Quantum-Inspired Support Vector Machine

Chen Ding, Tian-Yi Bao, and He-Liang Huang

Abstract—Support vector machine (SVM) is a particularly powerful and flexible supervised learning model that analyzes data for both classification and regression, whose usual algorithm complexity scales polynomially with the dimension of data space and the number of data points. To tackle the big data challenge, a quantum SVM algorithm was proposed, which is claimed to achieve exponential speedup for least squares SVM (LS-SVM). Here, inspired by the quantum SVM algorithm, we present a quantum-inspired classical algorithm for LS-SVM. In our approach, a improved fast sampling technique, namely indirect sampling, is proposed for sampling the kernel matrix and classifying. We first consider the LS-SVM with a linear kernel, and then discuss the generalization of our method to non-linear kernels. Theoretical analysis shows our algorithm can make classification with arbitrary success probability in logarithmic runtime of both the dimension of data space and the number of data points for low rank, low condition number and high dimensional data matrix, matching the runtime of the quantum SVM.

Index Terms—Quantum-inspired algorithm, machine learning, support vector machine, exponential speedup, matrix sampling.

I. INTRODUCTION

SINCE the 1980s, quantum computing has attracted wide attention due to its enormous advantages in solving hard computational problems [1], such as integer factorization [2]–[4], database searching [5], [6], machine learning [7]–[11] and so on. In 1997, Daniel R. Simon offered compelling evidence that the quantum model may have significantly more complexity theoretic power than the probabilistic Turing machine [12]. However, it remains an interesting question where is the border between classical computing and quantum computing. Although many proposed quantum algorithms have exponential speedups over the existing classical algorithms, is there any way we can accelerate such classical algorithms to the same complexity of the quantum ones?

In 2018, inspired by the quantum recommendation system algorithm proposed by Iordanis Kerenidis and Anupam Prakash [13], Ewin Tang designed a classical algorithm to produce a recommendation algorithm that can achieve an exponential improvement on previous algorithms [14], which is a breakthrough that shows how to apply the subsampling strategy based on Alan Frieze, Ravi Kannan, and Santosh Vempala's 2004 algorithm [15] to find a low-rank approximation of a matrix. Subsequently, Tang continued to use same techniques to dequantize two quantum machine learning algorithms, quantum principal component analysis [16] and quantum supervised clustering [17], and shows classical algorithms could also match the bounds and runtime of the corresponding quantum algorithms, with only polynomial slowdown [18].

Later, András Gilyén *et al.* [19] and Nai-Hui Chia *et al.* [20] independently and simultaneously proposed a quantum-inspired matrix inverse algorithm with logarithmic complexity of matrix size, which eliminates the speedup advantage of the famous Harrow-Hassidim-Lloyd (HHL) algorithm [21] on certain conditions. Recently, Juan Miguel Arrazola *et al.* studied the actual performance of quantum-inspired algorithms and found that quantum-inspired algorithms can perform well in practice under given conditions. However, the conditions should be further reduced if we want to apply the algorithms to practical datasets [22]. All of these works give a very promising future for designing the quantum-inspired algorithm in the machine learning area, where matrix inverse algorithms are universally used.

Support vector machine (SVM) is a data classification algorithm which is commonly used in machine learning area [23], [24]. Extensive studies have been conducted on SVMs to boost and optimize their performance, such as the sequential minimal optimization algorithm [25], the cascade SVM algorithm [26], and the SVM algorithms based on Markov sampling [27], [28]. These algorithms offer promising speedup either by changing the way of training a classifier, or by reducing the size of training sets. However, the time complexity of current SVM algorithms are all polynomial of data sizes. In 2014, Patrick Rebentrost, Masoud Mohseni and Seth Lloyd proposed the quantum SVM algorithm [29], which can achieve an exponential speedup compared to the classical SVMs. The time complexity of quantum SVM algorithm is polynomial of the logarithm of data sizes. Inspired by the quantum SVM algorithm, Tang's methods [14] and András Gilyén *et al.*'s work [19], we propose a quantum-inspired classical SVM algorithm, which also shows exponential speedup compared to previous classical SVM for low rank, low condition number and high dimensional data matrix. Both quantum SVM algorithm [29] and our quantum-inspired SVM algorithm are least squares SVM (LS-SVM), which reducing the optimization problem to finding the solution of a set of linear equations.

Our algorithm is a dequantization of the quantum SVM algorithm [29]. In quantum SVM algorithm, the labeled data vectors (x_j for $j = 1, ..., m$) are mapped to quantum vec-

Chen Ding is with CAS Centre for Excellence and Synergetic Innovation Centre in Quantum Information and Quantum Physics, University of Science and Technology of China, Hefei, Anhui 230026, China (e-mail: inrm@mail.ustc.edu.cn).

Tian-Yi Bao is with Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, OXFORD, OX1 3QD, UK (e-mail: tianyi.bao@exeter.ox.ac.uk).

He-Liang Huang is with Hefei National Laboratory for Physical Sciences at Microscale and Department of Modern Physics, University of Science and Technology of China, Hefei, Anhui 230026, China, and also with CAS Centre for Excellence and Synergetic Innovation Centre in Quantum Information and Quantum Physics, University of Science and Technology of China, Hefei, Anhui 230026, China (e-mail: quanhhl@ustc.edu.cn).

tors $|x_j\rangle = 1/|x_j| \sum (x_j)_k |k\rangle$ via a quantum random access memory (qRAM) and the kernel matrix is prepared using quantum inner product evaluation [17]. Then the solution of SVM is found by solving a linear equation system related to the quadratic programming problem of SVM using the quantum matrix inversion algorithm [21]. In our quantum-inspired SVM, the labeled vectors are stored in an arborescent data structure which provides the ability to random sampling within logarithmic time of the vector lengths. By performing sampling on these labeled vectors both by their numbers and lengths to get a much smaller dataset, we then find the approximate singular value decomposition of the kernel matrix. And finally, we solve the optimization problem and perform classification based on the solved parameters.

Our methods, particularly the sampling technique, is based on [14], [19]. However, the previous sampling techniques cannot be simply copied to solve the SVM tasks, since we don't have an efficient direct sampling access to the kernel matrix we want to perform matrix inversion on (see Section II-B for a more detailed explanation). Hence we have developed an indirect sampling technique to solve such problem. In the whole process, we need to avoid the direct multiplication on the vectors or matrices with the same size as the kernel, in case losing the exponential speedup. We first consider the LS-SVM with linear kernels, no regularization terms and no bias of the classification hyperplane, which could be regarded as the prototype for quantum-inspired techniques applied in various SVMs. Then we show that the regularization terms can be easily included in the algorithm in Section III. Finally, we discuss the generalization of our method to non-linear kernels in Section VII-C and the general case without the constraint on biases of classification hyperplanes in Section VII-D. Theoretical analysis shows that our quantum-inspired SVM can achieve exponential speedup over existing classical algorithms under several conditions. Experiments are carried out to demonstrate the feasibility of our algorithm. The indirect sampling developed in our work opens up the possibility of a wider application of the sampling methods into the field of machine learning.

II. PRELIMINARY

A. Notations

We list some matrix-related notations used in this paper.

B. Least squares SVM

Suppose we have m data points $\{(x_j, y_j) : x_j \in \mathbb{R}^n, y_j = \pm 1\}_{j=1,\ldots,m}$, where $y_j = \pm 1$ depending on the class which x_j belongs to. Denote $(x_1, ..., x_m)$ by X and $(y_1, \ldots, y_m)^T$ by y. A SVM finds a pair of parallel hyperplanes $x \cdot w + b = \pm 1$ that divides the points into two classes depending on the given data. Then for any new input points, it can make classification by its relative position with the hyperplanes.

We make a following assumption on the dataset so as to simplify the problem: Assume these data points are equally distributed on both sides of a hyperplane that passes through the origin and their labels are divided by such hyperplane.

TABLE I
THE NOTATIONS

Symbol	Meaning
A	matrix A
y	vector y or matrix y with only one column
A^+	pseudo inverse of A
A^T	transpose of A
A^{+T}	transpose of pseudo inverse of A
$A_{i,*}$	i-th row of A
$A_{*,j}$	j-th column of A
$\|A\|$	2-operator norm of A
$\|A\|_F$	Frobenius norm of A
$Q(\cdot)$	time complexity for querying an element of $\cdot$
$L(\cdot)$	time complexity for sampling an element of $\cdot$

Thus we assume $b = 0$. An generalized method for $b \neq 0$ is discussed in Section VII-D.

According to [24], the optimization problem of LS-SVM with linear kernel is

$$\min_{w,b,e} \mathcal{L}_1(w, b, e) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^{m} e_k^2,$$

subject to $\quad y_k(w^T x_k + b) = 1 - e_k, \quad k = 1, \ldots, m.$

Take $b = 0$, we get

$$\min_{w,e} \mathcal{L}_2(w, e) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^{m} e_k^2,$$

subject to $\quad y_k w^T x_k = 1 - e_k, \quad k = 1, \ldots, m.$

One defines the Lagrangian

$$\mathscr{L}(w, e, \mu) = \mathcal{L}_2(w, e) - \sum_{k=1}^{m} \mu_k (y_k w^T x_k - 1 + e_k).$$

The condition for optimality

$$\frac{\partial \mathscr{L}}{\partial w} = 0 \rightarrow w = \sum_{k=1}^{m} \mu_k y_k x_k,$$

$$\frac{\partial \mathscr{L}}{\partial e_k} = 0 \rightarrow \mu_k = \gamma e_k, k = 1, \ldots, m,$$

$$\frac{\partial \mathscr{L}}{\partial \mu_k} = 0 \rightarrow y_k w^T x_k - 1 + e_k = 0, k = 1, \ldots, m$$

can be written as the solution to the following set of linear equations $Z^T Z \mu + \gamma^{-1} \mu = 1$, where $Z = (x_1 y_1, \ldots, x_m y_m)$. Let $\alpha_k = \mu_k y_k$, we have

$$(X^T X + \gamma^{-1} I)\alpha = y. \qquad (1)$$

Once α is solved, the classification hyperplane will be $x^T X \alpha = 0$. Given query point x, we evaluate $\text{sgn}(x^T X \alpha)$ to make classification.

