

An Improved Quantum Algorithm for Ridge Regression

Chao-Hua Yu¹, Fei Gao², and Qiao-Yan Wen¹

Abstract—Ridge regression (RR) is an important machine learning technique which introduces a regularization hyperparameter α to ordinary multiple linear regression for analyzing data suffering from multicollinearity. In this paper, we present a quantum algorithm for RR, where the technique of parallel Hamiltonian simulation to simulate a number of Hermitian matrices in parallel is proposed and used to develop a quantum version of K -fold cross-validation approach, which can efficiently estimate the predictive performance of RR. Our algorithm consists of two phases: (1) using quantum K -fold cross-validation to efficiently determine a good α with which RR can achieve good predictive performance, and then (2) generating a quantum state encoding the optimal fitting parameters of RR with such α , which can be further utilized to predict new data. Since indefinite dense Hamiltonian simulation has been adopted as a key subroutine, our algorithm can efficiently handle non-sparse data matrices. It is shown that our algorithm can achieve exponential speedup over the classical counterpart for (low-rank) data matrices with low condition numbers. But when the condition numbers of data matrices are large to be amenable to full or approximately full ranks of data matrices, only polynomial speedup can be achieved.

Index Terms—Quantum algorithm, ridge regression, regularization hyperparameter, parallel Hamiltonian simulation, quantum K -fold cross validation

1 INTRODUCTION

DURING from the 80's of last century, quantum computing has been shown to be more computationally powerful in solving certain problems than classical computing [1], [2], [3], [4]. In the past decade, it has been brought into the field of machine learning, which is a sub-field of computer science and studies how to learn from data and make predictions on new data [5], giving birth to a new disciplinary research field—quantum machine learning. Since its inception, quantum machine learning has become a booming research field attracting worldwide attentions, and a number of efficient quantum algorithms have been proposed for various machine learning tasks [6], [7], [8], [9].

Linear regression (LR) is one of the most important machine learning tasks with wide applications in many scientific fields including biology, behavioristic, sociology, finance, and so on [5]. Given N data points $(\mathbf{x}_i, y_i)_{i=1}^N$, where

$\mathbf{x}_i = (x_{i1}, \dots, x_{iM})^T \in \mathbb{R}^M$ is a vector of M independent (exploratory, input) variables and $y_i \in \mathbb{R}$ is the scalar dependent (response, output) variable, LR assumes that \mathbf{x}_i and y_i are linearly correlated and attempts to construct a linear function $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ characterized by fitting parameters $\mathbf{w} = (w_1, \dots, w_M)^T$ that can best fit such relationship, i.e., making every $f(\mathbf{x}_i)$ as close as possible to y_i . It should be emphasized that \mathbf{x} can be generated by a nonlinear map on some original data, such as polynomial function, which enables LR to fit nonlinear function.

The simplest LR model is *ordinary linear regression* (OLR), where the optimal fitting parameters $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ are determined via least squares method of minimizing the sum of squared residuals. Here $\mathbf{y} = (y_1, \dots, y_N)^T$, and $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^T$ is called *design matrix*. However, OLR in practice is often far from satisfaction [5], [10], [11] when suffering multicollinearity of independent variables of data points (which makes $\mathbf{X}^T \mathbf{X}$ not invertible) or overfitting. These two difficulties substantially restrict the effectiveness of OLR when putting it into real-world applications. To circumvent them, Hoerl et al. [11] put forward a generalized version of OLR—*ridge regression* (RR), in which some regularization of \mathbf{w} is introduced into optimization. This leads to the optimal fitting parameters of RR being $\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \alpha \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$, where α denotes regularization hyperparameter and \mathbf{I} is the identity matrix. However, choosing an appropriate α with which RR can achieve the best (or approximately best) predictive performance is of great challenge.

As of now, a series of quantum algorithms for LR have been proposed. By building on the well-known quantum algorithm for solving linear systems of equations proposed by Harrow, Hassidim, and Lloyd (HHL) [12], Wiebe et al.

- C.-H. Yu is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China, with the State Key Laboratory of Cryptology, P. O. Box 5159, Beijing 100878, China, with the School of Information Technology, Jiangxi University of Finance and Economics, Nanchang 330032, China, and also with the School of Physics, University of Western Australia, Perth 6009, Australia. E-mail: quantum.ych@gmail.com.
- F. Gao is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the Center for Quantum Computing, Peng Cheng Laboratory, Shenzhen 518055, China. E-mail: gaof@bupt.edu.cn.
- Q.-Y. Wen is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: wqy@bupt.edu.cn.

Manuscript received 30 Apr. 2018; revised 13 Feb. 2019; accepted 13 Aug. 2019. Date of publication 26 Aug. 2019; date of current version 3 Feb. 2021. (Corresponding authors: Fei Gao and Qiao-Yan Wen.)

Recommended for acceptance by B. Moseley.

Digital Object Identifier no. 10.1109/TKDE.2019.2937491

[13] first provided a quantum algorithm that can efficiently determine the fitting quality of OLR over an exponentially large data set with a sparse design matrix. Their results were later improved and directly extended to tackle RR [14]. Lately, different from the previous algorithms [12], [14] which are efficient only for the data sets with sparse design matrices, Schuld et al. provided a quantum algorithm for prediction by OLR that can efficiently process low-rank non-sparse design matrices [15]. More recently, Wang suggested a quantum linear regression algorithm that works in the standard oracle model and can efficiently output the optimal fitting parameters in the classical form [16]. However, with the exception of prior work for quantum RR [14], almost all of these quantum linear regression algorithms are based on OLR rather than RR, thus cannot combat multicollinearity and overfitting mentioned above.

In this paper, to deeply explore how and to what extent RR can be done by quantum computing faster than by classical computing, we design a more comprehensive quantum algorithm for RR. Inspired by the technique of K -fold cross-validation [10] which has been widely used to evaluate the predictive performance of many machine learning algorithms [5], [28], we propose its quantum version to efficiently evaluate the predictive performance of RR. Our quantum algorithm will use the quantum K -fold cross-validation to determine a good α for RR, and then generate a quantum state encoding the fitting parameters of RR with such α in the amplitudes. It is shown that our algorithm is exponentially faster than the classical counterpart, when processing (low-rank) design matrices with relatively small elements and low condition numbers, but when design matrices have large condition numbers amenable to full or approximately full ranks of data matrices, only polynomial speedup can be achieved. Our algorithm improves the existing quantum algorithm for RR, i.e., LZ's algorithm [14], from two aspects. First, since our algorithm uses indefinite dense Hamiltonian simulation [22] as the key subroutine, our algorithm has no dependence on the sparsity of design matrices, but has slightly worse dependence on the error, whereas LZ's uses sparse Hamiltonian simulation as the subroutine and can only efficiently tackle sparse design matrices. Second, our algorithm presents an efficient procedure, i.e., quantum K -fold cross-validation, to determine a good α for RR, while LZ's algorithm does not address this important task.

Just as other HHL-based quantum machine learning algorithms with several caveats [17], our algorithm also faces similar caveats. First, our algorithm assumes that efficient quantum access to the entries of \mathbf{X} and \mathbf{y} is provided. This can be achieved by quantum random access memory (QRAM) [18], for which there is no general implementation in quantum hardware to date. However, if the entries can be efficiently computed by simple and explicit formula, the quantum access can be efficiently implemented directly without QRAM. Second, our algorithm does not output the classical form of the optimal fitting parameters \mathbf{w} , but a quantum state $|\mathbf{w}\rangle$ encoding \mathbf{w} in its amplitudes. Nevertheless, the state can be further used to efficiently predict new data via swap test [19], [20]. Finally, our algorithm is exponentially fast when the condition number of the design matrix \mathbf{X} is relatively low. The condition number may be reduced by preconditioning \mathbf{X} .

