Quantum algorithm for support matrix machines

Bojia Duan, Jiabin Yuan, Ying Liu, and Dan Li
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, No. 29 Jiangjun Avenue, 211106 Nanjing, China
(Received 25 March 2017; published 1 September 2017)

We propose a quantum algorithm for support matrix machines (SMMs) that efficiently addresses an image classification problem by introducing a least-squares reformulation. This algorithm consists of two core subroutines: a quantum matrix inversion (Harrow-Hassidim-Lloyd, HHL) algorithm and a quantum singular value thresholding (QSVT) algorithm. The two algorithms can be implemented on a universal quantum computer with complexity $O[\log(npq)]$ and $O[\log(pq)]$, respectively, where n is the number of the training data and pq is the size of the feature space. By iterating the algorithms, we can find the parameters for the SMM classfication model. Our analysis shows that both HHL and QSVT algorithms achieve an exponential increase of speed over their classical counterparts.

DOI: 10.1103/PhysRevA.96.032301

I. INTRODUCTION

As an emerging interdisciplinary field, quantum machine learning (QML) aims to aid the learning process via quantum computing and quantum information theory [1–4]. A range of quantum machine learning algorithms can offer an increase of speed over their classical counterparts. One of these quantum algorithms is phase estimation (PE), which can be used to solve matrix computing problems, namely eigenvalue decomposition or singular value decomposition of a matrix, or matrix inversion. It can help a variety of machine learning algorithms achieve an exponential speed increase, such as quantum principal component analysis [5], quantum singular value decomposition (QSVD) [6], and a quantum matrix inversion algorithm (Harrow-Hassidim-Lloyd, HHL) for solving linear systems of equations [7]. On the basis of these algorithms proposed in Refs. [5–7], quantum support vector machines, quantum algorithms for least-squares regression and statistical leverage scores, and pattern classification with linear regression have been presented [8–10]. In addition, other quantum algorithms also show good performance, such as quantum random access memory, which requires only a logarithmic memory call [11]; swap tests, which can solve inner product exponentially faster [12]; and generic Grover's algorithms, which can solve search problems with a quadratic speed increase [13–15]. These algorithms can also accelerate many machine learning procedures [16–24].

Herein, we focus our attention on support matrix machine (SMMs), an important model proposed in 2015 to address an image classification problem in machine learning [25]. Given a large number of classified training matrices, the task of a SMM is to classify a new matrix into one of two classes. For example, a SMM can be used to predict whether electroencephalogram (EEG) emotion is positive or negative in EEG classification, or whether a person is male or female in face classification. The core of a SMM is to preserve the structure information of the original feature matrices, which helps to increase the robustness and accuracy of the classification. The time complexity of the learning algorithm of SMMs is proportional to $O[\text{poly}(n, pq)]$, with n being the number of training data and pq the dimension of the feature space. However, when the size of the training set and/or the feature space is large (e.g., terabytes or even petabytes), a SMM may take a significant amount of time to execute.

In this paper we propose a quantum algorithm for SMMs. Schematically, for learning the parameters of the SMM classification model, we first introduce the least-squares method to re-express the quadratic programming program approximately to the linear programming program. This reformulation allows for a quantum solution with the quantum matrix inversion (HHL) algorithm. In addition, we propose a quantum singular value thresholding (QSVT) algorithm for updating the substitution of the regression matrix. Finally, a swap-test operation can then help to classify a query state. We have analyzed the complexity of our algorithm, and it shows that the two core subroutines (HHL and QSVT algorithms) can be implemented in time $O[\log(npq)]$ and $O[\log(pq)]$, respectively, achieving an exponential acceleration compared with its classical counterparts.

In detail, there are two contributions in our work. First, we introduced a least-squares reformulation in Sec. III A. Without this reformulation, the original problem could not be solved by the HHL algorithm directly. Second, we proposed a quantum algorithm (QSVT) for solving singular value thresholding (SVT) in Sec. III B 2. The QSVT algorithm achieves an exponential speedup compared to the classical counterparts.

The remainder of the paper is organized as follows: We give a brief overview of the SMM algorithm in Sec. II, put forward our quantum algorithm for SMMs and analyze the time complexity in Sec. III, and present our conclusions in Sec. IV.

II. REVIEW OF SMMS

In this section, we briefly review some basic notations of SMMs and describe the algorithmic procedures for the SMM learning process. More detailed information can be found in Ref. [25].

A. Notations

Let $\mathbf{I}_n$ denote the $n \times n$ identity matrix. For any matrix $\mathbf{A} \in \mathbb{R}^{p\times q}$, the Frobenius norm is defined as $\|\mathbf{A}\|_F^2 = \sum_{ij} a_{ij}^2 = \sum_i \sigma_i^2$, where a_{ij} are the (i,j)th elements of $\mathbf{A}$ and σ_i are the singular values. Given the matrix $\mathbf{A} \in \mathbb{R}^{p\times q}$ with rank $r \leqslant \min(p,q)$, let the thin singular value decomposition (SVD) of $\mathbf{A}$ be $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$,

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_r] \in \mathbb{R}^{p\times r}$ and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_r] \in \mathbb{R}^{q\times r}$ satisfy $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}_r$, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, with $\sigma_1 \geqslant \cdots \geqslant \sigma_r > 0$, are the singular values of $\mathbf{A}$. For any $\tau > 0$, let the singular value thresholding (SVT) of $\mathbf{A}$ be $\mathcal{D}_\tau(\mathbf{A}) = \mathbf{U}\mathcal{D}_\tau(\Sigma)\mathbf{V}^T$, where $\mathcal{D}_\tau(\Sigma) = \mathrm{diag}[(\sigma_1 - \tau)_+, \ldots, (\sigma_r - \tau)_+]$ and $(\sigma_i - \tau)_+ = \max(\sigma_i - \tau, 0)$ [26]. Obviously, $\mathcal{D}_\tau(\mathbf{A}) = \sum_{i:\sigma_i>\tau}(\sigma_i - \tau)\mathbf{u}_i\mathbf{v}_i^T$.

To convert a matrix into a column vector, let the vectorization of the matrix $\mathbf{A}$ be $\mathrm{vec}(\mathbf{A}^T) = (a_{11}, \ldots, a_{1q}, a_{21}, \ldots, a_{pq})^T \triangleq \mathbf{a} \in \mathbb{R}^{pq}$. Similarly, there are $\mathbf{x}_i \triangleq \mathrm{vec}(\mathbf{X}_i^T)_{i=1}^n$, $\mathbf{w} \triangleq \mathrm{vec}(\mathbf{W}^T)$, $\mathbf{s} \triangleq \mathrm{vec}(\mathbf{S}^T)$, and $\lambda_\Lambda \triangleq \mathrm{vec}(\Lambda^T)$.