We use our sampling techniques in solving Equation (1) and evaluating $\text{sgn}(x^T X \alpha)$ to avoid time complexity overhead of $\text{poly}(m)$ or $\text{poly}(n)$, which will kill the wanted exponential speedup. Note that the quantum-inspired algorithm for linear equations [19], [20] may inverse a low-rank matrix in logarithmic runtime. However, such algorithm cannot be invoked directly to solve Equation (1) here, since the complexity of

direct computing the matrix $X^T X + \gamma^{-1} I$ is polynomial, which would once again kill the exponential speedup. Thus we need to develop the indirect sampling technique to efficiently perform matrix inversion on $X^T X + \gamma^{-1} I$ with only sampling access of X.

C. The sampling technique

We show the definition and idea of our sampling method to get indices, elements or submatrices, which is the key technique used in our algorithm, as well as in [14], [15], [19].

Definition 1 (Sampling on vectors). *Suppose $v \in \mathbb{C}^n$, define $q^{(v)}$ as a probability distribution that:*

$$x \sim q^{(v)} : \quad \mathbb{P}[x = i] = \frac{|v_i|^2}{\|v\|^2}.$$

Picking an index according to the probability distribution $q^{(v)}$ is called a sampling on v.

Definition 2 (Sampling the indices from matrices). *Suppose $A \in \mathbb{C}^{n \times m}$, define $q^{(A)}$ as a 2-dimensional probability distribution that:*

$$(x, y) \sim q^{(v)} : \quad \mathbb{P}[x = i, y = j] = \frac{|A_{ij}|^2}{\|A\|_F^2}.$$

Picking a pair of indices (i, j) according to the probability distribution $q^{(A)}$ is called a sampling on A.

Definition 3 (Sampling the submatrices from matrices). *Suppose the target is to sample a submatrix $X'' \in \mathbb{C}^{c \times r}$ from $X \in \mathbb{C}^{n \times m}$. First we sample r times on the vector $(\|X_{*,j}\|)_{j=1,...,m}$ and get column indices $j_1, ..., j_r$. The columns $X_{*,j_1}, ..., X_{*,j_r}$ form submatrix X'. Then we sample c times on the j-th column of X and get row indices $i_1, ..., i_c$. In each time the j is sampled uniformly at random from $j_1, ..., j_r$. The rows $X'_{i_1,*}, ..., X'_{i_c}$ form submatrix X''. The matrices X' and X'' are normalized so that $\mathbb{E}[X' X'^T] = X X^T$ and $\mathbb{E}[X''^T X''] = X'^T X'$.*

The process of sampling the submatrices from matrices (as described in Def. 3) is shown in Fig. 1. To put it simple, it is taking several rows and columns out of the matrix by a random choice decided by the "importance" of the elements. Then normalize them so that they are unbiased from the original rows and columns.

To achieve fast sampling, we usually store vectors in an arborescent data structure (such as binary search tree) as suggested in [14] and store matrices by a list of their row trees or column trees. Actually, the sampling is an analog of quantum states measurements. It only reveals a low-dimensional projection of vectors and matrices in each computation. Rather than computing with the whole vector or matrix, we choose the most representative elements of them for calculation with a high probability (we choose the elements according to the probability of their squares, which is also similar to the quantum measurement of quantum states.). The sampling technique we use has the advantage of unbiasedly representing the original vector while consuming less computing resources.

We note that there are other kinds of sampling methods for SVM such as the Markov sampling [27], [28]. Different sampling methods may work well on different scenarios. Our algorithm is designed for low-rank datasets, while the algorithms based on Markov sampling [27], [28] may work well on the datasets that the columns form a uniformly ergodic Markov chain. In our algorithm, to achieve exponential speedup, the sampling technique is different from Markov sampling: (i) We sample both the rows and columns of matrix, rather than only sampling columns. (ii) We sample each elements according to norm-squared probability distribution. (iii) In each dot product computation (Alg. 1), we use sampling technique to avoid operations with high complexity.

D. The preliminary algorithms

We invoke two algorithms employing sampling techniques for saving complexity from [19]. They are treated as oracles that outputs certain outcomes with controlled errors in the main algorithm. Lemma 1 and Lemma 2 shows their correctness and efficiency. For the sake of convenience, some minor changes on the algorithms and lemmas are made.

1) Trace inner product estimation: Alg. 1 achieves computation of trace inner products with logarithmic time on the sizes of the matrices.

Algorithm 1 Trace Inner Product Estimation.

Input: $A \in \mathbb{C}^{m \times n}$ that we have sampling access in complexity $L(A)$ and $B \in \mathbb{C}^{n \times m}$ that we have query access in complexity $Q(B)$. Relative error bound ξ and success probability bound $1 - \eta$.

Goal: Estimate $\text{Tr}[AB]$.

1: Repeat step 2 $\lceil 6 \log_2(\frac{2}{\eta}) \rceil$ times and take the median of Y, noted as Z.

2: Repeat step 3 $\lceil \frac{9}{\xi^2} \rceil$ times and calculate the mean of X, noted as Y.

3: Sample i from row norms of A. Sample j from A_i. Let $X = \frac{\|A\|_F^2}{A_{ij}} B_{ji}$.

Output: Z.

Lemma 1 [19]. *Suppose that we have length-square sampling access to $A \in \mathbb{C}^{m \times n}$ and query access to the matrix $B \in \mathbb{C}^{n \times m}$ in complexity $Q(B)$. Then we can estimate $Tr[AB]$ to precision $\xi \|A\|_F \|B\|_F$ with probability at least $1 - \eta$ in time*

$$O \left(\frac{\log(1/\eta)}{\xi^2} (L(A) + Q(B)) \right).$$

Algorithm 2 Rejection sampling.

Input: $A \in \mathbb{C}^{m \times n}$ that we have length-square sampling access and $b \in \mathbb{C}^n$ that we have norm access and $y = Ab$ that we have query access.

Goal: Sample from length-square distribution of $y = Ab$.

1: Take $D \geq \|b\|^2$.

2: Sample a row index i by row norm squares of A.

3: Query $|y_i|^2 = |A_{i,*}b|^2$ and calculate $\frac{|A_{i,*}b|^2}{D\|A_{i,*}\|^2}$.

4: Sample a real number x uniformly distributed in $[0, 1]$. If $x < \frac{|A_{i,*}b|^2}{D\|A_{i,*}\|^2}$, output i, else, go to step 2.

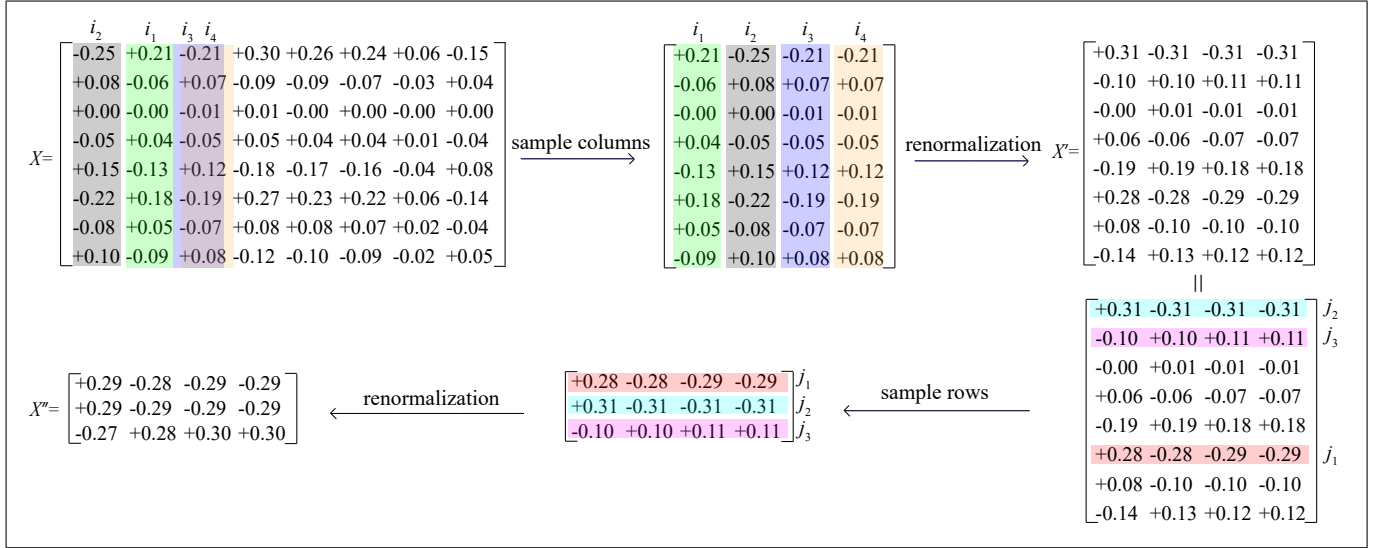**Output:** The row index i.

