Sparse SVM model using a training set $\{(x_i, y_i)\}$ with p features and m samples involves solving a minimization problem

$$\arg \min_{\beta \in \mathbb{R}^p} \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i \beta^T x_i) + \lambda \sum_{j=1}^{p} |\beta_j|. \tag{3.1}$$

Using standard techniques, for $\lambda > 0$, this non-linear unconstrained optimization problem can be transformed to an equivalent primal constrained linear program with

$n = m + 2p$ nonnegative variables and m linear inequality constraints

$$\min_{\xi, \beta^+, \beta^-} \quad \frac{1}{m} \sum_{i=1}^{m} \xi_i + \lambda \sum_{j=1}^{p} \beta_j^+ + \lambda \sum_{j=1}^{p} \beta_j^- \tag{3.2}$$

$$\text{s.t.} \quad \sum_{j=1}^{p} y_i x_i^j \beta_j^+ - \sum_{j=1}^{p} y_i x_i^j \beta_j^- \geq 1 - \xi_i, \quad i \in [m]$$

$$\xi_i, \beta_j^+, \beta_j^- \geq 0,$$

where $[m] = \{1, ..., m\}$. We can read out the solution as $\beta_j = \beta_j^+ - \beta_j^-$. We also have $|\beta_j| = \beta_j^+ + \beta_j^-$. The value of the hinge loss of i-th training sample is equal to ξ_i. The dual problem is

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i \quad \text{s.t.} \quad -\lambda \leq \sum_i \alpha_i y_i x_i^j \leq \lambda, \quad j \in [p]$$

$$0 \leq \alpha_i \leq 1/m \quad i \in [m]. \tag{3.3}$$

To solve the linear program associated with Sparse SVM, we employ a recently proposed quantum LP solver [26]. A classification problem with m samples in p-dimensional feature space leads to an LP with $n = m + 2p$ nonnegative variables and m linear inequality constraints. The quantum LP solver results in $\tilde{O}\left(\left(\sqrt{m} + \sqrt{n}\right)\left(R(r+1)/\varepsilon\right)^3\right)$ complexity of obtaining the solution to the LP to within ε additive error where $\tilde{O}$ notation hides logarithmic factors. The key question is whether the dependence on R and r, the L_1 norms of the primal and dual LP solutions, respectively, allows for achieving sublinear time complexity.

3.2.1 Linear Programs and Matrix Games

We employ a recently proposed quantum LP solver based on zero-sum games [26]. The solver is based on a series of zero-sum matrix games. For a given LP, each game in the series is constructed to indicate if the optimal value of the LP lies in a certain numerical range.

Each zero-sum matrix game involves two players, Alice with n possible strategies

and Bob with m possible strategies. If Alice chooses strategy $i \in [n]$ and Bob chooses $j \in [m]$, the pay-off to Alice is M_{ij} and to Bob is $-M_{ij}$, where M is the matrix defining the game. Given a randomized strategy for Alice, sampled from $u \in \mathbb{R}^n_+$, with $\| u \| = 1$, and a randomized strategy for Bob, sampled from $v \in \mathbb{R}^m_+$, the expected payoff to Alice is $u^T M v$. Bob can assume optimal strategy distribution u on Alice's part and chose his strategy distribution v according to $\min_v \max_u u^T M v$ to minimize Alice's and maximize his own expected pay-off.

For a given linear program $\mathrm{LP}(c, A, b) = \min c^T x \text{ s.t.} Ax \leq b, \ x \geq 0$ and a real value α, finding the expected pay-off for the optimal strategy in a game defined by a matrix M that contains α as an element and A, b, c as sub-blocks can be used to determine if the optimal value of the LP is lower than α. If it exists, a feasible LP solution with optimal value below α can also be obtained from the optimal strategy vectors u and v. Iteratively, this process can solve the LP up to a pre-determined error.

A classical algorithm [31] can solve the game up to ε additive error in $\tilde{O}\left((n+m)/\varepsilon^2\right)$ time, where $\tilde{O}$ notation hides logarithmic factors. The quantum version replaces Gibbs sampling step in the classical algorithm with its quantum counterpart, achieving quadratic speedup in terms of n, m, and a $1/\varepsilon^3$ dependence on the desired additive error ε of the solution. A query about the optimal value of an LP with m variables and n constraints being in a certain range reduces to a game with $(n+3) \times (m+2)$ matrix M. Further, assuming that the LP primal and dual solution vectors are bounded in L_1 norm by R and r, respectively, through the bisection argument only a logarithmic number of such range queries – matrix games – are needed, each solved to within error $R(r+1)/\varepsilon$ error, in order to obtain the solution to the LP to within ε additive error. Together, the computational complexity of the quantum LP solver is $\tilde{O}\left(\left(\sqrt{m} + \sqrt{n}\right)\left(R(r+1)/\varepsilon\right)^3\right)$, a polynomial speedup compared to classical methods, and lower exponent in dependence on R, r, ε than in previous quantum methods [27, 28, 29, 30].

3.2.2 Data Access Model

For a linear program $\text{LP}(c, A, b) = \min c^T x \ \text{s.t.} Ax \leq b, \ x \geq 0$, the quantum LP solver requires read-only access to the vectors and matrices (c, A, b) defining the LP, and read/write access to internal data. The solver assumes access to data using a quantum oracle implemented using quantum random access memory (QRAM). For example, the access to element A_{ij} of the LP constraint matrix A is given by an oracle associated with A, a unitary linear operator O_A capable of performing the mapping $O_A|i\rangle|j\rangle|z\rangle \to |i\rangle|j\rangle|z \oplus A_{ij}\rangle$ in superposition. The operator that takes three qubits on input, corresponding to indices i, j and a placeholder z, and output produces i, j and the exclusive alternative ($\oplus$) of the binary representation of the matrix element A_{ij} and the placeholder value of the third qubit, z. The algorithm assumes the oracle can operate in superposition, that is, given a superposition of indices, returns a superposition of array elements

$$O_A|i\rangle\big(\alpha|j\rangle + \beta|j'\rangle\big)|z\rangle \to \alpha|i\rangle|j\rangle|z \oplus A_{ij}\rangle + \beta|i\rangle|j'\rangle|z \oplus A_{ij'}\rangle,$$

and similarly for the first index. The computational complexity of the algorithm is measured with respect to number of oracle calls and the number of two-qubit quantum gates required for further processing.

Availability of quantum random access memory is a typical assumption in quantum algorithms, including recent quantum machine learning methods [16, 15, 17]. Workable, large-scale QRAM does not exist yet and feasibility of its constructing is still debated, but algorithmic models [2, 3, 10, 32] and experimental demonstrations [33, 11, 34, 35] of quantum memory alone, and as part of quantum networks or quantum learning systems, are emerging. Recent results indicate that loading classical data into quantum RAM can be done in logarithmic time [36, 37], at the cost of introducing small perturbations into the data.

In a fully-quantum RAM, operations of both reading and writing in quantum

superposition should be available. In quantum machine learning systems, typically the input data, as well as certain intermediate data, is processed classically; for example, we assume the feature values for each sample arrive over some classical information channel. Only the read operation is required to operate in quantum superposition, that is, with parallelism implied in linearity of the data access oracle. This type of access is referred to as quantum-read, classical-write RAM (QCRAM) [38, 39]. The standard model of QCRAM utilizes a tree structures over non-zero elements of an n-dimensional vector x to allow classical readout and update of a single vector entry x_i in $O(\log n)$ time. It also allows for accessing selected elements x_i in quantum superposition, sampling integers $i \in [n]$ according to $x_i / \| x \|$, and creating the quantum state corresponding to x.

For a linear program $\text{LP}(c, A, b) = \min c^T x$ s.t. $Ax \leq b$, $x \geq 0$, the quantum LP solver requires read-only access to the vectors and matrices (c, A, b) defining the LP, and read/write access to the strategy distribution vectors u, v as it iterates through a sequence of matrix games. The solver assumes access to data via a quantum oracle implemented efficiently using QRAM. Through iterations, the game solution vectors u, v are stored in QCRAM. Solving each game involves an iterative algorithm that results in at most $O(1/\varepsilon^2 \log mn)$ elements of u, v being non-zero [26]. The QCRAM stores only non-zero elements of u, v, thus, accessing all solution elements classically using sequential tree traversal adds $O(1/\varepsilon^2 \log mn)$ overhead. For sparse models, with few non-zero feature weights in the optimal solution, access can be even faster.

Iterations of the quantum LP solver include the scaled feasible solutions to the original LP. These are stores in QCRAM. Each iteration involves an algorithm that result in at most $O(1/\varepsilon^2 \log mn)$ elements of u, v being non-zero. The QCRAM stores only non-zero elements of u, v, thus, accessing all solution elements classically using sequential tree traversal adds $O(1/\varepsilon^2 \log mn)$ overhead. For sparse models, with few non-zero feature weights in the optimal solution, access can be even faster.

3.3 Lower Bound for Complexity of Quantum Sparse SVM

Assuming efficient oracle access to input, the computational complexity of quantum LP solver utilized in Quantum Sparse SVM shows improved dependence on n and m, but polynomial dependence on R and r may erase any gains compared to classical LP solvers.

For any training set, the minimum of the objective function of the SparseSVM optimization problem (eq. 3.1) is bounded from above by one, since an objective function value of one can be obtain by setting $\beta = 0$, which leads to unit loss for each training sample, and thus unit average loss. For some training sets, one is the minimum of the objective function – for example if training samples come in pairs, $(x, +1)$ and $(x, -1)$. In this case, the norm of the primal solution is $R = \sum_i |\xi_i| + \sum_j |\beta_j| = m$, and the norm of the dual solution is $r = \sum_i |\alpha_i| = 1$, since by eq. (3.3) and strong duality it is equal to the value of the primal objective function. The quantum solver we use includes $(R(r + 1)/\varepsilon)^3$ term in its complexity, and $Rr = O(m)$ erases any speedups compared to classical solvers.

A more realistic case in which we see $R = O(m)$ is a regular XOR problem, for example involving two features and four training samples, $[+1, +1]$ and $[-1, -1]$ with $y = +1$ and $[+1, -1]$, $[-1, +1]$ with $y = -1$. For any $\beta \in \mathbb{R}^2$, if there is a sample with loss $1 - \delta$, there is another sample with loss $1 + \delta$. Thus, sum of ξ_i variables is one for any β, and again $\beta = 0$ is the minimizer of the regularized empirical risk, leading to $Rr = O(m)$ and a $O(m^{3.5})$ term in the solver worst-case computational complexity.

The above negative results concern speedup of Sparse SVM utilizing a specific quantum LP solver. It can be shown that sublinear worst-case complexity, in terms of the smaller of m and p, is not possible in general. To provide lower bound on solving Sparse SVM using any quantum algorithm with black-box access to elements of feature vectors x and class variables y, we utilize reduction from majority problem, that is, the problem of finding the majority element in a binary vector f of size n. In the majority

problem, vector f has t unity and $s = n - t$ null elements, and the algorithm should return true if $t > s$ or false otherwise. Alternatively, it suffices to return the value of t or s. Given arbitrary majority problem instance, we show a simple procedure to treat it as the input training set for a Sparse SVM classifier, and prove that solving the Sparse SVM classification problem, for values of the regularization constant C leading to non-null feature weights vector β, allows us to provide the answer to the instance majority problem. The lower bound for computational complexity of the majority problem, assuming black-box quantum oracle access to elements f_i in superposition, is $\Omega(n)$ [40], leading to the lower bound on complexity of quantum Sparse SVM.

Theorem 1. *Assuming black-box quantum oracle access to a training set with m samples and p features, the lower bound for finding the optimal Sparse SVM solution β and its objective value $L^* = \min_\beta \hat{R}(\beta) + C \parallel \beta \parallel_1$, for $C > 1/m$ and $C = \tilde{O}\left(1/m\right)$, is $\tilde{\Omega}\left(\min(m, p)\right)$. The lower bound holds even if the optimization algorithm is allowed to return a suboptimal solution with objective value at most $L^* + \tilde{\Omega}\left(1\right)/16$.*

Proof. We will rely on reduction from majority problem, with oracle complexity Ωn for a problem of size n, both in the classical and in the quantum setting. We will also use a promise vartiant of majority, in which the minority class is guaranteed to have at most k elements, which has Ωk complexity lower bound.

