Communication-Efficient Quantum Algorithm for Distributed Machine Learning

Hao Tang,[1,*] Boning Li,[2,3,*] Guoqing Wang,[2,4] Haowei Xu,[4] Changhao Li,[2,4]
Ariel Barr,[1] Paola Cappellaro,[2,3,4,†] and Ju Li[1,4,‡]

[1]*Department of Materials Science and Engineering, Massachusetts Institute of Technology, Massachusetts 02139, USA*
[2]*Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*
[3]*Department of Physics, Massachusetts Institute of Technology, Massachusetts 02139, USA*
[4]*Department of Nuclear Science and Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*

(Received 11 September 2022; accepted 21 March 2023; published 12 April 2023)

The growing demands of remote detection and an increasing amount of training data make distributed machine learning under communication constraints a critical issue. This work provides a communication-efficient quantum algorithm that tackles two traditional machine learning problems, the least-square fitting and softmax regression problems, in the scenario where the dataset is distributed across two parties. Our quantum algorithm finds the model parameters with a communication complexity of $O(\log_2(N)/\epsilon)$, where N is the number of data points and ϵ is the bound on parameter errors. Compared to classical and other quantum methods that achieve the same goal, our methods provide a communication advantage in the scaling with data volume. The core of our methods, the quantum bipartite correlator algorithm that estimates the correlation or the Hamming distance of two bit strings distributed across two parties, may be further applied to other information processing tasks.

DOI: 10.1103/PhysRevLett.130.150602

The amount of training data is critical for machine learning to achieve high accuracy, generalization capabilities, and predictive power. Nowadays, data collection is growing with unprecedented speed around the world, so it becomes a challenge for algorithms to exploit such large-scale data within feasible time and memory [1,2]. Distributed machine learning emerges as a promising solution, where the training data and learning process are allocated to multiple machines [1,3,4]. This scales up computational power and is also suitable for intrinsically distributed data when collected [5,6]. However, these algorithms require extensive communication between different machines, which usually becomes a rate-limiting step [7]. Therefore, efficient communication schemes for distributed machine learning tasks are attracting broad interest. The communication necessary between two machines in a computation task is quantified by its communication complexity, either within classical [8–11] or quantum channels [12–15]. Compared to classical communication, even though quantum algorithms have been shown to reduce the communication complexity in some scenarios [16], machine learning tasks were not included. Quantum algorithms have been generally studied as accelerators for the computational complexity [17] in problems such as least-square fitting [18], statistical inference [19], feature engineering [20], and classification problems [21]. Whether quantum algorithms can accelerate communication in distributed learning tasks remains an open question.

Here, we propose a quantum communication algorithm for two typical data fitting subroutines in machine learning: least-square fitting and softmax regression, which are the common output layers of predictors and classifiers, respectively [22]. In this Letter we assume a training dataset contains N independent identically distributed (iid) data points. Each data point has an M-dimensional input $\vec{x}$ and a scalar output y. In the basic communication scenario [4], the training dataset, comprising the input attributes and labels, is distributed across two parties, Alice and Bob. Both least-square fitting and softmax regression aim at fitting a model $y \approx f(\vec{x}, \boldsymbol{\lambda})$ to the data, by estimating the parameters $\hat{\boldsymbol{\lambda}}$ that minimize a given loss function. The goal of a communication algorithm is to minimize the number of bits [8,9] or qubits [14,15] exchanged between Alice and Bob during model fitting, while keeping the accuracy of $\hat{\boldsymbol{\lambda}}$ within a standard error ϵ.

Least-square fitting has been extensively studied in both classical distributed algorithms and single-party (no communication) quantum algorithms. Using a classical algorithm based on correlation estimation, it has been proved that the classical communication complexity cannot be below $O(1/\epsilon^2)$ [23,24]. However, to reach such a lower bound requires an exponentially large number of data points. In the case of finite datasets, since the accuracy of the fitting parameters should be at least as small as its error ϵ, a classical deterministic method requires $O[N\log_2(1/\epsilon)]$ bits to be exchanged between two parties within a precision ϵ [25]. When high accuracy is not required, only $1/\epsilon^2$ data points with random indexes need to be transferred, which yields a $O\{[\log_2(1/\epsilon) + \log_2(N)](1/\epsilon^2)\}$ communication complexity [23]. Then, to achieve a statistical variance $\epsilon_s^2 = \mathrm{var}(|\lambda|) \propto 1/N$, these two classical algorithms have the same communication complexity $O[N\log_2(N)]$ or $O(\log_2(1/\epsilon_s)/\epsilon_s^2)$. In comparison, quantum computation methods for linear fitting based on the Harrow-Hassidim-Lloyd (HHL) algorithm [26] yield normalized parameters ($|\boldsymbol{\lambda}|^2 = 1$) from a quantum state $|\lambda\rangle = \sum_{j=1}^{M} \lambda_j |j\rangle$ with communication complexity of $O[\log_2(N)]$ [18,27,28]. However, to extract $\lambda_{j=1,\ldots,M}$, the HHL-based algorithm requires $O[M^2(1/\epsilon^2)]$ repeated measurements. In this case, the HHL-based fitting algorithm requires communicating $O(\log_2(N)/\epsilon^2)$ qubits [18,29], with no clear advantage over classical algorithms.