Fig. 1. A demonstration of sampling submatrices from matrices (The process described in Def. 3, which is also Step 2 and Step 3 in Alg. 3.). We sample columns from X to get X' and sample rows from X' to get X''. Note that X' and X'' are normalized such that $\mathbb{E}[X'X'^T] = XX^T$ and $\mathbb{E}[X''^T X''] = X'^T X'$.

2) Rejection sampling: Alg. 2 achieves sampling of a vector that we do not have full query access in time logarithmic of its length.

Lemma 2 [19]. *Suppose that we have length-square sampling access to $A \in \mathbb{C}^{m \times n}$ having normalized rows, and we are given $b \in \mathbb{C}^n$. Then we can implement queries to the vector $y := Ab \in \mathbb{C}^n$ with complexity $Q(y) = O(nQ(A))$ and we can length-square sample from $q^{(y)}$ with complexity $L(y)$ such that $E[L(y)] = O\left(\frac{n\|b\|^2}{\|y\|^2}(L(A) + nQ(A))\right)$.*

III. QUANTUM-INSPIRED SVM ALGORITHM

We show the main algorithm (Alg. 3) that makes classification as the classical SVMs do. Note that actual computation only happens when we use the expression "calculate" in this algorithm. Otherwise it will lose the exponential-speedup advantage for operations on large vectors or matrices. γ is temporarily taken as ∞. Fig. 2 shows the algorithm process.

Algorithm 3 Quantum-inspired SVM Algorithm.
___

Input: m training data points and their labels $\{(x_j, y_j) : x_j \in \mathbb{R}^n, y_j = \pm 1\}_{j=1,...,m}$, where $y_j = \pm 1$ depending on the class to which x_j belongs. Error bound ϵ and success probability bound $1 - \eta$. γ set as ∞.

Goal 1: Find $\tilde{\alpha}$ that $\|\tilde{\alpha} - \alpha\| \le \epsilon \|\alpha\|$ with success probability at least $1 - \eta$, in which $\alpha = (X^T X)^+ y$.

Goal 2: For any given $x \in \mathbb{R}^n$, find its class.

1: **Init:** Set r, c as described in (6) and (7).

2: **Sample columns:** Sample r column indices $i_1, i_2, ..., i_r$ according to the column norm squares $\frac{\|X_{*,i}\|^2}{\|X\|_F^2}$. Define X' to be the matrix whose s-th column is $\frac{\|X\|_F}{\sqrt{r}} \frac{X_{*,i_s}}{\|X_{*,i_s}\|}$. Define $A' = X'^T X'$.

Algorithm 3 Quantum-inspired SVM Algorithm.
___

3: **Sample rows:** Sample $s \in [r]$ uniformly, then sample a row index j distributed as $\frac{|X'_{js}|^2}{\|X'_{*,s}\|^2}$. Sample a total number of c row indices $j_1, j_2, ..., j_c$ this way. Define X'' whose t-th row is $\frac{\|X\|_F}{\sqrt{c}} \frac{X'_{j_t,*}}{\|X'_{j_t,*}\|}$. Define $A'' = X''^T X''$.

4: **Spectral decomposition:** Calculate the spectral decomposition of A''. Denote here by $A'' = V'' \Sigma^2 V''^T$. Denote the calculated eigenvalues by σ_l^2, $l = 1, ..., k$.

5: **Approximate eigenvectors:** Let $R = X'^T X$. Define $\tilde{V}_l = \frac{R^T V''_l}{\sigma_l^2}$ for $l = 1, ..., k$, $\tilde{V} = (\tilde{V}_l)_{l=1,...,k}$.

6: **Estimate matrix elements:** Calculate $\tilde{\lambda}_l = \tilde{V}_l^T y$ to precision $\frac{3\epsilon \sigma_l^2}{16\sqrt{k}}\|y\|$ by Alg. 1, each with success probability $1 - \frac{\eta}{4k}$. Let $u = \sum_{l=1}^{k} \frac{\tilde{\lambda}_l}{\sigma_l^4} V''_l$.

7: **Find query access:** Find query access of $\tilde{\alpha} = \tilde{R}^T u$ by $\tilde{\alpha}_p = u^T \tilde{R}_{*,p}$, in which $\tilde{R}_{ij}$ is calculated to precision $\frac{\epsilon \kappa^2}{4\|X\|_F}$ by Alg. 1, each with success probability $1 - \frac{\eta}{4\lceil 864/\epsilon^2 \log(8/\eta)\rceil}$.

8: **Find sign:** Calculate $x^T X \tilde{\alpha}$ to precision $\frac{\epsilon}{4}\|\alpha\|\|x\|$ with success probability $1 - \frac{\eta}{4}$ by Alg. 1. Tell its sign.

Output: The answer class depends on the sign. Positive corresponds to 1 while negative to -1.

The following theorem states the accuracy and time complexity of quantum-inspired support vector machine algorithm, from which we conclude the time complexity T depends polylogarithmically on m, n and polynomially on $k, \kappa, \epsilon, \eta$. It is to be proved in section IV and section V.

Theorem 1. *Given parameters $\epsilon > 0, 0 < \eta < 1$, and given the data matrix X with size $m \times n$, rank k, norm 1, and condition number κ, the quantum-inspired SVM algorithm will find the classification expression $x^T X \alpha$ for any vector $x \in \mathbb{C}^n$ with error less than $\epsilon \kappa^2 \sqrt{m}\|x\|$, success probability higher than*
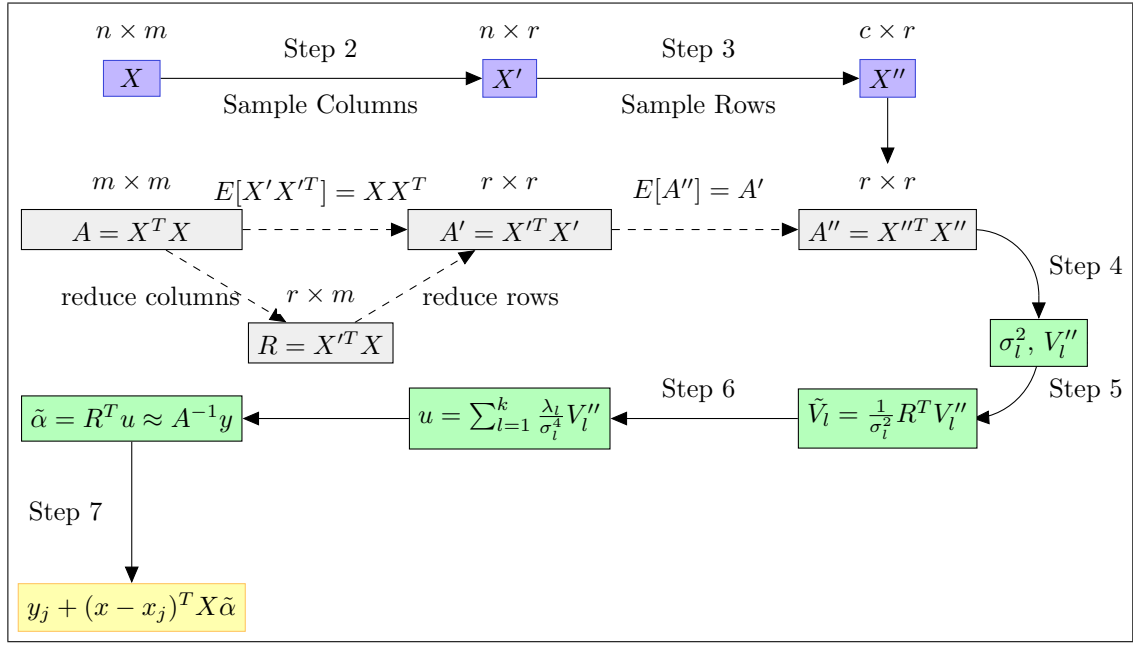
Fig. 2. The quantum-inspired SVM algorithm. In the algorithm, the subsampling of A is implemented by subsampling the matrix X, which is called the indirect sampling technique. After the indirect sampling, we estimate the approximation of the eigenvectors ($\tilde{V}_l$) of A, and finally get the classification expression.

$1 - \eta$ *and time complexity* $T(m, n, k, \kappa, \epsilon, \eta)$.

$$T = O(r \log_2 m + cr \log_2 n + r^3$$
$$+ \frac{\|X\|_F^2 k^2}{\epsilon^2} \log_2(\frac{8k}{\eta})(\log_2(mn) + k)$$
$$+ \frac{1}{\epsilon^2} \log_2 \frac{1}{\eta}(\log_2(mn) + rk \log_2(\frac{2}{\eta_1}) \frac{\|X\|_F^4}{\epsilon_1^2 r} \log_2(mn))),$$

in which

$$\epsilon_1 = \frac{\epsilon \|x\|}{2\sqrt{r} \lceil \frac{36}{\epsilon^2} \rceil \lceil 6 \log_2(\frac{16}{\eta}) \rceil},$$

$$\eta_1 = \frac{\eta}{8r \lceil \frac{36}{\epsilon^2} \rceil \lceil 6 \log_2(\frac{16}{\eta}) \rceil}.$$

In Alg. 3, γ is set as ∞, which makes the coefficient matrix $A = X^T X$. Notice that the eigenvectors of $X^T X + \gamma^{-1} I$ and $X^T X$ are the same, and the difference of their eigenvalues are γ^{-1}. Thus the algorithm can be easily extended to be applied to the coefficient matrix $X^T X + \gamma^{-1} I$ with arbitrary γ, by just simply adding γ^{-1} to the calculated eigenvalues in Step 4.