B. Support matrix machines

A SMM algorithm aims at classifying a regularized feature matrix into one of two classes. The ingredients of a SMM are a training set of already classified data and a learning algorithm. With the given training set $M = \{(\mathbf{X}_i, y_i) : \mathbf{X}_i \in \mathbb{R}^{p\times q}, y_i = \pm 1\}_{i=1}^n$, where $\mathbf{X}_i$ represented in matrix form is the ith input sample and y_i indicates the class which $\mathbf{X}_i$ belongs to, the learning algorithm outputs the parameters of the classification model: the regression matrix $\mathbf{W} \in \mathbb{R}^{p\times q}$ and the offset term $b \in \mathbb{R}$. Then, the class label of a new data $\mathbf{X}$ can be predicted by virtue of these parameters: $y = \mathrm{sgn}[\mathrm{tr}(\mathbf{W}^T\mathbf{X}) + b]$.

Formally, the matrix classification model SMM is defined as follows:

$$\arg\min_{\mathbf{W},b} \frac{1}{2}\mathrm{tr}(\mathbf{W}^T\mathbf{W}) + C\sum_{i=1}^{n}\{1 - y_i[\mathrm{tr}(\mathbf{W}^T\mathbf{X}_i) + b]\}_+ + \tau\|\mathbf{W}\|_*, \quad (1)$$

where $\{1 - y_i[\mathrm{tr}(\mathbf{W}^T\mathbf{X}_i) + b]\}_+$ is the hinge loss function and $\tau\|\mathbf{W}\|_*$ the penalty function.

To solve the objective function, Luo *et al.* [25] first rewrote Eq. (1) by introducing the constraint $\mathbf{S} - \mathbf{W} = 0$ and $G(\mathbf{S}) = \tau\|\mathbf{S}\|_*$ (where $\mathbf{S}$ can be regarded as a substitution for $\mathbf{W}$) and then applied the learning algorithm based on the ADMM (alternating direction method of multipliers). The ADMM solves the problem by using the augmented Lagrangian function:

$$L(\mathbf{W},b,\mathbf{S},\Lambda) = H(\mathbf{W},b) + G(\mathbf{S}) + \mathrm{tr}[\Lambda^T(\mathbf{S} - \mathbf{W})] + \frac{\rho}{2}\|\mathbf{S} - \mathbf{W}\|_F^2, \quad (2)$$

where ρ is a hyperparameter.

The overall procedure of the SMM learning algorithm is depicted in Fig. 1. In the kth iteration of the algorithm, the updating of the assistant parameters $\widehat{\mathbf{S}}^{(k+1)}$ and $\widehat{\Lambda}^{(k+1)}$ costs $O(1)$, which is much less costly than the first two procedures. Therefore, the core of the SMM learning process is the computations of $(\mathbf{W}^{(k)}, b^{(k)})$ and $\mathbf{S}^{(k)}$:

$$(\mathbf{W}^{(k)}, b^{(k)}) = \arg\min_{\mathbf{W},b} H(\mathbf{W},b) - \mathrm{tr}(\widehat{\Lambda}^{(k)T}\mathbf{W}) + \frac{\rho}{2}\|\mathbf{W} - \widehat{\mathbf{S}}^{(k)}\|_F^2, \quad (3)$$

$$\mathbf{S}^{(k)} = \arg\min_{\mathbf{S}} G(\mathbf{S}) + \mathrm{tr}(\widehat{\Lambda}^{(k)T}\mathbf{S}) + \frac{\rho}{2}\|\mathbf{W}^{(k)} - \mathbf{S}\|_F^2. \quad (4)$$

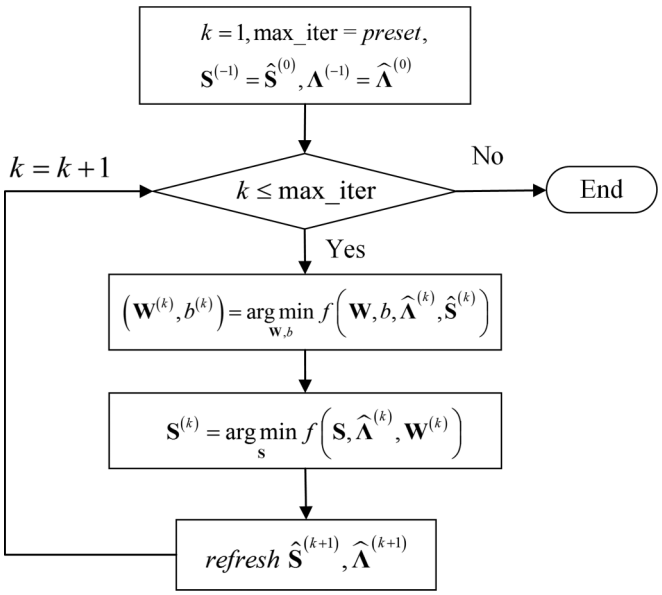

FIG. 1. Entire process of SMM learning algorithm.

In Ref. [25] it is shown that the update equation of $\mathbf{S}^{(k)}$ can be obtained by singular value thresholding [26]:

$$\mathbf{S}^{(k)} = \frac{1}{\rho}\mathcal{D}_\tau(\rho\mathbf{W}^{(k)} - \widehat{\Lambda}^{(k)}) = \frac{1}{\rho}\mathbf{U}_0(\Sigma_0 - \tau\mathbf{I})\mathbf{V}_0^T, \quad (5)$$

where $\mathbf{U}_0$ and $\mathbf{V}_0$ are matrices of the left and right singular vectors of $\rho\mathbf{W}^{(k)} - \widehat{\Lambda}^{(k)}$, with the corresponding diagonal matrix whose diagonal values are greater than τ.

In summary, the ultimate target of a SMM learning algorithm is to gain the regression matrix $\mathbf{W}$ and the offset b. This process of learning also involves an additional updating of substitution $\mathbf{S}$.

III. QUANTUM SMM

In this section, we design a quantum algorithm for a SMM based on a reformulation of the original SMM classification model. First, we show how to transform Eq. (3) to a linear problem that the quantum matrix inversion algorithm can address directly. Then, we sketch the basic idea of our quantum algorithm for a SMM and present in detail the key procedures of the algorithm. At the same time, we analyze its time complexity.

A. Least-squares reformulation

The quantum algorithm has shown good performance in solving the linear equation. In order to transform the original constraint quadratic programming problem in Eq. (3) to the solution of a linear equation system, we introduce the least-squares reformulation of the original SMM classification model. Here, we take a surrogate loss function, namely the square loss, instead of the hinge loss in Eq. (1).

First, we introduce a slack variable $e_i = 1 - y_i[\mathrm{tr}(\mathbf{W}^T\mathbf{X}_i) + b]$, $e_i \in \mathbb{R}$ to substitute the square loss $\frac{\eta}{2}\sum_{i=1}^n(1 - e_i - \mu)^2$ for the hinge loss $C\sum_{i=1}^n\{e_i\}_+$ in Eq. (1), as shown in Fig. 2. This naturally replaces the original inequality constraints with equality constraints.