First, we construct a dataset with $p' = n$ features and $m' = zn$ samples, for some number $z \in \mathbb{N}$ not growing with m', p' or growing very slowly; that is, we require $z = \tilde{O}\left(1\right)$. Based on chosen z, we represent C as $C = (2z - \gamma)/m'$ for some $1/2\gamma < 2z - 1$. This guarantees that $C > 1/m'$ and $C = \tilde{O}\left(1/m'\right)$.

Given n, let $f_{n,t}$ be an arbitrary instance of a majority problem of size n with t ones and $n - t$ nulls. We construct the training set based on $f_{n,t}$ in the following way. For $k = 1, ..., n$, if $f_{n,t}(k) = 1$, we add z pairs of samples with 1 value of the k-the feature and positive class for the even samples, and -1 and negative class for the odd samples, with null for other features. These samples can be perfectly classified

by setting $\beta_k = 1$. In the optimal objective value $L^* = \min_\beta \hat{R}(\beta) + C \|\beta\|_1$, for empirical risk $\hat{R}(\beta) = m'^{-1} \sum_i \left[1 - y_i \beta^T x_i\right]_+$, we will have null loss $\left[1 - y_i \beta^T x_i\right]_+$ for each sample, and a $C = (2z - \gamma)/m'$ contribution from the L_1 regularizer. Note that for $C > 2z/m'$, it is better to set β_k to null and instead have $2z/m'$ loss for the z pairs of samples. If $f_{n,t}(k) = 0$, we add z pairs of samples, with the same pattern of feature values and classes as for $f_{n,t}(k) = 0$, except both odd and even samples have $+1$ as the value of feature k. These samples cannot be classified correctly, the optimal regularized solution has $\beta_k = 0$. The optimal objective value will have $1/m'$ loss $1/m'$ per each sample corresponding to $f_{n,t}(k) = 0$.

In total, there are $2zt$ samples with $f_{n,t}(k) = 1$, together contributing $t(2z - \gamma)/m'$ to the objective function. There are $2z(n - t)$ samples with $f_{n,t}(k) = 0$, together contributing $2z(n - t)/m'$ to the objective function. We have $m' = 2zn$ samples in total. The optimal objective value is

$$L^* = \frac{t(2z - \gamma) + 2z(n - t)}{2zn}.$$

If the minimization algorithm returns the objective value $\hat{R}^*$, we can use it to calculate t for the underlying majority problem

$$t = n\frac{2z}{\gamma}(1 - L^*).$$

Given arbitrary p, m, we find the largest n, z for which $p' \leq p$ and $m' \leq m$. First m' samples with p' features are set as above. If $p \geq p'$, we add additional features with all-null values, which does not change the optimal solution's objective value. If $m \geq m'$, we add all-null samples, which adds a $1/m$ loss per sample. The solution L^* above needs to be adjusted for the added constant $(m - m')/m$, and scaled, since now the loss and the regularizer constant C both involve $1/m$ instead of $1/m'$, but otherwise remains the same.

Hence, among datasets with m samples and p features are those corresponding

to all possible majority problems of size $\tilde{\Omega}\left(\min(m, p)\right)$, some of which require number of data oracle calls proportional to n to be solved. Thus, quantum Sparse SVM is $\Omega(\min(m, p))$, since training the Sparse SVM classifier on the training set constructed above allows us to solve the underlying majority problem of size $n = \min(p, \lceil m/2z \rceil)$, with $z = \tilde{O}\left(1\right)$.

We assumed above that the algorithm for training the Sparse SVM is able to find the optimal solution and return its objective value L^*. If the algorithm instead returns a suboptimal solution with objective value within ε of the optimal L^*, the error in estimating t from it is at most

$$\Delta t = n\frac{2z}{\gamma}\varepsilon.$$

Consider a promise variant of the majority problem, in which the instances are promised to have less than $n/4$ elements in the minority class. These problems require on the order of $n/4$ oracle calls in the worst case. For such problems, making an $n/4$ error in calculating t does not change the correctness of the solution.

We can limit the error in estimating t to at most $n/4$ if the error in estimating $Ł^*$ is $\varepsilon < \gamma/8z$. Given that $\gamma > 1/2$ and $z = \tilde{O}\left(1\right)$, that is $1/z = \tilde{\Omega}\left(1\right)$, to achieve error in estimating t higher than $n/4$, we need

$$\varepsilon > \gamma/8z > \tilde{\Omega}\left(1\right)/16.$$

If the Sparse SVM solver guarantees error below $\tilde{\Omega}\left(1\right)/16$, the solution to Sparse SVM can be used to solve the $n/4$-promise majority problem, and thus has Ωn lower bound, same as in the no-error case. Note that $L^* \leq 1$, since $L^* = 1$ can always be achieved irrespective of the training set by an all-null feature weights vector β. $\qquad\square$

The result above shows that we cannot design a quantum algorithm that trains Sparse SVM in time sublinear in both the number of samples and the number of features. Notably, the dataset constructed above may have much fewer discriminative features

than p, indicating that the lower bound depends on the total number of features, not the number of non-zero features in the model.

3.4 Speedup in Quantum Sparse SVM for Special Cases

Worst-case analysis of the proposed method utilizing quantum LP solver [26], and of quantum Sparse SVM in general, shows that it cannot achieve sublinear computational complexity. This does not preclude sublinear complexity for some families of classification problems for specific quantum SVM solvers.

In the quantum LP-based approach, sublinear time can only be achieved if the L_1 norms of the primal and dual solution vectors are kept in check as the number of samples and features grows. The regularizing term $\| \beta \|_1$ can be expected to grow slowly in sparse models, and the average loss is normalized by the number of training samples and thus always below one. Yet, sparsity assumption alone is not enough to achieve speedup when class distributions overlap in the features space and the optimal error is not null. While the bound on the dual solution norm is a direct consequence of sparsity of the model, the $\sum_i \xi_i$ term in the primal solution norm in principle grows in proportion to the number of training samples and to the generalization risk, and may be further inflated by the stochastic nature of the training set. We show that under realistic assumptions about the family of classification problems for growing number of features, where new features occasionally bring new discriminative information, a bound on the primal solution norm that leads to sublinear training time can be formulated.

3.4.1 Truncated Subgaussian Classification Problems

The first defining characteristic of the family of classification problems we explore here is limited overlap between class distributions. Since we are focusing on a linear, sparse SVM classifier, the overlap will be with respect to a linear decision boundary – we will assume that the tails of class conditional distributions that reach across the

decision hyperplane are bounded, in a way formalized in Definition 2. The consequences of that assumption on the expected loss of the model are analyzed in Lemma 4 in the distribution setting, and then extended in Lemma 5 to the empirical risk on finite-sample datasets of fixed size (m, p) that are actually seen during training, providing basis for asymptotic analysis building up to Theorem 3.

In the asymptotic analysis of time complexity, we will assume that we are given a series of classification problems where both the number of samples, m, and the number of features, p, grows. The second defining characteristic of the scenarios leading to sublinear time complexity is the assumption that p', the number of discriminative features used by the sparse model, also grows, but at a slower rate than m or p. In an illustrative idealized case, these discriminative features have means $+c$ and $-c$ in the positive and the negative class, respectively – though the situation does not change if the signs of the means are swapped for some of the discriminative features. Thus, c represents how discriminative a feature is. With increasing number p' of discriminative features, each with means differing by at least $2c$, the distance between the means of the two multivariate distributions increases at the rate of at least $2c\sqrt{p'}$. Moving beyond the idealized case of means of each feature at $\pm c$, we will simply require that the multivariate class distribution means diverge at this rate. This defining feature of our scenario is key to asymptotic analysis starting in Lemmas 6 and 7 and concluding with the final complexity result in Theorem 3.

Definition 2. *A (Δ, μ)-truncated subgaussian classification problem, for $\mu > 1$, $\Delta > 0$, is defined by distribution D such that there is an underlying vector $\beta^\star \in \mathbb{R}^p$ with $\|\beta^\star\|_2 = 1$, for which*

- *the conditional distributions $D_{x|+}$ and $D_{x|-}$ of the samples from the positive and negative class, respectively, give rise to univariate distributions $D_{v|+}$ and $D_{v|-}$ on a line resulting from the projection $v = y\beta^{\star T}x$,*

- *the tails of $D_{v|+}$ and $D_{v|-}$ are bounded from above, in the region $v \in (-\infty, 1]$,*
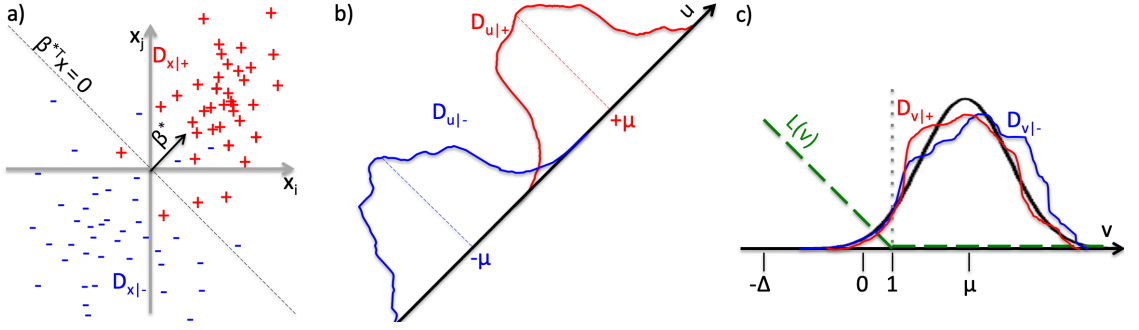
Figure 2: Graphical illustration of Definition 2: **a)** the initial multivariate two-class problem; **b)** intermediate step involving projection $u = \beta^{*T}x$ into one dimension; **c)** final univariate distributions after projection $v = yu$.

> *by the normal probability density function $\mathcal{N}_{\mu,1}(v)$ with unit standard deviation, centered at μ,*

- *the tails of $D_{v|+}$ and $D_{v|-}$ have zero mass for $v < -\Delta$.*

A p-dimensional (Δ, μ)-truncated subgaussian problem is called sparse if the number of non-zero components in the vector $\beta^\star$ is small compared to the number of features, p.

A graphical illustration of the definition is shown in Figure 2. Sparse (Δ, μ)-truncated subgaussian problems often arise in dataset coming from natural science. For example, in molecular biology, levels of raw mRNA transcripts of individual genes approximately follow a log-normal distribution within a class of samples [41, 42]. Raw data is typically processed via a log-like transform prior to analyses, then each class of a multi-gene dataset approximately follows a multivariate normal distribution, leading to sub-Gaussian tail. Biomedical data often has large number of features, and the samples are rarely sparse, that is, vast majority of feature values are non-zero. For example, classification problems involving gene expression measured using RNA-seq may have tens of thousands of features, only a relatively small number of genes contribute to between-class differences, leading to a sparse problem – sparse linear models have been successful in analyzing brain activity [43], microbiome [44], and in other biomedical

39

settings [45, 46, 47].

3.4.2 Complexity Analysis of Quantum Sparse SVM for Truncated Subgaussian Problems

We provide the following characterization of the computational complexity of training the Quantum Sparse SVM for sparse (Δ, μ)-truncated subgaussian problems.

