QUANTUM SEMI-SUPERVISED KERNEL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantum machine learning methods have the potential to facilitate learning using extremely large datasets. While the availability of data for training machine learning models is steadily increasing, oftentimes it is much easier to collect feature vectors that to obtain the corresponding labels. One of the approaches for addressing this issue is to use semi-supervised learning, which leverages not only the labeled samples, but also unlabeled feature vectors. 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.

1 INTRODUCTION

Data sets used for training machine learning models are becoming increasingly large, leading to continued 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 (Blencowe, 2010), work on quantum machine learning algorithms and on quantum learning theory is under way (see for example Refs. (Biamonte et al., 2017; Dunjko & Briegel, 2018; Schuld & Petruccione, 2018) and (Arunachalam & de Wolf, 2017) for review). An early example of this approach is Quantum LS-SVM (Rebentrost et al., 2014a), which achieves exponential speedup compared to classical LS-SVM algorithm. Quantum LS-SVM uses quadratic least-squares loss and squared-L_2 regularizer, and the optimization problem can be solved using the seminal HHL (Harrow et al., 2009) algorithm for solving quantum linear systems of equations. While progress has been made in quantum algorithms for supervised learning, it has been recently advocated that the focus should shift to unsupervised and semi-supervised setting (Perdomo-Ortiz et al., 2018).

In many domains, the most laborious part of assembling a training set is the collection of sample labels. Thus, in many scenarios, in addition to the labeled training set of size m 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. In the approach considered here, the training samples are connected by a graph that captures their similarity.

Here, we introduce a quantum algorithm for semi-supervised training of a kernel support vector machine classification model. We start with the existing Quantum LS-SVM (Rebentrost et al., 2014a), and use techniques from sample-based Hamiltonian simulation (Kimmel et al., 2017) to add a semi-supervised term based on Laplacian SVM (Melacci & Belkin, 2011). As is standard in quantum machine learning (Li et al., 2019), the algorithm accesses training points 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.

2 Preliminaries

2.1 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

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

where γ is a user define constant allowing for adjusting the regularization strength. The last term captures the squared norm of the graph gradient on the graph G that contains all training samples as vertices, and expresses similarity between samples through, possibly edges $G_{u,v}$

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

where $\bar{h}_u$ is the function value $h(x_i)$ for the vertex u corresponding to training point x_i, and L is the *combinatorial graph Laplacian matrix* such that $L[i,j] = D_j - G_{i,j}$.

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

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

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.

2.2 Quantum Computing and Quantum LS-SVM

Quantum computers are devices which perform computing according to the laws of quantum mechanics, a mathematical framework for describing physical theories, in language of linear algebra.

Quantum Systems. Any isolated, closed quantum physical system can be fully described by a unit-norm vector in a complex Hilbert space appropriate for that system; in quantum computing, the space is always finite-dimensional, $\mathbb{C}^d$. In quantum mechanics and quantum computing, Dirac notation for linear algebra is commonly used. In Dirac notation, a vector $\mathbf{x} \in \mathbb{C}^d$ and its complex conjugate $\mathbf{x}^T$, which represents a functional $\mathbb{C}^d \to \mathbb{R}$, are denoted by $|\mathbf{x}\rangle$ (called ket) and $\langle\mathbf{x}|$ (called bra), respectively. We call $\{|e_i\rangle\}_{i=1}^{d}$ the *computational basis*, where $|e_i\rangle = (0, ..., 1, ...0)^T$ with exactly one 1 entry in the i-th position. Any $|\mathbf{v}\rangle = (v_1, ..., v_d)^T$ can be written as $|\mathbf{v}\rangle = \sum_{i=1}^{d} v_i|e_i\rangle$; coefficient $v_i \in \mathbb{C}$ are called *probability amplitudes*. Any unit vector $|x\rangle \in \mathbb{C}^d$ describes a d-level quantum state. Such a system is called a *pure state*. An inner product of $|x_1\rangle, |x_2\rangle \in \mathbb{C}^d$ is written as $\langle x_1|x_2\rangle$. A two-level quantum state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $|0\rangle = (1,0)^T$, $|1\rangle = (0,1)^T$ and $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2$, is called a *quantum bit*, or qubit for short. When both α and β are nonzero, we say $|\psi\rangle$ is in a *superposition* of the computational basis $|0\rangle$ and $|1\rangle$; the two superposition states $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ are very common in quantum computing.

A composite quantum state of two distinct quantum systems $|x_1\rangle \in \mathbb{C}^{d_1}$ and $|x_2\rangle \in \mathbb{C}^{d_2}$ is described as *tensor product* of quantum states $|x_1\rangle \otimes |x_2\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$. Thus, a state of an n-qubit system is a vector in the tensor product space $\left(\mathbb{C}^2\right)^{\otimes n} = \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes ... \otimes \mathbb{C}^2$, and is written as $\sum_{i=0}^{2^n-1} \alpha_i|i\rangle$, where i is expressed using its binary representation; for example for $n = 4$, we have $|2\rangle = |0010\rangle = |0\rangle \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle$.