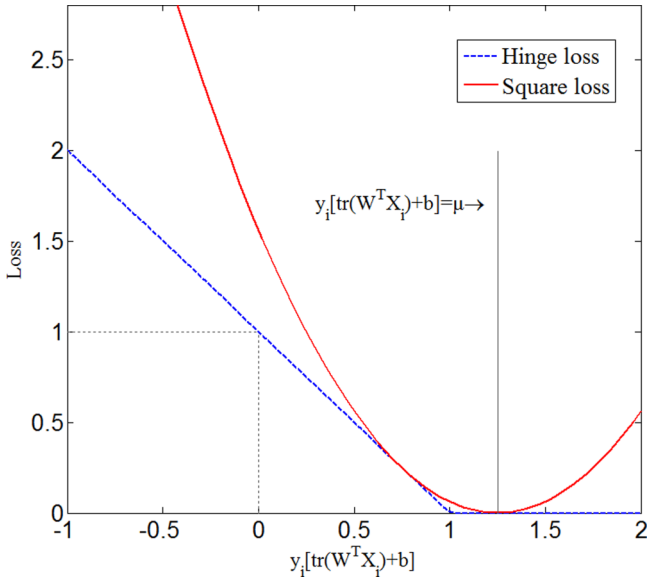

FIG. 2. Loss functions in a SMM. The blue dashed line represents the hinge loss ℓ^{hinge} in the original objective function and the red solid line the square loss ℓ^{square} used in our method. Here we introduce a new parameter μ to make ℓ^{hinge} upper bounded by ℓ^{square} strictly.

Theorem 1. There exists a square loss function

$$\ell^{\text{square}}(e_i) := (1 - e_i - \mu)^2,$$

which is a convex surrogate for the hinge loss:

$$\ell^{\text{hinge}}(e_i) := \max\{0, e_i\}.$$

Using the surrogate loss transforms the objective function (3) to the following problem:

$$\begin{aligned}\arg\min_{\mathbf{W},b,e_i} \quad & \frac{1}{2}\text{tr}(\mathbf{W}^T\mathbf{W}) + \frac{\eta}{2}\sum_{i=1}^{n}(1 - e_i - \mu)^2 - \text{tr}(\widehat{\Lambda}^{(k)T}\mathbf{W}) \\ & + \frac{\rho}{2}\|\mathbf{W} - \widehat{\mathbf{S}}^{(k)}\|_F^2, \\ \text{such that} \quad & y_i[\text{tr}(\mathbf{W}^T\mathbf{X}_i) + b] = 1 - e_i, \end{aligned} \tag{6}$$

where η is the regularization parameter, which plays the same role as the parameter C in Eq. (1).

Theorem 2. One of the solutions of the problem (6) is

$$\mathbf{W}^{(k)} = \frac{1}{\rho + 1}\left(\widehat{\Lambda}^{(k)} + \rho\widehat{\mathbf{S}}^{(k)} + \sum_{i=1}^{n}\alpha_i^{(k)} y_i \mathbf{X}_i\right), \tag{7}$$

and the offset $b^{(k)}$ and $\alpha^{(k)} = [\alpha_i{}^{(k)}] \in \mathbb{R}^n$ are the solutions of the following linear equation:

$$\mathbf{F}\begin{pmatrix} b^{(k)} \\ \tilde{\alpha}^{(k)} \end{pmatrix} \equiv \begin{pmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & \mathbf{K} + \mathbf{I}/\gamma \end{pmatrix}\begin{pmatrix} b^{(k)} \\ \tilde{\alpha}^{(k)} \end{pmatrix} = \begin{pmatrix} 0 \\ \Phi^{(k)} \end{pmatrix}, \tag{8}$$

where $\mathbf{K} = [K_{ij}] \in \mathbb{R}^{n\times n}$ and $\Phi^{(k)} = [\Phi_i{}^{(k)}] \in \mathbb{R}^n$ are independent of $\tilde{\alpha}^{(k)} = [\alpha_i^{(k)} y_i] \in \mathbb{R}^n$; specifically,

$$\begin{aligned} K_{ij} &= \frac{\text{tr}(\mathbf{X}_i^T\mathbf{X}_j)}{\rho + 1}, \\ \Phi_i^{(k)} &= \mu y_i - \frac{\text{tr}[(\widehat{\Lambda}^{(k)} + \rho\widehat{\mathbf{S}}^{(k)})^T\mathbf{X}_i]}{\rho + 1}. \end{aligned} \tag{9}$$

The proofs of the two theorems are shown in Appendices A and B.

By Theorem 2, updating $(\mathbf{W}^{(k)}, b^{(k)})$ can be done by solving Eq. (8). This simplification fits the situation in which the quantum matrix inversion algorithm works.

B. Algorithm

Our quantum algorithm for SMMs is based on the following model. Assume that the matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{p\times q}$ is given by a quantum oracle $O_{\mathbf{A}}$:

$$O_{\mathbf{A}}|i\rangle|j\rangle|0\rangle = |i\rangle|j\rangle|a_{ij}\rangle, \forall i \in \{1,\ldots,p\}, \forall j \in \{1,\ldots,q\}. \tag{10}$$

In addition, we assume that the vector $\mathbf{y} = [y_i] \in \mathbb{R}^n$ is given by a quantum oracle $O_{\mathbf{y}}$:

$$O_{\mathbf{y}}|i\rangle|0\rangle = |i\rangle|y_i\rangle, \quad \forall i \in \{1,\ldots,n\}. \tag{11}$$

That is, given the indexes i and j, the oracles $O_{\mathbf{A}}$ and $O_{\mathbf{y}}$ can return the values of $\mathbf{A}$ and $\mathbf{y}$, respectively.

Then, via the method outlined in Ref. [22], the following quantum state can be generated:

$$|\psi_{\mathbf{A}}\rangle = \sum_{i=1}^{p}\sum_{j=1}^{q} a_{ij}|i\rangle|j\rangle, \tag{12}$$

$$|\psi_{\mathbf{y}}\rangle = \sum_{i=1}^{p} y_i|i\rangle. \tag{13}$$

Note that our algorithm works with the normalized data, namely matrices and vectors, as these data in many machine learning algorithms are normalized.

Based on this model, our algorithm consists of two subroutines: a quantum matrix inversion (HHL) algorithm for estimating $\mathbf{W}^{(k)}$ and $b^{(k)}$ and a QSVT algorithm for estimating $\mathbf{S}^{(k)}$.

The overall procedure of our algorithm includes the following steps.

Algorithm. $(\mathbf{W}, b) = QSMM(\Lambda, \mathbf{S}, \mathbf{X}, \mathbf{y})$.