Theorem 3. *For $p \to \infty$, consider a family of p-dimensional (Δ_p, μ_p)-truncated subgaussian problems D_p with underlying vectors $\beta_p^\star$. Assume that the vector $\beta_p^\star$ is sparse, it only has $p' = 1 + 2\log p$ non-zero coefficients. Further, assume that the mean μ_p diverges with the number of discriminative features p' as $\mu_p > c\sqrt{p'}$ for some $c > 1$ that captures how separated class centers are for individual informative features. As p grows, we allow scattering of the samples farther into the region dominated by the other class, $\Delta_p \le 2\log p$. For growing p, assume efficient oracle access to the training data, with the training set sizes m growing proportionally with p, $m/p = O(1)$. Then, training QsSVM has computational complexity of*

$$\tilde{O}\left(\sqrt{m + 2p}\,\mathrm{poly}\left(\log p, 1/\varepsilon\right)\right).$$

Proof. Training QsSVMs translates to solving an LP problem (eq. 3.2) with m constraints and $n = 2p + m$ variables. The quantum LP solver proposed of van Apeldoorn and Gilyén [26] has complexity $\tilde{O}\left(\left(\sqrt{m} + \sqrt{n}\right)\left(R_p(r_p + 1)/\varepsilon\right)^3\right)$. When $m/p = O(1)$, by virtue Lemma 7, neither R_p nor r_p grow with m, and both grow with p as $O(\log p)$, yielding the complexity result. $\square$

Scenarios in which the number of features is larger or at least comparable to the number of samples are of great practical importance – it is common in biomedical data analysis, for example in classification of molecular profiles such as gene expression or methylation, or classification of 3D brain scans. In these $p > m$ scenarios, in which L_1 regularization is especially useful, Theorem 3 shows that the

complexity is $\tilde{O}\left(\sqrt{m + 2p}\,\mathrm{poly}\left(\log p, 1/\varepsilon\right)\right)$, that is, using quantum LP solvers offers speedup compared to classical solvers. More generally, the computational complexity is $\tilde{O}\left(\sqrt{m + 2p}\,\mathrm{poly}\left(\frac{m \log p}{p}, \log p, 1/\varepsilon\right)\right)$, leading to speedup even in some cases beyond the $p > m$ scenario, such as $m = O\left(p \log p\right)$.

The path from Definition 2 to Theorem 3 leads through a series of technical Lemmas, from generalization risk to the characteristics of the empirical solution.

For the hinge loss, the generalization risk $R(\beta^\star)$ associated with model $h(x) = \beta^{\star T} x$ on the (Δ, μ)-truncated subgaussian problem D is bounded through the following lemma.

Lemma 4. *Let D be a (Δ, μ)-truncated subgaussian classification problem with underlying vector $\beta^\star$ leading to univariate distributions $D_{v|+}$ and $D_{v|-}$ as described above. Let $L = \max(0, 1 - v)$ be a univariate random variable capturing hinge loss of the model $h(x) = \beta^{\star T} x$ for samples from D. Then, the expectation and standard deviation of L are bounded from above by*

$$R(\beta^\star) = \mathbb{E}[L] \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{(1-\mu)^2}{2}} = \mathcal{N}_\mu(1), \tag{3.4}$$

$$\mathrm{Var}[L] \leq \left[(1 - \mu)^2 + 1\right]\left[1 + \mathrm{erf}\left(\frac{1 - \mu}{\sqrt{2}}\right)\right]. \tag{3.5}$$

Also, values of L are in the range $[0, \Delta + 1]$.

Proof. The proof relies on properties of integrals of $x^k \mathcal{N}_0(x)$. Let

$$G(x) = \mathcal{N}_0(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \quad G_k(x) = \int_{-\infty}^{x} t^k G(t)\,\mathrm{d}t.$$

Then, we have Let p_+ and p_- by the probabilities, under D, of the positive and the

negative class, respectively. We have

$$\mathbb{E}[L] = \int_{-\infty}^{\infty} \max(0, 1 - v)[p_+ D_{v|+}(v) + p_- D_{v|-}(v)]\, dv$$

$$\leq (1 - \mu)G_0(1 - \mu) - G_1(1 - \mu) \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{(1-\mu)^2}{2}}.$$

$$\mathbb{E}[L^2] \leq [(1 - \mu)^2 + 1]G_0(1 - \mu) + (1 - \mu)G(1 - \mu)$$

$$\leq \left[(1 - \mu)^2 + 1\right]\left[1 + \operatorname{erf}\left(\frac{1 - \mu}{\sqrt{2}}\right)\right].$$

The range of L follows immediately from null mass of $D_{v|y}$ for $v \leq -\Delta$, and from null loss for any $v \geq 1$. $\qquad\square$

The result above gives the bound on the expected value of the hinge loss for the model $h(x) = \beta^{\star T} x$ on the distribution D, that is, it bounds from above the generalization risk of that model, $R(\beta^\star) = \mathbb{E}[L]$. However, it does not give an upper bound on the empirical risk for the model $h(x) = \beta^{\star T} x$ on a specific training set with m samples and p features, sampled from D. This bound is given be the following lemma.

Lemma 5. *Let D be a (Δ, μ)-truncated subgaussian problem based on $\beta^\star$. Let $\hat{R}(\beta^\star)$ be the empirical risk associated with model $\beta^\star$ over a m-sample training set sampled* i.i.d. *from D. Then, with probability at least $1 - \delta$*

$$\hat{R}(\beta^\star) \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{(1-\mu)^2}{2}} + 4\frac{(\Delta + 1)\log(2/\delta)}{m} \tag{3.6}$$

$$+ 4\frac{\sqrt{\log(2/\delta)}}{\sqrt{m}}\left[(1 - \mu)^2 + 1\right]\left[1 + \operatorname{erf}\left(\frac{1 - \mu}{\sqrt{2}}\right)\right]$$

Proof. Consider m values $l_1, ..., l_m$ drawn from a univariate random variable L taking values in range in $[a, b] = [0, \Delta + 1]$, and with finite variance $s = \mathbb{V}\mathrm{ar}[L]$ and finite mean $R = \mathbb{E}[L]$. Let $\hat{R} = \frac{1}{m}\sum_{i=1}^{m} l_i$ be the empirical mean. Bernstein's inequality states that

$$\mathbb{P}(|\hat{R} - R| \geq t) \leq 2\exp\left(\frac{mt^2}{2(s^2 + (b - a)t)}\right).$$

That is, with probability at least $1 - \delta$,

$$\hat{R} \leq R + 4s\sqrt{\frac{\log(2/\delta)}{m}} + \frac{4(b-a)\log(2/\delta)}{m}.$$

We thus have

$$\hat{R}(\beta^\star) \leq \mathbb{E}[L] + 4\mathbb{V}\mathrm{ar}[L]\sqrt{\frac{\log(2/\delta)}{m}} + \frac{4(\Delta + 1)\log(2/\delta)}{m}$$

The bound follows from plugging in the bounds on expected value (eq. 3.4) and variance (eq. 3.5) of the loss. $\qquad\square$

We are now ready to analyze the behavior of empirical risk of models $\beta_p^\star$ on problems D_p as the number of all features p and the number of discriminative features p' grow.

Lemma 6. *For $p \to \infty$, consider a family of p-dimensional (Δ_p, μ_p)-truncated sub-gaussian problems D_p with underlying vectors $\beta_p^\star$. Assume that the vector $\beta_p^\star$ is sparse, it only has $p' = 1 + 2\log p$ non-zero coefficients. Further, assume that the mean μ_p diverges with the number of discriminative features p' as $\mu_p > c\sqrt{p'}$ for some $c > 1$ that captures how separated class centers are for individual informative features. As p grows, we allow scattering of the samples farther into the region dominated by the other class – specifically, we allow $\Delta_p \leq 2\log p$. Then, with probability at least $1 - \delta$, we have*

$$\hat{R}(\beta_p^\star) \leq \frac{1}{\sqrt{2\pi p}} + 4\frac{(2\log p + 1)\log(2/\delta)}{m}. \tag{3.7}$$

Proof. Under the assumption that μ_p grows at least as $c\sqrt{p'} = c\sqrt{1 + 2\log p}$, we have $\mu_p \geq 1 + \sqrt{2\log p}$, which leads to the bound on the first term of eq. (3.6), and to the second term approaching null limit.

Specifically, we have the following limit $\lim_{x\to\infty} c\sqrt{1 + kx}/[1 + \sqrt{kx}] = c$. Thus, under the assumption that μ_p grows at least as $c\sqrt{p'} = c\sqrt{1 + 2\log p}$, for $c > 1$, for sufficiently large p, we have $\mu_p \geq c\sqrt{1 + 2\log p} \geq 1 + \sqrt{2\log p}$, that is, $1 - \mu_p \leq -\sqrt{2\log p}$.

For the first term in eq. (3.6), e^x is an increasing function of x, we thus have the following upper bound

$$\frac{1}{\sqrt{2\pi}}e^{-\frac{(1-\mu_p)^2}{2}} \leq \frac{1}{\sqrt{2\pi}}e^{-\log p} = \frac{1}{\sqrt{2\pi}}\frac{1}{p}.$$

For the second term in eq. (3.6), we can show that $1 + \mathrm{erf}\left(\frac{1-\mu_p}{\sqrt{2}}\right)$ approaches null with the rate faster than $\frac{1}{p^2}$. Since $1 - \mu_p \leq -\sqrt{2\log p}$ and $\mathrm{erf}(x)$ is an increasing function of x, we have $1 + \mathrm{erf}\left(\frac{1-\mu_p}{\sqrt{2}}\right) \leq 1 + \mathrm{erf}(-\sqrt{2\log p})$. We also have

$$\frac{\mathrm{d}\left[1 + \mathrm{erf}\left(-\sqrt{2\log p}\right)\right]}{\mathrm{d}p} = -\frac{\sqrt{\frac{2}{\pi}}}{p^3\sqrt{\log p}}.$$

From the L'Hôpital's rule,

$$\lim_{p\to\infty} \frac{1 + \mathrm{erf}\left(-\sqrt{2\log p}\right)}{p^{-2}} = \lim_{p\to\infty} \frac{1}{2\sqrt{\pi}}\frac{1}{\sqrt{\log p}} = 0.$$

Thus, $\left[(1-\mu_p)^2 + 1\right]\left[1 + \mathrm{erf}\left(\frac{1-\mu_p}{\sqrt{2}}\right)\right] = O\left(\left[(-\sqrt{2\log p})^2 + 1\right]/p^2\right) = O\left(\log p/p^2\right)$. The second term quickly approaches null as p grows. $\qquad\square$

Sparse SVM involves regularized empirical risk, that is, minimization of a weighted sum of the empirical risk and the L_1 norm of the model β. Under the scenario of slowly increasing number of discriminative features, the Sparse SVM regularized empirical risk minimization is characterized by the following lemma.

Lemma 7. *For $p \to \infty$, consider a family of p-dimensional classification problems D_p as described in Lemma 6. For each D_p, consider the SparseSVM regularized empirical minimization problem (eq. 3.1)*

$$\arg\min_{\beta}\ \frac{1}{m}\sum_{i=1}^{m}\max(0, 1 - y_i\beta^T x_i) + \lambda\|\beta\|_1,$$

involving m-sample training set sampled from D_p. Then, for each p, with probability $1 - \delta$, there exist an empirical minimizer $\hat{\beta}_p$ of the problem above that can be found using a linear program (eq. 3.2), with L_1 norms of the primal and dual solutions, R_p

and r_p, respectively, bounded from above by

$$R_p \leq \frac{1}{\sqrt{2\pi}}\frac{m}{p} + 4(1 + 2\log p)\log(2/\delta)] + \lambda\sqrt{1 + 2\log p},$$

$$r_p \leq \frac{1}{\sqrt{2\pi}}\frac{1}{p} + \frac{4(1 + 2\log p)\log(2/\delta)}{m} + \lambda\sqrt{1 + 2\log p}.$$

Proof. For any unit L_2-norm vector β with p' non-zero entries, the highest L_1 norm is achieved if all the p' coordinates are equal to $1/\sqrt{p'}$. Thus, $\|\beta_p^\star\|_1 \leq \sqrt{p'}$ for $\beta_p^\star$ with unit L_2-norm and at most p' non-zero coefficients. On the training set, the minimizer $\hat{\beta}_p$ has lowest objective function of all possible β, including $\beta_p^\star$. Thus, we have

$$\hat{R}(\hat{\beta}_p) + \lambda\|\hat{\beta}_p\|_1 \leq \hat{R}(\beta_p^\star) + \lambda\|\beta_p^\star\|_1$$

$$\leq \frac{1}{\sqrt{2\pi}}\frac{1}{p} + 4\frac{(1 + 2\log p)\log(2/\delta)}{m} + \lambda\sqrt{1 + 2\log p}.$$

From strong duality, we have $r_p = \hat{R}(\hat{\beta}_p) + \lambda\|\hat{\beta}_p\|_1$. The norm R_p of the primal solution does not involve averaging the losses $\max(0, 1 - y_i\beta^T x_i)$. Instead, the losses are added up, that is, R_p includes the term $m\hat{R}(\hat{\beta}_p)$ instead of the empirical risk. $\square$

Lemma 7 directly leads to Theorem 3.