We designed a *quantum counting*-based [30,31] communication algorithm that achieves a reduced communication complexity of $O(\log_2(N)/\epsilon)$ for both least-square fitting and softmax regression (Table I). At its core, the direct action of our algorithm is to estimate the correlation or the Hamming distance of two bit strings distributed across two parties. Embedding this algorithm into a hybrid computing scheme enables the data fitting tasks beyond the theoretical limit of classical algorithms, and we expect it could benefit other scenarios not analyzed here.

Estimating correlation.—We first present the core subroutine of our methods, the quantum bipartite correlator (QBC) algorithm. The problem is stated as follows: Alice and Bob have N-dimensional vectors $\vec{x}^{\mathrm{b}}, \vec{y}^{\mathrm{b}} \in \{0,1\}^N$, respectively, that can only take binary values (denoted by superscript $^{\mathrm{b}}$). This is not as restrictive as it sounds, as real numbers can always be expanded as binary floating point numbers (see Sec. "*Least-square fitting*"). The task is to estimate the correlation $\hat{\rho} \equiv \left[(\overline{x^{\mathrm{b}}y^{\mathrm{b}}} - \overline{x^{\mathrm{b}}} \cdot \overline{y^{\mathrm{b}}}) / \sqrt{\overline{x^{\mathrm{b}}}(1-\overline{x^{\mathrm{b}}})\overline{y^{\mathrm{b}}}(1-\overline{y^{\mathrm{b}}})}\right]$, in which the communication-intensive step is to evaluate $\overline{x^{\mathrm{b}}y^{\mathrm{b}}} = (1/N)\sum_{i=1}^{N} x_i^{\mathrm{b}} y_i^{\mathrm{b}}$ within a standard deviation error ϵ [23].

We assume that Alice and Bob have access to quantum computers with oracles. The oracle of Alice's computer performs a unitary transformation $\hat{U}_{\vec{x}^{\mathrm{b}}}^{1,2}$: $|i\rangle_1|0\rangle_2 \mapsto |i\rangle_1|x_i^{\mathrm{b}}\rangle_2$ that encodes the data x_i^{b}, where $|i\rangle$ is an $n \equiv \lceil \log_2(N) \rceil$-qubit state $|i_1 i_2 \cdots i_n\rangle$, representing the index of the queried component, and $|x_i^{\mathrm{b}}\rangle$ is a single-qubit state. Bob has an oracle $\hat{U}_{\vec{y}^{\mathrm{b}}}$ of the same type that encodes the data y_i^{b}. This type of oracle is a common building block in quantum algorithms [18,26,36], which can be realized through quantum random access memory [37] or other data-loading procedures [38,39].

Estimating the correlation $\overline{x^{\mathrm{b}}y^{\mathrm{b}}}$ is based on quantum counting, in which the phase oracle is realized cooperatively by Alice and Bob through communication, as shown in Fig. 1. We sketch the framework here and provide the algorithm details in the Supplemental Material [40] (which includes Refs. [31,33,35,41]), Sec. I. The algorithm works on an n-qubit vector index space ($|\cdot\rangle_n$), a t-qubit register space ($|\cdot\rangle_t$), and a 2-qubit oracle workspace ($|\cdot\rangle_o$). Initially, all qubits are set to zero: $|\psi_0\rangle \equiv |0\rangle_t|0\rangle_n|00\rangle_o$. Hadamard gates are applied to create superposition in both t and n space $|\psi_1\rangle = 2^{-(t+n)/2}\sum_{i,\tau}|\tau\rangle_t|i\rangle_n|00\rangle_o$. A phase oracle on the state $|\cdot\rangle_n$ can be realized through the following unitary operation:

$$\hat{O}_{\vec{x}^{\mathrm{b}},\vec{y}^{\mathrm{b}}} \equiv \hat{U}_{\vec{x}^{\mathrm{b}}}^{n,o_1}\hat{U}_{\vec{y}^{\mathrm{b}}}^{n,o_2} CZ^{o_1,o_2}\hat{U}_{\vec{y}^{\mathrm{b}}}^{n,o_2}\hat{U}_{\vec{x}^{\mathrm{b}}}^{n,o_1}, \tag{1}$$

which yields $\hat{O}_{\vec{x}^{\mathrm{b}},\vec{y}^{\mathrm{b}}}|i\rangle_n|00\rangle_o = (-1)^{x_i^{\mathrm{b}}y_i^{\mathrm{b}}}|i\rangle_n|00\rangle_o$. Here o_1, o_2 are the two qubits in the oracle space, and CZ^{o_1,o_2} is a control-Z gate acting on them. Each oracle call requires about