IV. ACCURACY

We prove that the error of computing the classification expression $x^T X \tilde{\alpha}$ in the quantum-inspired SVM algorithm will not exceed $\epsilon \kappa^2 \sqrt{m} \|x\|$. Although when performing classifications, we are more concerned with the classification rate than with the errors in each computation of classifications expression, the errors can directly affect the classification rate. Therefore, evaluating the error plays an important role in verifying the algorithm and further improving the performance of it. We take $\gamma = \infty$ in the analysis because adding γ^{-1} to the eigenvalues won't cause error and thus the analysis is the same in the case of $\gamma \neq \infty$.

We divide the calculated error into parts. Let $\alpha = (X^T X)^+ y$, $\alpha' = \sum_{l=1}^k \frac{\lambda_l}{\sigma_l^2} \tilde{V}_l = \tilde{V} \Sigma^{-2} \tilde{V}^T y$, in which $\lambda_l = \tilde{V}_l^T y$ and $\alpha'' = \sum_{l=1}^k \frac{\tilde{\lambda}_l}{\sigma_l^2} \tilde{V}_l$. Then the total error of the classification expression is [1]

$$\begin{aligned} E &= \Delta(x^T X \alpha) \\ &\leq \Delta(x^T X \tilde{\alpha}) + |x^T X(\tilde{\alpha} - \alpha)| \\ &\leq \Delta(x^T X \tilde{\alpha}) + \|x\|(\|\tilde{\alpha} - \alpha''\| + \|\alpha'' - \alpha'\| + \|\alpha' - \alpha\|) \\ &= E_1 + E_2 + E_3 + E_4. \end{aligned}$$

Here $E_1 \leq \frac{\epsilon}{4} \|\alpha\| \|x\|$ with probability no less than $1 - \frac{\eta}{4}$ is guaranteed by step 8 of Alg. 3.

$E_2 \leq \frac{\epsilon}{4} \|\alpha\| \|x\|$ with probability no less than $1 - \frac{\eta}{4}$ is guaranteed by step 7 of Alg. 3.

$E_3 \leq \frac{\epsilon}{4} \|\alpha\| \|x\|$ with probability no less than $1 - \frac{\eta}{4}$ is shown in subsection IV-A.

$E_4 \leq \frac{\epsilon}{4} \|\alpha\| \|x\|$ with probability no less than $1 - \frac{\eta}{4}$ is shown in subsection IV-B.

Thus

$$\begin{aligned} E &\leq E_1 + E_2 + E_3 + E_4 \\ &\leq \epsilon \kappa^2 \sqrt{m} \|x\|, \end{aligned}$$

with success probability no less than $1 - \eta$.

For achieving accurate classification, we only need a relative error $\frac{E}{x^T X \alpha}$ less than 1. Thus by lessening ϵ, we can achieve this goal in any given probability range.

---

[1] For any expression f, $\Delta(f)$ represents the difference of the exact value of f and the value calculated by the estimation algorithms Alg. 1 and Alg. 3 (These two algorithms cannot get the exact values because randomness is introduced.).
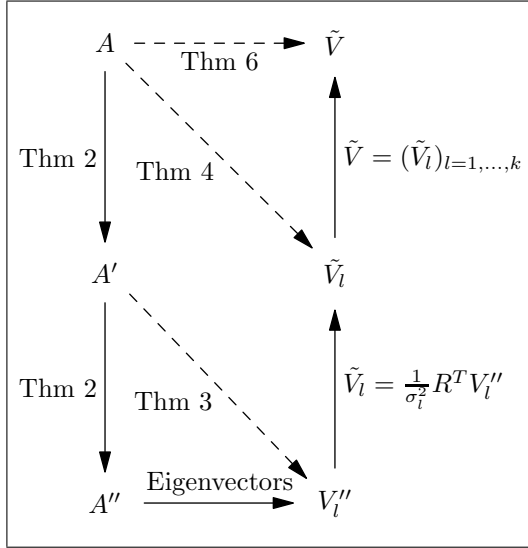
Fig. 3. The whole procedure of proving $\|\tilde{V}\Sigma^{-2}\tilde{V}^T A - I_m\| \le \frac{\epsilon}{2}$.

A. Proof of $E_3 \le \frac{\epsilon}{4}\|\alpha\|\|x\|$

Notice that

$$
\begin{aligned}
E_3 &= \|x\|\|\alpha - \alpha'\| \\
&= \|x\|\|\alpha - \tilde{V}\Sigma^{-2}\tilde{V}^T A\alpha\| \\
&\le \|\alpha\|\|x\|\|\tilde{V}\Sigma^{-2}\tilde{V}^T A - I_m\|.
\end{aligned}
$$

Here we put 5 theorems (from 2 to 6) to prove $\|\tilde{V}\Sigma^{-2}\tilde{V}^T A - I_m\| \le \frac{\epsilon}{4}$, in which theorem 2 and 5 are invoked from [19]. We offer proofs for Theorem 3,4 and 6 in appendix A. The purpose of these theorems is to show that $\tilde{V}\Sigma^{-2}\tilde{V}^T$ is functionally close to the inverse of matrix A, as $\|\tilde{V}\Sigma^{-2}\tilde{V}^T A - I_m\| \le \frac{\epsilon}{4}$ suggests.

Theorem 2 states the norm distance between A, A' and A''. According to the norm distance, and the fact that V_l'' are the eigenvectors of A'', Theorem 3 finds the relation between A' and V_l''. We define $\tilde{V}_l = \frac{1}{\sigma_l^2}R^T V_l''$, and Theorem 6 finally gives the relation between A and $\tilde{V}$. The procedure is shown in Fig. 3.

Theorem 2 [19]. *Let $X' \in \mathbb{C}^{n \times r}$, $X'' \in \mathbb{C}^{c \times r}$ is the sampling outcome of X'. Suppose X'' is normalized that $\mathbb{E}[X''^T X''] = X'^T X'$, then $\forall \epsilon \in [0, \frac{\|X'\|}{\|X'\|_F}]$, we have*

$$
\mathbb{P}\left[\|X'^T X' - X''^T X''\| \ge \epsilon\|X'\|\|X'\|_F\right] \le 2re^{-\frac{\epsilon^2 c}{4}}.
$$

Hence, for $c \ge \frac{4\log_2(\frac{2r}{\eta})}{\epsilon^2}$, with probability at least $1 - \eta$ we have

$$
\|X'^T X' - X''^T X''\| \le \epsilon\|X'\|\|X'\|_F.
$$

When a submatrix X'' is randomly subsampled from X', it is a matrix of multiple random variables. Theorem 2 is the Markov inequality for X''. It points out that the operator norm distance between $X'^T X'$ and $X''^T X''$ is short with a high probability.

Theorem 3. *Suppose V_l'' is a system of orthogonal vectors while*

$$
A'' = \sum_{l=1}^k \sigma_l^2 V_l'' V_l''^T.
$$

Suppose $\|A' - A''\| \le \beta$. Then

$$
|V_i''^T A' V_j'' - \delta_{ij}\sigma_i^2| \le \beta.
$$

Theorem 3 points out that if matrix A' and A'' are close in operator norm sense, A'''s eigenvectors will approximately work as eigenvectors for A' too.

Theorem 4. *Suppose that V_l'' is a system of orthogonal vectors that*

$$
|V_i''^T A' V_j'' - \delta_{ij}\sigma_i^2| \le \beta.
$$

Suppose $\|XX^T - X'X'^T\| \le \epsilon'$, $rank(X') = k$, $\frac{1}{\kappa} \le \sigma_i^2 \le 1$. Let $\tilde{V}_l = \frac{R^T V_l''}{\sigma_i^2}$, then

$$
|\tilde{V}_i^T \tilde{V}_j - \delta_{ij}| \le \kappa^2((k+1)\beta + \epsilon'),
$$

and

$$
\begin{aligned}
|\tilde{V}_i^T A \tilde{V}_j - \delta_{ij}\sigma_i^2| \le\,&2\epsilon'\kappa^6 + (k^2 - 2k + 2)\beta^3\kappa^4 + (3k - 4)\beta^2\kappa^4 \\
&+ 3\beta\kappa^4.
\end{aligned}
$$

in which $A' = X'^T X'$, $A = X^T X$.

Theorem 4 points out that if A'''s eigenvectors approximately work as eigenvectors for A' and $\|XX^T - X'X'^T\| \le \epsilon'$, $\tilde{V}_l^T$ approximately work as eigenvectors for A.

Theorem 5 [19]. *If $rank(B) \le k$, $\tilde{V}$ has k columns that spans the row and column space of B, then*

$$
\|B\| \le \|(\tilde{V}^T \tilde{V})^+\|\|\tilde{V}^T B\tilde{V}\|.
$$

Under the condition that $\tilde{V}_l^T$ approximately work as eigenvectors for A, the following Theorem 6 points out that $\tilde{V}\Sigma^{-2}\tilde{V}^T$ is functionally close to the inverse of matrix A.