3.4.3 Speedup Compared to Classical Sparse SVM for Truncated Subgaussian Problems

For truncated subgaussian classification problems, the proposed algorithm for solving Sparse SVM optimization problem using quantum LP solver has sublinear complexity. We show here that classical algorithms for the same class of problems cannot be faster than linear, through reduction from the search problem. As a result, Quantum Sparse SVM holds polynomial complexity advantage to any classical algorithm for Sparse SVM on this class of problems.

Consider the following truncated subgaussian problem with m samples and p features. All samples from the positive class are vectors with p ones. On the other hand, all samples from the negative class have negative one on the same subset of $p' = O(logp)$

features, and have the value of positive one for all other features. This setup meets the criteria for a sparse truncated subgaussian classification problem according to Definition 2. Finding the optimal classification model involves finding at least one of the p' features where a positive sample differs in value from a negative sample, out of p possibilities. Successfully training the classifier for this particular classification problem translates to successfully solving the search problem. Search problem has classical complexity of $O(p)$, providing lower bound on classical algorithms for training Sparse SVMs for truncated subgaussian problems. In the quantum setting, the problem can be solved by Groveer's search algorithm, which attains the quantum lower bound of $O\left(\sqrt{p}\right)$. The quantum lower bound shows that the proposed method involving a quantum LP solver is optimal, up to slower growing polylog terms.

We also note that since all the sample feature vectors are composed of elements with the same, unit magnitude, and thus all rows and all columns have the same norm, it is not possible to use sampling methods to construct a dequantized version of Quantum Sparse SVM with sublinear complexity for the truncated subgaussian problems.

3.5   Related Work

Considerable effort has been devoted into designing fast classical algorithms for training SVMs, although the focus is on the traditional SVM that involves the L_2 regularizer, which results in a strongly convex objective function even for hinge loss, as opposed to Sparse SVM, where the objective function is piece-wise linear, and thus convex but not strongly convex. The decomposition-based methods such as SMO [48] are able to efficiently manage problems with large number of features p, but their computational complexities are super-linear in m. Other training strategies [7, 49, 50] are linear in m but scale quadratically in p in the worst case. In more recent work, the number of features p is replaced by the average sparsity of samples, s, that is, the average number of non-zero feature values per sample. The SVM^{perf} algorithm [51]

for linear SVM scales as $O(sm)$. The Pegasos algorithm [52] improves the complexity to $\tilde{O}\left(s/(\lambda\epsilon)\right)$, where λ, and ϵ are the regularization parameter of SVM and the error of the solution, respectively. However, sublinear sparsity of input samples is a strong assumption, and holds only in some special scenarios, such as Bag-of-Words encoding in natural language processing, but rarely holds for data describing physical phenomena, including biomedical data.

Beyond the classical realm, instead of the solver [26] we used, two alternative quantum solvers based on the interior point approach were also introduced recently, an SDP/LP [53] and a dedicated LP [54] solver. However, both of the quantum interior point solvers come with computational complexity challenges that make them less appealing for training sparse models. The interior point dedicated LP solver [54] has complexity of $O(\sqrt{n}L/\epsilon)$, where $L = O(\log n)$ on average but $L = mn$ in the worst case. Crucially, the complexity also depends on the sparsity of the constraint matrix A. However, in SVMs, A is composed of elements $y_i x_i^j$, the feature values, which are unlikely to be spare in many applications, including those involving biomedical data. The SDP/LP solver [53] has cubic dependence on the condition number κ of the intermediate solution matrices of the SDP. In the LP variant, κ is ratio of largest to smallest feature weight magnitude, which can easily explode in machine learning applications, where some feature weights are null or close to null. Even if the problem is well-conditioned, the interior-point SDP/LP solver has complexity of $\tilde{O}(m^2)$.

An alternative to using a quantum LP solver is to use quantum gradient descent. However, most quantum gradient descent approaches only work reliably for small number of steps, with probability of following the gradient path decreasing exponentially with the number of gradient updates. Recently, a quantum descent method circumventing this problem has been proposed [55], but only for objective functions where the norm of the gradient decreases with each step, such as quadratic functions. Sparse SVM objective function is piece-wise linear and non-differentiable, with possibility of

subgradient norm increase when a step from one linear region to another is made.

Recently, construction of fast classical algorithm based on quantum machine learning methods using sampling-based data structure has been proposed [56], and has led to a dequantization of LS SVM and other machine learning methods under low-rank assumption [57]. A dequantized SDP solver has also been proposed [58] for SDP problems with low rank input matrices. While a linear program $\mathrm{LP}(c, A, b)$ can be seen as an SDP with diagonal matrices formed by c and rows of A, for the LP resulting from Sparse SVM these diagonal matrices are full-rank. Thus, the proposed quantum Sparse SVM cannot be dequantized using existing approaches as a variant of dequantized SDP.

CHAPTER 4

QUANTUM SEMI-SUPERVISED KERNEL LEARNING

4.1   Introduction

In the previous chapter, we explored the possibility of achieving quantum speedup for Sparse SVM, which is categorized as a supervised machine learning problem. While much progress has been made in developing quantum algorithms for supervised learning, it has been recently advocated that the focus should shift to the unsupervised and semi-supervised setting. To maximize the potential of finding applications on near-term quantum computers, it is suggested to explore problems that are considered hard and intractable for the classical machine learning community, such as generative models in unsupervised and semi-supervised learning [59]. In this chapter, we focus on a setting where accessing the labels for all data points is hard to obtain and present a semi-supervised quantum algorithm for a kernel learning problem.

In many domains, the most laborious part of assembling a training set is collecting sample labels. Thus, in many scenarios, in addition to the labeled training set, we have access to many more feature vectors with missing labels. One way of utilizing these additional data points to improve the classification model is through semi-supervised learning. In semi-supervised learning, we are given m observations $x_1, ..., x_m$ drawn from the marginal distribution $p(x)$, where the l ($l \ll m$) first data points come with labels $y_1, ..., y_l$ drawn from conditional distribution $p(y|x)$. Semi-supervised learning algorithms exploit the underlying distribution of the data to improve classification accuracy on unseen samples.

4.1.1 Our Contribution

In this chapter, we introduce a quantum algorithm for semi-supervised training of a kernel support vector machine classification model. We start with the existing Quantum Least Squares Support Vector Machine (Quantum LS-SVM) [18], and use techniques from sample-based Hamiltonian simulation [60] to add a semi-supervised term based on Laplacian SVM [61]. As is standard in quantum machine learning [16], the algorithm accesses training samples and the adjacency matrix of the graph connecting samples via a quantum oracle. We show that, with respect to the oracle, the proposed algorithm achieves the same quantum speedup as LS-SVM, that is, adding the semi-supervised term does not lead to increased computational complexity.

4.2 Preliminaries

In this section, we setup the background that we need for describing quantum semi-supervised algorithm. We first describe briefly the semi-supervised kernel machines and show training a semi-supervised LS-SVM problem can be reformulated as a system of linear equations. LS-SVM minimizes quadratic loss and quadratic regularizer $\sum_{i=1}^{m}(y_i - \beta^T x_i)^2 + \lambda\|\beta\|_2^2$ over the training set $\{(x_i, y_i)\}$ and with standard techniques can be re-written as a system of linear equations. Quantum LS-SVM utilizes a quantum linear system solver known as HHL algorithm [18]. Second, for preparing the quantum setting, we describe the utilized quantum subroutines such as HHL algorithm and LMR technique.

4.2.1 Semi-Supervised Kernel Machines

Reproducing Kernel Hilbert Spaces. Binary classification models take the form of a function $h : X \to \mathbb{R}$ from some Hilbert space of functions $\mathcal{H}$. Consider $L_2(X)$, the space of all square-integrable functions $X \to \mathbb{R}$. In L_2, closeness in norm does not imply everywhere pointwise closeness of two functions. Difference between $f(x)$ and

$g(x)$ can be arbitrary large for $x \in S \subset X$ even if $||f - g||_{\mathcal{H}} = 0$, as long as S has measure of 0.

Not all classes of functions exhibit this problem. Consider a family of Dirac evaluation functionals $\mathcal{H} \to \mathbb{R}$, indexed by $t \in X$, and defined for Hilbert spaces of functions as $F_t[h] = h(t)$. A linear functional $F : \mathcal{H} \to \mathbb{R}$ is bounded if $\exists M \in \mathbb{R}$: $\forall h \in \mathcal{H} \quad |F[h]| \leq M||h||_{\mathcal{H}}$. Consider a space $\mathcal{H}$ in which evaluation functionals for all $t \in X$ are bounded. Then, for any $f, g \in \mathcal{H}$, small $||f - g||_{\mathcal{H}}$ implies small $|f(t) - g(t)|$, everywhere on X. Riesz representation theorem guarantees that for each bounded evaluation functional F_t, there exists a unique function $K_t \in \mathcal{H}$ such that $F_t[h] = h(t) = \langle K_t, h \rangle_{\mathcal{H}}$. Function $K_t : X \to \mathbb{R}$ is called the *representer* of $t \in X$. Any space of functions in which every evaluation operator F_t is bounded, and thus has a corresponding representer K_t, is called a *Reproducing Kernel Hilbert Space* (RKHS). In RKHS, by symmetry of inner product, $K_t(s) = \langle K_t, K_s \rangle_{\mathcal{H}} = K_s(t)$. We can thus define a function $K : X \times X \to \mathbb{R}$ with values $K(s, t) = K_s(t)$, such that $\forall x, y \in X \quad \exists K_x, K_y \in \mathcal{H} : \quad \langle K_x, K_y \rangle_{\mathcal{H}} = K(x, y)$.

Any symmetric and positive definite function $K : X \times X \to \mathbb{R}$ (that is, a function that fulfills the Mercer condition, $\int_X \int_X c(x)K(x, y)c(y)dxdy \geq 0 \quad \forall c \in \mathcal{H}$) is called a *reproducing kernel*. A reproducing kernel gives rise to functions $K_t : X \to \mathbb{R}$ defined by fixing t and defining $K_t(x) = K(t, x)$. We can construct an inner product space as the span of functions $K_t : X \to \mathbb{R}$ for all $t \in X$, with the inner product defined through representers as $\langle f, g \rangle_{\mathcal{H}} = \sum_{j=1}^{n} \sum_{j=1}^{n'} c_j c_j' K(t_j, t_j')$. The Moore-Aronszajn theorem states that the space defined this way can be completed, the resulting RKHS space is unique, and K is the reproducing kernel in that space.

Discrete-topology Gradients of Functions in RKHS. A simple way of measuring local variability of a function $X \to \mathbb{R}$ is through its gradient; for example, functions with Lipschitz-continuous gradient cannot change too rapidly. In manifold learning, instead of analyzing gradient using the topology of X, some other topology is used. In

particular, in Laplacian SVM and related methods, a discrete topology of a graph G connecting training points in X is used.

For a given undirected graph G with a set of m vertices, V, and a set of n edges, E, let us define a Hilbert space $\mathcal{H}_V$ of functions $h : V \to \mathbb{R}$ with inner product $\langle f, g \rangle_V = \sum_{v \in V} f(v)g(v)$, and a Hilbert space $\mathcal{H}_E$ of functions $\psi : E \to \mathbb{R}$ with inner product $\langle \psi, \phi \rangle_E = \sum_{e \in E} \psi(e)\phi(e) = \sum_{u \sim v} \psi([u,v])\phi([u,v])$. We can define a linear operator $\nabla : \mathcal{H}_V \to \mathcal{H}_E$ such that:

$$\nabla h([u,v]) = \sqrt{G_{u,v}} h(u) - \sqrt{G_{u,v}} h(v) = -\nabla h([v,u]).$$

The operator ∇ can be seen as a discrete counterpart to the gradient of a function – given a function h over vertices, for a given point u in the domain of h, ∇ over different vertices v gives us a set of values showing the change of h over all directions from u, that is, all edges incident to it. We define graph equivalent of divergence, a linear operator $\mathrm{div} : \mathcal{H}_E \to \mathcal{H}_V$ such that $-\mathrm{div}$ is the adjoint of ∇, that is, $\langle \nabla[h], \psi \rangle_E = \langle h, -\mathrm{div}[\psi] \rangle_V$. Finally, we define *graph Laplacian*, a linear operator $\Delta : \mathcal{H}_V \to \mathcal{H}_V$

$$\Delta[h] = -\frac{1}{2}\mathrm{div}[\nabla[h]] = D_v h(v) - \sum_{u \sim v} G_{u,v} h(u).$$

The graph Laplacian operator Δ is self-adjoint and positive semi-definite, and the squared norm of the graph gradient can be captured through it as

$$\frac{1}{2}||\nabla h||_E^2 = \langle \Delta[h], h \rangle_V = \frac{1}{2} \sum_{u \sim v} G_{u,v} (\bar{h}_u - \bar{h}_v)^2 = \bar{h}^T L \bar{h},$$

where $\Delta[h]$ can be seen as $\Delta[h] = L\bar{h}$, a multiplication of vector of function values over vertices, $\bar{h}$, by *combinatorial graph Laplacian matrix* L such that $L[i,j] = D_j - G_{i,j}$.

