Quantum circuit learning

K. Mitarai,[1,*] M. Negoro,[1,2] M. Kitagawa,[1,3] and K. Fujii[2,4,†]

[1]*Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan*

[2]*JST, PRESTO, 4-1-8 Honcho, Kawaguchi, Saitama 332-0012, Japan*

[3]*Quantum Information and Quantum Biology Division, Institute for Open and Transdisciplinary Research Initiatives, Osaka University, Osaka 560-8531, Japan*

[4]*Graduate School of Science, Kyoto University, Yoshida-Ushinomiya-cho, Sakyo-ku, Kyoto 606-8302, Japan*

(Received 12 January 2018; published 10 September 2018)

We propose a classical-quantum hybrid algorithm for machine learning on near-term quantum processors, which we call quantum circuit learning. A quantum circuit driven by our framework learns a given task by tuning parameters implemented on it. The iterative optimization of the parameters allows us to circumvent the high-depth circuit. Theoretical investigation shows that a quantum circuit can approximate nonlinear functions, which is further confirmed by numerical simulations. Hybridizing a low-depth quantum circuit and a classical computer for machine learning, the proposed framework paves the way toward applications of near-term quantum devices for quantum machine learning.

DOI: 10.1103/PhysRevA.98.032309

I. INTRODUCTION

In recent years, machine learning has acquired much attention in a wide range of areas including the field of quantum physics [1–5]. Since quantum information processing is expected to bring us exponential speedups on some problems [6,7], the usual machine learning tasks might be improved when carried out on a quantum computer. Also, for the purpose of learning a complex quantum system, it is natural to utilize a quantum system as our computational resource. A variety of machine learning algorithms for quantum computers have been proposed [8–11], since the Harrow-Hassidim-Lloyd (HHL) algorithm [12] enabled us to perform basic matrix operations on a quantum computer. These HHL-based algorithms have the quantum phase estimation algorithm [7] at their heart, which requires a high-depth quantum circuit. To circumvent a high-depth quantum circuit, which is still a long-term goal on the hardware side, classical-quantum hybrid algorithms consisting of a relatively low-depth quantum circuit such as the quantum variational eigensolver [13,14] (QVE) and quantum approximate optimization algorithm [15–17] (QAOA) have been suggested. In these methods, a problem is encoded in an Hermitian matrix A. Its expectation value $\langle A\rangle$ with respect to an ansatz state $|\psi(\boldsymbol{\theta})\rangle$ is iteratively optimized by tuning the parameter $\boldsymbol{\theta}$. The central idea of hybrid algorithms is dividing the problem into two parts, each of which can be performed easily on a classical and a quantum computer.

In this paper, we present a new hybrid framework, which we call quantum circuit learning (QCL), for machine learning with a low-depth quantum circuit. In QCL, we provide input data to a quantum circuit and iteratively tune the circuit parameters so that the optimized circuit gives the desired output. A gradient-based systematic optimization of parameters is introduced for the tuning just like the backpropagation method [18] utilized in feedforward neural networks. We theoretically show that a quantum circuit driven by the QCL framework can approximate any analytical function, if the circuit has a sufficient number of qubits. The ability of the QCL framework to learn nonlinear functions and perform a simple classification task is demonstrated by numerical simulations. Also, we show by simulation that a six-qubit circuit is capable of fitting the dynamics of 3 spins of a 10-spin system with a fully connected Ising Hamiltonian. We stress here that the proposed framework is easily realizable on near-term devices.

II. QUANTUM CIRCUIT LEARNING

A. Algorithm

Our QCL framework aims to perform supervised or unsupervised learning tasks [18]. In supervised learning, an algorithm is provided with a set of input $\{\boldsymbol{x}_i\}$ and corresponding teacher data $\{f(\boldsymbol{x}_i)\}$. The algorithm learns to output $y_i = y(\boldsymbol{x}_i, \boldsymbol{\theta})$ that is close to the teacher $f(\boldsymbol{x}_i)$, by tuning $\boldsymbol{\theta}$. The output and the teacher can be vector-valued. QCL assigns the calculation of the output y_i to a quantum circuit and the update of the parameter $\boldsymbol{\theta}$ to a classical computer. The objective of learning is to minimize a cost function, which is a measure of how close the teacher and the output are, by tuning $\boldsymbol{\theta}$. As an example, the quadratic cost $L = \sum_i \|f(\boldsymbol{x}_i) - y_i\|^2$ is often used in regression problems. On the other hand, in unsupervised learning (e.g., clustering), only input data are provided, and some objective cost function that does not involve the teacher is minimized.