Theorem 6. *Suppose that $\tilde{V}_l$ is a system of approximated orthogonal vectors that*

$$
|\tilde{V}_i^T \tilde{V}_j - \delta_{ij}| \le \frac{1}{4k}, \tag{2}
$$

$$
|\tilde{V}_i^T A \tilde{V}_j - \delta_{ij}\sigma_i^2| \le \zeta,
$$

in which $A = X^T X$, $rankX = k$, $\|X\| \le 1$, $\|X^+\| \le \kappa$. Then

$$
\|\tilde{V}\Sigma^{-2}\tilde{V}^T A - I_m\| \le \frac{5}{3}\kappa k\zeta.
$$

To conclude, for $\mathbb{P}[\|\alpha' - \alpha\| > \frac{\epsilon}{4}\|\alpha\|] \le \frac{\eta}{4}$, we need to pick ϵ' and β such that

$$
5k\kappa^5(2\epsilon'\kappa^2 + 4\beta) \le 3\frac{\epsilon}{2}, \tag{3}
$$

$$
4k\kappa^2((k+1)\beta + \epsilon') \le 1, \tag{4}
$$

$$
(k^2 - 2k + 2)\beta^2 + (3k - 4)\beta \le 1, \tag{5}
$$

and decide the sampling parameter as

$$r = \lceil \frac{4 \log_2(\frac{8n}{\eta})}{\epsilon'^2} \rceil, \tag{6}$$

$$c = \lceil \frac{4\kappa^2 \log_2(\frac{8r}{\eta})}{\beta^2} \rceil. \tag{7}$$

B. Proof of $E_4 \le \frac{\epsilon}{4}\|\alpha\|\|x\|$

Notice that

$$E_4 = \|x\|\|\alpha - \tilde{\alpha}\|.$$

For $y = X^T X \alpha$ and $\alpha = X^+ X^{+T} y$, we have $\|y\| \le \|\alpha\| \le \kappa^2 \|y\|$.

For $\|\tilde{\alpha} - \alpha'\|$, let z be the vector that $z_l = \frac{\lambda_l - \tilde{\lambda}_l}{\sigma_l^2}$, we have

$$
\begin{aligned}
\|\tilde{\alpha} - \alpha'\| &= \| \sum_{l=1}^{k} \frac{\lambda_l - \tilde{\lambda}_l}{\sigma_l^2} \tilde{V}_l \| \\
&= \|\tilde{V} z\| \\
&\le \sqrt{\|\tilde{V}^T \tilde{V}\|}\|z\| \\
&\le \frac{4}{3} \frac{3\epsilon\sigma_l^2}{8\sqrt{k}} \|y\| \frac{1}{\sigma_l^2} \sqrt{k} \\
&\le \frac{1}{4}\epsilon\|\alpha\|.
\end{aligned}
$$

in which $\|\tilde{V}^T \tilde{V}\| \le \frac{4}{3}$ as shown in proof of theorem 6.

V. COMPLEXITY

In this section, we will analyze the time complexity of each step in the main algorithm. Note that in the main algorithm the variables R, $\tilde{V}_l$, $\tilde{\alpha}$ are queried instead of calculated. We will include the corresponding query complexity in analysis of the steps where we queried these variables.

A. Sampling of columns and rows

It is easy to find the time complexity of sampling of columns and rows is $O(r \log_2 m + cr \log_2 n)$.

B. The spectral decomposition

For $r \times r$ symmetric matrix A, the fastest classical spectral decomposition is through classical spectral symmetric QR method, of which the complexity is $O(r^3)$.

C. Computation of $\tilde{\lambda}_l$

Inspecting Alg. 1, we can easily find its complexity is

$$O(\frac{1}{\xi^2} \log_2(\frac{2}{\eta})(L(A) + Q(B))).$$

For computation of $\tilde{\lambda}_l$ by Alg. 1, we have

$$\lambda_l = \frac{1}{\sigma_l^2} V_l''^T R y = \frac{1}{\sigma_l^2} \text{Tr}[V_l''^T X'^T X y] = \frac{1}{\sigma_l^2} \text{Tr}[X y V_l''^T X'^T].$$

Observe that $\|y V_l''^T X'^T\|_F = \|y\|\|V_l''^T X'^T\| \le \|y\|$, and we can query the (i, j) matrix element of $y V_l''^T X'^T$ in cost $O(r)$. Thus the complexity in step 6 is

$$T_6 = O(\frac{\|X\|_F^2 k^2}{\epsilon^2} \log_2(\frac{8k}{\eta})(\log_2(mn) + k)).$$

D. Computation of $x^T X \tilde{\alpha}$

Computation of $x^T X \tilde{\alpha}$ is the last step of the algorithm, and also the most important step for saving time complexity. In the Step 8 of Alg. 3, we need to calculate $x^T X \tilde{\alpha}$, which is equal to $\text{Tr}[X \tilde{\alpha} x^T]$, with precision $\epsilon\|\alpha\|\|x\|$ and success probability $1 - \frac{\eta}{4}$ using Alg. 1. Let the A and B in Alg. 1 be X and $\tilde{\alpha} x^T$, respectively. To calculate $\text{Tr}[X \tilde{\alpha} x^T]$, we first establish the query access for $\tilde{\alpha} x^T$ (we already have the sampling access of X), and then using the Alg. 1 as an oracle:

1) Query of R: First we find query access of $R = X'^T X$. For any $s = 1, ..., r$, $j = 1, ..., m$, $R_{sj} = e_s^T X'^T X e_j = \text{Tr}[X e_j e_s^T X'^T]$, we calculate such trace by Alg. 1 to precision ϵ_1 with success probability $1 - \eta_1$. The time complexity for one query will be

$$Q(R) = O(\log_2(\frac{2}{\eta_1}) \frac{\|X\|^4}{\epsilon_1^2 r} \log_2(mn)).$$

2) Query of $\tilde{\alpha}$: For any $i = 1, ..., m$, we have $\tilde{\alpha}_j = \sum_{s=1}^{r} R_{sj} u_s$. One query of $\tilde{\alpha}$ will cost time $r k Q(R)$, with error $\epsilon_1 \sum_{s=1}^{r} |u_s|$ and success probability more than $1 - r\eta_1$.

3) Computation of $x^T X \tilde{\alpha}$: We use Alg. 1 to calculate $x^T X \tilde{\alpha} = \text{Tr}[X \tilde{\alpha} x^T]$ to precision $\frac{\epsilon}{2}\|\alpha\|\|x\|$ with success probability $1 - \frac{\eta}{8}$. Notice the query of $\tilde{\alpha}$ is with error and success probability. We only need

$$\epsilon_1 \sum_{s=1}^{r} |u_s| \lceil \frac{36}{\epsilon^2} \rceil \lceil 6 \log_2(\frac{16}{\eta}) \rceil \le \frac{\epsilon}{2}\|\alpha\|\|x\|,$$

$$r\eta_1 \lceil \frac{36}{\epsilon^2} \rceil \lceil 6 \log_2(\frac{16}{\eta}) \rceil \le \frac{\eta}{8}$$

to fulfill the overall computing task. Notice $\sum_{s=1}^{r} |u_s| \le \sqrt{r}\|u\|$ and $\alpha = R^T u$ We set

$$\epsilon_1 = \frac{\epsilon\|x\|}{2\sqrt{r}\lceil \frac{36}{\epsilon^2} \rceil \lceil 6 \log_2(\frac{16}{\eta}) \rceil},$$

$$\eta_1 = \frac{\eta}{8r\lceil \frac{36}{\epsilon^2} \rceil \lceil 6 \log_2(\frac{16}{\eta}) \rceil}.$$

And the overall time complexity for computing $x^T X \tilde{\alpha}$ is

$$
\begin{aligned}
T_7 &= O(\frac{1}{\epsilon^2} \log_2 \frac{1}{\eta} (\log_2(mn) + rk Q(R))) \\
&= O(\frac{1}{\epsilon^2} \log_2 \frac{1}{\eta} (\log_2(mn) + rk \log_2(\frac{2}{\eta_1}) \frac{\|X\|_F^4}{\epsilon_1^2 r} \log_2(mn))).
\end{aligned}
$$

VI. EXPERIMENTS

In this section, we demonstrate the proposed quantum-inspired SVM algorithm in practice by testing the algorithm on artificial datasets. The feasibility and efficiency of some other quantum-inspired algorithms (quantum-inspired algorithms for recommendation systems and linear systems of equations) on large datasets has been benchmarked, and the results indicate that quantum-inspired algorithms can perform well in practice under its specific condition: low rank, low condition number, and very large dimension of the input matrix [22]. Here we show the feasibility of the quantum-inspired SVM. Firstly, we test the quantum-inspired SVM algorithm on low-rank and

low-rank approximated datasets and compare it to an existing classical SVM implementation. Secondly, we discuss the characteristics of the algorithm by analyzing its dependence on the parameters and datasets. In our experiment, we use the arborescent data structure instead of arrays for storage and sampling [22], making the experiment conducted in a more real scenario compared to the previous work [22]. All algorithms are implemented in Julia [30]. The source code and data are available at https://github.com/helloinrm/qisvm.

A. Experiment I: Comparison with LIBSVM

In this experiment, we test quantum-inspired SVM algorithm on large datasets and compare its performance to the well-known classical SVM implementation LIBSVM [31].

We generate datasets of size 10000×11000, which represent 11000 vectors (6000 vectors for training and 5000 vectors for testing) with length 10000. All the data vectors in training and testing sets are chosen uniformly at random from the generated data matrix, so that they are statistically independent and identically distributed. We test quantum-inspired SVM and LIBSVM on two kinds of datasets: low-rank datasets (rank= 1) and high-rank but low-rank approximated datasets (rank= 10000). Each scenario is repeated for 5 times. The construction method for data matrices is described in Appendix B. And the parameters for quantum-inspired SVM are choosen as $\epsilon = 5, \eta = 0.1$ and $b = 1$ (We explain the parameters and their setting in Experiment II.).

The average classification rates are shown in Table II, from which we observe the advantage of quantum-inspired SVM on such low-rank approximated datasets (on average about 5% higher). We also find that both quantum-inspired SVM and LIBSVM performs better on low-rank datasets than low-rank approximated datasets.