Semi-Supervised Least-Squares Kernel Support Vector Machines. Consider a problem where we are aiming to find predictors $h(x) : X \to \mathbb{R}$ that are functions from a RKHS defined by a kernel K. In Semi-Supervised LS-SVMs in RKHS, we are looking

for a function $h \in \mathcal{H}$ that minimizes

$$\min_{h \in \mathcal{H}, b \in \mathbb{R}} \quad \frac{\gamma}{2} \sum_{i=1}^{l} (y_i - (h(x_i) + b))^2 + \frac{1}{2}||h||_{\mathcal{H}}^2 + \frac{1}{2}||\nabla h||_E^2,$$

where γ is a user define constant allowing for adjusting the regularization strength. The Representer Theorem states that if $\mathcal{H}$ is RKHS defined by kernel $K : X \times X \to \mathbb{R}$, then the solution minimizing the problem above is achieved for a function h that uses only the representers of the training points, that is, a function of the form $h(x) = \sum_{j=1}^{m} c_j K_{x_j}(x) = \sum_{j=1}^{m} c_j K(x_j, x)$. Thus, we can translate the problem from RKHS into a constrained quadratic optimization problem over finite, real vectors

$$\min_{c, \xi, b} \quad \frac{\gamma}{2} \sum_{i=1}^{m} \xi_i^2 + \frac{1}{2} c^T K c + \frac{1}{2} c^T K L K c \qquad \text{s.t.} \quad 1 - y_i \left[b + \sum_{j=1}^{m} c_j K[i,j] \right] = \xi_i$$

where $l \leq m$ is the number of training points with labels (these are grouped at the beginning of the training set), and $\bar{h} = Kc$, since function h is defined using representers K_{x_i}. The semi-supervised term, the squared norm of the graph gradient of h, $1/2||\nabla h||_E^2$, penalizes large changes of function h over edges of graph G. In defining the kernel K and the Laplacian L and in the two regularization terms, we use all m samples. On the other hand, in calculating the empirical quadratic loss, we only use the first l samples.

The solution to the Semi-Supervised LS-SVMs is given by solving the following $(m+1) \times (m+1)$ system of linear equations

$$\begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & K + KLK + \gamma^{-1}\mathbb{1} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \tag{4.1}$$

where $\mathbf{y} = (y_1, ..., y_m)^T$, $\mathbf{1} = (1, ..., 1)^T$, $\mathbb{1}$ is identity matrix, K is kernel matrix, L is the graph Laplacian matrix, γ is a hyperparameter and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_m)^T$ is the vector of Lagrangian multipliers.

4.2.2 Quantum Computing and Quantum LS-SVM

Quantum Linear Systems of Equations. Given an input matrix $A \in \mathbb{C}^{n \times n}$ and a vector $b \in \mathbb{C}^n$, the goal of linear system of equations problem is finding $x \in \mathbb{C}^n$ such that $Ax = b$. When A is Hermitian and full rank, the unique solution is $x = A^{-1}b$. If A is not a full rank matrix then A^{-1} is replaced by the Moore-Penrose pseudo-inverse. HHL algorithm introduced an analogous problem in quantum setting: assuming an efficient algorithm for preparing b as a quantum state $b = \sum_{i=1}^{n} b_i |i\rangle$ using $\lceil \log n \rceil + 1$ qubits, the algorithm applies quantum subroutines of phase estimation, controlled rotation, and inverse of phase estimation to obtain the state

$$|x\rangle = \frac{A^{-1}|b\rangle}{\| A^{-1}|b\rangle \|}. \tag{4.2}$$

Intuitively, HHL algorithm works as follows: if A has spectral decomposition $A = \sum_{i=1}^{n} \lambda_i v_i v_i^T$ (where λ_i and v_i are corresponding eigenvalues and eigenstates of A), then A^{-1} maps $\lambda_i v_i \mapsto \frac{1}{\lambda_i} v_i$. The vector b also can be written as the linear combination of the A's eigenvectors v_i as $b = \sum_{i=1}^{n} \beta_i v_i$ (we are not required to compute β_i). Then $A^{-1}b = \sum_{i=1}^{n} \beta_i \frac{1}{\lambda_i} v_i$. In general A and A^{-1} are not unitary (unless all A's eigenvalues have unit magnitude), therefore we are not able to apply A^{-1} directly on $|b\rangle$. However, since $U = e^{iA} = \sum_{i=1}^{n} e^{i\lambda_i} v_i v_i^T$ is unitary and has the same eigenvectors as A and A^{-1}, one can implement U and powers of U on a quantum computer by Hamiltonian simulation techniques; clearly for any expected speed-up, one need to enact e^{iA} efficiently. The HHL algorithm uses the phase estimation subroutine to estimate an approximation of λ_i up to a small error. The Next step computes a conditional rotation on the approximated value of λ_i and an auxiliary qubit $|0\rangle$ and outputs $\frac{1}{\lambda_i}|0\rangle + \sqrt{1 - \frac{1}{\lambda_i^2}}|1\rangle$. The last step involves the inverse of phase estimation and quantum measurement for eliminating of garbage qubits and for returning the desired state $|x\rangle = A^{-1}|b\rangle = \sum_{i=1}^{n} \beta_i \frac{1}{\lambda_i} v_i$.

Quantum Estimation of the Kernel Matrix. Quantum LS-SVM uses density operator formalizm and partial trace (see Section 2.2.1) to represent the computation involving the kernel matrix. To obtain kernel matrix K as a density matrix, quantum LS-SVM [62] relies on partial trace, and on a quantum oracle that can convert, in superposition, each data point $\{x_i\}_{i=1}^m$, $x_i \in \mathbb{R}^p$ to a quantum state $|x_i\rangle = \frac{1}{\|x_i\|} \sum_{t=1}^p (x_i)_t |t\rangle$, where $(x_i)_t$ refers to the tth feature value in data point x_i and assuming the oracle is given $\|x_i\|$ and y_i. Vector of the labels is given in the same fashion as $|y\rangle = \frac{1}{\|y\|} \sum_{i=1}^m y_i |i\rangle$. For preparation the normalized Kernel matrix $K' = \frac{1}{\text{tr}(K)} K$ where $K = X^T X$, we need to prepare a quantum state combining all data points in quantum superposition $|X\rangle = \frac{1}{\sqrt{\sum_{i=1}^m \|x_i\|^2}} \sum_{i=1}^m |i\rangle \otimes \|x_i\| |x_i\rangle$. The normalized Kernel matrix is obtained by discarding the training set state,

$$K' = \text{Tr}_2(|X\rangle\langle X|) = \frac{1}{\sum_{i=1}^m \|x_i\|^2} \sum_{i,j=1}^m \|x_i\| \|x_j\| \langle x_i|x_j\rangle |i\rangle\langle j|. \tag{4.3}$$

The approach used above to construct density matrix corresponding to linear kernel matrix can be extended to polynomial kernels [62].

LMR Technique for Density Operator Exponentiation. In HHL-based quantum machine learning algorithms , including in the method proposed here, matrix A for the Hamiltonian simulation within the HHL algorithm is based on data. For example, A can contain the kernel matrix K captured in the quantum system as a density matrix. Then, one need to be able to efficiently compute $e^{-iK\Delta t}$, where K is scaled by the trace of kernel matrix. Since K is not sparse, a strategy similar to [63] is adapted for the exponentiation of a non-sparse density matrix:

$$\text{Tr}_1\left\{e^{-iS\Delta t}(K \otimes \sigma)e^{iS\Delta t}\right\} = \sigma - i\Delta t[K, \sigma] + O\left(\Delta t^2\right) \approx e^{-iK\Delta t}\sigma e^{iK\Delta t}, \tag{4.4}$$

where $S = \sum_{i,j} |i\rangle\langle j| \otimes |j\rangle\langle i|$ is the swap operator and the facts $\text{Tr}_1\{S(K \otimes \sigma)\} = K\sigma$ and $\text{Tr}_1\{(K \otimes \sigma)S\} = \sigma K$ are used. The equation (4.4) summarizes the LMR

technique: approximating $e^{-iK\Delta t}\sigma e^{iK\Delta t}$ up to error $O(\Delta t^2)$ is equivalent to simulating a swap operator S, applying it to the state $K\otimes\sigma$ and discarding the first system by taking partial trace operation. Since the swap operator is sparse, its simulation is efficient. Therefore the LMR trick provides an efficient way to approximate exponentiation of a non-sparse density matrix.

Quantum LS-SVM. Quantum LS-SVM [62] uses partial trace to construct density operator corresponding to the kernel matrix K. Once the kernel matrix K becomes available as a density operator, the quantum LS-SVM proceeds by applying the HHL algorithm for solving the system of linear equations associated with LS-LSVM, using the LMR technique for performing the density operator exponentiation $e^{-iK\Delta t}$ where the density matrix K encodes the kernel matrix.

4.3   Quantum Semi-Supervised Least Square SVM

We proposed here a quantum algorithm for solving Semi-Supervised Least Square SVM. Semi-supervised LS SVM involves solving the following system of linear equations

$$\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & K + KLK + \gamma^{-1}\mathbb{1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} = A^{-1} \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \tag{4.5}$$

In quantum setting the task is to generate $|b, \boldsymbol{\alpha}\rangle = \hat{A}^{-1}|0, \mathbf{y}\rangle$, where the normalized $\hat{A} = \dfrac{A}{Tr(A)}$. The linear system differs from the one in LS-SVM in that instead of K, we have $K + KLK$. While this difference is of little significance for classical solvers, in quantum systems we cannot just multiply and then add the matrices and then apply quantum LS-SVM – we are limited by the unitary nature of quantum transformations.

In order to obtain the solution to the quantum Semi-Supervised Least Square SVM, we will use the following steps. First, we will read in the graph information to obtain scaled graph Laplacian matrix as a density operator. Next, we will use polynomial Hermitian exponentiation for computing the matrix inverse $(K + KLK)^{-1}$.