(1) Initialize $k = 1$, apply the quantum algorithm $(b^{(k)}, \tilde{\alpha}^{(k)}) = \text{HHL}(\Phi^{(k)}, \mathbf{F})$, where $(\Phi^{(k)}, \mathbf{F})$ can be prepared in terms of $(\mathbf{X}, \mathbf{y}, \widehat{\Lambda}^{(k)}, \widehat{\mathbf{S}}^{(k)})$ in Eq. (9).

(2) Extract $b^{(k)}$ and $\tilde{\alpha}^{(k)}$ to construct the matrix $\mathbf{W}^{(k)}$ according to Eq. (7).

(3) Apply the quantum algorithm $\mathbf{S}^{(k)} = \text{QSVT}(\mathbf{A}^{(k)}, \tau)$, where $\mathbf{A}^{(k)}$ is a normalized r-rank approximate of $\rho\mathbf{W}^{(k)} - \widehat{\Lambda}^{(k)}$.

(4) Update the parameters $\widehat{\Lambda}^{(k+1)}$ and $\widehat{\mathbf{S}}^{(k+1)}$ and set $k = k + 1$.

(5) Repeat Steps 1–4 until the number of iterations $k =$ max_iter; then, one can obtain the regression matrix $\mathbf{W}$ and the offset b.

We begin to introduce the quantum algorithms $(b, \tilde{\alpha}) = \text{HHL}(\Phi, \mathbf{F})$ and $\mathbf{S} = \text{QSVT}(\mathbf{A}, \tau)$ in detail in the following subsections.

1. HHL algorithm

To obtain $\mathbf{W}^{(k)}$ in Eq. (7), we should solve the normalized $(b^{(k)}, \tilde{\alpha}^{(k)})^T = \mathbf{F}^{-1}(0, \Phi^{(k)})^T$, where $\|\mathbf{F}\| \leqslant 1$. The HHL

algorithm was proposed for solving matrix inversion problems [7]. But the matrix should be s-sparse in Ref. [7]. In this paper, we adopted a strategy in Ref. [5], therefore the HHL algorithm can be used when the matrix is nonsparse. The detailed description is shown in Appendix C. The core subroutine of the algorithm is phase estimation (denoted $\mathbf{U}_{\mathrm{PE}}$). Apply $\mathbf{U}_{\mathrm{PE}}$ to the registers C and B, which is represented as follows:

$$\begin{aligned}\mathbf{U}_{\mathrm{PE}} &= \mathbf{U}_{\mathrm{PE}}(\mathbf{A}) \\ &= (\mathbf{F}_{\mathbf{T}}^{\dagger}\otimes\mathbf{I}^{B})\left(\sum_{\tau=0}^{T-1}|\tau\rangle\langle\tau|^{C}\otimes e^{i\mathbf{A}\tau t_0/T}\right)(\mathbf{H}^{\otimes t}\otimes\mathbf{I}^{B}),\end{aligned} \tag{14}$$

where register C with t qubits is used to store the estimated eigenvalues of an Hermite matrix $\mathbf{A}$, register B with s qubits stores the input state $|\psi\rangle$, $\mathbf{F}_{\mathbf{T}}^{\dagger}$ is the inverse quantum Fourier transform, and $\sum_{\tau=0}^{T-1}|\tau\rangle\langle\tau|^{C}\otimes e^{i\mathbf{A}\tau t_0/T}$ is the conditional Hamiltonian evolution [7].

We now give the HHL algorithm for solving the reformulated SMM to estimate the parameters $\mathbf{W}^{(k)}$ and $b^{(k)}$. In Appendix C, we show how to prepare the input $|\psi_{\Phi}\rangle$ and $e^{-i\mathbf{F}t_0}$.

Input. A quantum state $|\psi_{\Phi}\rangle$ and a unitary $e^{-i\mathbf{F}t_0}$.

Output. A quantum state proportional to $|b,\tilde{\alpha}\rangle \approx \mathbf{F}^{-1}|\psi_{\Phi}\rangle$.

Algorithm. $(b,\tilde{\alpha}) = \mathrm{HHL}(\Phi,\mathbf{F})$.

(1) Prepare three quantum registers: register C for storing the estimated eigenvalues $|\lambda_i\rangle$ of $\mathbf{F}$, register B for storing the input state $|\psi_{\Phi}\rangle$, and register R with an ancilla qubit for storing the inverse of the eigenvalues λ_i^{-1} of $\mathbf{F}$. Prepare the three registers in the state

$$|\psi_0\rangle = |0\rangle^{R}(|0\rangle|0\rangle\cdots|0\rangle)^{C}(|\psi_{\Phi}\rangle)^{B}, \tag{15}$$

where $|\psi_{\Phi}\rangle = |0,\Phi\rangle = \sum_{i=1}^{n+1}\phi_i|i\rangle$.

(2) Perform the unitary operation $\mathbf{U}_{\mathrm{PE}}(\mathbf{F})$ on the state. $\mathbf{U}_{\mathrm{PE}}(\mathbf{F})$ expands $|\psi_{\Phi}\rangle$ into the eigenstates $|\mathbf{u}_i\rangle$ of $\mathbf{F}$ with corresponding eigenvalues λ_i and we have the state

$$|\psi_1\rangle = |0\rangle^{R}\sum_{i=1}^{n+1}\langle\mathbf{u}_i|\psi_{\Phi}\rangle|\lambda_i\rangle^{C}|\mathbf{u}_i\rangle^{B}. \tag{16}$$

(3) Apply a controlled rotation $\mathbf{R}_f^{\mathrm{RC}}$,

$$|0\rangle^{R}|z\rangle^{C} \to \left(\frac{\gamma}{z}|1\rangle + \sqrt{1-\frac{\gamma^2}{z^2}}|0\rangle\right)^{R}|z\rangle^{C}, \tag{17}$$

to the register R, controlled by the register C, where γ is a constant. This rotation transforms the state to

$$|\psi_2\rangle = \left(\sqrt{1-\frac{\gamma^2}{\lambda_i^2}}|0\rangle + \frac{\gamma}{\lambda_i}|1\rangle\right)^{R}\sum_{i=1}^{n+1}\langle\mathbf{u}_i|\psi_{\Phi}\rangle|\lambda_i\rangle^{C}|\mathbf{u}_i\rangle^{B}. \tag{18}$$

(4) Uncompute the registers C and B to obtain the state

$$|\psi_3\rangle = \left(\sqrt{1-\frac{\gamma^2}{\lambda_i^2}}|0\rangle + \frac{\gamma}{\lambda_i}|1\rangle\right)^{R}(|0\cdots0\rangle)^{C}\sum_{i=1}^{n+1}\langle\mathbf{u}_i|\psi_{\Phi}\rangle|\mathbf{u}_i\rangle^{B}. \tag{19}$$

Remove the register C and measure the register R to be $|1\rangle$. We have the state proportional to

$$|\psi_4\rangle = \sum_{i=1}^{n+1}\langle\mathbf{u}_i|\psi_{\Phi}\rangle/\lambda_i|\mathbf{u}_i\rangle. \tag{20}$$

(5) Finally, represent the state in the basis of training set labels to acquire the desired parameters that exist in the coefficients of the state [8]:

$$|b,\tilde{\alpha}\rangle = \frac{1}{\sqrt{l}}\left(b|0\rangle + \sum_{i=1}^{n}\tilde{\alpha}_i|i\rangle\right), \tag{21}$$

with $l = b^2 + \sum_{i=1}^{n}\tilde{\alpha}_i^2$.