B. Experiment II: Discussion on algorithm parameters

As analyzed in Section IV and Section V, there are two main parameters for the quantum-inspired algorithm: relative error ϵ and success probability $1 - \eta$. Based on them we set subsampling size r, c and run the algorithm. However, for datasets that are not large enough, setting r, c by Equation (6) and Equation (7) is rather time costly. For instance, when the condition number of data matrix is 1.0, taking $\eta = 0.1$ and $\epsilon = 5.0$, theoretically, the r, c for 10000×10000 dataset should be set as 1656 and 259973 to assure that the algorithm calculates the classification expression with relative error less than ϵ and success probability higher than $1 - \eta$. For practical applications of not too large datasets, we set r, c as $r = b\lceil 4\log_2(2n/\eta)/\epsilon^2 \rceil$ and $c = b\lceil 4\log_2(2r/\eta)/\epsilon^2 \rceil$, in which b is the subsampling size control parameter. When $b = 1$, our practical choice of r, c assures the relative error of subsampling (Step 2 and Step 3 in Alg. 3) won't exceed ϵ (guaranteed by Theorem 2).

In Experiment I, we took the practical setting of r, c, where we already found advantage compared to LIBSVM. Our choice of ϵ, η and b is $\epsilon = 5$, $\eta = 0.1$ and $b = 1$. Here, we test the algorithm on other choices of ϵ, η and b and check the classification rate of the algorithm. We test each parameter choice for 50 times. The variation intervals

of each parameter are ϵ from 1 to 10, η from 0.1 to 1, and b from 1 to 10. The results are shown in Fig. 4. We find the average classification rates of the algorithm in each experiment are close. We notice when using the practical r, c, which are much smaller than the theoretical ones, the algorithm maintains its performance (classification rate around 0.90). This phenomenon indicates a gap between our theoretical analysis and the actual performance, as [22] reports "the performance of these algorithms is better than the theoretical complexity bounds would suggest".

VII. DISCUSSION

In this section, we will present some discussions on the proposed algorithm. And we will also discuss the potential applications of our techniques to other types of SVMs, such as non-linear SVM and least square SVM, but more works on the complexity and errors are required in future work if we want to realize these extensions.

A. The cause of exponential speedup

An interesting fact is that we can achieve exponential speedup without using any quantum resources, such as superposition or entanglement. This is a somewhat confusing but reasonable result that can be understood as follows: Firstly, the advantage of quantum algorithms, such as HHL algorithm, is that high-dimensional vectors can be represented using only a few qubits. By replacing qRAM to the arborescent data structure for sampling, we can also represent the low-rank matrices by its normalized submatrix in a short time. By using the technique of sampling, large-size computations are avoided, and we only need to deal with the problem that has the logarithmic size of the original data. Secondly, the relative error of matrix subsampling algorithm is minus-exponential on the matrix size, which ensures the effectiveness of such logarithmic-complexity algorithm (e.g. Theorem 2 shows the error of matrix row subsampling).

B. Improving sampling for dot product

Remember in Alg. 1 we can estimate dot products for two vectors. However, it does not work well for all the conditions, like when $\|x\|$ and $\|y\|$ are donminated by one element. For randomness, [32] implies that we can apply a spherically random rotation R to all x, which does not change the kernel matrix K, but will make all the elements in the dataset matrix be in a same distribution.

C. LS-SVM with non-linear kernels

In Section II, we have considered the LS-SVM with the linear kernel $K = X^T X$. When data sets are not linear separable, non-linear kernels are usually needed. To deal with non-linear kernels with Alg. 3, we only have to show how to establish sampling access for the non-linear kernel matrix K from the sampling access of X.

We first show how the sampling access of polynomial kernel $K_p(x_i, x_j) = (x_j^T x_i)^p$ can be established. The corresponding kernel matrix is $K_p = ((x_j^T x_i)^p)_{i=1,\ldots,m, j=1,\ldots,m}$.

TABLE II
THE AVERAGE VALUES AND STANDARD DEVIATIONS OF CLASSIFICATION RATES (%) OF QISVM AND LIBSVM IN FIVE EXPERIMENTS.

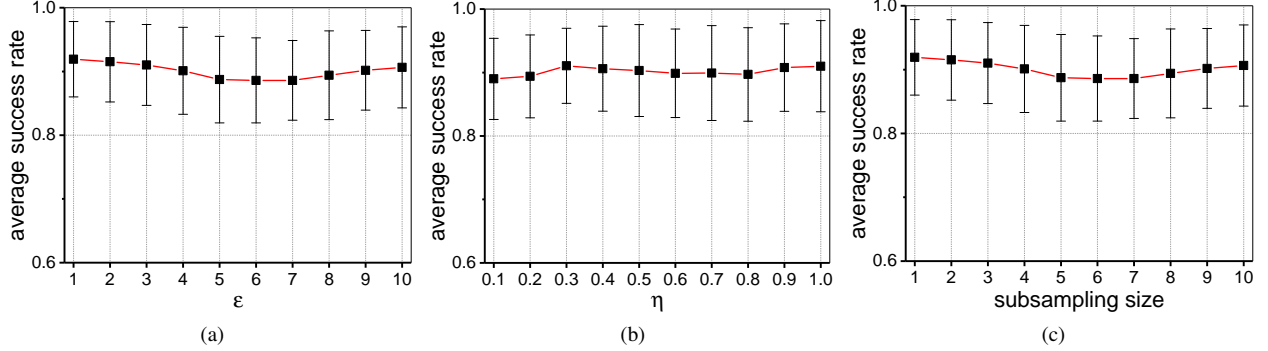|	Testing Set		Training Set	
	qiSVM	LIBSVM	qiSVM	LIBSVM
Low-rank	91.45±3.17	86.46±2.00	91.35±3.64	86.45±2.15
Low-rank approximated	89.82±4.38	84.90±3.20	89.92±4.23	84.69±2.87

Fig. 4. The average classification rate of quantum-inspired SVM algorithm with different parameters on the dataset with rank 1. Each point represents an average classification rate for 50 trials, and the error bar shows the standard deviation of the 50 trials. (a) Algorithm performance when the parameter ϵ is taken from 1 to 10. (b) Algorithm performance when the parameter η is taken from 0.1 to 1. (c) Algorithm performance when the parameter b is taken from 1 to 10.

We take

$$Z = (x_1^{\otimes p}, x_2^{\otimes p}, ..., x_m^{\otimes p}),$$

in which the j-column Z_j is the p-th tensor power of x_j.

Notice that $Z^T Z = K_p$. Once we have sampling access of Z, we can sample K_p as Step 2 and Step 3 in Alg. 3 do. The sampling access of Z can be established by (The effectiveness of Alg. 4 is shown in Appendix C.):

Algorithm 4 Polynomial kernel matrices sampling.

Input: The sampling access of X in logarithmic time of m and n.

Goal: Sample a column index j from the column norm vector $(\|x_1\|^p, \|x_2\|^p, \ldots, \|x_m\|^p)$ of Z, and them sample a row index i from column $x_j^{\otimes p}$ of Z.

1: Sample on column norm vector $(\|x_1\|, \|x_2\|, \ldots, \|x_m\|)$ of X to get index j.

2: Query $\|x_j\|$ from $(\|x_1\|, \|x_2\|, \ldots, \|x_m\|)$. Calculate $\|x_j\|^p$.

3: Sample a real number a uniformly distributed in $[0, 1]$. If $a \geq \|x_j\|^p$, go to Step 1. If not, output index j as the column index and continue.

4: Repeat sampling on x_j for p times. Denote the outcome indices as $i_1, i_2, \ldots, i_p$.

5: Output column index $\sum_{\tau=1}^{p}(i_\tau - 1)n^{p-\tau} + 1$.

For general non-linear kernels, we note that they can always be approximated by linear combination of polynomial kernels (and thus can be sampled based on sampling access of polynomial kernels) the corresponding non-linear feature function is continous. For instance, the popular radial basis function

(RBF) kernel

$$K_{\mathrm{RBF}}(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$$

can be approximated by

$$\tilde{K}_{\mathrm{RBF}}(x_i, x_j) = \sum_{p=0}^{N} \frac{1}{p!} \left(-\frac{x_i^T x_i - 2x_j^T x_i + x_j^T x_j}{2\sigma^2} \right)^p$$

$$= \sum_{p=0}^{N} \left(-\frac{1}{2\sigma^2} \right)^p \sum_{q,l=0}^{p} \binom{p}{q} K_q(x_i, x_i)$$

$$+ (-2)^l \binom{p}{l} K_l(x_i, x_j) + \binom{p}{p-q-l} K_{p-q-l}(x_j, x_j).$$

D. General LS-SVM

In the former sections, we began with a LS-SVM with $b = 0$ and linear kernels in Section II. And we showed how the method can be extended to nonlinear kernels in Section VII-C. Finally, we deal with the last assumption $b = 0$. We show how a general LS-SVM can be tackled using techniques alike in Alg. 3:

A general LS-SVM equation [24] is

$$\begin{pmatrix} 0 & 1^T \\ 1 & K + \gamma^{-1}I \end{pmatrix} \begin{pmatrix} b \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ y \end{pmatrix}, \quad (8)$$

in which K is the kernel matrix.

Equation (8) can be solved as follows:

(i) Firstly, by methods in Section VII-C, we establish the sampling access for kernel matrix K. Suppose a sampling outcome of K is K''.

(ii) Secondly, take

$$A = \begin{pmatrix} 0 & 1^T \\ 1 & K + \gamma^{-1}I \end{pmatrix}.$$

and

$$A'' = \begin{pmatrix} 0 & 1^T \\ 1 & K'' + \gamma^{-1}I \end{pmatrix}.$$

We establish the eigen relations between A and A'' by theorems which are similar to Theorem 2 and Theorem 4.