2 REVIEW OF RIDGE REGRESSION

Given a set of N data points $(\mathbf{x}_i, y_i)_{i=1}^N$ as described above, RR aims at finding a linear function $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} = \sum_{j=1}^M x_j w_j$ characterized by the fitting parameters $\mathbf{w} = (w_1, \dots, w_M)^T$ that makes all $f(\mathbf{x}_i)$ as close as possible to y_i [5], [10], [11]. Different from OLR where the sum of squared residuals is minimized, RR minimizes the sum of squared residuals plus a fraction of regularization of \mathbf{w} and has the optimal fitting parameters

$$\begin{aligned} \mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^N |f(\mathbf{x}_i) - y_i|^2 + \alpha \|\mathbf{w}\|^2 \\ &= (\mathbf{X}^T \mathbf{X} + \alpha \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}, \end{aligned} \quad (1)$$

where $\|\mathbf{v}\|$ is the 2-norm of any vector \mathbf{v} . Evidently, OLR is a special case of RR with $\alpha = 0$. Write \mathbf{X} in the reduced singular value decomposition [21] form $\mathbf{X} = \sum_{j=1}^R \lambda_j |\mathbf{u}_j\rangle \langle \mathbf{v}_j|$, where R is the rank of \mathbf{X} , λ_j are the nonzero singular values, and $|\mathbf{u}_j\rangle$ ($|\mathbf{v}_j\rangle$) are the corresponding left (right) normalized singular vectors. Adding another $N - R$ normalized vectors $|\mathbf{u}_{R+1}\rangle, \dots, |\mathbf{u}_N\rangle$ that make $|\mathbf{u}_1\rangle, \dots, |\mathbf{u}_N\rangle$ become an orthonormal basis spanning the whole space \mathbb{R}^N , $\mathbf{y}/\|\mathbf{y}\|$ can be written as a linear combination of $\{|\mathbf{u}_j\rangle\}_1^N$, $\mathbf{y}/\|\mathbf{y}\| = \sum_{j=1}^N \beta_j |\mathbf{u}_j\rangle$ with $\sum_{j=1}^N \beta_j^2 = 1$, and thus \mathbf{w} can be rephrased as

$$\mathbf{w} = \sum_{j=1}^R \frac{\lambda_j}{\lambda_j^2 + \alpha} \beta_j \|\mathbf{y}\| |\mathbf{v}_j\rangle, \quad (2)$$

which depends on the choice of α . After attaining \mathbf{w} , one can predict the output \tilde{y} of any new input $\tilde{\mathbf{x}}$ via computing $\tilde{y} = \mathbf{w}^T \tilde{\mathbf{x}}$. So the predictive squared error sum for all the training data points is

$$\begin{aligned} &\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 \\ &= \|\mathbf{y}\|^2 \left(\sum_{j=1}^R \left(1 - \frac{\lambda_j^2}{\lambda_j^2 + \alpha} \right)^2 \beta_j^2 + \sum_{j=R+1}^N \beta_j^2 \right) \\ &\geq \|\mathbf{y}\|^2 \left(1 - \Lambda(2 - \Lambda) \left(\sum_{j=1}^R \beta_j^2 \right) \right), \end{aligned}$$

since $\sum_{j=R+1}^N \beta_j^2 = 1 - (\sum_{j=1}^R \beta_j^2)$, where $\Lambda = \max_{j=1, \dots, R} \frac{\lambda_j^2}{\lambda_j^2 + \alpha}$ and $0 < \Lambda < 1$. If $\sum_{j=1}^R \beta_j^2$ is small, the error sum would be very large, meaning that the model is badly constructed; otherwise, the error sum is small. As a result, when the RR model is well constructed, the support of $\mathbf{y}/\|\mathbf{y}\|$ in the space spanned by $\{|\mathbf{u}_j\rangle\}_{j=1}^R$, i.e., $\sum_{j=1}^R \beta_j^2$, should be large to be close to 1.

Therefore, it is of great importance to choose a good α so that RR with such α can achieve good predictive performance, and then to obtain the \mathbf{w} of RR with such α .

3 QUANTUM ALGORITHM

In the following, we design a quantum algorithm for RR. It consists of two subroutines: a quantum algorithm for generating the quantum state encoding the optimal fitting parameters \mathbf{w} (Eqs. (1) and (2)), and a quantum algorithm for

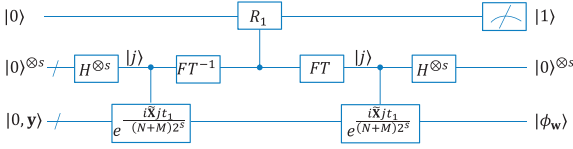


Fig. 1. Quantum circuit for Algorithm 1. Here, the ‘/’ denotes a bundle of wires, H denotes the Hadamard operation, FT represents the quantum Fourier transformation and FT^{-1} is its inverse [1], and s is the number of qubits for estimating the eigenvalues of $\frac{\tilde{\mathbf{X}}}{N+M}$ in step (2), controlled- R_1 denotes controlled rotation in step (3).

finding a good α . Throughout the algorithm, we assume we are provided the quantum oracles

$$O_X : |j\rangle|k\rangle|0\rangle \mapsto |j\rangle|k\rangle|x_{jk}\rangle$$

and

$$O_Y : |j\rangle|0\rangle \mapsto |j\rangle|y_j\rangle,$$

which can efficiently access the entries of \mathbf{X} and \mathbf{y} in time $O(\text{polylog}(MN))$ and $O(\text{polylog}(N))$, respectively. This holds when the entries of \mathbf{X} and \mathbf{y} are efficiently computable or are stored in QRAM [18]. In general, \mathbf{X} is not too much skewed, and $\|\mathbf{X}\|_{\max}$ and $\|\mathbf{y}\|_{\max}$ are not too large, hence we assume $M = \Theta(N)$ and $\|\mathbf{X}\|_{\max}, \|\mathbf{y}\|_{\max} = \Theta(1)$ hereafter.

3.1 Algorithm 1: Generating a Quantum State Encoding the Optimal Fitting Parameters

We first give a quantum algorithm to generate a quantum state $|\mathbf{w}\rangle$ that approximates the normalized \mathbf{w} within error ϵ . From Eq. (2), it is easy to see that, to obtain \mathbf{w} , we need perform singular value decomposition on \mathbf{X} . To achieve this, the recently invented technique of indefinite non-sparse Hamiltonian simulation [22] is adopted. Given a Hermitian matrix $A \in \mathbb{C}^{N \times N}$ and efficient quantum access to its entries, by embedding A into a larger one-sparse Hermitian matrix, it is able to simulate the unitary matrix e^{-iAt} for time t within error ϵ in time $O(\text{polylog}(N)t^2 \|A\|_{\max}^2/\epsilon)$, where $\|A\|_{\max} := \max_{i,j} |A_{ij}|$. However, in our problem, since \mathbf{X} is generally not Hermitian, we extend it to a larger but Hermitian matrix

$$\tilde{\mathbf{X}} = \begin{bmatrix} 0 & \mathbf{X} \\ \mathbf{X}^T & 0 \end{bmatrix} \in \mathbb{R}^{(N+M) \times (N+M)}, \quad (3)$$

which is of $2R$ nonzero eigenvalues $\{\pm\lambda_j\}_{j=1}^R$ and corresponding normalized eigenvectors $\{|\mathbf{u}_j, \pm\mathbf{v}_j\rangle := (|0, \mathbf{u}_j\rangle \pm |1, \mathbf{v}_j\rangle)/\sqrt{2} \in \mathbb{R}^{N+M}\}_{j=1}^R$, where

$$|0, \mathbf{u}_j\rangle = \begin{bmatrix} |\mathbf{u}_j\rangle \\ \mathbf{0} \end{bmatrix}, |1, \mathbf{v}_j\rangle = \begin{bmatrix} \mathbf{0} \\ |\mathbf{v}_j\rangle \end{bmatrix}. \quad (4)$$

Without loss of generality, we assume $\frac{\lambda_j}{N+M} \in [1/\kappa, 1]$, where κ is the condition number of \mathbf{X} . In addition, from Eq. (2), it is easy to see that too small α will make RR reduced to OLR and too large α will make the optimal fitting parameters approach zero, thus we choose α satisfying $\Theta(\frac{(N+M)^2}{\kappa^2}) \leq \alpha \leq \Theta((N+M)^2)$.