Here, amplitude estimation can then be used to extract $b^{(k)}$ and $\tilde{\alpha}^{(k)}$, and now we can obtain the matrix $\mathbf{W}^{(k)}$ according to Eq. (7).

In summary, the HHL algorithm can be represented as the following unitary operation:

$$\mathbf{U}_{\mathrm{HHL}} = (\mathbf{I}^{R}\otimes\mathbf{U}_{\mathrm{PE}}^{\dagger})(\mathbf{R}_f^{\mathrm{RC}}\otimes\mathbf{I}^{B})(\mathbf{I}^{R}\otimes\mathbf{U}_{\mathrm{PE}}), \tag{22}$$

where $\mathbf{R}_f^{\mathrm{RC}}$ is given in Eq. (17).

Analysis. We now discuss the error of the HHL algorithm. Define the condition number of $\mathbf{F}$ as κ, and $t_0 = O(\kappa/\varepsilon)$. Therefore, only the eigenvalues in the interval $1/\kappa \leqslant |\lambda_j| \leqslant 1$ are taken into account. The estimation of eigenvalue $\tilde{\lambda}_j := 2\pi j/t_0$ in step (2) introduces error $O(1/t_0) = O(\varepsilon/\kappa)$ which translates into error $O(\varepsilon/(\lambda\kappa))$ in λ^{-1} in step (3) [7]. With $\lambda \geqslant 1/\kappa$ taking $t_0 = O(\kappa/\varepsilon)$ induces a final error of $O(\varepsilon)$.

We now analyze the time complexity. In phase estimation, $O(t^2)$ operations and one call to the conditional Hamiltonian evolution ($\sum_{\tau=0}^{T-1}|\tau\rangle\langle\tau|^{C}\otimes e^{i\mathbf{F}\tau t_0/T}$) are needed, where t is the number of qubits in register C [4]. The preparation of the unitary $e^{-i\mathbf{F}t_0}$ costs time $O[t_0^2\varepsilon^{-1}\log(npq)]$ which is shown in Appendix C. As t decides the accuracy of the estimated eigenvalues, t is not very large to some extent. When the size of the training data and feature space is large, the preparation of the unitary $e^{-i\mathbf{F}t_0}$ dominates the time complexity of phase estimation.

The probability of obtaining λ^{-1} determines the number of iterations of the algorithm. This probability is determined by the amplitude square γ/λ_i, and this value is at least $\Omega(1/\kappa^2)$ with $\gamma = O(1/\kappa)$ and $\lambda \leqslant 1$. Hence, $O(\kappa^2)$ repetitions are needed to ensure success with high probability. By virtue of the amplitude amplification algorithm [15], only $O(\kappa)$ repetitions are sufficient to achieve a constant success probability of the postselection process.

In conclusion, the total evolution time of the algorithm is $O[t_0^2\varepsilon^{-1}\log(npq)]O(\kappa) = O[\kappa^3\varepsilon^{-3}\log(npq)]$. In contrast, the matrix inversion via classical method needs time $O[\mathrm{poly}(npq)]$.

2. *QSVT algorithm*

Herein we propose a QSVT algorithm for estimating $\mathbf{S}^{(k)}$. Suppose $\mathbf{A}^{(k)}$ is a normalized matrix that is a good r-rank approximate of $\rho\mathbf{W}^{(k)} - \widehat{\Lambda}^{(k)}$, with $\|\mathbf{A}^{(k)}\| \leqslant 1$. In the following, we simplify the notation $\mathbf{A}^{(k)}$ as $\mathbf{A}$. Then, using the

Gram-Schmidt decomposition, we have

$$|\psi_{\mathbf{A}}\rangle = \sum_{i=1}^{p}\sum_{j=1}^{q} a_{ij}|i\rangle|j\rangle = \sum_{k=1}^{r}\sigma_k|\mathbf{u}_k\rangle|\mathbf{v}_k\rangle, \tag{23}$$

where r is the rank of $\mathbf{A}$, and σ_k are just the singular values of $\mathbf{A}$, with $\mathbf{u}_k$ and $\mathbf{v}_k$ being the left and right singular vectors. Note that $\mathbf{u}_k$ are just the eigenvectors of $\mathbf{A}\mathbf{A}^\dagger$ and $\mathbf{v}_k$ are the eigenvectors of $\mathbf{A}^\dagger\mathbf{A}$. The eigenvalues of $\mathbf{A}\mathbf{A}^\dagger$ or $\mathbf{A}^\dagger\mathbf{A}$ are both $\lambda_k = \sigma_k^2$. Based on quantum Random Access Memories (QRAM), the preparation of $|\psi_{\mathbf{A}}\rangle$ costs time $O[\log(pq)]$.

By taking a partial trace of $|\psi_{\mathbf{A}}\rangle\langle\psi_{\mathbf{A}}|$, we can obtain a mixed state $\rho_{\mathbf{A}\mathbf{A}^\dagger}$ [10]:

$$\rho_{\mathbf{A}\mathbf{A}^\dagger} = \mathrm{tr}_j(|\psi_{\mathbf{A}}\rangle\langle\psi_{\mathbf{A}}|) = \sum_{i,i'=1}^{p}\sum_{j=1}^{q} a_{ij}a^*_{i'j}|i\rangle\langle i'|, \tag{24}$$

where $\rho_{\mathbf{A}\mathbf{A}^\dagger}$ represents the positive Hermitian matrix $\mathbf{A}\mathbf{A}^\dagger$.

The QSVT algorithm is now presented as follows:

Input. A quantum state $|\psi_{\mathbf{A}}\rangle$, a unitary $e^{-i\rho_{\mathbf{A}\mathbf{A}^\dagger}t_0}$, and a constant τ.

Output. A quantum state proportional to $|\psi_{\mathbf{S}}\rangle = \sum_{k=1}^{r'}(\sigma_k - \tau)|\mathbf{u}_k\rangle|\mathbf{v}_k\rangle$.

Algorithm. $\mathbf{S} = QSVT(\mathbf{A},\tau)$.