4.3.1 Quantum Input Model for the Graph Laplacian

In the semi-supervised model used here, we assume that we have information on the similarity of the training samples, in a form of graph G that uses n edges to connect similar training samples, represented as m vertices. We assume that for each sample, G contains its k most similar other samples, that is, the degree of each vertex is d. To have the graph available as a quantum density operator, we observe that the graph Laplacian L is the Gram matrix of the rows of the $m \times n$ graph incidence matrix G_I, $L = G_I G_I^T$. We assume oracle access to the graph adjacency list, allowing us to construct, in superposition, states corresponding to rows of the graph incidence matrix G_I

$$|v_i\rangle = \frac{1}{\sqrt{d}} \sum_{t=1}^{n} G_I[i,t] |t\rangle.$$

That is, state $|v_i\rangle$ has probability amplitude $\frac{1}{\sqrt{d}}$ for each edge $|t\rangle$ incident with vertex i, and null probability amplitude for all other edges. In superposition, we prepare a quantum state combining rows of the incidence matrix for all vertices, to obtain

$$|G_I\rangle = \frac{1}{\sqrt{md}} \sum_{i=1}^{m} |i\rangle \otimes |v_i\rangle$$

The graph Laplacian matrix L, composed of inner products of the rows of G_I, is obtained by discarding the second part of the system,

$$L = Tr_2(|G_I\rangle\langle G_I|) = \frac{1}{md} \sum_{i,j=1}^{m} |i\rangle\langle j| \otimes d\langle v_i|v_j\rangle = \frac{1}{m} \sum_{i,j=1}^{m} \langle v_i|v_j\rangle |i\rangle\langle j|. \qquad (4.6)$$

4.3.2 Polynomial Hermitian Exponentiation for Semi Supervised Learning

For computing the matrix inverse $(K + KLK)^{-1}$ on a quantum computer that runs our quantum machine algorithm and HHL algorithm as a subroutine, we need to efficiently compute $e^{-i(K+KLK)\Delta t} \sigma e^{i(K+KLK)\Delta t}$. For this purpose we adapt the generalized LMR technique for simulating Hermitian polynomials proposed in [60] to the specific

57

case of $e^{-i(K+KLK)\Delta t}\sigma e^{i(K+KLK)\Delta t}$. Simulation of $e^{-iK\Delta t}$ follows from the original LMR algorithm, and therefore we focus here only on simulation $e^{-iKLK\Delta t}$. The final dynamics $(K+KLK)^{-1}$ can be obtained by sampling from the two separate output states for $e^{-iKLK\Delta t}$ and $e^{-iK\Delta t}$.

Simulating $e^{iKLK\Delta t}$. Let $D(\mathcal{H})$ denote the space of density operators associated with state space $\mathcal{H}$. Let $K^\dagger, K, L \in D(\mathcal{H})$ be the density operators associated with the kernel matrix and the Laplacian, respectively. We will need two separate systems with the kernel matrix K, to distinguish between them we will denote the first as $K^\dagger$ and the second as K; since K is real and symmetric, these are indeed equal. The kernel and Laplacian matrices $K^\dagger, K, L$ are not sparse therefore we adapt the generalized LMR technique for simulating Hermitian polynomials for our specific case $B = K^\dagger L K$.

For adapting the generalized LMR technique to our problem we need to generate a quantum state $\rho' = |0\rangle\langle 0| \otimes \rho'' + |1\rangle\langle 1| \otimes \rho'''$ with $Tr(\rho'' + \rho''') = 1$, such that

$$\text{Tr}_1\left\{\text{Tr}_3\left\{e^{-iS'\Delta}\left(\rho' \otimes \sigma\right)e^{iS'\Delta}\right\}\right\} = \sigma - i[B,\sigma] + O(\Delta^2) = e^{-iBt}\sigma e^{iBt} + O(\Delta^2), \quad (4.7)$$

where $B = \rho'' - \rho''' = \frac{1}{2}K^\dagger L K + \frac{1}{2}KLK^\dagger = KLK$ and $S' := |0\rangle\langle 0|\otimes S + |1\rangle\langle 1|\otimes(-S)$ is a controlled partial swap in the forward $(+S)$ and backward direction $(-S)$ in time, and

$$e^{-iS'\Delta} = |0\rangle\langle 0| \otimes e^{-iS\Delta} + |1\rangle\langle 1| \otimes e^{iS\Delta}.$$

Therefore with one copy of ρ', we obtain the simulation of $e^{-iB\Delta}$ up to error $O(\Delta^2)$. If we choose the time slice $\Delta = \delta/t$ and repeating the above procedure for t^2/δ times, we are able to simulate e^{-iBt} up to error $O(\delta)$ using $n = O(t^2/\delta)$ copies of ρ'.

Generating $\rho' = |0\rangle\langle 0| \otimes \rho'' + |1\rangle\langle 1| \otimes \rho'''$. Figure 3 shows the quantum circuit for creating $\rho' = |0\rangle\langle 0|\otimes\rho''+|1\rangle\langle 1|\otimes\rho'''$ such that $Tr(\rho''+\rho''') = 1$ and $B = \rho'' - \rho''' = KLK$.

The analysis of the steps preformed by the circuit depicted in Fig.3 is as follows.
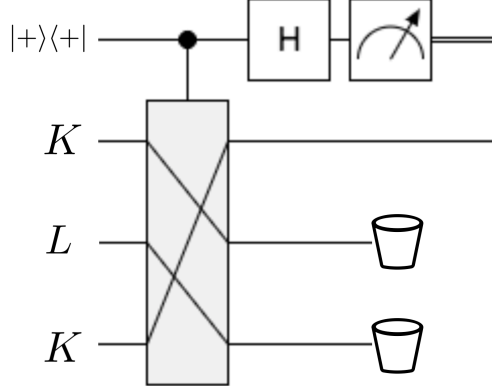
Figure 3: Quantum circuit for creating $\rho' = |0\rangle\langle 0| \otimes \rho'' + |1\rangle\langle 1| \otimes \rho'''$. The circuit is to be read from left-to-right. Each wire at left shows its corresponding input state. The vertical rectangle denotes the cyclic permutation operator P on K, L, K defined in (4.8). H is the Hadamard gate, and the waste bins show partial trace. The measurement on the first quantum state is in computational basis.

Let P be the cyclic permutation of three copies of $\mathcal{H}_A$ that operates as $P |j_1, j_2, j_3\rangle = |j_3, j_1, j_2\rangle$. In operator form it can be written as

$$P := \sum_{j_1, j_2, j_3 = 1}^{\dim \mathcal{H}_A} |j_3\rangle \langle j_1| \otimes |j_1\rangle \langle j_2| \otimes |j_2\rangle \langle j_3| \tag{4.8}$$

The input state to the circuit depicted in Fig. 3 is

$$|+\rangle\langle+| \otimes K^\dagger \otimes L \otimes K = \frac{1}{2} \sum_{i,j \in \{0,1\}} |i\rangle\langle j| \otimes K^\dagger \otimes L \otimes K.$$

Applying P on $K^\dagger, L, K$ gives

$$I = \frac{1}{2}[|0\rangle\langle 0| \otimes K^\dagger \otimes L \otimes K + |0\rangle\langle 1| \otimes \left(K^\dagger \otimes L \otimes K\right) P$$

$$+ |1\rangle\langle 0| \otimes P \left(K^\dagger \otimes L \otimes K\right) + |1\rangle\langle 1| \otimes P \left(K^\dagger \otimes L \otimes K\right) P.$$

After discarding the third and second register sequentially by applying corresponding

partial trace operators, we get

$$II = \text{Tr}_2\left[\text{Tr}_3(I)\right] = |0\rangle\langle 0| \otimes \frac{1}{2}K^\dagger + |0\rangle\langle 1| \otimes \frac{1}{2}K^\dagger LK + |1\rangle\langle 0| \otimes \frac{1}{2}KLK^\dagger + |1\rangle\langle 1| \otimes \frac{1}{2}K,$$

in this step KLK term where the last line obtained from

$$\text{Tr}_2\left[\text{Tr}_3\left[\left(K^\dagger \otimes L \otimes K\right)\text{P}\right]\right] = K^\dagger LK,$$

$$\text{Tr}_2\left[\text{Tr}_3\left[\text{P}(K^\dagger \otimes L \otimes K)\right]\right] = KLK^\dagger,$$

$$\text{Tr}_2\left[\text{Tr}_3\left[\text{P}(K^\dagger \otimes L \otimes K)\,\text{P}\right]\right] = K.$$

After applying a Hadamard gate $H = \frac{1}{\sqrt{2}}[(|0\rangle + |1\rangle)\langle 0| + (|0\rangle - |1\rangle)\langle 1|]$ on the first qubit of II, we get

$$III = H \otimes \mathbb{1}(II)H \otimes \mathbb{1} =$$

$$\frac{1}{2}\left(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| + |1\rangle\langle 1|\right) \otimes \frac{1}{2}K^\dagger + \frac{1}{2}\left(|0\rangle\langle 0| - |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|\right) \otimes \frac{1}{2}K^\dagger LK$$

$$+ \frac{1}{2}\left(|0\rangle\langle 0| + |0\rangle\langle 1| - |1\rangle\langle 0| - |1\rangle\langle 1|\right) \otimes \frac{1}{2}KLK^\dagger + \frac{1}{2}\left(|0\rangle\langle 0| - |0\rangle\langle 1| - |1\rangle\langle 0| + |1\rangle\langle 1|\right) \otimes \frac{1}{2}K$$

$$= |0\rangle\langle 0| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger + \frac{1}{2}K^\dagger LK + \frac{1}{2}KLK^\dagger + \frac{1}{2}K\right) + |0\rangle\langle 1| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger - \frac{1}{2}K^\dagger LK + \frac{1}{2}KLK^\dagger - \frac{1}{2}K\right)$$

$$+ |1\rangle\langle 0| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger + \frac{1}{2}K^\dagger LK - \frac{1}{2}KLK^\dagger - \frac{1}{2}K\right) + |1\rangle\langle 1| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger - \frac{1}{2}K^\dagger LK - \frac{1}{2}KLK^\dagger + \frac{1}{2}K\right).$$

The last step is applying a measurement in computational basis $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ on the first register to obtain our desired state ρ',

$$IV = |0\rangle\langle 0| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger + \frac{1}{2}K^\dagger LK + \frac{1}{2}KLK^\dagger + \frac{1}{2}K\right)$$

$$+ |1\rangle\langle 1| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger - \frac{1}{2}K^\dagger LK - \frac{1}{2}KLK^\dagger + \frac{1}{2}K\right)$$

We can see that by defining $\rho'' = \frac{1}{2}\left(\frac{1}{2}K^\dagger + \frac{1}{2}K^\dagger LK + \frac{1}{2}KLK^\dagger + \frac{1}{2}K\right)$ and $\rho''' = \frac{1}{2}\left(\frac{1}{2}K^\dagger - \frac{1}{2}K^\dagger LK - \frac{1}{2}KLK^\dagger + \frac{1}{2}K\right)$ the final state is in the form of $\rho' = |0\rangle\langle 0| \otimes \rho'' + |1\rangle\langle 1| \otimes \rho'''$ where $Tr(\rho'' + \rho''') = 1$, and we obtain $\rho'' - \rho''' = \frac{1}{2}K^\dagger LK + \frac{1}{2}KLK^\dagger = B$.

Now with having the output state ρ' we are ready to apply the generalized LMR in

(4.7) to simulate $e^{-iKLK\Delta t}\sigma e^{iKLK\Delta t}$ up to error $O(\Delta^2)$. Comparing the LMR technique in equation (4.4) with the generalized LMR for the spacial case of KLK in equation (4.7), we see approximating $e^{-iKLK\Delta t}\sigma e^{iKLK\Delta t}$ up to error $O(\Delta t^2)$ is equivalent to simulating the controlled partial swap operator S', applying it to the state $\rho' \otimes \sigma$ and discarding the third and first systems by taking partial trace operations, respectively. Since S' is also sparse, and its simulation is efficient, the generalized LMR technique offers an efficient approach for simulating $e^{iKLK\Delta t}$.

4.3.3 Quantum Semi-Supervised LS-SVM Algorithm and its Complexity

---

Algorithm 1 Quantum Semi-Supervised LS-SVM

Input: The datapoint set $\{x_1, ...x_l, ...x_m\}$ with the first l data points labeled and the rest unlabeled, $\mathbf{y} = (y_1, ..., y_l)$ and the graph G

Output: The classifier $|\alpha, b\rangle = A^{-1}|y\rangle$

1: **Quantum data preparation.** Encode classical data points into quantum data points using quantum oracles $O_x : \{x_1, ...x_l, ...x_m\} \mapsto |X\rangle = \frac{1}{\sqrt{\sum_{i=1}^{m}\|x_i\|^2}} \sum_{i=1}^{m} |i\rangle \otimes \| x_i \| |x_i\rangle$ and $O_x : \mathbf{y} \mapsto |y\rangle$.

2: **Quantum Laplacian preparation.** Prepare quantum density matrix using oracle access to G.

3: **Matrix inversion.** Compute the matrix inversion $|\alpha, b\rangle = A^{-1}|y\rangle$ via HHL algorithm. A quantum circuit for the HHL algorithm has three main steps:

4:     *Phase estimation,* including efficient Hamiltonian simulation involving KLK (Section 4.3.2)

5:     *Controlled rotation*

6:     *Uncomputing*

7: **Classification.** Based on **Swap test** algorithm, same as in Quantum LS-SVM.