The first algorithm proceeds as following steps and the schematic is given in Fig. 1:

- (1) Prepare the $(N+M)$ -dimensional quantum state $|0, \mathbf{y}\rangle = (|\mathbf{y}\rangle^T, \mathbf{0})^T = \sum_{j=1}^N \beta_j |0, \mathbf{u}_j\rangle$ by directly expanding the state $|\mathbf{y}\rangle := \mathbf{y}/\|\mathbf{y}\|$.

Here we assume $|\mathbf{y}\rangle$ can be generated efficiently in time $O(\text{polylog}(N))$. As shown in Appendix A, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TKDE.2019.2937491>, when \mathbf{y} is *balanced* [16] in the sense that

$$\frac{\sum_{j=1}^N |y_j|^2}{N\|\mathbf{y}\|_{\max}^2} = \Omega(1),$$

$|\mathbf{y}\rangle$ can be efficiently generated in time $O(\text{polylog}(N))$ via O_Y . Alternatively, $|\mathbf{y}\rangle$ can also be efficiently prepared when for any i_1, i_2 , $\sum_{i=i_1}^{i_2} |y_i|^2$ are efficiently computable [23].

- (2) Add another register in the state $|0 \cdots 0\rangle$ to the above state $|0, \mathbf{y}\rangle$, and perform phase estimation by simulating $e^{-i\tilde{\mathbf{X}}t_1/(N+M)}$ [22] for some evolution time t_1 to reveal the eigenvalues and eigenvectors of $\frac{\tilde{\mathbf{X}}}{N+M}$ as

$$\sum_{j=1}^R \beta_j |\mathbf{u}_j, \pm\mathbf{v}_j\rangle \left| \frac{\pm\lambda_j}{N+M} \right\rangle / \sqrt{2}. \quad (5)$$

Here for convenience we assume $|\mathbf{y}\rangle$ fully lies in the subspace $\{|\mathbf{u}_j\rangle\}_{j=1}^R$, namely $|\mathbf{y}\rangle = \sum_{j=1}^R \beta_j |\mathbf{u}_j\rangle$ with $\sum_{j=1}^R \beta_j^2 = 1$ and thus $|0, \mathbf{y}\rangle = \sum_{j=1}^R \beta_j |0, \mathbf{u}_j\rangle = \sum_{j=1}^R \beta_j |\mathbf{u}_j, \pm\mathbf{v}_j\rangle / \sqrt{2}$. If $|\mathbf{y}\rangle$, more generally, does not fully lie in the subspace $\{|\mathbf{u}_j\rangle\}_{j=1}^R$, the state of Eq. (5) would be

$$\sum_{j=1}^R \beta_j |\mathbf{u}_j, \pm\mathbf{v}_j\rangle \left| \frac{\pm\lambda_j}{N+M} \right\rangle / \sqrt{2} + \sum_{j=R+1}^N \beta_j |0, \mathbf{u}_j\rangle |0 \cdots 0\rangle. \quad (6)$$

Nonetheless, we can efficiently transform the state of Eq. (6) to the state of Eq. (5): introduce another qubit $|0\rangle$ and rotate it to $|1\rangle$ when the eigenvalue stored in the second register (eigenvalue register) is nonzero, and then measure the qubit to see the outcome $|1\rangle$. The measurement probability is $\sum_{j=1}^R \beta_j^2 \approx 1$ as discussed in Section 2, so the transformation is quite efficient. After successful measurement, we obtain the state of Eq. (5) by letting $\beta_j \leftarrow \beta_j / \sqrt{\sum_{j=1}^R \beta_j^2}$ for $j = 1, \dots, R$.

- (3) Add one qubit and rotate it from $|0\rangle$ to $\sqrt{1 - C_1^2 h^2(\pm\lambda_j, \alpha)} |0\rangle + C_1 h(\pm\lambda_j, \alpha) |1\rangle$ controlled on $|\frac{\pm\lambda_j}{N+M}\rangle$, where $h(\lambda, \alpha) := \frac{(N+M)\lambda}{\lambda^2 + \alpha}$ and $C_1 = O(\max_{\lambda_j} h(\lambda_j, \alpha)^{-1}) = O(1/\kappa)$. As shown in Appendix B, available in the online supplemental material, the maximum of $h(\lambda_j, \alpha)$ as well as C_1 depends on the actual choice of α , but $C_1 h(\lambda_j, \alpha) = \Omega(1/\kappa)$ for all possible α . Then we undo phase estimation and obtain

$$\sum_{j=1}^R \beta_j |\mathbf{u}_j, \pm \mathbf{v}_j\rangle \left(\sqrt{1 - C_1^2 h^2(\pm \lambda_j, \alpha)} |0\rangle + C_1 h(\pm \lambda_j, \alpha) |1\rangle \right). \quad (7)$$

- (4) Measure the last qubit to get $|1\rangle$ and project the first register onto the \mathbf{v}_j part. The final state of the first register approximates

$$|\phi_{\mathbf{w}}\rangle := \frac{\sum_{j=1}^R C_1 \beta_j h(\lambda_j, \alpha) |\mathbf{v}_j\rangle}{\sqrt{\sum_{j=1}^R C_1^2 \beta_j^2 h^2(\lambda_j, \alpha)}} \propto \mathbf{w}, \quad (8)$$

which is proportional to Eqs. (1) and (2) as desired. The success probability of getting $|1\rangle$ is $\sum_{j=1}^R C_1^2 \beta_j^2 h^2(\lambda_j, \alpha) = \Omega(1/\kappa^2)$, which implies that $O(\kappa^2)$ repetitions are enough to yield the desirable state with a large probability, and this can be improved by amplitude amplification [24] with $O(\kappa)$ repetitions.

Given the quantum form $|\tilde{\mathbf{x}}\rangle$ of a new input data $\tilde{\mathbf{x}}$, the state $|\phi_{\mathbf{w}}\rangle$ can be used to predict the output $\tilde{\mathbf{y}} = \mathbf{w}^T \tilde{\mathbf{x}}$ up to some factor by evaluating the inner product of $|\tilde{\mathbf{x}}\rangle$ and $|\phi_{\mathbf{w}}\rangle$ via swap test [19], [20].

3.2 Time Complexity of Algorithm 1

The time complexity of Algorithm 1 is dominated by phase estimation and amplitude amplification. In step (2), the eigenvalues $\pm \frac{\lambda_j}{N+M} \in \pm[1/\kappa, 1]$ are estimated within error $O(1/t_1)$ via phase estimation. Consequently, the relative error of estimating $h(\pm \lambda_j, \alpha)$ scales as $O(\kappa/t_1)$ no matter how α is chosen, but its actual scale depends on α as detailed in Appendix B, available in the online supplemental material. Therefore, $t_1 = O(\kappa/\epsilon)$ is taken to ensure the final state approximates $|\phi_{\mathbf{w}}\rangle$ within error ϵ . Thus, according to [22], phase estimation takes time $O(\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa^2/\epsilon^3)$. Considering amplitude amplification of $O(\kappa)$ repetitions in step (4), it takes total time $O(\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa^3/\epsilon^3)$ to generate $|\phi_{\mathbf{w}}\rangle$. Since $\text{Tr}(\mathbf{X}^T \mathbf{X}) = \sum_{j=1}^R \lambda_j^2 = \sum_{ij} x_{ij}^2 \leq NM \|\mathbf{X}\|_{\max}^2$ and $\lambda_j \in [\frac{N+M}{\kappa}, N+M]$, we have $\frac{R}{\kappa^2} \leq \frac{NM \|\mathbf{X}\|_{\max}^2}{(N+M)^2}$, and thus the rank R of \mathbf{X} is upper bounded as $R = O(\kappa^2)$ due to $M = \Theta(N)$ and $\|\mathbf{X}\|_{\max} = \Theta(1)$.