Here we summarize the QCL algorithm on an N-qubit circuit:

(1) Encode input data $\{\boldsymbol{x}_i\}$ into some quantum state $|\psi_{\text{in}}(\boldsymbol{x}_i)\rangle$ by applying a unitary input gate $U(\boldsymbol{x}_i)$ to initialized qubits $|0\rangle$.

*mitarai@qc.ee.es.osaka-u.ac.jp

†fujii.keisuke.2s@kyoto-u.ac.jp

(2) Apply a $\boldsymbol{\theta}$-parameterized unitary $U(\boldsymbol{\theta})$ to the input state and generate an output state $|\psi_{\rm out}(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})\,|\psi_{\rm in}(\boldsymbol{x}_i)\rangle$.

(3) Measure the expectation values of some chosen observables. Specifically, we use a subset of Pauli operators $\{B_j\} \subset \{I, X, Y, Z\}^{\otimes N}$. Using some output function F, output $y_i = y(\boldsymbol{x}_i, \boldsymbol{\theta})$ is defined to be $y(\boldsymbol{x}_i, \boldsymbol{\theta}) \equiv F(\{\langle B_j(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle\})$.

(4) Minimize the cost function $L(f(\boldsymbol{x}_i), y(\boldsymbol{x}_i, \boldsymbol{\theta}))$ of the teacher $f(\boldsymbol{x}_i)$ and the output y_i, by tuning the circuit parameters $\boldsymbol{\theta}$ iteratively.

(5) Evaluate the performance by checking the cost function with respect to a data set that is taken independently from the training one.

B. Relation with the existing algorithms

Minimization of the quadratic cost can be performed using a high-depth quantum circuit with HHL-based algorithms. For example, Ref. [19] shows a detailed procedure. This matrix inversion approach is similar to the quantum support vector machine [10]. As opposed to this, QCL applied to a regression problem minimizes the cost by iterative optimization, successfully circumventing a high-depth circuit.

Quantum reservoir computing (QRC) [20] shares a similar idea, in the sense that it passes the central optimization procedure to a classical computer. There, output is defined to be $y(\boldsymbol{x}_i) \equiv \boldsymbol{w} \cdot \langle \boldsymbol{B}\rangle$, where $\boldsymbol{B}$ is a set of observables taken from quantum many-body dynamics driven with a fixed Hamiltonian, and $\boldsymbol{w}$ is the weight vector, which is tuned on a classical device to minimize the cost function. The idea stems from a so-called echo-state network approach [21]. If one views QRC as a quantum version of the echo-state network, QCL, which tunes the whole network, can be regarded as a quantum counterpart of a basic neural network. In the QVE/QAOA, the famous hybrid quantum algorithms, the weighted sum of measured expectation values $\boldsymbol{w}_{\rm fixed} \cdot \langle \boldsymbol{B}(\boldsymbol{\theta})\rangle$ is minimized by tuning the parameter $\boldsymbol{\theta}$. There, an input $\boldsymbol{x}$ of a problem, such as the geometry of a molecule or topology of a graph, is encoded to the weight vector $\boldsymbol{w}_{\rm fixed}$ as $\boldsymbol{w}_{\rm fixed}(\boldsymbol{x})$. This procedure corresponds to a special case of QCL where we do not use the input unitary $U(\boldsymbol{x})$, and a cost function $L = \boldsymbol{w}_{\rm fixed} \cdot \langle \boldsymbol{B}\rangle$ is utilized. Figure 1 summarizes a comparison of the QVE/QAOA, QRC, and presented QCL frameworks.

C. Ability to approximate a function

First, we consider the case where input data are one-dimensional, for simplicity. It is straightforward to generalize the following argument for higher-dimensional inputs.