TABLE I. Communication complexity of classical distributed algorithm, quantum counting-based algorithm developed in this work, and other quantum algorithms. Listed problems include estimating correlation and Hamming distance of two separate bit strings, distributed linear fitting, and distributed softmax regression. In the first column, (c) and (q) mean the problem requires output as classical data or quantum states, respectively. In the table, ϵ, N, and M are the standard error of solution, number of data points, and number of attributes in Alice's data; κ and s are the condition number and sparseness of the matrix $\boldsymbol{X}$ in linear regression problems; and q is the number of classes in softmax regression problems. (See derivation in Sec. III.). All the classical algorithms and the LOCC algorithm transfer classical bits, and the rest of the quantum algorithms transfer qubits.

Problem (output)	Classical algorithm	Quantum counting	Other quantum algorithm
Correlation (c)	$O(1/\epsilon^2)$ lower-bound) Ref. [23]	$O(\log_2(N)/\epsilon)$	$O(\log_2(N)/\epsilon^2)$ (swap-test, [32]) $O[\log_2(N)\max\{(1/\epsilon^2),(\sqrt{N}/\epsilon)\}]$ (LOCC, [33])
Hamming distance (c)	$O(N)$ [34]	$O(\log_2(N)/\epsilon)$	$O(\log_2(N)/\epsilon^2)$ (classical shadows, [35])
Linear-fitting (c)	$O[N\log_2(\kappa^2/\epsilon)]$ (deterministic [25]) $O\{[\log_2(N)+\log_2(\kappa^2/\epsilon)]/(\epsilon/\kappa^2)^2\}$ (stochastic [23])	$O(M\kappa(\log_2(N)/\epsilon))$	$O(M^2\kappa^5(\log_2(N)/\epsilon^2))$ (HHL, [18])
Linear-fitting (q)	$\cdots$	$O(M\kappa(\log_2(N)/\epsilon))$	$O[\kappa^5\log_2(N)]$ (HHL, [18])
Softmax regression (c)	$O(N\log_2 q)$	$O\{Mq\kappa[\log_2(N)/\epsilon]\}$	$\cdots$

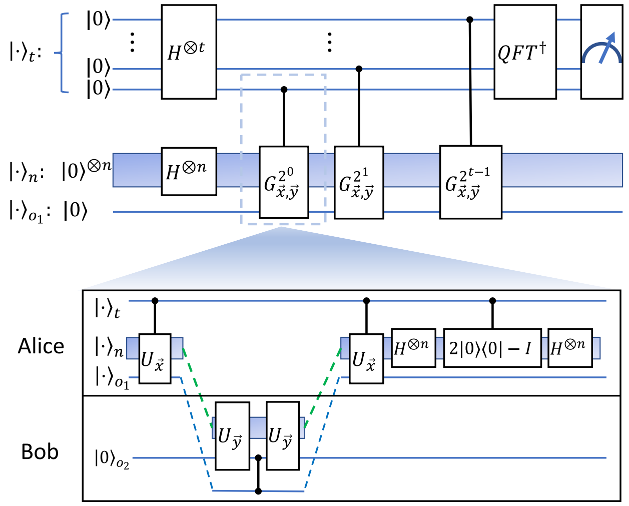

FIG. 1. Quantum circuits for the distributed quantum counting or QBC scheme. H, G, and QFT† represent the Hadamard gate, the Grover operator, and the inverse QFT, respectively. The t-qubit register is measured after the inverse QFT. The inset shows the biparty distributed scheme of the Grover operation, where U_{x_l} and V_{y_k} are defined in Eqs. (7) and (8).

$2n$-qubit communication, as Alice needs to send the $(n+1)$ qubits to Bob after applying $\hat{U}^{n,o_1}_{\vec{x}^b}$ and Bob needs to send the $(n+1)$ qubits back after applying $\hat{U}^{n,o_2}_{\vec{y}^b}CZ^{o1,o2}\hat{U}^{n,o_2}_{\vec{y}^b}$; finally, Alice applies $\hat{U}^{n,o_1}_{\vec{x}^b}$ to finish the whole oracle $\hat{O}_{\vec{x}^b,\vec{y}^b}$. The Grover operation needed for counting is then constructed as $\hat{G}_{\vec{x}^b,\vec{y}^b} \equiv \hat{H}^{\otimes n}(2|0\rangle_n\langle 0|_n - \hat{I})\hat{H}^{\otimes n}\hat{O}_{\vec{x}^b,\vec{y}^b}$. The QBC scheme applies the Grover operation iteratively on the initial state:

$$|\psi_2\rangle = \frac{1}{2^{(t+n)/2}}\sum_\tau |\tau\rangle_t \otimes (\hat{G}_{\vec{x}^b,\vec{y}^b})^\tau \sum_i |i\rangle_n |00\rangle_o. \qquad (2)$$