The best known classical RR algorithm has time complexity $O(NM + N^2 R \log(\frac{R}{\epsilon})/\epsilon^2)$ [25]. Under the assumptions that $M = \Theta(N)$ and $\|\mathbf{X}\|_{\max} = \Theta(1)$, and letting $1/\epsilon = O(\text{polylog}(N))$, our Algorithm 1 takes time $O(\text{polylog}(N)\kappa^3)$, while the classical algorithm takes time $\tilde{O}(\text{polylog}(N)N^2 R)$, where \tilde{O} is used to suppress the relatively small quantity of $\log(R/\epsilon)$. When κ is large with $\kappa = O(\sqrt{N})$ which is amenable to full or approximately full rank of \mathbf{X} (i.e., $R = O(N)$), Algorithm 1 achieves (approximately) quadratic speedup over the classical algorithm, because in this case Algorithm 1 has time complexity $O(\text{polylog}(N)N^{3/2})$, whereas the classical algorithm has time complexity $\tilde{O}(\text{polylog}(N)N^3)$. However, when κ is small with $\kappa = \text{polylog}(N)$ which implies low rank of \mathbf{X} ($R = \text{polylog}(N)$), Algorithm 1 has time complexity $O(\text{polylog}(N))$, while the classical algorithm has time complexity $\tilde{O}(\text{polylog}(N)N^2)$, so our Algorithm 1 in this case is exponentially faster than the classical algorithm.

Compared with LZ's quantum RR algorithm [14] whose time complexity is $O(\log(N+M)s^2\kappa_R^3/\epsilon^2)$, where s is the sparsity of design matrix and $\kappa_R = \frac{\max\{1, \frac{\sqrt{\alpha}}{N+M}\}}{\min\{1/\kappa, \frac{\sqrt{\alpha}}{N+M}\}} = O(\kappa)$ (Note that the singular values of design matrix in our algorithm are assumed to be in $[(N+M)/\kappa, N+M]$, while those in LZ's algorithm are assumed to be in $[1/\kappa, 1]$), the time complexity of our Algorithm 1 has the same dependence on κ as LZ's result, whereas the dependence on ϵ is worse by a factor ϵ^{-1} . However, our Algorithm 1 has no dependence on s , which makes it capable to efficiently handle non-sparse design matrices, and is exponentially faster than LZ's algorithm for non-sparse design matrices with $s = O(N)$ when $\kappa, 1/\epsilon = O(\text{polylog}(N))$.

Just as LZ's algorithm, our Algorithm 1 can also output the scale parameter $\|\mathbf{w}\|^2$ which tells how \mathbf{w} is rescaled to $|\mathbf{w}\rangle$. From Eq. (2), we can see that

$$\begin{aligned} \|\mathbf{w}\|^2 &= \sum_{j=1}^R \left(\frac{\lambda_j}{\lambda_j^2 + \alpha} \right)^2 \beta_j^2 \|\mathbf{y}\|^2 \\ &= \sum_{j=1}^R \frac{h^2(\lambda_j, \alpha) \beta_j^2 \|\mathbf{y}\|^2}{(N+M)^2} \\ &= \frac{P \|\mathbf{y}\|^2}{C_1^2 (N+M)^2}, \end{aligned} \quad (9)$$

where $P = \sum_{j=1}^R C_1^2 \beta_j^2 h^2(\lambda_j, \alpha) = \Omega(1/\kappa^2)$ represents the measurement probability in step (4) of Algorithm 1. Just as estimating $\|\mathbf{y}\|^2$ shown in Appendix A, available in the online supplemental material, P (as well as $\|\mathbf{w}\|^2$) is estimated via amplitude estimation [24] within relative error ϵ with $O(\sqrt{\frac{1-P}{P}} \frac{1}{\epsilon}) = O(\frac{\kappa}{\epsilon})$ repetitions of steps (1)-(3). Since the runtime of each repetition is dominated by that of phase estimation as discussed above, i.e., $O(\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa^2/\epsilon^3)$, $\|\mathbf{w}\|^2$ can be estimated within relative error ϵ in time $O(\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa^3/\epsilon^4)$. Considering $M = \Theta(N)$ and $\|\mathbf{X}\|_{\max} = \Theta(1)$, this procedure would be efficient when $\kappa, 1/\epsilon = \text{polylog}(N)$.

3.3 Algorithm 2: Choosing a Good α

Choosing a good value of α which allows the prediction of future data is a critical part of RR. A common and efficient method for choosing a good α is to choose the best one out of a number of candidate α 's, so that RR with such α has the best predictive performance [11]. The most common method for evaluating the predictive performance of RR as well as other linear regression tasks is K -fold cross-validation [11]. Let us outline how to combine these two methods to determine the best α . First, the set of N data points is divided into K ($2 \leq K \leq N$) subsets and the l th ($l = 1, \dots, K$) subset contains the data points (\mathbf{x}_j, y_j) with $j \in S_l$, where

$$S_l := \{(l-1)N/K + 1, \dots, lN/K\} \quad (10)$$

is used to mark the numbers of data points assigned to the l th subset. Then K turns of training-test procedures are run, where in the l th turn the l th subset is taken as the test set and the others are taken as the training set. After that, the squared residual sum over all data points are calculated to

evaluate the predictive performance of this model for a certain α . The α over all the candidates corresponding to the best predictive performance is chosen as the final α . The details are shown as follows.

Let $\mathbf{X}_l \in \mathbb{R}^{N/K \times M}$ be the matrix containing the rows S_l of \mathbf{X} which corresponds to the l th subset, and $\mathbf{X}_{-l} \in \mathbb{R}^{N \times M}$ be the matrix \mathbf{X} but replacing the elements in the rows of S_l with zeros. Evidently, the rank of \mathbf{X}_{-l} is equal or less than that of \mathbf{X} . \mathbf{X}_{-l} can be written in the singular value decomposition $\mathbf{X}_{-l} = \sum_{j=1}^{R_l} \lambda_{lj} |\mathbf{u}_{lj}\rangle \langle \mathbf{v}_{lj}|$, where λ_{lj} are its singular values, $|\mathbf{u}_{lj}\rangle$ ($|\mathbf{v}_{lj}\rangle$) are their corresponding left (right) singular vectors, R_l is its rank and $\leq R$ obviously. All λ_{lj} lie in $(\frac{N+M}{\kappa'}, N+M)$, and $\kappa' = O(\kappa)$ by taking $K = \Omega(\frac{NM\|\mathbf{X}\|_{\max}^2 \kappa^2}{(N+M)^2})$, a good example of choosing such K being leave-one-out cross-validation; see Appendix C, available in the online supplemental material, for more details on the scale of λ_{lj} . Similarly, we define \mathbf{y}_l and \mathbf{y}_{-l} .

In the l th turn, according to the Eq. (2), the optimal fitting parameters are

$$\mathbf{w}_l = (\mathbf{X}_{-l}^T \mathbf{X}_{-l} + \alpha \mathbf{I})^{-1} \mathbf{X}_{-l}^T \mathbf{y}_{-l}. \quad (11)$$

Consequently, the squared residual sum of prediction of l th turn is $\|\mathbf{y}_l - \mathbf{X}_l \mathbf{w}_l\|^2$ and the predictive performance of RR with a certain α is quantified as the total sum over K turns,

$$E(\alpha) = \sum_{l=1}^K \|\mathbf{y}_l - \mathbf{X}_l \mathbf{w}_l\|^2 \quad (12)$$

$$= \sum_{l=1}^K (\|\mathbf{y}_l\|^2 + \|\mathbf{X}_l \mathbf{w}_l\|^2 - 2\mathbf{y}_l^T \mathbf{X}_l \mathbf{w}_l) \\ = E_1(\alpha) + E_2(\alpha) + E_3(\alpha). \quad (13)$$

Given a set of candidate α 's, $\{\alpha_1, \dots, \alpha_L\}$, our objective is to choose $\hat{\alpha}$ such that

$$\hat{\alpha} = \underset{\alpha \in \{\alpha_1, \dots, \alpha_L\}}{\operatorname{argmin}} E(\alpha). \quad (14)$$

Normally, we uniformly take these L candidate values of α in the prespecified range $[\alpha_{\min} = \Theta(\frac{(N+M)^2}{\kappa^2}), \alpha_{\max} = \Theta((N+M)^2)]$, e.g., $[\frac{(N+M)^2}{10\kappa^2}, \frac{(N+M)^2}{2}]$. That is to say, $\alpha_j = \alpha_{\min} + \frac{(j-1)(\alpha_{\max} - \alpha_{\min})}{L-1}$, for $j = 1, \dots, L$.