(iii) Once $A \in \mathbb{R}^{m \times m}$ is subsampled to $A'' \in \mathbb{R}^{r \times r}$, we can continue Step 3-7 of Alg. 3.

(iv) Once Equation (8) is solved in Step 7 of Alg. 3, which means we can establish the query access for α. According to Equation 8, $b = y_j - x_j^T X \alpha - \gamma^{-1}\alpha_j$ for any j such that $\alpha_j \neq 0$. We can then evaluate the classification expression $y_j + (x - x_j)^T X \alpha - \gamma^{-1}\alpha_j$ and make classification using Alg. 1. There are two ways to find j: One is executing the rejection sampling on α using Alg. 2. The other is checking if $\alpha_j = 0$ after each sampling of X in Step 3 of Alg. 1.

VIII. CONCLUSION

We have proposed a quantum-inspired SVM algorithm that achieves exponential speedup over the previous classical algorithms. The feasibility of the proposed algorithm is demonstrated by experiments. Our algorithm works well on low-rank datasets or datasets that can be well approximated by low-rank matrices, which is similar with quantum SVM algorithm [29] as "when a low-rank approximation is appropriate". Certain investigations on the application of such an algorithm are required to make quantum-inspired SVM operable in solving questions like face recognition [23] and signal processing [33].

We hope that the techniques developed in our work can lead to the emergence of more efficient classical algorithms, such as applying our method to support vector machines with more complex kernels [24], [34] or other machine learning algorithms. The technique of indirect sampling can expand the application area of fast sampling techniques. And it will make contribution to the further competition between classical algorithms and quantum ones.

Some improvements on our work would be made in the future, such as reducing the conditions on the data matrix, further reducing the complexity, and tighten the error bounds in the theoretical analysis, which can be achieved through a deeper investigation on the algorithm and the error propagation process. The investigation of quantum-inspired non-linear SVMs and least squares SVM as discussed in Section VII also requires theoretical analysis and empirical evaluations.

We note that our work, as well as the previous quantum-inspired algorithms, are not intended to demonstrate that quantum computing is uncompetitive. We want to find out where the boundaries of classical and quantum computing are, and we expect new quantum algorithms to be developed to beat our algorithm.

APPENDIX A
PROOF OF THEOREMS IN IV

A. Proof of Theorem 3

Proof:

$$|V_i''^T A'V_j'' - \delta_{ij}\sigma_i^2| \leq |V_i''^T (A' - A'')V_j''| + |V_i''^T A''V_j'' - \delta_{ij}\sigma_i^2|$$
$$\leq \|V_i''^T\| \cdot \|(A' - A'')V_j''\|$$
$$\leq \beta.$$

∎

B. Proof of Theorem 4

Proof:

$$\Delta_1 = |\tilde{V}_i^T \tilde{V}_j - \delta_{ij}|$$
$$= |\frac{V_i''^T RR^T V_j'' - \delta_{ij}\sigma_i^4}{\sigma_i^2 \sigma_j^2}|$$
$$\leq \frac{1}{\sigma_i^2 \sigma_j^2} \left(|V_i''^T A'A'V_j'' - \delta_{ij}\sigma_i^4| + |V_i''^T (RR^T - A'A')V_j''| \right)$$
$$\leq (\sigma_i^2 + \sigma_j^2 + k - 1)\beta + \|X'^T (XX^T - X'X'^T)X'\|$$
$$\leq \frac{1}{\sigma_i^2 \sigma_j^2}((\sigma_i^2 + \sigma_j^2 + k - 1)\beta + \epsilon')$$
$$\leq \kappa^2((k + 1)\beta + \epsilon').$$

$$\Delta_2 = |\tilde{V}_i^T A\tilde{V}_j - \delta_{ij}\sigma_i^2|$$
$$= \frac{1}{\sigma_i^2 \sigma_j^2}|V_i''^T RAR^T V_j'' - \delta_{ij}\sigma_i^6|$$
$$\leq \frac{1}{\sigma_i^2 \sigma_j^2}(|V_i''^T (RAR^T - A'A'A')^T V_j''|$$
$$+ |V_i''^T A'A'A'^T V_j'' - \delta_{ij}\sigma_i^6|)$$
$$\leq \frac{1}{\sigma_i^2 \sigma_j^2}(\kappa^2\|XX^T XX^T - X'X'^T X'X'^T\|$$
$$+ |V_i''^T A'(\sum_{l_1=1}^{k} V_{l_1}'' V_{l_1}''^T)A'(\sum_{l_2=1}^{k} V_{l_2}'' V_{l_2}''^T)A'V_j'' - \delta_{ij}\sigma_i^6|).$$

For

$$\|XX^T XX^T - X'X'^T X'X'^T\| \leq \|XX^T (XX^T - X'X'^T)\|$$
$$+ \|(XX^T - X'X'^T)X'X'^T\|$$
$$\leq 2\epsilon'.$$

and

$$\Delta_3 = |V_i''^T A'(\sum_{l_1=1}^{k} V_{l_1}'' V_{l_1}''^T)A'(\sum_{l_2=1}^{k} V_{l_2}'' V_{l_2}''^T)A'V_j'' - \delta_{ij}\sigma_i^6|$$
$$\leq (k - 1)^2\beta^3 + \sigma_i^2 \sigma_j^2\beta + \sigma_i^2((k - 1)\beta^2 + \sigma_i^2\beta)$$
$$+ \sigma_j^2((k - 1)\beta^2 + \sigma_j^2\beta) + \beta^2((k^2 - 3k + 3)\beta + k - 2)$$
$$\leq (k^2 - 2k + 2)\beta^3 + (3k - 4)\beta^2 + 3\beta.$$

Thus

$$\Delta_4 = |\tilde{V}_i^T A\tilde{V}_j - \delta_{ij}\sigma_i^2|$$
$$\leq \kappa^4(2\epsilon'\kappa^2 + (k^2 - 2k + 2)\beta^3 + (3k - 4)\beta^2 + 3\beta).$$

∎

C. Proof of Theorem 6

Proof:

For $|\tilde{V}_i^T \tilde{V}_j - \delta_{ij}| \le \frac{1}{4k}$, we assume $|\tilde{V}_i^T \tilde{V}_j - \delta_{ij}| \le \gamma_1 \le \frac{1}{4k}$. By (2) we have $\|\tilde{V}^T \tilde{V} - I\| \le k\gamma_1 \le \frac{1}{4}$, thus $\|(\tilde{V}^T \tilde{V})^+\| \le \frac{4}{3}$.

$$\begin{aligned}
|\tilde{V}_i^T B \tilde{V}_j| = & |\sum_{l=1}^{k} \frac{\tilde{V}_i^T \tilde{V}_l \cdot \tilde{V}_l^T A \tilde{V}_j}{\sigma_l^2} - \tilde{V}_i^T \tilde{V}_j| \\
\le & |\sum_{l=1}^{k} \frac{\tilde{V}_i^T \tilde{V}_l}{\sigma_l^2}(\tilde{V}_l^T A \tilde{V}_j - \delta_{lj}\sigma_l^2)| \\
& + |\sum_{l=1}^{k} \tilde{V}_i^T \tilde{V}_l \delta_{lj} - \tilde{V}_i^T \tilde{V}_j| \\
\le & \zeta((k-1)\gamma_1\kappa + (\gamma_1+1)\kappa) \\
\le & \frac{5}{4}\zeta\kappa.
\end{aligned}$$

Let $B = \tilde{V}\Sigma^{-2}\tilde{V}^T A - I_m$ then by Theorem 5

$$\|B\| \le \frac{5}{3}\kappa k \zeta.$$

$\blacksquare$

APPENDIX B
THE CONSTRUCTION METHOD OF DATASETS

In our experiment, we constructed artificial datasets which are low-rank or can be low-rank approximated. Here we put up our construction mehtod:

1. Firstly, we multiply a random matrix A of size $n \times k$ with another random matrix B of size $k \times m$. The elements in both of them are evenly distributed in $[-0.5, 0.5]$. Denote the multiplication outcome as X. Then the rank of X is at most k.

2. We add turbulence to the matrix X by adding a random number evenly distributed in $[-0.1x, 0.1x]$ to all the elements in X, in which x is the average of all the absolute values of X. After adding turbulence, X is no more low-rank but still low-rank approximated.

3. We normalize X such that X has operator norm 1.

4. We divide the column vectors of X into two classes by a random hyperplane $w^T x = 0$ that passes the origin (By random hyperplane we mean the elements in w are uniformly sampled from $[0, 1]$ at random.), while making sure that both classes are not empty.

5. Since now we have m linear-separable labeled vectors, each with length n. We choose uniformly at random m_1 of them for training, and let the other $m_2 = m - m_1$ for testing, while making sure that the training set includes vectors of both classes.

APPENDIX C
THE EFFECTIVENESS OF ALG. 4

The goal of Alg. 4 is to sample a column index and a row index from Z. We show it achieves this goal.

Step 1-3 are for sampling out the column index. They are essentially Alg. 2 with $A = \text{Diag}(\|x_1\|^{p-1}, \dots, \|x_m\|^{p-1})$ and $b = (\|x_1\|, \dots, \|x_m\|)$, which sample from the column norm vector $b = (\|x_1\|^p, \dots, \|x_m\|^p)$ of Z to get the column index j.

We note that in practical applications, Step 1-3 can be adjusted for speedup, such as frugal rejection sampling suggested in [35].