Expanding the Grover operator in its eigenbasis gives $(\hat{G}_{\vec{x}^b,\vec{y}^b})^\tau \sum_i |i\rangle_n = (e^{i\tau\theta}|\phi_+\rangle\langle\phi_+| + e^{-i\tau\theta}|\phi_-\rangle\langle\phi_-|)\sum_i |i\rangle_n$, where $|\phi_\pm\rangle$ are the two eigenstates of $\hat{G}_{\vec{x}^b,\vec{y}^b}$, and $\theta = 2\arcsin\left(\sqrt{\overline{x^b y^b}}\right)$. Applying the inverse quantum Fourier transform (QFT†) to $|\cdot\rangle_t$ yields the final state:

$$|\psi_3\rangle = \frac{1}{\sqrt{2^{t+n}}}\sum_{\eta=\pm,i} \langle\phi_\eta|i\rangle|\phi_\eta\rangle_n|00\rangle_o \mathrm{QFT}^\dagger\left(\sum_\tau |\tau\rangle_t e^{i\eta\tau\theta}\right). \qquad (3)$$

Measuring the t register will project into a state $|j\rangle_t$ resulting in the phase $2\pi j \cdot 2^{-t}$ which encodes either $\hat{\theta}$ or $2\pi - \hat{\theta}$ with an equivalent standard deviation: $\Delta\hat{\theta} = 2^{-t+1}$.

Both cases give the same estimated correlation $\widehat{\overline{x^b y^b}} = \sin^2(\hat{\theta}/2)$, with standard deviation $\epsilon = \sqrt{\overline{x^b y^b}(1 - \overline{x^b y^b})}2^{-t+1}$ (see the Supplemental Material [40], Sec. II, for details). The overall communication complexity $\mathcal{C}$ is the Grover operation's $2(n+1)$ qubit communication repeated for $2^t - 1$ iterations:

$$\mathcal{C} = 2(n+1)(2^t - 1) = O\left(\frac{\log_2(N)}{\epsilon}\right), \qquad (4)$$

where we choose t to satisfy the desired error bound. The computational complexity is the total number of oracle calls by Alice and Bob, which is $\mathcal{C}_{\rm comp} = 4(2^t - 1) = O(1/\epsilon)$.

We note that the QBC algorithm solves the problem of estimating $\overline{x^b y^b}$, which is equivalent to computing the inner product. The inner product of quantum states is usually accomplished by the swap test algorithm [32,33]. However, the swap test method costs $O(\log_2(N)/\epsilon^2)$ bits of communication, due to the requirement of repeated measurements. Recently, Anshu *et al.* [33] proposed an algorithm to estimate the inner product of two quantum states using local quantum operations and classical communication (LOCC). With respect to communication complexity, neither the original SWAP test that transfers qubits, nor LOCC that transfers bits, achieves an advantage over the classical algorithms. The QBC algorithm achieves the communication advantage by utilizing quantum counting and a distributed implementation of the Grover iterator.

Estimating the Hamming distance.—The QBC algorithm can be used to estimate the Hamming distance d between $\vec{x}^b$ and $\vec{y}^b$ (that is, the number of positions i where $x^b_i \neq y^b_i$). The key is to replace the oracle in Eq. (1) by

$$\hat{O}'_{\vec{x}^b,\vec{y}^b} \equiv \hat{U}^{n,o_1}_{\vec{x}^b}\hat{U}^{n,o_2}_{\vec{y}^b} C^{o_1,o_2}_{\rm NOT} Z^{o_2} C^{o_1,o_2}_{\rm NOT} \hat{U}^{n,o_2}_{\vec{y}^b}\hat{U}^{n,o_1}_{\vec{x}^b}, \qquad (5)$$

where $C^{o_1,o_2}_{\rm NOT}$ represents a Control-NOT (CNOT) gate gate with o_1 as control qubit, and Z^{o_2} represents a σ_Z gate acting on the o_2 qubit. This phase oracle acts as $\hat{O}'_{\vec{x}^b,\vec{y}^b}|i\rangle_n|00\rangle_o = (-1)^{x^b_i \oplus y^b_i}|i\rangle_n|00\rangle_o$, and the QBC scheme counts the number of indexes i such that $x^b_i \oplus y^b_i = 1$, returning (d/N) with the same communication complexity as for estimating the correlation.