Let x and $\rho_{\rm in}(x) = |\psi_{\rm in}(x)\rangle\langle\psi_{\rm in}(x)|$ be an input datum and a corresponding density operator of the input state. $\rho_{\rm in}(x)$ can be expanded by a set of Pauli operators $\{P_k\} = \{I, X, Y, Z\}^{\otimes N}$ with $a_k(x)$ as coefficients, $\rho_{\rm in}(x) = \sum_k a_k(x) P_k$. A parametrized unitary transformation $U(\boldsymbol{\theta})$ acting on $\rho_{\rm in}(x)$ creates the output state, which can also be expanded by $\{P_k\}$ with $\{b_k(x, \boldsymbol{\theta})\}$. Now let $u_{ij}(\boldsymbol{\theta})$ be such that $b_m(x, \boldsymbol{\theta}) = \sum_k u_{mk}(\boldsymbol{\theta}) a_k(x)$. b_m is the expectation value of the Pauli observable itself, and therefore, the output is a linear combination of input coefficient functions a_k under unitarity constraints imposed on $\{u_{ij}\}$.

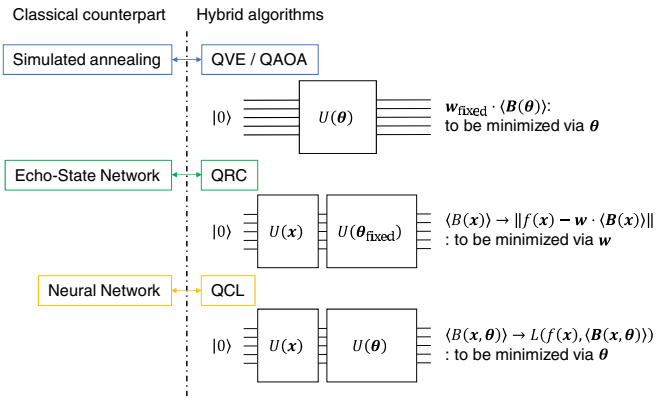

FIG. 1. Comparison of the QVE/QAOA, QRC, and presented QCL frameworks. In the QVE, the output of the quantum circuit is directly minimized. The QRC and QCL both optimize the output to the teacher $f(\boldsymbol{x})$. QRC optimization is done by tuning the linear weight $\boldsymbol{w}$, as opposed to the QCL approach, which tunes the circuit parameter $\boldsymbol{\theta}$.

When the teacher $f(x)$ is an analytical function, we can show, at least in principle, that QCL is able to approximate it by considering a simple case with an input state created by single-qubit rotations. The tensor product structure of the quantum system plays an important role in this analysis. Let us consider a state of N qubits:

$$\rho_{\rm in}(x) = \frac{1}{2^N} \bigotimes_{i=1}^{N} [I + x X_i + \sqrt{1 - x^2} Z_i]. \tag{1}$$

This state can be generated for any $x \in [-1, 1]$ with single-qubit rotations, namely, $\prod_{i=1}^{N} R_i^Y(\sin^{-1} x)$, where $R_i^Y(\phi)$ is the rotation of the ith qubit around the y axis with angle ϕ. The state given by Eq. (1) has higher-order terms up to the Nth with respect to x. Thus an arbitrary unitary transformation on this state can provide us with an arbitrary Nth-order polynomial as the expectation value of an observable. Terms like $x\sqrt{1 - x^2}$ in Eq. (1) can enhance its ability to approximate a function.

An important note regarding the example given above is that the highest-order term x^N is hidden in an observable $X^{\otimes N}$. To extract x^N from Eq. (1), one needs to transfer the nonlocal observable $X^{\otimes N}$ to a single-qubit observable using an entangling gate such as the controlled-NOT gate. Entangling nonlocal operations are the key ingredients of nonlinearity of an output.

The above argument can readily be generalized to multidimensional inputs. Assume that we are given, with d-dimensional data, $\boldsymbol{x} = \{x_1, x_2, .., x_d\}$ and want higher terms up to the n_kth $(k = 1, \ldots, d)$ for each datum, then encode this datum in an $N = \sum_k n_k$-qubit quantum state as $\rho_{\rm in}(\boldsymbol{x}) = \frac{1}{2^N} \bigotimes_{k=1}^{d} (\bigotimes_{i=1}^{n_k} [I + x_k X_i + \sqrt{1 - x_k^2} Z_i])$. These input states automatically have an exponentially large number of independent functions as coefficients set to the number of qubits. The tensor product structure of the quantum system readily "calculates" a product such as $x_1 x_2$.