Transforming and Measuring Quantum States. Quantum operations manipulate quantum states in order to obtain some desired final state. Two types of manipulation of a quantum system are allowed by laws of physics: unitary operators and measurements. Quantum measurement, if done in the computational basis, stochastically transforms the state of the system into one of the computational basis states, based on squared magnitudes of probability amplitudes; that is, $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ will result in $|0\rangle$ and $|1\rangle$ with equal chance. Unitary operators are deterministic, invertible, norm-preserving linear transforms. A unitary operator $\mathcal{U}$ models a transformation of a quantum state $|u\rangle$ to $|v\rangle = \mathcal{U}|u\rangle$. Note that $\mathcal{U}|u_1\rangle + \mathcal{U}|u_2\rangle = \mathcal{U}(|u_1\rangle + |u_2\rangle)$, applying a unitary to a superposition of states has the same effect as applying it separately to element of the superposition. In quantum circuit model unitary transformations are referred to as quantum gates – for example, one of the most common gates, the single-qubit *Hadamard* gate, is a unitary operator represented in the computational basis by the matrix

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

Note that $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$.

Quantum Input Model. Quantum computation typically starts from all qubits in $|0\rangle$ state. To perform computation, access to input data is needed. In quantum computing, input is typically given by a unitary operator that transforms the initial state into the desired input state for the computation – such unitaries are commonly referred to as oracles, and the computational complexity of quantum algorithms is typically measured with access to an oracle as the unit. 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 (Giovannetti et al., 2008; Arunachalam et al., 2015; Biamonte et al., 2017), but it is still an open challenge, and subtle differences in qRAM architecture may erase any gains in computational complexity of a quantum algorithm Aaronson (2015).

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

$$|x\rangle = \frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|}. \tag{3}$$

Intuitively and at the risk of over-simplifying, 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 $\lambda_i v_i \mapsto \dfrac{1}{\lambda_i} v_i$. 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 $A^{-1}b = \sum_{i=1}^{n} \beta_i \dfrac{1}{\lambda_i} v_i$. 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 $\dfrac{1}{\lambda_i}|0\rangle + \sqrt{1 - \dfrac{1}{\lambda_i^2}}|1\rangle$. The last step involves the inverse of phase estimation and quantum measurement for getting rid of garbage qubits and outputs our desired state $|x\rangle = A^{-1}|b\rangle = \sum_{i=1}^{n} \beta_i \dfrac{1}{\lambda_i} v_i$.

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*. A mixed state that describes an ensemble $\{p_i, |\psi\rangle_i\}$ is written as

$$\rho = \sum_{i=1}^{k} p_i |\psi_i\rangle\langle\psi_i|, \tag{4}$$

where $\sum_{i=1}^{k} p_i = 1$ forms a probability distribution and ρ is called *density operator*, which in a finite-dimensional system, in computational basis, is a semi-definite positive matrix with $Tr(\rho) = 1$.

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

Partial Trace of Composite Quantum System. Consider a two-part quantum system in a state described by tensor product of two density operators $\rho \otimes \sigma$. A *partial trace*, tracing out the second part of the quantum system, is defined as the linear operator that leaves the first part of the system in a state $\mathrm{Tr}_2\left(\rho \otimes \sigma\right) = \rho \, \mathrm{tr}\left(\sigma\right)$, where $\mathrm{Tr}\left(\sigma\right)$ is the trace of the matrix σ.

To obtain Kernel matrix K as a density matrix, quantum LS-SVM (Rebentrost et al., 2014b) 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 $|x_i\rangle = \frac{1}{\parallel x_i \parallel} \sum_{t=1}^{p} (x_i)_t |t\rangle$, where $(x_i)_t$ refers to the tth feature value in data point x_i and assuming the oracle is given $\parallel x_i \parallel$ and y_i . Vector of the labels is given in the same fashion as $|y\rangle = \frac{1}{\parallel y \parallel} \sum_{i=1}^{m} y_i |i\rangle$. For preparation the normalized Kernel matrix $K' = \frac{1}{\mathrm{tr}(K)} K$ where $K = X^T X$, we need to prepare a quantum state combining all data points in quantum superposition $|X\rangle = \frac{1}{\sqrt{\sum_{i=1}^{m} \parallel x_i \parallel^2}} \sum_{i=1}^{m} |i\rangle \otimes \parallel x_i \parallel |x_i\rangle$. The normalized Kernel matrix is obtained by discarding the training set state,

$$K' = \mathrm{Tr}_2(|X\rangle\langle X|) = \frac{1}{\sum_{i=1}^{m} \parallel x_i \parallel^2} \sum_{i,j=1}^{m} \parallel x_i \parallel \parallel x_j \parallel \langle x_i | x_j \rangle |i\rangle\langle j|. \tag{5}$$

The approach used above to construct density matrix corresponding to linear kernel matrix can be extended to polynomial kernels (Rebentrost et al., 2014b).