This result provides a quantum solution to the widely studied gap-Hamming problem in theoretical computer science [34,42]. Multiple proofs conclude that it is impossible for a classical protocol to output the Hamming distance d within $\sqrt{N}$ using less than $O(N)$ bits of communication [23,34,43]. By setting $\epsilon = (1/\sqrt{N})$, our quantum scheme performs the estimation using $O[\sqrt{N}\log_2(N)]$ qubits of communication, exhibiting a square-root speedup over classical algorithms. The Hamming distance can also be estimated via the "classical shadows" algorithm [35] (an established quantum algorithm) with communication complexity of $O(\log N/\epsilon^2)$ (see the Supplemental Material [40], Sec. V, for details), which has a higher order to $(1/\epsilon)$ than the QBC algorithm. As estimating the Hamming distance under communication constraints has applications in database searching [42], networking [44], and streaming algorithms [45], the QBC algorithm may be embedded into other diverse applications in the future.

Least-square fitting.—When machine learning models are used to predict the central value of Gaussian distributed continuous variables, the common setting is a linear output layer $f(\boldsymbol{x_i}, \boldsymbol{\lambda}) = \lambda_0 + \vec{\lambda} \cdot \vec{x} = \boldsymbol{\lambda}^T \boldsymbol{x}$ (where $\boldsymbol{x_i} \equiv (1, x_{i,1}, \ldots, x_{i,M-1})^T$ and $\boldsymbol{\lambda} \equiv (\lambda_0, \lambda_1, \ldots, \lambda_{M-1})^T$) that performs the least-square fitting. The model fitting is reduced to solving a linear least-square problem $\boldsymbol{X\lambda} = \boldsymbol{y}$, where $\boldsymbol{X} \equiv (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^T$ is an $N \times M$ matrix belonging to Alice and $\boldsymbol{y}$ is Bob's $N \times 1$ column vector, both of which have real-number components. The goal is to estimate $\hat{\boldsymbol{\lambda}}$ with standard error ϵ using minimal communications. Here we assume $M \ll N$, as the number of model parameters or attributes is usually much smaller than the number of data points to avoid overfitting.

The least-square solution of the equation is $\boldsymbol{\lambda} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = (1/N)(N\boldsymbol{X}^\dagger)\boldsymbol{y}$, where $\boldsymbol{X}^\dagger$ is the Moore-Penrose pseudoinverse of $\boldsymbol{X}$, and $N\boldsymbol{X}^\dagger$ should scale as $O(N^0)$ in the case of the iid dataset. As $N\boldsymbol{X}^\dagger$ can be computed by Alice locally, only the calculation of $(1/N)(N\boldsymbol{X}^\dagger)\boldsymbol{y}$ involves communication. The jth component of λ can be represented by correlations (inner product) $\lambda_j = (1/N)\sum_i (NX^\dagger_{ji})y_i$, $j = 0, \ldots, M-1$, which can be calculated by expanding the real numbers as binary floating point numbers. For example, following the IEEE 754 standard [46], each $NX^\dagger_{ji}$ and y_i can be written as binary floating point numbers: $NX^\dagger_{ji} \equiv \sum_{k=0}^\infty 2^{u-k} x^{\mathrm{b}k}_{ji}$, $y_i \equiv \sum_{k=0}^\infty 2^{v-k} y^{\mathrm{b}k}_i$, where u and v are the highest digits of the elements of $NX^\dagger_{ji}$ and y_i, and $x^{\mathrm{b}k}_{ji}$ and $y^{\mathrm{b}k}_i$ are the kth digits, respectively. Then λ_j can be written as

$$\begin{aligned} \lambda_j &= \frac{1}{N} \sum_{r=0}^{\infty} 2^{u+v-r} \sum_{k=0}^{r} \sum_{i=1}^{N} x^{\mathrm{b}k}_{ji} y^{\mathrm{b}(r-k)}_i \\ &= 2^{u+v} \sum_{r=0}^{\infty} 2^{-r}(r+1) f_{jr}. \end{aligned} \tag{6}$$

As $x^{\mathrm{b}k}_{ji}$ and $y^{\mathrm{b}k}_i$ are binary quantity, the inner product $f_{jr} = [1/N(r+1)] \sum_{k=0}^{r} \sum_{i=1}^{N} x^{\mathrm{b}k}_{ji} y^{\mathrm{b}(r-k)}_i$ can be directly estimated by the QBC algorithm. The overall communication complexity is $\mathcal{C} = \sum_{j=1}^{M} \sum_{r=0}^{\infty} 2[\log_2(N)/\epsilon_{jr}]$, where ϵ_{jr} is the standard deviation error of f_{jr}. The infinite series in r is cut off according to the target accuracy ϵ of each component λ_j, setting ϵ_{jr} to $\epsilon_{jr} = \epsilon[(0.449/2^{u+v})(r+1)^{2/3}]2^{(2/3)r}$. If r is large enough so that $\epsilon_{jr} > 1$, the quantum algorithm is no longer pertinent, as the number t of ancilla qubits in the quantum phase estimation algorithm drops to less than one, since $\epsilon_{jr} = 2^{-t+1}$. In that case, f_{jr} can be simply dropped because these f_{jr} terms are multiplied by 2^{-r} in Eq. (6); they do not contribute substantially to the total error of λ_j. Rewriting $\mathcal{C}$ in terms of the condition number $\kappa = \|\mathbf{A}^{-1}\|_\infty \|\mathbf{A}\|_\infty$ of the matrix $\mathbf{A} = (1/N)\boldsymbol{X}^T\boldsymbol{X}$ gives