The quantum LS-SVM in [62] offers exponential speedup $O(\log mp)$ over the classical time complexity for solving SVM as a quadratic problem, which requires time

$O(log(\epsilon^{-1})poly(p, m))$, where ϵ is the desired error. The exponential speedup in p occurs as the result of fast quantum computing of kernel matrix, and relies on the existence of efficient oracle access to data. The speedup on m is due to applying quantum matrix inversion for solving LS-SVM, which is inherently due to fast algorithm for exponentiation of a resulting non-sparse matrix. Our algorithm introduces two additional steps: preparing the Laplacian density matrix, and Hamiltonian simulation for KLK. The first step involves oracle access to a sparse graph adjacency list representation, which is at least as efficient as the oracle access to non-sparse data points. The Hamiltonian simulation involves simulating a sparse conditional partial swap operator, which results an efficient strategy for applying $e^{-iKLK\Delta t}$ in time $\tilde{O}(\log(m)\Delta t)$, where the notation $\tilde{O}$ hides more slowly growing factors in the simulation [64].

4.4   Related Work

Classical SVM.   Considerable effort has been devoted into designing fast classical algorithms for training SVMs. The decomposition-based methods such as SMO [48] are able to efficiently manage problems with large number of features p, but their computational complexities are super-linear in m. Other training strategies [7, 49, 50] are linear in m but scale quadratically in p in the worst case. The Pegasos algorithm [52] for non-linear kernel improves the complexity to $\tilde{O}\left(m/(\lambda\epsilon)\right)$, where λ, and ϵ are the regularization parameter of SVM and the error of the solution, respectively.

Quantum SVM.   Beyond the classical realm, three quantum algorithms for training linear models have been proposed, the quantum LS-SVM that involves L_2 regularizer [18], and a quantum training algorithm that solves a maximin problem resulting from a maximum – not average – loss over the training set [16], as well as the quantum Sparse SVM we introduced in Chapter 3.

Quantum-inspired SVM. Although quantum machine learning has been considered a potential application for building scalable quantum computers, its potentially-exponential speedup has been less manifested in practice than the well-known quantum algorithms such as Shor's algorithm. One of the main reason is that most quantum machine learning algorithms are based on a strong assumption: the existence of an efficient approach for preparing input data (mainly via QRAM), and learning partially from the quantum output state (via quantum measurement).

Quantum-inspired machine learning started by Tang's work [56] is a new line of research that tries to *dequantize* quantum machine learning algorithms when a low-rank approximation is feasible. Namely, a quantum-inspired algorithm, also known as a dequantized algorithm, solves a classical equivalent of a quantum machine learning problem in a setting where the data input and output model is design to mimic the assumptions underlying QRAM and quantum measurements. These assumptions make the comparison between quantum algorithms and their classical, dequantized counterparts more fair. Specifically, analogous to using QRAM input model and quantum measurement, these classical algorithms exploit ℓ^2-*norm sampling and query access*, also known as *importance sampling* or *length-square sampling* in randomized linear algebra literature. However, the dequantized algorithms are efficient when dealing with low-rank matrices, whereas quantum computers may still exhibit an exponential speedup over all known classical algorithms for sparse and high-to-full-rank matrix problems. Dequantizing these cases would imply that classical computers can efficiently simulate quantum computers, which is currently considered to be unlikely.

ℓ^2-norm Sampling and Query Access Model. Analogous to the assumption one can efficiently prepare a quantum state $|x\rangle$ proportional (up to normalization) to some input vector x, a quantum-inspired algorithm is assisted by an input model called sampling and query access. One has sampling and query access to a vector $x \in \mathbb{C}^n$ if the following queries can be done in $\tilde{O}(1)$:

(a) given an index $i \in [n]$, output the ith element x_i,

(b) sample an index $j \in [n]$ with probability $\frac{|x_j^2|}{\|x\|^2}$,

(c) output the ℓ^2-norm $\|x\|$.

For a matrix $A \in \mathbb{C}^{m \times n}$ the model gives the sampling and query access to each row $A(i,.)$ and access to a vector with elements corresponding to ℓ^2-norm of A's rows. The fundamental difference between quantum-inspired algorithms and traditional classical algorithms is that via importance sampling, their runtime is independent of the dimension of input data, and thus it builds a setting comparable with quantum machine learning algorithms aided by QRAM. Recently [65] introduced an algorithmic framework for quantum-inspired classical algorithms on low-rank matrices that generalizes a series of previous work, recovers existing quantum-inspired algorithms such as quantum-inspired recommendation systems [56], quantum-inspired principal component analysis [66], quantum-inspired low-rank matrix inversion [67], and quantum-inspired support vector machine [68].

It is natural to ask how our proposed quantum algorithm's complexity can be interpreted in the quantum-inspired classical setting. As earlier mentioned, our quantum algorithm's complexity is asymptotically the same as the complexity of the quantum LS-SVM. Here we briefly compare the complexity of quantum-inspired LS-SVM with its corresponding quantum version.

Quantum LS-SVM speedup is rooted in utilizing two quantum subroutines: first, efficient input kernel matrix preparation using quantum inner product; second, efficient quantum matrix inversion algorithm with the asymptotic final runtime $\tilde{O}\left(\epsilon_\kappa^{-3}\epsilon^{-3}\log mp\right)$. The matrix to be inverted is $\hat{F} = F/tr(F)$, where $F = \begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & K + \gamma^{-1}\mathbb{1} \end{bmatrix}$ and assuming the operator norm $\left\|\hat{F}\right\| \leq 1$. Here, ϵ_κ is a user-defined constant such that the absolute values of $\hat{F}$'s eigenvalues satisfy $\epsilon_\kappa \leq |\lambda_i| \leq 1$, and ϵ is the error occurred while computing $e^{-i\hat{F}\Delta t}$.

Similarly, quantum-inspired LS-SVM assumes $\hat{F}$ has a singular value threshold σ and $\left\|\hat{F}\right\|_F \in O(1)$ i.e, the Frobenius norm is independent from $\hat{F}$'s dimension. We also have $\sigma_{max} = \left\|\hat{F}\right\| \leq \left\|\hat{F}\right\|_F$, where σ_{max} is the largest singular value of matrix $\hat{F}$. These conditions simultaneously bounds the rank and condition number of $\hat{F}$. Quantum-Inspired LS-SVM gives the final output by applying the quantum-inspired low-rank matrix inversion on the corresponding system of linear equations in time $\tilde{O}\left(\frac{\|\hat{F}\|_F^6 \|\hat{F}\|^{22}}{\sigma^{28} \eta^6 \varepsilon^6} \log^3 \frac{1}{\delta}\right)$, which with assumptions on operator and Frobenius norms, it can be simplified as $\tilde{O}\left(\sigma^{-28} \eta^{-6} \varepsilon^{-6} \log^3 \frac{1}{\delta}\right)$. Here σ has the same role as ϵ_κ in quantum LS-SVM runtime; ε and ϵ are the accuracy of quantum-inspired and quantum LS-SVM, respectively. Parameters η and δ have no corresponding meaning in quantum LS-SVM; without loss of generality we consider their values as a constant.

For the purpose of understanding how both quantum-inspired and quantum LS-SVM behaves in edge cases, we consider the following special cases. Recall that by definition, the operator norm $\|A\| = \sigma_{max}$ and $\|A\|_F = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A)}$. We consider the ratio $\frac{\epsilon_\kappa^{-3}}{\sigma^{-28}}$ as the speedup ratio of quantum over quantum-inspired LS-SVM and consider the following cases:

1. Extremely low-rank cases: for example, when $\hat{F}$ has only one nonzero singular value equal to one, we have $\frac{\epsilon_\kappa^{-3}}{\sigma^{-28}} = O(1)$.

2. Full rank cases: for example, when $\hat{F}$ is equal to scaled Identity matrix, all singular value are equal to $1/p$ and $\frac{\epsilon_\kappa^{-3}}{\sigma^{-28}} = \frac{O(p^3)}{O(p^{28})}$, thus we expect polynomial speedup for quantum LS-SVM.

3. Low-rank cases: for example, when $\hat{F}$ has a scaled Identity block of size q: in this case $\frac{\epsilon_\kappa^{-3}}{\sigma^{-28}} = \frac{O(q^3)}{O(q^{28})}$. Note that, if $p = 1000000, q = \sqrt[6]{p} = 10$, we gain sub-linear speedup in both cases; as $\frac{\epsilon_\kappa^{-3}}{\sigma^{-28}} = \frac{O(p^{3/6})}{O(p^{28/6})}$.

We see that for problems involving low-rank matrices, quantum-inspired LS-SVM offers an asymptotic exponential speed up compared to previously known classical algorithms.

However, it exhibits still hefty polynomial overhead compared to quantum LS-SVM. A recent work [69] studies the performance of quantum-inspired algorithms in practice and concluded that their performance degrades significantly when the rank and condition number of the input matrix are increased, and high performance requires very low rank and condition number.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this dissertation, we have analyzed two quantum machine learning problems and provided quantum algorithms that, under certain assumptions, achieve speedup compared to existing classical algorithms for training the predictive models. The problems we chose were selected based on their expected utility for very large data.

In the first problem, we analyzed the computational complexity of quantum sparse support vector machine, a linear classifier that minimizes the hinge loss and the L_1 norm of the feature weights vector and relies on a quantum linear programming solver instead of a classical solver. Sparse SVM leads to sparse models that use only a small fraction of the input features in making decisions, and is especially useful when the total number of features, p, approaches or exceeds the number of training samples, m. We prove a $\Omega(\min(m, p))$ worst-case lower bound for computational complexity of any quantum training algorithm relying on black-box access to training samples; quantum sparse SVM has at least linear worst-case complexity. However, we proved that there are realistic scenarios in which a sparse linear classifier is expected to have high accuracy, and can be trained in sublinear time in terms of both the number of training samples and the number of features. We believe sparse supervised learning will be important once we reach data sizes that merit the use of a quantum computer for model training, because we would want to have a small, compact model that can be executed on a classical computer once its trained. A sparse model can achieve that: even if the original feature space dimensionality is huge, the number of features that are needed for classification once the model is trained is much smaller.

The second area that we focused on, weakly- or semi-supervised learning, is likely to be equally important in the huge-data regime. It is unlikely that all training samples

will have class labels attached to them, since often assigning correct labels is much more laborious than collecting the feature vectors. We provided a quantum algorithm for semi-supervised SVM that can be trained even if many labels are missing. The algorithm uses recent advances in quantum sample-based Hamiltonian simulation to extend the existing Quantum LS-SVM algorithm to handle the semi-supervised term in the loss, while maintaining the same quantum speedup as the Quantum LS-SVM. We also analyzed the offered speedup by our algorithm compare to the quantum-inspired classical equivalent.

Our work leaves several open questions: for both quantum sparse LS-SVM and semi-supervised LS-SVM, we showed there are polynomial speedup in some cases, and no speedup in other. It is natural to ask if there could be a general speedup, or if not, how broad is the set of families of special cases that allow speedup? In the first problem we analyzed, while speedup is not possible in the worst case, we showed that for the family of truncated subgaussian classification problems speed up is achievable. Here a natural question is, can one find other useful distribution that shows quantum advantage? For the second problem we analyzed, we showed speedup in some moderate-to-low rank input cases, but not in very low rank cases. It remains to be seen if there other characterizations of the input data that lead to possibility of speedup.

REFERENCES

[1] David Deutsch and Peter T Landsberg. "The fabric of reality". In: *Nature* 388.6638 (1997), pp. 136–136.

[2] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Quantum random access memory". In: *Physical Review Letters* 100.16 (2008), p. 160501.

[3] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. "On the robustness of bucket brigade quantum RAM". In: *New Journal of Physics* 17.12 (2015), p. 123010.

[4] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. "Quantum machine learning". In: *Nature* 549.7671 (2017), p. 195.

[5] Scott Aaronson. "Read the fine print". In: *Nature Physics* 11.4 (2015), p. 291.

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning.* Vol. 1. 10. Springer series in statistics New York, 2001.