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 (Lloyd et al., 2014) is adapted for the exponentiation of a non-sparse density matrix:

$$\mathrm{Tr}_1\left\{e^{-iS\Delta t}(K \otimes \sigma)e^{iS\Delta t}\right\} = \sigma - i\Delta t[K, \sigma] + O\left(\Delta t^2\right) \approx e^{-iK\Delta t}\sigma e^{iK\Delta t}, \tag{6}$$

where $S = \sum_{i,j} |i\rangle\langle j| \otimes |j\rangle\langle i|$ is the swap operator and the facts $\mathrm{Tr}_1 \{S(K \otimes \sigma)\} = K\sigma$ and $\mathrm{Tr}_1 \{(K \otimes \sigma)S\} = \sigma K$ are used. The equation (6) 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 (Rebentrost et al., 2014b) 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.

3 Quantum Semi-Supervised Least Square SVM.

Semi-Supervised Least Square SVM involves solving the following system of linear equations

$$\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & K + KLK + \gamma^{-1}\mathbb{1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} = A^{-1} \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \tag{7}$$

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 $(K + KLK)^{-1}$.

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,

$$L = Tr_2(|G_I\rangle\langle G_I|) = \frac{1}{md} \sum_{i,j=1}^{m} |i\rangle\langle j| \otimes d\langle v_i|v_j\rangle = \frac{1}{m} \sum_{i,j=1}^{m} \langle v_i|v_j\rangle |i\rangle\langle j|. \tag{8}$$

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 (Kimmel et al., 2017) to the specific 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 $B = K^\dagger LK$.

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

$$\mathrm{Tr}_1\left\{\mathrm{Tr}_3\left\{e^{-iS'^{\Delta}}\left(\rho' \otimes \sigma\right)e^{iS'^{\Delta}}\right\}\right\} = \sigma - i[B,\sigma] + O(\Delta^2) = e^{-iBt}\sigma e^{iBt} + O(\Delta^2), \qquad (9)$$

where $B = \rho'' - \rho''' = \frac{1}{2}K^\dagger LK + \frac{1}{2}KLK^\dagger = KLK$ 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 1 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 $B = \rho'' - \rho''' = KLK$.

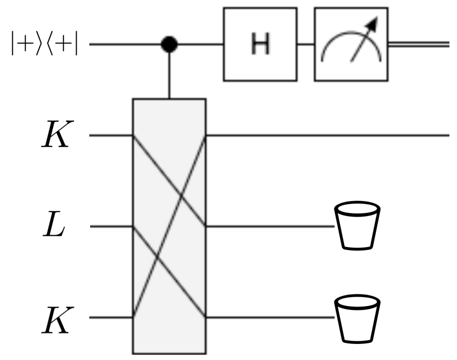

Figure 1: Quantum circuit for creating $\rho' = |0\rangle\langle 0| \otimes \rho'' + |1\rangle\langle 1| \otimes \rho'''$. The circuit is to be read from left-to-right. Each wire at left shows its corresponding input state. The vertical rectangle denotes the cyclic permutation operator P on K, L, K defined in (10). H is the Hadamard gate, and the waste bins show partial trace. The measurement on the first quantum state is in computational basis.

The analysis of the steps preformed by the circuit depicted in Fig.1 is as follows. Let P be the cyclic permutation of three copies of $\mathcal{H}_A$ that operates as $\mathrm{P}\,|j_1, j_2, j_3\rangle = |j_3, j_1, j_2\rangle$. In operator form it can be written as

$$\mathrm{P} := \sum_{j_1,j_2,j_3=1}^{\dim \mathcal{H}_\mathrm{A}} |j_3\rangle\langle j_1| \otimes |j_1\rangle\langle j_2| \otimes |j_2\rangle\langle j_3| \qquad (10)$$

The input state to the circuit depicted in Fig. 1 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

$$\begin{aligned} I = \frac{1}{2}[&|0\rangle\langle 0| \otimes K^\dagger \otimes L \otimes K + |0\rangle\langle 1| \otimes \left(K^\dagger \otimes L \otimes K\right) \mathrm{P} \\ &+ |1\rangle\langle 0| \otimes \mathrm{P} \left(K^\dagger \otimes L \otimes K\right) + |1\rangle\langle 1| \otimes \mathrm{P} \left(K^\dagger \otimes L \otimes K\right) \mathrm{P}\,. \end{aligned}$$