$$\mathcal{C} = 11.026 \times 2^{v+1} 2^u M \frac{\log_2(N)}{\epsilon} = O\left(\frac{M\kappa \log_2(N)}{\epsilon}\right), \tag{7}$$

where the absolute magnitude of 2^{v+u} in $\mathcal{C}$ is on the same order of $(\kappa|y|_\infty/\|X\|_\infty)$ (see the Supplemental Material [40], Sec. III, for details). The total number of oracle queries is $\mathcal{C}_{\mathrm{comp}} = (M\kappa/\epsilon)$.

An HHL-based quantum algorithm has been previously developed for data fitting without the communication bottleneck [18]. The algorithm produces a quantum state $|\lambda\rangle \equiv \sum_j \lambda_j |j\rangle$ with $O[(s^3\kappa^6/\epsilon)\log_2(N)]$ computational complexity, where $0 \leq s \leq 1$ is the sparseness of the matrix A. As explained above, this method is, however, inefficient in extracting classical data from the quantum states. In the communication-restricted scenario, the HHL-based algorithm requires sharing $O[(\kappa^5 M^2/\epsilon^2)\log_2(N)]$ qubits. For a target statistical precision $\epsilon = 1/\sqrt{N}$, the QBC based scheme again obtains a square-root speedup from $O(N)$ to $O[\sqrt{N}\log_2(N)]$ compared with the classical theoretical limit. A summary of the communication complexity of different schemes is presented in Table I.

After demonstrating that the QBC algorithm can reduce the communication complexity to N, we numerically assess the practical conditions when the quantum algorithm shows

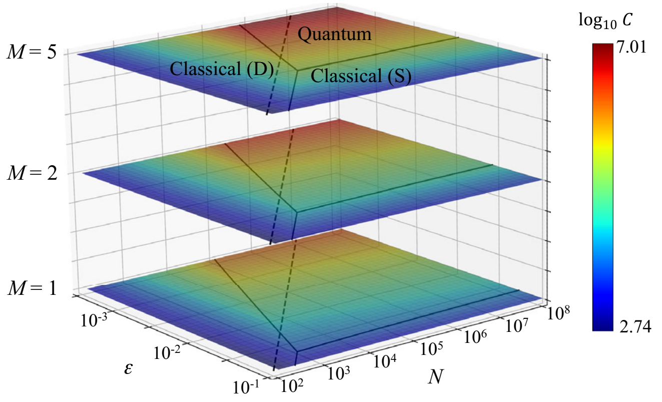

FIG. 2. Communication complexity phase diagram of the QBC algorithm, deterministic, and stochastic classical algorithms in parameter space of N, ϵ, and M. Without loss of generality, we assume that both $\vec{x}$ and y are normalized and different components of $\vec{x}$ are iid. The color map represents the minimal communication complexity of the three algorithms in the logarithmic scale. Black lines divide the space into three regions denoted as Classical (D), Classical (S), and Quantum, representing the region where the deterministic classical, stochastic classical, and QBC algorithm have the smallest communication complexity. The black dashed line in each layer indicates the statistical variance $\epsilon = 1/\sqrt{N}$.

an advantage compared with classical algorithms (Fig. 2). In general, the QBC algorithm starts showing an advantage when $N \geq 10^3 \sim 10^4$, which is a reasonable range in fitting problems. The quantum advantage requires ϵ to be in an intermediate level: too-small or too-large ϵ make deterministic or stochastic classical algorithms have a lower communication complexity.

The quality of a fitted model can be characterized by the mean square error $E \equiv (1/N)(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\lambda}})^2 = (1/N)(\boldsymbol{y}^2 + \hat{\boldsymbol{y}}^2 - 2\boldsymbol{y}^T\hat{\boldsymbol{y}})$. Only the calculation of $(1/N)\boldsymbol{y}^T\hat{\boldsymbol{y}}$ involves communication, which can again be realized through the correlation estimation scheme, requiring $O(\log_2(N)/\epsilon)$-qubit communication.

The applications of the QBC algorithm are not restricted to fitting linear functions, as a general function of $\vec{x}$ can be expanded as a linear combination of a series of basis functions $y = \sum_j \lambda_j f_j(\vec{x})$. The matrix $F_{ij} \equiv f_j(\vec{x}_i)$ can be computed locally, and the problem is then reduced to the linear fitting problem $\boldsymbol{F}\boldsymbol{\lambda} = \boldsymbol{y}$. Furthermore, this scheme can be used as the linear output layer of a neural network in high-expressivity machine learning models [22].