(1) Prepare three quantum registers in the state

$$|\psi_0\rangle = |0\rangle^R(|0\rangle|0\rangle\cdots|0\rangle)^C(|\psi_{\mathbf{A}}\rangle)^B. \tag{25}$$

(2) Perform the unitary operation $\mathbf{U}_{\mathrm{PE}}(\rho_{\mathbf{A}\mathbf{A}^\dagger})$ on the state. According to the ideas of *quantum principal component analysis* in Refs. [5,10], we have the state

$$|\psi_1\rangle = |0\rangle^R\sum_{k=1}^{r}\sigma_k\left|\sigma_k^2\right\rangle^C|\mathbf{u}_k\rangle|\mathbf{v}_k\rangle. \tag{26}$$

(3) Apply a controlled rotation $\mathbf{R}^{\mathrm{RC}}_{f'}$ to the register R, controlled by the register C. $\mathbf{R}^{\mathrm{RC}}_{f'}$ is defined as follows: if $\sigma_j > \tau$,

$$\begin{aligned}&|0\rangle^R|z\rangle^C\\ &\to\left(\frac{\gamma(\sqrt{z}-\tau)}{\sqrt{z}}|1\rangle + \sqrt{1-\frac{\gamma^2(\sqrt{z}-\tau)^2}{z}}|0\rangle\right)^R|z\rangle^C;\end{aligned} \tag{27}$$

otherwise do nothing.

This rotation transforms the state to

$$\begin{aligned}|\psi_2\rangle = &\left(\frac{\gamma(\sigma_k-\tau)}{\sigma_k}|1\rangle + \sqrt{1-\frac{\gamma^2(\sigma_k-\tau)^2}{\sigma_k^2}}|0\rangle\right)^R\\ &\otimes\sum_{k=1}^{r'}\sigma_k\left|\sigma_k^2\right\rangle^C|\mathbf{u}_k\rangle|\mathbf{v}_k\rangle\\ &+|0\rangle^R\sum_{k=r'+1}^{r}\sigma_k\left|\sigma_k^2\right\rangle^C|\mathbf{u}_k\rangle|\mathbf{v}_k\rangle.\end{aligned} \tag{28}$$

(4) Uncompute the registers C and B, remove the register C, and measure the register R to be $|1\rangle$. Then, we have the state proportional to

$$|\psi_3\rangle = \sum_{k=1}^{r'}(\sigma_k-\tau)|\mathbf{u}_k\rangle|\mathbf{v}_k\rangle = |\psi_{\mathbf{S}}\rangle. \tag{29}$$

In summary, the QSVT algorithm can be represented as the following unitary operation:

$$\mathbf{U}_{\mathrm{QSVT}} = (\mathbf{I}^R\otimes\mathbf{U}^\dagger_{\mathrm{PE}})\left(\mathbf{R}^{\mathrm{RC}}_{f'}\otimes\mathbf{I}^B\right)(\mathbf{I}^R\otimes\mathbf{U}_{\mathrm{PE}}), \tag{30}$$

where $\mathbf{R}^{\mathrm{RC}}_{f'}$ is proposed in Eq. (27).

Analysis. We now discuss the error of the QSVT algorithm. Define the condition number of $\rho_{\mathbf{A}\mathbf{A}^\dagger}$ as κ, and $t_0 = O(\kappa/\varepsilon)$. Therefore, only the eigenvalues σ_k^2 in the interval $1/\kappa \leqslant \sigma_k^2 \leqslant 1$ are taken into account. The estimation of eigenvalues $\tilde{\sigma}_k^2 := 2\pi k/t_0$ in step (2) introduces error $O(1/t_0) = O(\varepsilon/\kappa)$. This error translates into error $O[\varepsilon/(\sigma_k\kappa)]$ in $(1-\tau/\sigma_k)$ after the controlled rotation $\mathbf{R}^{\mathrm{RC}}_{f'}$ in step (3). With $\sigma_k^2 \geqslant 1/\kappa$ taking $t_0 = O(\kappa/\varepsilon)$ induces a final error of $O(\sqrt{\varepsilon/\kappa})$.

We now discuss the time complexity of the QSVT algorithm. Clearly, the preparation of the unitary operation $e^{-i\rho_{\mathbf{A}\mathbf{A}^\dagger}t_0}$ dominates the time complexity of phase estimation, and it costs time $O[t_0^2\varepsilon^{-1}\log(pq)]$. The probability of obtaining $(\sigma_k-\tau)$ determines the number of iterations of the algorithm, and it is determined by the amplitude square $\gamma(\sigma_k-\tau)/\sigma_k$. This value is at least $\Omega(1/\kappa^2)$ with $\gamma = O(1/\kappa)$ and $\sigma_k \leqslant 1$. Hence, $O(\kappa^2)$ repetitions are needed to ensure success with high probability. Using amplitude amplification [15], to achieve a constant success probability of the postselection process only needs $O(\kappa)$ repetitions. Therefore, the total evolution time of the algorithm is $O[t_0^2\varepsilon^{-1}\log(pq)]O(\kappa) = O[\kappa^3\varepsilon^{-3}\log(pq)]$. In contrast, the matrix inversion via the classical method needs time $O[\mathrm{poly}(pq)]$.

C. Classification

With the parameters $\mathbf{W}$ and b, we can classify a query state $\mathbf{X}$ via the method presented in Ref. [8]. First, we construct the training-data state via quantum oracles:

$$|\psi_{\tilde{\mathbf{t}}}\rangle = \frac{1}{\sqrt{N_{\mathbf{t}}}}\left(b|0\rangle|0\rangle + |0\rangle|\lambda_\Lambda\rangle + \sum_{\tilde{\alpha}_i y_i>0}\tilde{\alpha}_i|i\rangle|\mathbf{x}_i\rangle\right), \tag{31}$$

and the query state

$$|\psi_{\tilde{\mathbf{x}}}\rangle = \frac{1}{\sqrt{N_{\mathbf{x}}}}\left(|0\rangle|0\rangle + \sum_{i=0}^{n}|i\rangle|\mathbf{x}\rangle\right), \tag{32}$$

where $N_{\mathbf{t}}$ and $N_{\mathbf{x}}$ are the norms of these states. The inner product of the two states

$$\langle\psi_{\tilde{\mathbf{t}}}|\psi_{\tilde{\mathbf{x}}}\rangle = \frac{1}{\sqrt{N_{\mathbf{t}}N_{\mathbf{x}}}}\left(b + \langle\lambda_\Lambda|\mathbf{x}\rangle + \sum_{\tilde{\alpha}_i y_i>0}\tilde{\alpha}_i\langle\mathbf{x}_i|\mathbf{x}\rangle\right) \tag{33}$$

reveals the label of the classification problem

$$y_j = \mathrm{sgn}\left[\left(\lambda_\Lambda + \sum_{\tilde{\alpha}_i y_i>0}\tilde{\alpha}_i\mathbf{x}_i\right)^{\mathrm{T}}\mathbf{x} + \mathrm{b}\right]. \tag{34}$$

Then, a swap-test operation can be used to find the value, and the algorithm achieves an exponential increase of speed over its classical counterpart.