[7] Johan AK Suykens and Joos Vandewalle. "Least squares support vector machine classifiers". In: *Neural Processing Letters* 9.3 (1999), pp. 293–300.

[8] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (1995), pp. 273–297.

[9] Miles Blencowe. "Quantum computing: Quantum RAM". In: *Nature* 468.7320 (2010), p. 44.

[10] Daniel K Park, Francesco Petruccione, and June-Koo Kevin Rhee. "Circuit-based quantum random access memory for classical data". In: *Scientific reports* 9.1 (2019), pp. 1–8.

[11] N Jiang, Y-F Pu, W Chang, C Li, S Zhang, and L-M Duan. "Experimental realization of 105-qubit random access quantum memory". In: *npj Quantum Information* 5.1 (2019), pp. 1–6.

[12] Vedran Dunjko and Hans J Briegel. "Machine learning & artificial intelligence in the quantum domain: a review of recent progress". In: *Reports on Progress in Physics* 81.7 (2018), p. 074001.

[13] M. Schuld and F. Petruccione. *Supervised Learning with Quantum Computers*. Springer Nature, 2018.

[14] Srinivasan Arunachalam and Ronald de Wolf. "A survey of quantum learning theory". In: *ACM SIGACT News* 48.2 (2017), pp. 41–67.

[15] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. "q-means: A quantum algorithm for unsupervised machine learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 4136–4146.

[16] Tongyang Li, Shouvanik Chakrabarti, and Xiaodi Wu. "Sublinear quantum algorithms for training linear and kernel-based classifiers". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 3815–3824.

[17] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. "Quantum Algorithms for Deep Convolutional Neural Networks". In: *International Conference on Learning Representations*. 2020.

[18] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification". In: *Physical Review Letters* 113.13 (2014), p. 130503.

[19] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". In: *Physical Review Letters* 103.15 (2009), p. 150502.

[20] Rolando Somma, Andrew Childs, and Robin Kothari. "Quantum linear systems algorithm with exponentially improved dependence on precision". In: *APS Meeting Abstracts*. 2016.

[21] Sathyawageeswar Subramanian, Steve Brierley, and Richard Jozsa. "Implementing smooth functions of a Hermitian matrix on a quantum computer". In: *arXiv preprint arXiv:1806.06885* (2018).

[22] Kristin Bennett. "Combining support vector and mathematical programming methods for induction". In: *Advances in Kernel Methods: Support Vector Learning* (1999), pp. 307–326.

[23] Vojislav Kecman and Ivana Hadzic. "Support vectors selection by linear programming". In: *Proc. International Joint Conference on Neural Networks (IJCNN'00)*. Vol. 5. IEEE. 2000, pp. 193–198.

[24] Jinbo Bi, Kristin Bennett, Mark Embrechts, Curt Breneman, and Minghu Song. "Dimensionality reduction via sparse support vector machines". In: *Journal of Machine Learning Research* 3.Mar (2003), pp. 1229–1243.

[25] Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. "1-norm support vector machines". In: *Advances in Neural Information Processing Systems (NIPS'04)*. MIT Press Boston, MA, 2004, pp. 49–56.

[26] Joran van Apeldoorn and András Gilyén. "Quantum algorithms for zero-sum games". In: *arXiv:1904.03180* (2019).

[27] Fernando GSL Brandão and Krysta M Svore. "Quantum speed-ups for solving semidefinite programs". In: *Proc. IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*. IEEE. 2017, pp. 415–426.

[28] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. "Quantum SDP-Solvers: Better upper and lower bounds". In: *Proc. IEEE 58th*

Annual Symposium on Foundations of Computer Science (FOCS'17). IEEE. 2017, pp. 403–414.

[29] FGSL Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M Svore, and Xiaodi Wu. "Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning". In: *arXiv preprint arXiv:1710.02581* 6 (2017).

[30] Joran van Apeldoorn and András Gilyén. "Improvements in Quantum SDP-Solving with Applications". In: *arXiv preprint arXiv:1804.05058* (2018).

[31] Michael D. Grigoriadis and Leonid Khachiyan. "A sublinear-time randomized approximation algorithm for matrix games". In: *Operations Research Letters* 18 (1994), pp. 53–58.

[32] Connor T. Hann, Chang-Ling Zou, Yaxing Zhang, Yiwen Chu, Robert J. Schoelkopf, S. M. Girvin, and Liang Jiang. "Hardware-Efficient Quantum Random Access Memory with Hybrid Quantum Acoustic Systems". In: *Physical Review Letters* 123 (25 2019), p. 250501.

[33] Michał Parniak, Michał Dabrowski, Mateusz Mazelanik, Adam Leszczyński, Michał Lipka, and Wojciech Wasilewski. "Wavevector multiplexed atomic quantum memory via spatially-resolved single-photon detection". In: *Nature Communications* 8.1 (2017), pp. 1–9.

[34] Yong Yu, Fei Ma, Xi-Yu Luo, Bo Jing, Peng-Fei Sun, Ren-Zhou Fang, Chao-Wei Yang, Hui Liu, Ming-Yang Zheng, Xiu-Ping Xie, et al. "Entanglement of two quantum memories via metropolitan-scale fibers". In: *Nature* 578 (2020), 240–245.

[35] X-L Ouyang, X-Z Huang, Y-K Wu, W-G Zhang, X Wang, H-L Zhang, L He, X-Y Chang, and L-M Duan. "Experimental demonstration of quantum-enhanced machine learning in a nitrogen-vacancy-center system". In: *Physical Review A* 101.1 (2020), p. 012307.

[36] Zhikuan Zhao, Jack K Fitzsimons, Patrick Rebentrost, Vedran Dunjko, and Joseph F Fitzsimons. "Smooth input preparation for quantum and quantum-inspired machine learning". In: *arXiv:1804.00281* (2018).

[37] Ryan LaRose and Brian Coyle. "Robust data encodings for quantum classifiers". In: *arXiv:2003.01695* (2020).

[38] Lov Grover and Terry Rudolph. "Creating superpositions that correspond to efficiently integrable probability distributions". In: *arXiv quant-ph/0208112* (2002).

[39] Iordanis Kerenidis and Anupam Prakash. "Quantum recommendation systems". In: *arXiv:1603.08675* (2016).

[40] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. "Quantum lower bounds by polynomials". In: *Journal of the ACM* 48.4 (2001), pp. 778–797.

[41] Martin Bengtsson, Anders Ståhlberg, Patrik Rorsman, and Mikael Kubista. "Gene expression profiling in single cells from the pancreatic islets of Langerhans reveals lognormal distribution of mRNA levels". In: *Genome Research* 15.10 (2005), pp. 1388–1392.

[42] Vasilis Ntranos, Lynn Yi, Páll Melsted, and Lior Pachter. "A discriminative learning approach to differential expression analysis for single-cell RNA-seq". In: *Nature Methods* 16.2 (2019), pp. 163–166.

[43] Yunzhe Liu, Raymond J Dolan, Zeb Kurth-Nelson, and Timothy EJ Behrens. "Human Replay Spontaneously Reorganizes Experience". In: *Cell* 178.3 (2019), pp. 640–652.

[44] Jennifer M Fettweis, Myrna G Serrano, J Paul Brooks, David J Edwards, Philippe H Girerd, Hardik I Parikh, Bernice Huang, Tom J Arodz, Laahirie Edupuganti, Abigail L Glascock, et al. "The vaginal microbiome and preterm birth". In: *Nature Medicine* 25 (2019), 1012—1021.

[45] Helen Davies, Dominik Glodzik, Sandro Morganella, Lucy R Yates, Johan Staaf, Xueqing Zou, Manasa Ramakrishna, Sancha Martin, Sandrine Boyault, Anieta M Sieuwerts, et al. "HRDetect is a predictor of BRCA1 and BRCA2 deficiency based on mutational signatures". In: *Nature Medicine* 23.4 (2017), p. 517.

[46] Margaret E Ackerman, Jishnu Das, Srivamshi Pittala, Thomas Broge, Caitlyn Linde, Todd J Suscovich, Eric P Brown, Todd Bradley, Harini Natarajan, Shu Lin, et al. "Route of immunization defines multiple mechanisms of vaccine-mediated protection against SIV". In: *Nature Medicine* 24.10 (2018), p. 1590.

[47] Peggie Cheung, Francesco Vallania, Hayley C Warsinske, Michele Donato, Steven Schaffert, Sarah E Chang, Mai Dvorak, Cornelia L Dekker, Mark M Davis, Paul J Utz, et al. "Single-cell chromatin modification profiling reveals increased epigenetic variations with aging". In: *Cell* 173.6 (2018), pp. 1385–1397.

[48] John Platt. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.* Tech. rep. MSR-TR-98-14. Microsoft, 1998.

[49] Glenn M Fung and Olvi L Mangasarian. "Multicategory proximal support vector machine classifiers". In: *Machine Learning* 59.1-2 (2005), pp. 77–97.

[50] S Sathiya Keerthi and Dennis DeCoste. "A modified finite Newton method for fast solution of large scale linear SVMs". In: *Journal of Machine Learning Research* 6 (2005), pp. 341–361.

[51] Thorsten Joachims. "Training linear SVMs in linear time". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM. 2006, pp. 217–226.

[52] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. "Pegasos: Primal estimated sub-gradient solver for svm". In: *Mathematical Programming* 127.1 (2011), pp. 3–30.

[53] Iordanis Kerenidis and Anupam Prakash. "A quantum interior point method for LPs and SDPs". In: *arXiv preprint arXiv:1808.09266* (2018).

[54] PAM Casares and MA Martin-Delgado. "A Quantum IP Predictor-Corrector Algorithm for Linear Programming". In: *arXiv preprint arXiv:1902.06749* (2019).

[55] Iordanis Kerenidis and Anupam Prakash. "Quantum gradient descent for linear systems and least squares". In: *Physical Review A* 101.2 (2020), p. 022316.

[56] Ewin Tang. "A quantum-inspired classical algorithm for recommendation systems". In: *Proc. 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC'19)*. 2019, pp. 217–228.

[57] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. "Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning". In: *arXiv:1910.06151* (2019).

[58] Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang. "Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches". In: *arXiv:1901.03254* (2019).

[59] Alejandro Perdomo-Ortiz, Marcello Benedetti, John Realpe-Gómez, and Rupak Biswas. "Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers". In: *Quantum Science and Technology* 3.3 (2018), p. 030502.

[60] Shelby Kimmel, Cedric Yen-Yu Lin, Guang Hao Low, Maris Ozols, and Theodore J Yoder. "Hamiltonian simulation with optimal sample complexity". In: *npj Quantum Information* 3.1 (2017), p. 13.

[61] Stefano Melacci and Mikhail Belkin. "Laplacian support vector machines trained in the primal". In: *Journal of Machine Learning Research* 12.Mar (2011), pp. 1149–1184.

[62] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification". In: *Physical Review Letters* 113.13 (2014), p. 130503.

[63] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". In: *Nature Physics* 10.9 (2014), p. 631.

[64] Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. "Efficient quantum algorithms for simulating sparse Hamiltonians". In: *Communications in Mathematical Physics* 270.2 (2007), pp. 359–371.

[65] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. "Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning". In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing.* 2020, pp. 387–400.

[66] Ewin Tang. "Quantum-inspired classical algorithms for principal component analysis and supervised clustering". In: *arXiv preprint arXiv:1811.00414* (2018).

[67] András Gilyén, Seth Lloyd, and Ewin Tang. "Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension". In: *arXiv preprint arXiv:1811.04909* (2018).

[68] Chen Ding, Tian-Yi Bao, and He-Liang Huang. "Quantum-Inspired Support Vector Machine". In: *arXiv preprint arXiv:1906.08902* (2019).

[69] Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd. "Quantum-inspired algorithms in practice". In: *arXiv preprint arXiv:1905.10415* (2019).

VITA

Seyran Saeedi is currently pursuing a Ph.D. degree in computer science at Virginia Commonwealth University. Her research interests include quantum machine learning, quantum optimization, and theoretical machine learning. She received her Bachelor of Science in applied mathematics from the University of Tabriz in 2007 and a Master's degree in computer science from the University of Tehran, Iran, in 2011. After earning her master's degree, she has taught as an instructor at the University of Tehran for two years. She joined Virginia Commonwealth University in 2015.