In the following, we present an efficient quantum algorithm to choose $\hat{\alpha}$. Taking full advantage of quantum parallelism, our quantum algorithm can efficiently estimate $E(\alpha)$ for a given candidate α . Since the algorithm is inspired by above K -fold cross-validation, we name it *quantum K -fold cross-validation*.

Given a certain α , the first term of $E(\alpha)$ (Eq. (12)), $E_1(\alpha) = \sum_{l=1}^K \|\mathbf{y}_l\|^2 = \|\mathbf{y}\|^2$ can be estimated easily as shown in Appendix A, available in the online supplemental material. From Eq. (12), we can see that, to estimate the second and third terms, i.e., $E_2(\alpha)$ and $E_3(\alpha)$, \mathbf{w}_l for $l = 1, \dots, K$ need to be revealed. Moreover, every data point in the l th subset is assigned to the same \mathbf{w}_l . Therefore, we intend to generate the quantum state approximating

$$|\psi_{\mathbf{w}}\rangle = \frac{\sum_{l=1}^K \left(\sum_{\tau \in S_l} |\tau\rangle \right) \otimes \mathbf{w}_l}{\sqrt{\sum_{l=1}^K N \|\mathbf{w}_l\|^2 / K}} \quad (15)$$

which encodes \mathbf{w}_l in parallel, within error ϵ .

The details of the second algorithm are described in the following steps.

- (1) Prepare the initial quantum state

$$|\psi_0\rangle = \frac{\sum_{l=1}^K \left(\sum_{\tau \in S_l} |\tau\rangle \right) \otimes \|\mathbf{y}_{-l}\| |0, \mathbf{y}_{-l}\rangle}{\sqrt{\sum_{l=1}^K N \|\mathbf{y}_{-l}\|^2 / K}}, \quad (16)$$

which can be efficiently generated in time $O(\text{polylog} N)$ as shown in Appendix A, available in the online supplemental material.

- (2) Perform phase estimation on the above state by simulating the unitary operation

$$\sum_{l=1}^K \left(\sum_{\tau \in S_l} |\tau\rangle \langle \tau| \right) \otimes e^{-\frac{i \tilde{\mathbf{X}}_{-l} t_2}{N+M}} \quad (17)$$

for some evolution time t_2 to reveal the eigenvalues of $\frac{\tilde{\mathbf{X}}_{-l}}{N+M}$ in parallel, where

$$\tilde{\mathbf{X}}_{-l} = \begin{bmatrix} 0 & \mathbf{X}_{-l} \\ \mathbf{X}_{-l}^T & 0 \end{bmatrix} \in \mathbb{R}^{(N+M) \times (N+M)}, \quad (18)$$

which has eigenvalues $\{\pm \lambda_{lj}\}_{j=1}^{R_l}$ and corresponding eigenvectors $\{|\mathbf{u}_{lj}, \pm \mathbf{v}_{lj}\rangle\}_{j=1}^{R_l}$. Similar to the state (5), the resultant state becomes

$$\frac{\sum_{l=1}^K \left(\sum_{\tau \in S_l} |\tau\rangle \right) \left(\sum_j \|\mathbf{y}_{-l}\| \beta_{lj} |\mathbf{u}_{lj}, \pm \mathbf{v}_{lj}\rangle \left| \frac{\pm \lambda_{lj}}{N+M} \right\rangle \right)}{\sqrt{\sum_{l=1}^K 2N \|\mathbf{y}_{-l}\|^2 / K}}, \quad (19)$$

where $\beta_{lj} := \langle 0, \mathbf{u}_{lj} | 0, \mathbf{y}_{-l} \rangle$.

- (3) Similar to step (3) of Algorithm 1, an auxiliary qubit is added and rotated from $|0\rangle$ to $\sqrt{1 - C_2^2 h^2(\pm \lambda_{lj}, \alpha)} |0\rangle + C_2 h(\pm \lambda_{lj}, \alpha) |1\rangle$. Here $C_2 = O(\max_{\lambda} h(\lambda, \alpha))^{-1} = O(1/\kappa')$ with $\frac{\lambda}{N+M} \in [\frac{1}{\kappa'}, 1]$.
- (4) Undo phase estimation and measure the auxiliary qubit to see the outcome $|1\rangle$ with probability

$$P_{\mathbf{w}} = \frac{\sum_{l=1}^K \sum_j C_2^2 \beta_{lj}^2 h^2(\lambda_{lj}, \alpha) \|\mathbf{y}_{-l}\|^2}{\sum_{l=1}^K \|\mathbf{y}_{-l}\|^2} \\ = \frac{\sum_{l=1}^K C_2^2 (N+M)^2 \|\mathbf{w}_l\|^2}{(K-1) \|\mathbf{y}\|^2}, \quad (20)$$

which scales as $\Omega(1/\kappa'^2 \kappa^2)$ as shown in Appendix D, available in the online supplemental material. To reduce the complexity, amplitude amplification is applied with $O(\kappa' \kappa)$ repetitions. Then we get the desired state $|\psi_{\mathbf{w}}\rangle$ of (15).

- (5) Append two additional registers $|0 \dots 0\rangle |0\rangle$ to the state $|\psi_{\mathbf{w}}\rangle$ which can be rewritten as

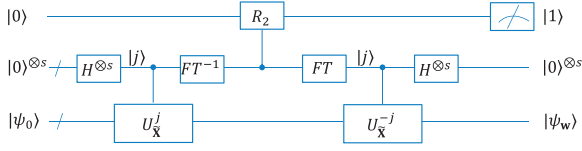


Fig. 2. Quantum circuit for steps (1)-(4) of Algorithm 2. Here, $U_X = \sum_{l=1}^K \left(\sum_{\tau \in S_l} |\tau\rangle\langle\tau| \right) \otimes e^{-\frac{i\mathbf{X}_l \cdot \mathbf{t}_2}{(N+M)^{2s}}}$.

$$|\psi_w\rangle = \frac{\sum_{l=1}^K \left(\sum_{\tau \in S_l} |\tau\rangle \right) \otimes \left(\sum_{k=1}^M \mathbf{w}_{lk} |k\rangle \right)}{\sqrt{\sum_{l=1}^K N \|\mathbf{w}_l\|^2 / K}}, \quad (21)$$

where \mathbf{w}_{lk} is the k th entry of \mathbf{w}_l . Then implement the following procedures.

First, perform O_X to implement

$$\sum_{\tau \in S_l} \mathbf{w}_{lk} |\tau\rangle |k\rangle |0 \cdots 0\rangle |0\rangle \mapsto \sum_{\tau \in S_l} \mathbf{w}_{lk} |\tau\rangle |k\rangle |x_{\tau k}\rangle |0\rangle.$$

Second, perform a controlled rotation denoted by controlled- R_X to generate

$$\sum_{\tau \in S_l} \mathbf{w}_{lk} |\tau\rangle |k\rangle |x_{\tau k}\rangle \left(\frac{x_{\tau k}}{\|\mathbf{X}\|_{\max}} |1\rangle + \sqrt{1 - \frac{x_{\tau k}^2}{\|\mathbf{X}\|_{\max}^2}} |0\rangle \right).$$

Third, perform the inverse of O_X and the state becomes

$$\sum_{\tau \in S_l} \mathbf{w}_{lk} |\tau\rangle |k\rangle \left(\sqrt{1 - \frac{x_{\tau k}^2}{\|\mathbf{X}\|_{\max}^2}} |0\rangle + \frac{x_{\tau k}}{\|\mathbf{X}\|_{\max}} |1\rangle \right).$$