Step 4-5 are for sampling out the row index. Suppose $l = \sum_{\tau=1}^{p}(i_\tau - 1)n^{p-\tau} + 1$. According the definition of tensor power, the l-th element of $x_j^{\otimes p}$ is

$$(x_j^{\otimes p})_l = \Pi_{\tau=1}^{p} x_{i_\tau j}.$$

When Step 4 executes p times of sampling on x_j, the probability of getting the outcome $i_1, i_2, \dots, i_p$ is $|\Pi_{\tau=1}^{p} x_{i_\tau j}|^2$, which is exactly the probability of sampling out $(x_j^{\otimes p})_l$ in $x_j^{\otimes p}$. Thus in Step 5 we output index $l = \sum_{\tau=1}^{p}(i_\tau - 1)n^{p-\tau} + 1$.

REFERENCES

[1] H.-L. Huang, D. Wu, D. Fan, and X. Zhu, "Superconducting quantum computing: a review," *Science China Information Sciences*, vol. 63, no. 8, pp. 1–32, 2020.
[2] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annual Symposium Foundations Computer Sci.* Santa Fe, NM, USA: IEEE, Nov. 1994, pp. 124–134. [Online]. Available: https://ieeexplore.ieee.org/document/365700
[3] C.-Y. Lu, D. E. Browne, T. Yang, and J.-W. Pan, "Demonstration of a compiled version of shor's quantum factoring algorithm using photonic qubits," *Physical Review Letters*, vol. 99, no. 25, p. 250504, 2007. [Online]. Available: https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.99.250504
[4] H.-L. Huang, Q. Zhao, X. Ma, C. Liu, Z.-E. Su, X.-L. Wang, L. Li, N.-L. Liu, B. C. Sanders, C.-Y. Lu *et al.*, "Experimental blind quantum computing for a classical client," *Physical review letters*, vol. 119, no. 5, p. 050503, 2017. [Online]. Available: https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.119.050503
[5] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 21th Annual ACM Symposium Theory Computing*. Philadelphia, Pennsylvania, USA: ACM, May 1996, pp. 212–219. [Online]. Available: http://doi.acm.org/10.1145/237814.237866
[6] T. Li, W.-S. Bao, H.-L. Huang, F.-G. Li, X.-Q. Fu, S. Zhang, C. Guo, Y.-T. Du, X. Wang, and J. Lin, "Complementary-multiphase quantum search for all numbers of target items," *Physical Review A*, vol. 98, no. 6, p. 062308, 2018. [Online]. Available: https://journals.aps.org/pra/abstract/10.1103/PhysRevA.98.062308
[7] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, p. 195–202, Sept. 2017. [Online]. Available: https://doi.org/10.1038/nature23474
[8] H.-L. Huang, X.-L. Wang, P. P. Rohde, Y.-H. Luo, Y.-W. Zhao, C. Liu, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan, "Demonstration of topological data analysis on a quantum processor," *Optica*, vol. 5, no. 2, pp. 193–198, 2018. [Online]. Available: https://www.osapublishing.org/optica/abstract.cfm?uri=optica-5-2-193
[9] J. Liu, K. H. Lim, K. L. Wood, W. Huang, C. Guo, and H.-L. Huang, "Hybrid quantum-classical convolutional neural networks," *arXiv preprint*, 2019. [Online]. Available: https://arxiv.org/abs/1911.02998
[10] H.-L. Huang, Y.-W. Zhao, T. Li, F.-G. Li, Y.-T. Du, X.-Q. Fu, S. Zhang, X. Wang, and W.-S. Bao, "Homomorphic encryption experiments on ibm's cloud quantum computing platform," *Frontiers of Physics*, vol. 12, no. 1, p. 120305, 2017. [Online]. Available: https://link.springer.com/article/10.1007/s11467-016-0643-9
[11] H.-L. Huang, Y. Du, M. Gong, Y. Zhao, Y. Wu, C. Wang, S. Li, F. Liang, J. Lin, Y. Xu *et al.*, "Experimental quantum generative adversarial networks for image generation," *arXiv:2010.06201*, 2020.
[12] D. R. Simon, "On the power of quantum computation," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1474–1483, July 1997. [Online]. Available: https://doi.org/10.1137/S0097539796298637

[13] I. Kerenidis and A. Prakash, "Quantum recommendation systems," in *8th Innovations Theoretical Computer Sci. Conf.*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 67, Berkeley, CA, USA, Jan. 2017, pp. 49:1–49:21. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2017/8154

[14] E. Tang, "A quantum-inspired classical algorithm for recommendation systems," in *Proc. 51st Annual ACM SIGACT Symposium Theory Computing*, vol. 25. New York, NY, USA: ACM, June 2019, pp. 217–228. [Online]. Available: https://doi.org/10.1145/3313276.3316310

[15] A. Frieze, R. Kannan, and S. Vempala, "Fast monte-carlo algorithms for finding low-rank approximations," *J. Assoc. Comput. Mach.*, vol. 51, no. 6, pp. 1025–1041, Nov. 2004. [Online]. Available: http://doi.acm.org/10.1145/1039488.1039494

[16] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nat. Phys.*, vol. 10, no. 9, p. 631–633, July 2014. [Online]. Available: https://doi.org/10.1038/nphys3029

[17] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," *arXiv preprint*, Nov. 2013. [Online]. Available: https://arxiv.org/abs/1307.0411

[18] E. Tang, "Quantum-inspired classical algorithms for principal component analysis and supervised clustering," *arXiv preprint*, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1811.00414

[19] A. Gilyén, S. Lloyd, and E. Tang, "Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension," *arXiv preprint*, Nov. 2018. [Online]. Available: http://arxiv.org/abs/1811.04909

[20] N.-H. Chia, H.-H. Lin, and C. Wang, "Quantum-inspired sublinear classical algorithms for solving low-rank linear systems," *arXiv preprint*, Nov. 2018. [Online]. Available: https://arxiv.org/abs/1811.04852

[21] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, no. 15, p. 150502, Oct. 2009.

[22] J. M. Arrazola, A. Delgado, B. R. Bardhan, and S. Lloyd, "Quantum-inspired algorithms in practice," *arXiv preprint*, May 2019. [Online]. Available: http://arxiv.org/abs/1905.10415

[23] P. J. Phillips, "Support vector machines applied to face recognition," in *Advances Neural Inform. Processing Systems*, vol. 48, no. 6241, Gaithersburg, MD, USA, Nov. 1999, pp. 803–809. [Online]. Available: https://doi.org/10.6028/nist.ir.6241

[24] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, June 1999. [Online]. Available: https://doi.org/10.1023/A:1018628609742

[25] J. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," Apr. 1998. [Online]. Available: https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vecto

[26] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel Support Vector Machines: The Cascade SVM," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. MIT Press, 2005, pp. 521–528. [Online]. Available: http://papers.nips.cc/paper/2608-parallel-support-vector-machines-the-cascade-svm.pdf

[27] J. Xu, Y. Y. Tang, B. Zou, Z. Xu, L. Li, Y. Lu, and B. Zhang, "The generalization ability of svm classification based on markov sampling," *IEEE transactions on cybernetics*, vol. 45, no. 6, pp. 1169–1179, 2014. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6881630

[28] B. Zou, C. Xu, Y. Lu, Y. Y. Tang, J. Xu, and X. You, "k-times markov sampling for svmc," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 4, pp. 1328–1341, 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7993056/

[29] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Phys. Rev. Lett.*, vol. 113, p.

[32] D. Achlioptas, F. McSherry, and B. Schölkopf, "Sampling techniques for kernel methods," in *Advances Neural Inform. Processing Systems*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Vancouver, British Columbia, Canada: MIT Press, Dec.

130503, Sept. 2014. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.113.130503

[30] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017. [Online]. Available: https://doi.org/10.1137/141000671

[31] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

2002, pp. 335–342. [Online]. Available: https://papers.nips.cc/paper/2072-sampling-techniques-for-kernel-methods

[33] L. Wang, *Support Vector Machines for Signal Processing*, 1st ed. The Netherlands: Springer, Berlin, Heidelberg, 2005, ch. 15, pp. 321–342. [Online]. Available: https://doi.org/10.1007/b95439

[34] L. Wang, *Multiple Model Estimation for Nonlinear Classification*, 1st ed. The Netherlands: Springer, Berlin, Heidelberg, 2005, ch. 2, pp. 49–76. [Online]. Available: https://doi.org/10.1007/b95439

[35] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, "Quantum Supremacy Is Both Closer and Farther than It Appears," *arXiv preprint*, Sep. 2018. [Online]. Available: http://arxiv.org/abs/1807.10749

Chen Ding received the B.S. degree from University of Science and Technology of China, Hefei, China, in 2019.

He is currently a graduate student in CAS Centre for Excellence and Synergetic Innovation Centre in Quantum Information and Quantum Physics. His current research interests include quantum machine learning, quantum-inspired algorithm designing and variational quantum computing.

Tian-Yi Bao received the B.S. degree from University of Michigan, Ann Arbor, USA, in 2020.

She is currently a graduate student in Oxford University. Her current research interests include the machine learning and human-computer interaction.

He-Liang Huang received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2018.

He is currently an Assistant Professor of Henan Key Laboratory of Quantum Information and Cryptography, Zhengzhou, China, and the Postdoctoral Fellow of University of Science and Technology of China, Hefei, China. He has authored or co-authored over 30 papers in refereed international journals and co-authored 1 book. His current research interests include secure cloud quantum computing, big data quantum computing, and the physical implementation of quantum computing architectures, in particular using linear optical and superconducting systems.