Softmax classifier.—Besides fitting continuous data, the QBC scheme can also be used for fitting discrete labels (classification). A common output layer of classification models is the softmax classifier. The basic scenario is that the data of Bob y_i has discrete possible values in a set of classes $Y = \{c_1, c_2, \ldots, c_q\}$. The model outputs the probabilities for a given data point $\vec{x}$ to be in each class $P(y = c_j|\boldsymbol{x}, \boldsymbol{\Lambda})$ with ansatz $P(y = c_j|\boldsymbol{x}, \boldsymbol{\Lambda}) = (e^{\boldsymbol{\lambda}_j^T\boldsymbol{x}}/\sum_l e^{\boldsymbol{\lambda}_l^T\boldsymbol{x}})$, where the coefficient matrix is $\boldsymbol{\Lambda} \equiv (\boldsymbol{\lambda}_0, \ldots, \boldsymbol{\lambda}_q)$. The cross-entropy loss function $L(\boldsymbol{\Lambda}) \equiv -\sum_{ij} 1_{y_i=c_j} \log_2 P(y_i = c_j|\boldsymbol{x}_i, \boldsymbol{\Lambda})$ is to be minimized, where $1_{y=c_j}$ is a 1 when $y = c_j$ and 0 otherwise. $\hat{\boldsymbol{\lambda}}$ can be obtained from a set of equations:

$$\sum_{i=1}^{N} \frac{\boldsymbol{x}_i e^{\hat{\boldsymbol{\lambda}}_j^T \boldsymbol{x}_i}}{\sum_{k=1}^{q} e^{\hat{\boldsymbol{\lambda}}_k^T \boldsymbol{x}_i}} = \sum_{i=1}^{N} 1_{y_i=c_j}\boldsymbol{x}_i, \qquad j = 1, 2, \ldots, q. \tag{8}$$

The equation's right-hand side can be estimated as the inner product between $1_{y=c_j}$ and the vector $\boldsymbol{x}$ following our previous scheme, with communication complexity $\mathcal{C} = O(qM\log_2(N)/\epsilon)$ (see the Supplemental Material [40], Sec. IV, for details). As the left-hand side of the equations does not involve y, the equations can be solved without any further communication. We note that logistic regression for the two-class classification problems can be derived as a special case of the softmax regression scheme with $q = 2$.

We can further quantify the communication complexity of evaluating the quality of a fitted classifier. The quality can be determined by comparing the model outputs $\hat{y}_i = \mathrm{argmax}_{c_j} P(y_i = c_j|x_i, \boldsymbol{\Lambda})$ and labels y_i on the training or testing dataset. Alice and Bob encode $\hat{y}_i$ and y_i into Nq-bit strings $\hat{b}_{ij} \equiv 1_{\hat{y}_i=c_j}$ and $b_{ij} \equiv 1_{y_i=c_j}$, respectively. Then the correctness of the model can be determined by estimating the Hamming distance d between $\hat{b}$ and b as $1 - (d/2N)$ (as each error in classification contributes a two-bit difference). The communication complexity is $\mathcal{C} = O(\log_2(Nq)/\epsilon)$, showing no dependence on dimension M and insensitive dependence on the number of classes q.

Conclusion and outlook.—In this work, we developed a distributed Grover-quantum counting-based scheme that performs distributed least-square fitting or softmax regression with a communication complexity $O(\log_2(N)/\epsilon)$, a square-root improvement over classical algorithms. The quantum advantage comes from reduced communication requirements by encoding information in the phases of a superposition state, a unique attribute of quantum systems. Some previous quantum schemes [18,29,32] encode the information in the weight of superposition: as extracting the superposition weight by state tomography also requires $O(1/\epsilon^2)$ repetitions of state preparation and measurements, these methods do not show significant advantage in deriving classical fitting parameters compared with classical schemes. The core of our algorithm, a communication-efficient "quantum bipartite correlator," is expected to be useful in other communication and information-processing contexts as well. This method is expected to preserve privacy between two parties. Neither Alice nor Bob can determine the other party's attributes of a specific data point, as only the statistical average is encoded in the phase during communication. This meets the security requirement of distributed computing [47].

We thank Prof. Isaac Chuang for insightful comments. This work was supported by NSF CMMI-1922206 and DTRA (Grant No. HDTRA1-20-2-0002) Interaction of Ionizing Radiation with Matter (IIRM) University Research Alliance (URA). The calculations in this work were performed in part on the Texas Advanced Computing Center (TACC) and MIT engaging cluster.

[*] These authors contributed equally.

[†] pcappell@mit.edu

[‡] liju@mit.edu

[1] M. Gheisari, G. Wang, and M. Z. A. Bhuiyan, in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)* (IEEE, Guangzhou, 2017), Vol. 2, pp. 173–180.

[2] L. Bottou and O. Bousquet, Adv. Neural Inf. Process. Syst. **20**, 161 (2007).