IV. CONCLUSIONS

We have shown that quantum mechanics can be used to speed up the learning and classification process of an important matrix classifier SMM in machine learning. The algorithmic complexity of the two key subroutines can be $O[\log(npq)]$ and $O[\log(pq)]$, respectively. Aiming at the two cores of the SMM learning procedure, the least-squares reformulation has been utilized for estimating the parameters $(\mathbf{W}^{(k)}, b^{(k)})$; therefore, the HHL algorithm can be used to address the reformulated SMM classification model. Furthermore, a QSVT algorithm is proposed for estimating $\mathbf{S}^{(k)}$. This algorithm can also be a subroutine in other quantum machine learning algorithms for solving the singular value thresholding problem. Finally, we give the classification process, which can be implemented quantum mechanically. Our analysis shows that the core subprocedures can achieve an exponential acceleration over classical counterparts. We hope that our algorithm can inspire more fruitful results in quantum machine learning.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grants No. 61571226 and No. 61701229), the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20140823), the Jiangsu Innovation Program for Graduate Education (Grant No. KYLX15_0326), and the Fundamental Research Funds for the Central Universities.

APPENDIX A: PROOF OF THEOREM 1

Proof. Obviously, for all the e_i, $\ell^{\text{hinge}}(e_i) \leqslant \ell^{\text{square}}(e_i)$ when $\mu \geqslant 5/4$. Then, the square loss is the upper bound of the hinge loss. Along with the convexity of the square function, $\ell^{\text{square}}(e_i)$ meets the surrogate property [27].

APPENDIX B: PROOF OF THEOREM 2

Proof. To solve Eq. (6), we construct the Lagrange function as follows:

$$\begin{aligned} L(\mathbf{W},b,\mathbf{e};\alpha) = {} & \frac{1}{2}\mathrm{tr}(\mathbf{W}^T\mathbf{W}) + \frac{\gamma}{2}\sum_{i=1}^{n}(1-e_i-\mu)^2 \\ & - \mathrm{tr}(\widehat{\Lambda}^{(k)T}\mathbf{W}) + \frac{\rho}{2}\|\mathbf{W}-\widehat{\mathbf{S}}^{(k)}\|_F^2 \\ & - \sum_{i=1}^{n}\alpha_i\{y_i[tr(\mathbf{W}^T\mathbf{X}_i)+b]-1+e_i\}, \end{aligned} \tag{B1}$$

where α_i are the Lagrange multipliers.

Taking partial derivatives of L with respect to $\mathbf{W}$, b, e_i, and α_i to be zero, respectively, we have

$$\begin{aligned} & \frac{\partial L}{\partial \mathbf{W}} = 0 \to \mathbf{W} = \frac{1}{\rho+1}\left(\widehat{\Lambda}^{(k)} + \rho\widehat{\mathbf{S}}^{(k)} + \sum_{i=1}^{n}\alpha_i y_i \mathbf{X}_i\right), \\ & \frac{\partial L}{\partial b} = 0 \to \sum_{i=1}^{n}\alpha_i y_i = 0, \\ & \frac{\partial L}{\partial e_i} = 0 \to e_i = \frac{\alpha_i}{\gamma} - \mu + 1,\ i = 1,\ldots,n, \\ & \frac{\partial L}{\partial \alpha_i} = 0 \to y_i[\mathrm{tr}(\mathbf{W}^T\mathbf{X}_i)+b] - 1 + e_i = 0,\ i = 1,\ldots,n. \end{aligned} \tag{B2}$$

Then, with $y_i^2 = 1$, substituting the first and third equations into the fourth in Eqs. (B2) to eliminate $\mathbf{W}$ and e_i, we obtain $b + \sum_{j=1}^{n}\alpha_j y_j \frac{\mathrm{tr}(\mathbf{X}_i^T\mathbf{X}_j)}{\rho+1} + \frac{\alpha_i y_i}{\gamma} = \mu y_i - \frac{\mathrm{tr}[(\Lambda^{(k)}+\rho\mathbf{S}^{(k)})^T\mathbf{X}_i]}{\rho+1}$, $(i = 1,\ldots,n)$; thus, with the second equation in Eqs. (B2), the solution can be straightforwardly given by Eq. (8).

APPENDIX C: PREPARATION OF $|\psi_\Phi\rangle$, $e^{-i\mathbf{F}t_0}$, AND $e^{-i\rho_{\mathbf{AA}^\dagger}t_0}$

Preparation of the input $|\psi_\Phi\rangle$ of the HHL algorithm. The main part of Φ_i in Eq. (9) can be reworded as the inner product of two vectors:

$$\begin{aligned} & \mathrm{tr}[(\widehat{\Lambda}^{(k)} + \rho\widehat{\mathbf{S}}^{(k)})^T\mathbf{X}_i] \\ & \quad = \mathrm{vec}(\widehat{\Lambda}^{(k)T} + \rho\widehat{\mathbf{S}}^{(k)T})^T\mathrm{vec}(\mathbf{X}_i^T) \\ & \quad := \langle\lambda_\Lambda^{(k)}|\mathbf{x}_i\rangle + \rho\langle\mathbf{s}^{(k)}|\mathbf{x}_i\rangle, \end{aligned} \tag{C1}$$

where $\{|\mathbf{x}_i\rangle\}_{i=1}^n$, $|\mathbf{s}^{(k)}\rangle$, and $|\lambda_\Lambda^{(k)}\rangle$ are the quantum states in the basis of the training set space. Here, we have

$$|\psi_\Phi\rangle = \frac{1}{\sqrt{N_{\psi_\Phi}}}\left[|0\rangle + \sum_{i=1}^{n}\left(\mu y_i - \frac{\langle\lambda_\Lambda^{(k)}|\mathbf{x}_i\rangle + \rho\langle\mathbf{s}^{(k)}|\mathbf{x}_i\rangle}{\rho+1}\right)|i\rangle\right], \tag{C2}$$

with N_{ψ_Φ} being the norms of $|\psi_\Phi\rangle$. Based on QRAM [11], a swap test can be used to evaluate these inner products. Therefore, the preparation of $|\psi_\Phi\rangle$ costs time $O[\log(npq)]$ [12].