Finally, perform the projective measurement on the last two registers to see if they are in the state $\left(\sum_{k=1}^M |k\rangle \right) / \sqrt{M} |1\rangle$, and if success we get the state (of the first register)

$$\begin{aligned} |\hat{\mathbf{y}}\rangle &= \frac{\sum_{l=1}^K \sum_{\tau \in S_l} \left(\sum_{k=1}^M \mathbf{w}_{lk} x_{\tau k} \right) |\tau\rangle}{\sqrt{\sum_{l=1}^K \sum_{\tau \in S_l} \left(\sum_{k=1}^M \mathbf{w}_{lk} x_{\tau k} \right)^2}} \\ &= \frac{\sum_{l=1}^K \sum_{\tau \in S_l} \mathbf{w}_l^T \mathbf{x}_\tau |\tau\rangle}{\sqrt{\sum_{l=1}^K \sum_{\tau \in S_l} (\mathbf{w}_l^T \mathbf{x}_\tau)^2}} \end{aligned} \quad (22)$$

encoding the prediction of \mathbf{y} . The success probability is

$$P_1 = \frac{\sum_{l=1}^K \sum_{\tau \in S_l} (\mathbf{w}_l^T \mathbf{x}_\tau)^2}{M \|\mathbf{X}\|_{\max}^2 \left(\sum_{l=1}^K N \|\mathbf{w}_l\|^2 / K \right)}, \quad (23)$$

which, as shown in Appendix F, available in the online supplemental material, scales as $\Omega(1/k^2)$ when RR achieves good predictive performance. This implies that the second term of $E(\alpha)$ (Eq. (12)) can be estimated as

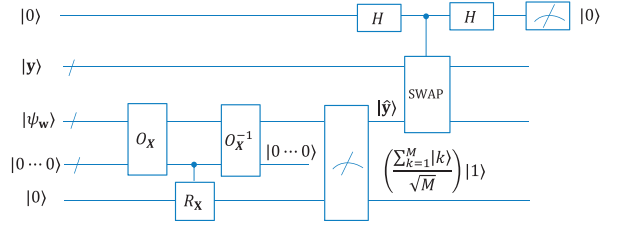


Fig. 3. Quantum circuit for steps (5)-(6) of Algorithm 2. Here, SWAP denotes the SWAP operation.

$$\begin{aligned} E_2(\alpha) &= \sum_{l=1}^K \|\mathbf{X}_l \mathbf{w}_l\|^2 \\ &= \sum_{l=1}^K \sum_{\tau \in S_l} (\mathbf{w}_l^T \mathbf{x}_\tau)^2 \\ &= \frac{P_1 P_w N M (K-1) \|\mathbf{X}\|_{\max}^2 \|\mathbf{y}\|^2}{C_2^2 (N+M)^2 K}. \end{aligned} \quad (24)$$

Note that $N, M, K, \|\mathbf{X}\|_{\max}$ and C_2 are known, and $\|\mathbf{y}\|^2$ can be estimated as shown in Appendix A, available in the online supplemental material.

(6) Perform a swap test [19], [20] on the states $|\mathbf{y}\rangle$ and $|\hat{\mathbf{y}}\rangle$, with the success probability of getting $|0\rangle$ being

$$\begin{aligned} P_2 &= \frac{1}{2} + \frac{1}{2} |\langle \mathbf{y} | \hat{\mathbf{y}} \rangle|^2 \\ &= \frac{1}{2} + \frac{1}{2} \frac{(\sum_{l=1}^K \sum_{\tau \in S_l} y_\tau \mathbf{w}_l^T \mathbf{x}_\tau)^2}{\|\mathbf{y}\|^2 \left(\sum_{l=1}^K \sum_{\tau \in S_l} (\mathbf{w}_l^T \mathbf{x}_\tau)^2 \right)}. \end{aligned} \quad (25)$$

So the third term of $E(\alpha)$ can be estimated as

$$\begin{aligned} E_3(\alpha) &= \sum_{l=1}^K \mathbf{y}_l^T \mathbf{X}_l \mathbf{w}_l = \sum_{l=1}^K \sum_{\tau \in S_l} y_\tau \mathbf{x}_\tau^T \mathbf{w}_l \\ &= \pm \sqrt{\frac{(2P_2 - 1) P_1 P_w N M (K-1) \|\mathbf{y}\|^2 \|\mathbf{X}\|_{\max}}{K C_2 (N+M)}}, \end{aligned} \quad (26)$$

which is ambiguous in the sign. A more deliberate method revealing the sign is to conditionally prepare these two states to make them entangled with an ancilla qubit, $\frac{|0\rangle|\mathbf{y}\rangle + |1\rangle|\hat{\mathbf{y}}\rangle}{\sqrt{2}}$, and perform the swap test on the ancilla qubit with $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ [15], [20]. The success probability is $1 - \langle \mathbf{y} | \hat{\mathbf{y}} \rangle$, which reveals the exact value of $\langle \mathbf{y} | \hat{\mathbf{y}} \rangle$. In fact, when the RR model is well constructed, the predictive outputs $\mathbf{X}_l \mathbf{w}_l$ should be close to the actual outputs \mathbf{y}_l , for $l = 1, \dots, K$, thus in this case the sum of their inner products, $\sum_{l=1}^K \mathbf{y}_l^T \mathbf{X}_l \mathbf{w}_l$, will be positive. Now that all three terms of Eq. (12) can be estimated, the sum of them, $E(\alpha)$, can be directly estimated as well.

(7) For every $\alpha \in \{\alpha_1, \dots, \alpha_L\}$, execute steps (1)-(6), and then pick out the best α with minimum $E(\alpha)$ as the final regularization hyperparameter $\hat{\alpha}$ for RR.

The schematic quantum circuit of steps (1)-(4) of Algorithm 2 is given in Fig. 2 and that of steps (5)-(6) is shown in Fig. 3

In order to implement the unitary operation (17) for phase estimation in step (2), we propose the technique of *parallel Hamiltonian simulation*, which is to simulate a chain of $N \times N$ Hermitian matrices A_1, \dots, A_Q in quantum parallel, i.e., to implement the unitary operation $\sum_{q=1}^Q |q\rangle\langle q| \otimes e^{-\frac{iA_q t}{N}}$, within some error. This technique is detailed by the following theorem whose proof is given in Appendix E, available in the online supplemental material.

Theorem 1 (Parallel Hamiltonian Simulation). *Given Q Hermitian $N \times N$ matrices (Hamiltonians) A_1, \dots, A_Q and efficient quantum oracles that can access the elements of these matrices, the unitary operation $\sum_{q=1}^Q |q\rangle\langle q| \otimes e^{-\frac{iA_q t}{N}}$ can be simulated for time t within spectral-norm error ϵ in time $O(M_A^2 t^2 \text{polylog}(N^2 Q)/\epsilon)$, where the states $|q\rangle$ are the Q computational basis states of a Q -dimensional quantum system and M_A is the maximum absolute value of all the elements of these matrices.*

It is worth noting that the method for implementing parallel Hamiltonian simulation in Theorem 1 is much more advantageous than the intuitive method: put A_1, \dots, A_Q into the diagonal of a larger matrix times Q , $A = \sum_{q=1}^Q |q\rangle\langle q| \otimes QA_q$, which is of size $NQ \times NQ$, and then simulate A via the indefinite non-sparse Hamiltonian simulation [22], i.e., implement the desired unitary operation $e^{-\frac{iAt}{NQ}} = \sum_{q=1}^Q |q\rangle\langle q| \otimes e^{-\frac{iA_q t}{N}}$ within error ϵ . However, since $\|A\|_{\max} = QM_A$, the time complexity of this intuitive method is $O(Q^2 M_A^2 t^2 \text{polylog}(NQ)/\epsilon)$, which is roughly Q^2 times more than that of the method presented in Theorem 1. Therefore, our method in Theorem 1 is much more efficient than the intuitive method, especially when Q is large.

According to Theorem 1, by setting $Q = N$, $M_A = \|\mathbf{X}\|_{\max}$ and $A_q = \tilde{\mathbf{X}}_{-l}$ for any $q \in S_l$, the unitary operation (17) can be implemented within error ϵ in time $O(\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)t^2/\epsilon)$.