[3] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, ACM Comput. Surv. (CSUR) **53**, 1 (2020).

[4] D. Peteiro-Barral and B. Guijarro-Berdiñas, Prog. Artif. Intell. **2**, 1 (2013).

[5] J. Erickson, *Database Technologies: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications* (IGI Global, Hershey PA, 2009).

[6] S. E. Haupt and B. Kosovic, in *2015 IEEE Symposium Series on Computational Intelligence* (IEEE, Cape Town, 2015), pp. 496–501.

[7] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, IEEE Trans. Inf. Theory **64**, 109 (2017).

[8] H. Abelson, J. Appl. Comput. Mech. **27**, 384 (1980).

[9] A. C.-C. Yao, in *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, 1979), pp. 209–213.

[10] E. Kushilevitz, in *Advances in Computers* (Elsevier, New York, 1997), Vol. 44, pp. 331–360.

[11] A. Rao and A. Yehudayoff, *Communication Complexity: And Applications* (Cambridge University Press, Cambridge, England, 2020).

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

[13] D. Martínez, A. Tavakoli, M. Casanova, G. Canas, B. Marques, and G. Lima, Phys. Rev. Lett. **121**, 150504 (2018).

[14] G. Brassard, Found. Phys. **33**, 1593 (2003).

[15] H. Buhrman, R. Cleve, and A. Wigderson, in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, 1998), pp. 63–68.

[16] H. Buhrman, R. Cleve, S. Massar, and R. de Wolf, Rev. Mod. Phys. **82**, 665 (2010).

[17] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature (London) **549**, 195 (2017).

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

[19] G. H. Low, T. J. Yoder, and I. L. Chuang, Phys. Rev. A **89**, 062315 (2014).

[20] S. Lloyd, M. Mohseni, and P. Rebentrost, Nat. Phys. **10**, 631 (2014).

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

[22] Y. LeCun, Y. Bengio, and G. Hinton, Nature (London) **521**, 436 (2015).

[23] U. Hadar, J. Liu, Y. Polyanskiy, and O. Shayevitz, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery, New York, 2019), pp. 792–803.

[24] D. A. Freedman, *Statistical Models: Theory and Practice* (Cambridge University Press, Cambridge, England, 2009).

[25] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis* (Cengage Learning, Boston MA, 2015).

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

[27] D.-B. Zhang, Z.-Y. Xue, S.-L. Zhu, and Z. D. Wang, Phys. Rev. A **99**, 012331 (2019).

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

[29] G. Wang, Phys. Rev. A **96**, 012335 (2017).

[30] G. Brassard, P. HØyer, and A. Tapp, in *Automata, Languages and Programming*, edited by K. G. Larsen, S. Skyum, and G. Winskel (Springer Berlin Heidelberg, Berlin, Heidelberg, 1998), pp. 820–831.

[31] M. A. Nielsen and I. L. Chuang, Phys. Today **54**, No. 11, 60 (2001).

[32] M. Fanizza, M. Rosati, M. Skotiniotis, J. Calsamiglia, and V. Giovannetti, Phys. Rev. Lett. **124**, 060503 (2020).

[33] A. Anshu, Z. Landau, and Y. Liu, in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery, New York, 2022), pp. 44–51.

[34] A. Chakrabarti and O. Regev, SIAM J. Comput. **41**, 1299 (2012).

[35] H.-Y. Huang, R. Kueng, and J. Preskill, Nat. Phys. **16**, 1050 (2020).

[36] N. Wiebe, D. W. Berry, P. Høyer, and B. C. Sanders, J. Phys. A **44**, 445308 (2011).

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

[38] X.-M. Zhang, M.-H. Yung, and X. Yuan, Phys. Rev. Res. **3**, 043200 (2021).

[39] J. A. Cortese and T. M. Braje, arXiv:1803.01958.

[40] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.130.150602 for details of the algorithm and complexities' derivations.

[41] H. Abdi and L. J. Williams, Wiley Interdiscip. Rev.: Syst. Biol. Med. **2**, 433 (2010).

[42] P. Indyk and D. Woodruff, in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings* (IEEE, New York, 2003), pp. 283–288.

[43] A. A. Sherstov, Theor. Comput. Sci. **8**, 197 (2012).

[44] A. Akella, A. Bharambe, M. Reiter, and S. Seshan, in *Proceedings of the Workshop on Management and Processing of Data Streams* (Citeseer, New York, 2003).

[45] A. Chakrabarti, G. Cormode, and A. McGregor, ACM Trans. Algorithms (TALG) **6**, 1 (2010).

[46] IEEE Standard for Floating-Point Arithmetic, in IEEE Std 754-2019 (Revision of IEEE 754-2008) (2019), pp. 1-84, 10.1109/IEEESTD.2019.8766229.

[47] M. Al-Rubaie and J. M. Chang, IEEE Secur. Privacy **17**, 49 (2019).