After discarding the third and second register sequentially by applying corresponding partial trace operators, we get

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

in this step KLK term where the last line obtained from

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

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

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

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

$$\begin{aligned} &III = H \otimes \mathbb{1}(II)H \otimes \mathbb{1} = \\ &\frac{1}{2}\left(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| + |1\rangle\langle 1|\right) \otimes \frac{1}{2}K^\dagger + \frac{1}{2}\left(|0\rangle\langle 0| - |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|\right) \otimes \frac{1}{2}K^\dagger LK \\ &+ \frac{1}{2}\left(|0\rangle\langle 0| + |0\rangle\langle 1| - |1\rangle\langle 0| - |1\rangle\langle 1|\right) \otimes \frac{1}{2}KLK^\dagger + \frac{1}{2}\left(|0\rangle\langle 0| - |0\rangle\langle 1| - |1\rangle\langle 0| + |1\rangle\langle 1|\right) \otimes \frac{1}{2}K \\ &= |0\rangle\langle 0| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger + \frac{1}{2}K^\dagger LK + \frac{1}{2}KLK^\dagger + \frac{1}{2}K\right) + |0\rangle\langle 1| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger - \frac{1}{2}K^\dagger LK + \frac{1}{2}KLK^\dagger - \frac{1}{2}K\right) \\ &+ |1\rangle\langle 0| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger + \frac{1}{2}K^\dagger LK - \frac{1}{2}KLK^\dagger - \frac{1}{2}K\right) + |1\rangle\langle 1| \otimes \frac{1}{2}\left(\frac{1}{2}K^\dagger - \frac{1}{2}K^\dagger LK - \frac{1}{2}KLK^\dagger + \frac{1}{2}K\right). \end{aligned}$$

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',

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

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 (9) to simulate $e^{-iKLK\Delta t}\sigma e^{iKLK\Delta t}$ up to error $O(\Delta^2)$. Comparing the LMR technique in equation (6) with the generalized LMR for the spacial case of KLK in equation (9), 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 $e^{iKLK\Delta t}$.

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 3.2)
5:     Controlled rotation
6:     Uncomputing
7: Classification. Based on Swap test algorithm, same as in Quantum LS-SVM.
```

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

The quantum LS-SVM in (Rebentrost et al., 2014b) 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 (Berry et al., 2007).

3.4 Comparison with Alternative Approaches

Considerable effort has been devoted into designing fast classical algorithms for training SVMs. The decomposition-based methods such as SMO (Platt, 1998) are able to efficiently manage problems with large number of features p, but their computational complexities are super-linear in m. Other training strategies (Suykens & Vandewalle, 1999; Fung & Mangasarian, 2005; Keerthi & DeCoste, 2005) are linear in m but scale quadratically in p in the worst case. The Pegasos algorithm (Shalev-Shwartz et al., 2011) 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.

Beyond the classical realm, three quantum algorithms for training linear models have been proposed, the quantum LS-SVM that involves L_2 regularizer (Rebentrost et al., 2014a), a recently proposed Quantum Sparse SVM which is limited to a linear kernel (Arodz & Saeedi, 2019), and a quantum training algorithm that solves a maximin problem resulting from a maximum – not average – loss over the training set (Li et al., 2019).

References

Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291, 2015.

Tom Arodz and Seyran Saeedi. Quantum sparse support vector machines. *arXiv preprint arXiv:1902.01879*, 2019.

Srinivasan Arunachalam and Ronald de Wolf. A survey of quantum learning theory. *ACM SIGACT News*, 48(2):41–67, 2017.

Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12):123010, 2015.

Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2): 359–371, 2007.

Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017.

Miles Blencowe. Quantum computing: Quantum RAM. *Nature*, 468(7320):44, 2010.

Vedran Dunjko and Hans J Briegel. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics*, 81(7):074001, 2018.

Glenn M Fung and Olvi L Mangasarian. Multicategory proximal support vector machine classifiers. *Machine Learning*, 59(1-2):77–97, 2005.

Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16):160501, 2008.

Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009.

S Sathiya Keerthi and Dennis DeCoste. A modified finite newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

Shelby Kimmel, Cedric Yen-Yu Lin, Guang Hao Low, Maris Ozols, and Theodore J Yoder. Hamiltonian simulation with optimal sample complexity. *npj Quantum Information*, 3(1):13, 2017.

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*, pp. 3815–3824, 2019.

Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631, 2014.

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

Alejandro Perdomo-Ortiz, Marcello Benedetti, John Realpe-Gómez, and Rupak Biswas. Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Science and Technology*, 3(3):030502, 2018.

John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft, 1998.

Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, 2014a.

Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, 2014b.

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

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.