3.4 Time Complexity of Algorithm 2

In steps (1)-(4), the time cost is mainly taken for phase estimation and amplitude amplification for generating the state $|\psi_w\rangle$. Similar to Algorithm 1, in step (2), $t_2 = O(\kappa'/\epsilon)$ is required to make the error of $|\psi_w\rangle$ be within ϵ , and thus phase estimation takes time $O(\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^2/\epsilon^3)$. Plus amplitude amplification in step (4) with $O(\kappa'\kappa)$ repetitions, it takes total time $O(\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^3\kappa/\epsilon^3)$ to generate $|\psi_w\rangle$.

In step (5), the second term of $E(\alpha)$, i.e., $E_2(\alpha)$, can be estimated by estimating P_w and P_1 as shown in Eq. (24). Just as estimating $\|\mathbf{y}\|^2$ by amplitude estimation described in Appendix A, available in the online supplemental material, P_w can be estimated within relative error ϵ_w by amplitude estimation with $O(\sqrt{\frac{1-P_w}{P_w}} \frac{1}{\epsilon_w}) = O(\frac{1}{\sqrt{P_w}\epsilon_w})$ repetitions of steps (1)-(3), resulting in the runtime

$$O\left(\frac{\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^2/\epsilon^3}{\sqrt{P_w}\epsilon_w}\right). \quad (27)$$

Similarly, P_1 can be estimated within relative error ϵ_1 by amplitude estimation with $O(\frac{1}{\sqrt{P_1}\epsilon_1})$ repetitions of in total.

generating $|\psi_w\rangle$ and calling O_X and O_X^{-1} as shown in step (5), which results in the runtime

$$O\left(\frac{\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^3\kappa/\epsilon^3}{\sqrt{P_1}\epsilon_1}\right). \quad (28)$$

It should be noted that it is hard to estimate the scale of P_1 (and P_2) because it depends on the closeness between the prediction $\mathbf{w}_\tau^T \mathbf{x}_\tau$ and the actual output y_τ for $l = 1, \dots, K$ and $\tau \in S_l$. But when RR achieves good predictive performance with $\mathbf{w}_\tau^T \mathbf{x}_\tau \approx y_\tau$ for most τ , $P_1 = \Omega(1/\kappa'^2)$ (and $P_2 \approx 1$) as shown in Appendix F, available in the online supplemental material. Moreover, the relative errors for estimating P_w and P_1 , i.e., ϵ_w and ϵ_1 , make the relative error for estimating $P_1 P_w$, as well as that for estimating $E_2(\alpha)$ as shown in Eq. (24), be within $O(\epsilon_w + \epsilon_1)$. So, in conclusion, putting the runtime for estimating these three probability together, $E_2(\alpha)$ can be estimated within relative error $O(\epsilon_w + \epsilon_1)$ in total time

$$\begin{aligned} & O\left[\frac{\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^2}{\epsilon^3} \left(\frac{1}{\sqrt{P_w}\epsilon_w} + \frac{\kappa'\kappa}{\sqrt{P_1}\epsilon_1}\right)\right] \\ &= O\left[\frac{\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^3\kappa}{\epsilon^3} \left(\frac{1}{\epsilon_w} + \frac{\kappa'}{\epsilon_1}\right)\right], \end{aligned} \quad (29)$$

since $P_w = \Omega(1/\kappa'^2\kappa^2)$ and $P_1 = \Omega(1/\kappa'^2)$.

In step (6), P_2 can be estimated within relative error ϵ_2 by amplitude estimation with $O(\frac{1}{\sqrt{P_2}\epsilon_2})$ repetitions of generating $|\mathbf{y}\rangle$ and $|\hat{\mathbf{y}}\rangle$. The state $|\mathbf{y}\rangle$ is generated in time $O(\text{polylog}N)$ as shown in Appendix A, available in the online supplemental material. With help of amplitude amplification, generating $|\hat{\mathbf{y}}\rangle$ in step (5) takes time $O\left(\frac{1}{\sqrt{P_1}} \frac{\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^3\kappa}{\epsilon^3}\right)$. As a result, P_2 can be estimated within relative error ϵ_2 with runtime $O\left(\frac{\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^4\kappa}{\epsilon^3\epsilon_2}\right)$ since $P_1 = \Omega(1/\kappa'^2)$ and $P_2 \approx 1$ as shown in Appendix F, available in the online supplemental material. The relative errors for estimating P_w , P_1 and P_2 , ϵ_w , ϵ_1 and ϵ_2 , respectively, make the relative error for estimating $\sqrt{(2P_2-1)P_1P_w}$, as well as that for estimating $E_3(\alpha)$ as shown in Eq. (26), be within $O(2\epsilon_2 + \epsilon_w + \epsilon_1)$ due to $P_2 \approx 1$. So, $E_3(\alpha)$ can be estimated within relative error $O(2\epsilon_2 + \epsilon_w + \epsilon_1)$ in time $O\left(\frac{\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^4\kappa}{\epsilon^3\epsilon_2}\right)$.

Consider that $E_1(\alpha) = \sum_{l=1}^K \|\mathbf{y}_l\|^2 = \|\mathbf{y}\|^2$ can be estimated within relative error ϵ_y taking runtime $O(\text{polylog}(N)/\epsilon_y)$ as shown in Appendix A, available in the online supplemental material. Letting $\epsilon_y = \epsilon$, $\epsilon_w = \epsilon_1 = \epsilon/3$, and $\epsilon_2 = \epsilon/6$, each of the three terms $E_1(\alpha)$, $E_2(\alpha)$ and $E_3(\alpha)$ has relative error $O(\epsilon)$, and thus $E(\alpha) = E_1(\alpha) + E_2(\alpha) + E_3(\alpha)$ has relative error $O(\epsilon)$, and the runtime for estimating it scales as $O\left(\frac{\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^4\kappa}{\epsilon^3}\right)$. Furthermore, step (7) involves estimating $E(\alpha_1), \dots, E(\alpha_L)$, so Algorithm 2 takes runtime

$$O\left(\frac{L\|\mathbf{X}\|_{\max}^2 \text{polylog}(N+M)\kappa'^4\kappa}{\epsilon^4}\right) \quad (30)$$

The best classical counterpart of Algorithm 2 consists of L iterations and each one evolves a K -fold cross validation. In j th ($j = 1, 2, \dots, L$) iteration, two phases are involved: (1) K turns of RR with α_j are run and in l th turn \mathbf{w}_l (Eq. (11)) is output in time $O\left(\frac{(K-1)NM}{K} + \frac{(K-1)^2 N^2 R_l \log(R_l/\epsilon)}{K^2 \epsilon^2}\right)$ [25]; (2) $E(\alpha_j)$ is calculated according to Eq. (12), which is easy to see the time complexity scales as $O(NM)$. So the total time complexity scales as $O\left(LNM + \frac{LN^2(\sum_{l=1}^K R_l \log(R_l/\epsilon))}{\epsilon^2}\right)$ due to $K \geq 2$.

Considering the assumptions of $\|\mathbf{X}\|_{\max} = \Theta(1)$ and $N = \Theta(M)$, and $\kappa' = O(\kappa)$ by setting $K = \Omega\left(\frac{NM\|\mathbf{X}\|_{\max}^2 \kappa^2}{(N+M)^2}\right) = O(\kappa^2)$, our Algorithm 2 has runtime $O(L \text{polylog}(N) \kappa^5 / \epsilon^4)$, while the best classical counterpart has runtime $\tilde{O}\left[LN^2\left(\sum_{l=1}^K R_l\right)/\epsilon^2\right]$, where \tilde{O} is used to suppress the relatively small quantities of $\log(R_l/\epsilon)$. When $\kappa = O(\sqrt{N})$ which is amenable to full or approximately full rank of \mathbf{X} and \mathbf{X}_l , i.e., $R_l = O(N)$, our Algorithm 2 has runtime $O(L \text{polylog}(N) N^{2.5} / \epsilon^4)$, whereas the classical counterpart takes time $O(L \text{polylog}(N) N^4 / \epsilon^2)$, so polynomial speedup over the classical counterpart can be achieved when $L, 1/\epsilon = O(\text{polylog}(N))$. However, when $\kappa = O(\text{polylog}(N))$ which implies low rank of \mathbf{X} as well as \mathbf{X}_l , i.e., $R_l = \text{polylog}(N)$, our Algorithm 2 has runtime $O(L \text{polylog}(N) / \epsilon^4)$, while the classical counterpart has runtime $O(L \text{polylog}(N) N^2 / \epsilon^2)$, so in this case exponential speedup can be achieved when $L, 1/\epsilon = O(\text{polylog}(N))$.