Preparation of the input $e^{-i\mathbf{F}t_0}$ of the HHL algorithm. The unitary $e^{-i\mathbf{F}t_0}$ can be simplified to the direct preparation and exponentiation of the matrix $\mathbf{K}$ [8]. To prepare the matrix $\mathbf{K}$ with elements

$$K_{ij} = \frac{\mathrm{tr}(\mathbf{X}_i^T\mathbf{X}_j)}{\rho+1} = \frac{\mathrm{vec}(\mathbf{X}_i^T)^T\mathrm{vec}(\mathbf{X}_j^T)}{\rho+1} = \frac{\mathbf{x}_i\cdot\mathbf{x}_j}{\rho+1},$$

we can take a partial trace of a quantum state

$$|\psi\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle|\mathbf{x}_i\rangle$$

to acquire the normalized

$$\mathbf{K} = \mathrm{tr}_2\{|\psi\rangle\langle\psi|\} := \sum_{i,j=1}^{n}\langle\mathbf{x}_i|\mathbf{x}_j\rangle|\mathbf{x}_i||\mathbf{x}_j||i\rangle\langle j|. \tag{C3}$$

Here, $|\psi\rangle$ can be prepared in time $O[log(npq)]$.

Then, the unitary $e^{-i\mathbf{F}t_0}$ can be simulated via the methods outlined in Ref. [8]. Here, t_0 is the total evolution time with $t_0 = \Delta t T$, i.e., the interval Δt times the number of steps T in the phase estimation. For the time slice Δt, the unitary $e^{-i\mathbf{F}\Delta t}$ can be simplified to the direct preparation and exponentiation

of the matrix $\mathbf{K}$ with error $O(\Delta t^2)$:

$$e^{-i\mathbf{F}\Delta t} = e^{-i\Delta t\gamma^{-1}\mathbf{I}}e^{-i\Delta t\mathbf{J}}e^{-i\Delta t\mathbf{K}} + O(\Delta t^2),$$

with $\mathbf{J} = \left(\begin{smallmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & 0 \end{smallmatrix}\right)$. As $\mathbf{K}$ in Eq. (C3) is Hermitian, the exponentiation of the matrix $\mathbf{K}$ can be prepared by applying the algorithm in Ref. [5] in terms of a density matrix description with error $O(\Delta t^2)$: $e^{-i\mathbf{K}\Delta t}(\sigma) \approx \mathrm{tr}_1\{e^{-iS_{\mathrm{wap}}\Delta t}\mathbf{K}\otimes\sigma e^{iS_{\mathrm{wap}}\Delta t}\} = \sigma - i\Delta t[\mathbf{K},\sigma] + O(\Delta t^2)$, where $S_{\mathrm{wap}} = \sum_{p,q=1}^{n}|p\rangle\langle q|\otimes|q\rangle\langle p|$. Thus, using $T = O(t_0^2\varepsilon^{-1})$ copies of $\mathbf{K}$ to implement $e^{-i\mathbf{K}t_0}$ to accuracy ε is in time $O[t_0^2\varepsilon^{-1}\log(npq)]$.

Preparation of the input $e^{-i\rho_{\mathbf{AA}^\dagger}t_0}$ of the QSVT algorithm. The unitary $e^{-i\rho_{\mathbf{AA}^\dagger}t_0}$ can also be simulated via $O(t_0^2\varepsilon^{-1})$ copies of $e^{-i\rho_{\mathbf{AA}^\dagger}\Delta t}$ for the time slice Δt: $e^{-i\rho_{\mathbf{AA}^\dagger}\Delta t}(\sigma) \approx tr_1\{e^{-iS_{\mathrm{wap}}\Delta t}\rho_{\mathbf{AA}^\dagger}\otimes\sigma e^{iS_{\mathrm{wap}}\Delta t}\}$. As the preparation of $|\psi_{\mathbf{A}}\rangle$ is $O[\log(pq)]$, the time complexity of implementing $e^{-i\ \rho_{\mathbf{AA}^\dagger}t_0}$ to accuracy ε is $O[t_0^2\varepsilon^{-1}\log(pq)]$.

[1] J. Adcock, E. Allen, M. Day, S. Frick, J. Hinchliff, M. Johnson, S. Morleyshort, S. Pallister, A. Price, and S. Stanisic, arXiv:1512.02900v1.

[2] M. Schuld, I. Sinayskiy, and F. Petruccione, Contemp. Phys. **56**, 172 (2014).

[3] P. Wittek, *Quantum Machine Learning What Quantum Computing Means to Data Mining* (Elsevier, Singapore, 2014).

[4] M. A. Nielsen and I. L. Chuang, in *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, Cambridge, England, 2010), pp. 1–59.

[5] S. Lloyd, M. Mohseni, and P. Rebentrost, Nat. Phys. **10**, 108 (2014).

[6] P. Rebentrost, A. Steffens, and S. Lloyd, arXiv:1607.05404v1.

[7] A. W. Harrow, A. Hassidim, and S. Lloyd, Phys. Rev. Lett. **103**, 150502 (2009).

[8] P. Rebentrost, M. Mohseni, and S. Lloyd, Phys. Rev. Lett. **113**, 130503 (2014).

[9] Y. Liu and S. Zhang, Theor. Comput. Sci. **657**, 38 (2016).

[10] M. Schuld, I. Sinayskiy, and F. Petruccione, Phys. Rev. A **94**, 022342 (2016).

[11] V. Giovannetti, S. Lloyd, and L. Maccone, Phys. Rev. Lett. **100**, 160501 (2008).

[12] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, Phys. Rev. Lett. **87**, 167902 (2001).

[13] L. K. Grover, Phys. Rev. Lett. **79**, 325 (1997).

[14] C. Durr and P. Hoyer, arXiv:quant-ph/9607014v2 (1996).

[15] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, Contemp. Math. **305**, 53 (2002).

[16] E. Aimeur, G. Brassard, and S. Gambs, Mach. Learn. **90**, 261 (2013).

[17] S. Lloyd, M. Mohseni, and P. Rebentrost, arXiv:1307.0411v2.

[18] N. Wiebe, A. Kapoor, and K. Svore, Quantum Inf. Comput. **15**, 0318 (2015).

[19] I. Kerenidis and A. Prakash, arXiv:1603.08675v3.

[20] H. K. Lau, R. Pooser, G. Siopsis, and C. Weedbrook, Phys. Rev. Lett. **118**, 080501 (2017).

[21] N. Wiebe, D. Braun, and S. Lloyd, Phys. Rev. Lett. **109**, 050505 (2012).

[22] A. Prakash, Ph.D thesis, Dissertations and Theses - Gradworks, 2014.

[23] I. Cong and L. Duan, New J. Phys. **18**, 073011 (2016).

[24] C. H. Yu, F. Gao, Q. L. Wang, and Q. Y. Wen, Phys. Rev. A **94**, 042311 (2016).

[25] L. Luo, Y. Xie, Z. Zhang, and W. J. Li, in *Proceedings of the 32nd International Conference on Machine Learning*, JMLR Workshop and Conference Proceedings 37 (JMLR.org, Lille, France, 2015), pp. 938–947.

[26] J. F. Cai, E. J. Candès, and Z. Shen, SIAM J. Optim. **20**, 1956 (2008).

[27] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms* (Cambridge University Press, Cambridge, England, 2014).