3.5 The Whole Quantum Algorithm for RR

Our quantum algorithm for RR will start with Algorithm 2 to find a good α , and then plug such α into Algorithm 1 to estimate the optimal fitting parameters in the quantum state form. The quantum state can further be applied to efficiently predict new data via swap test. It is easy to see the time complexity of the whole algorithm is dominated by Algorithm 2, and thus the speedup over the classical algorithm also depends on the condition number of design matrix as discussed in Algorithm 2 above.

4 CONCLUSIONS

In summary, we have described a quantum algorithm that can efficiently implement RR over an exponentially large data set. In particular, we propose the technique of parallel Hamiltonian simulation and use it to develop the quantum K -fold cross-validation that can efficiently evaluate the predictive performance of RR. The algorithm first uses quantum K -fold cross-validation to efficiently determine a good α with which RR can achieve good predictive performance, and then generates a quantum state encoding the optimal fitting parameters of RR with such α . The state can be further used to efficiently predict new data. It is shown that our algorithm can handle data sets with non-sparse design matrices, and is able to be exponentially faster than the classical algorithm for (low-rank) design matrices with low condition numbers, but be polynomially faster than the classical algorithm for (full or approximately full) design matrices with large condition numbers.

We hope our algorithm and especially the key techniques used in our algorithm, parallel Hamiltonian simulation and

quantum K -fold cross-validation, can inspire more efficient quantum machine learning algorithms. For example, since cross-validation is an important technique being widely used to estimate the predictive performance of various machine learning algorithms [5], [28] other than RR, it is promising that our quantum K -fold cross-validation can be applicable in these fields. We explore these possibilities in the future.

ACKNOWLEDGMENTS

The authors would like to thank J. B. Wang, L.-C. Wan, S.-J. Pan, H.-L. Liu, D. Li, B. J. Duan, and S. Hua for helpful discussions. This work is supported by NSFC (Grant Nos. 61976024, 61972048, 61671082, and 61672110).

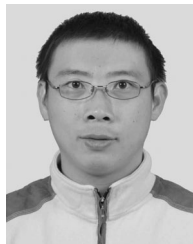
REFERENCES

- [1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2010.
- [2] A. Montanaro, "Quantum algorithms: An overview," *npj Quantum Inf.*, 2, 2016, Art. no. 15023.
- [3] C.-H. Yu, F. Gao, Q.-L. Wang, and Q.-Y. Wen, "Quantum algorithm for association rules mining," *Phys. Rev. A*, vol. 94, 2016, Art. no. 042311.
- [4] L.-C. Wan, C.-H. Yu, S.-J. Pan, F. Gao, Q.-Y. Wen, and S.-J. Qin, "Asymptotic quantum algorithm for the Toeplitz systems," *Phys. Rev. A*, vol. 97, 2018, Art. no. 062322.
- [5] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
- [6] P. Wittek, *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. New York, NY, USA: Academic, 2014.
- [7] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Phys.*, vol. 56, 2015, Art. no. 172.
- [8] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, pp. 195–202, 2017.
- [9] B. Duan, J. Yuan, Y. Liu, and D. Li, "Quantum algorithm for support matrix machines," *Phys. Rev. A*, vol. 96, 2017, Art. no. 032301.
- [10] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, 1970, Art. no. 55.
- [11] W. N. van Wieringen, "Lecture notes on ridge regression," *arXiv:1509.09169*, 2015.
- [12] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, 2009, Art. no. 150502.
- [13] N. Wiebe, D. Braun, and S. Lloyd, "Quantum algorithm for data fitting," *Phys. Rev. Lett.*, vol. 109, 2012, Art. no. 050505.
- [14] Y. Liu and S. Zhang, "Fast quantum algorithms for least squares regression and statistic leverage scores," *Theor. Comput. Sci.*, vol. 657, 2017, Art. no. 38.
- [15] M. Schuld, I. Sinayskiy, and F. Petruccione, "Prediction by linear regression on a quantum computer," *Phys. Rev. A*, vol. 94, 2016, Art. no. 022342.
- [16] G. Wang, "Quantum algorithm for linear regression," *Phys. Rev. A*, vol. 96, 2017, Art. no. 012335.
- [17] S. Aaronson, "Read the fine print," *Nature Phys.*, vol. 11, 2015, Art. no. 291.
- [18] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Phys. Rev. Lett.*, vol. 100, 2008, Art. no. 160501.
- [19] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, "Quantum fingerprinting," *Phys. Rev. Lett.*, vol. 87, 2001, Art. no. 167902.
- [20] S. Lloyd, M. Mohseni and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," *arXiv:1307.0411v2*, 2013.
- [21] L. Hogben, *Handbook of Linear Algebra*. Boca Raton, FL, USA: CRC Press, 2006.
- [22] P. Rebentrost, A. Steffens, I. Marvian and S. Lloyd, "Quantum singular-value decomposition of nonsparse low-rank matrices," *Phys. Rev. A*, vol. 97, 2018, Art. no. 012327.
- [23] L. Grover and T. Rudolph, "Creating superpositions that correspond to efficiently integrable probability distributions," *arXiv: quant-ph/0208112*, 2002.

- [24] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum Amplitude Amplification and Estimation*. New York, NY, USA: AMS, 2002.
- [25] S. Chen, Y. Liu, M. Lyu, I. King, and S. Zhang, "Fast relative-error approximation algorithm for ridge regression," in *Proc. 31th Conf. Uncertainty Artif. Intell.*, 2015., Art. no. 201.
- [26] J. N. Franklin, *Matrix Theory*. New York, NY, USA: Dover, 1993.
- [27] D. W. Berry, G. Ahokas, R. Cleve and B. C. Sanders, "Efficient quantum algorithms for simulating sparse Hamiltonians," *Comm. Math. Phys.*, vol. 270, 2007, Art. no. 359.
- [28] X. He, C. Zhang, L. Zhang and X. Li, "A-optimal projection for image representation," *IEEE Trans. Pattern Anal. Mach. Learn.*, vol. 38, no. 5, pp. 1009–1015, May 2016.



Chao-Hua Yu received the BS degree in information and computing science from Shenyang Agricultural University, Shenyang, China, in 2011, and the ME degree in computer applied technology from Fujian Normal University, Fuzhou, China, in 2014. He is currently working toward the doctorate degree in cryptography at the Beijing University of Posts and Telecommunication (BUPT). In 2014, he joined NSRC, SKLNST, Beijing, China, where he participated in the research of quantum algorithms. He is currently working on the quantum algorithms for machine learning and data mining.



Fei Gao received the BE degree in communication engineering and PhD degree in cryptography from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2002 and 2007, respectively. He is a professor of BUPT. In 2002, he joined NSRC, SKLNST, Beijing, China, where he participated in the research of quantum cryptography and quantum information. He is currently working on the practical quantum cryptographic protocols, quantum nonlocality, and quantum algorithms. He is a member of the Chinese Association for Cryptologic Research (CACR).



Qiao-Yan Wen received the BS and MS degrees in mathematics from Shaanxi Normal University, Xi'an, China, in 1981 and 1984, respectively, and the PhD degree in cryptography from Xidian University, Xi'an, China, in 1997. She is currently a professor of the Beijing University of Posts and Telecommunications and the leader of NSRC, SKLNST, Beijing, China. Her current research interests include cryptography, information security, Internet security, and applied mathematics. She is a senior member of CACR.

▷ **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.**