The unitarity condition of u_{ij} may have the effect of eliminating the overfitting problem, which is crucial for its performance in machine learning or in regression methods.

One way to handle it in classical machine learning methods is to add a regularization term to the cost function. For example, ridge regression adds the regularization term $\|\boldsymbol{w}\|^2$ to the quadratic cost function. Overall, $L = \sum_i \|f(\boldsymbol{x}_i) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i)\|^2 + \|\boldsymbol{w}\|^2$ is minimized. The weight vector $\boldsymbol{w}$ corresponds to the matrix element u_{ij} in QCL. The norm of the row vector $\|\boldsymbol{u}_i\|$, however, is restricted to unity by the unitarity condition, which prevents overfitting, from the unitarity of quantum dynamics. Simple examples of this are given in the Appendix.

D. Possible quantum advantages

We have shown in the above discussion that approximation of any analytical function is possible with the use of nonlinearity created by the tensor product. In fact, nonlinear basis functions are crucial for many methods utilized in classical machine learning. They require a large number of basis functions to create a complex model that makes predictions with a high precision. However, the computational cost of learning increases with respect to an increasing number of basis functions. To avoid this problem, the so-called kernel-trick method, which circumvents the direct use of a large number of them, is utilized [18]. In contrast, QCL directly utilizes the exponential number of functions with respect to the number of qubits to model the teacher, which is basically intractable on classical computers. This is a possible quantum advantage of our framework, which was not obvious from previous approaches like the QVE and QAOA.

Moreover, let us now argue about the potential power of QCL representing complex functions. Suppose we want to learn the output of QCL that is allowed to use unlimited resources in the learning process, via classical neural networks. Then it has to learn the relation between inputs and outputs of a quantum circuit, which, in general, includes universal quantum cellular automata [22,23]. This certainly could not be achieved using a polynomial-sized classical computational resource to the number of qubits used for QCL. This implies that QCL has the potential power to represent more complex functions than the classical counterpart. Further investigations are needed, including of the learning costs and which actual learning problem enjoys such an advantage.

E. Optimization procedure

In the QVE [13], it has been suggested to use gradient-free methods like Nelder-Mead. However, gradient-based methods are generally more preferred when the parameter space becomes large. In neural networks, the backpropagation method [18], which is basically gradient descent, is utilized in the learning procedure.

To calculate the gradient of the expectation value of an observable with respect to a circuit parameter $\boldsymbol{\theta}$, suppose the unitary $U(\boldsymbol{\theta})$ consists of a chain of unitary transformations $\prod_{j=1}^{l} U_j(\theta_j)$ on a state ρ_{in} and we measure an observable B. For convenience, we use the notation $U_{j:k} = U_j \dots U_k$. Then $\langle B(\boldsymbol{\theta})\rangle$ is given as $\langle B(\boldsymbol{\theta})\rangle = \text{Tr}(BU_{l:1}\rho_{\text{in}}U_{l:1}^\dagger)$. We assume that U_j is generated by a Pauli product P_j, that is, $U_j(\theta) = \exp(-i\theta_j P_j/2)$. The gradient is calculated to be $\frac{\partial\langle B\rangle}{\partial\theta_j} = -\frac{i}{2}\text{Tr}(BU_{l:j}[P_j, U_{j-1:1}\rho_{\text{in}}U_{j-1:1}^\dagger]U_{l:j}^\dagger)$. While we cannot evaluate the commutator directly, the following property of the commutator for an arbitrary operator ρ enables us to compute the gradient on a quantum circuit:

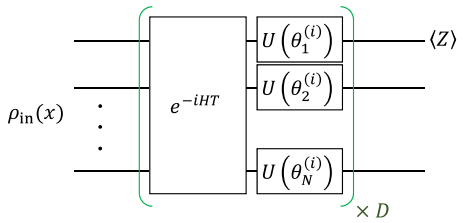

FIG. 2. Quantum circuit used in numerical simulations. The parameters $\boldsymbol{\theta}$ of single-qubit arbitrary unitaries $U(\theta_j^{(i)})$ are optimized to minimize the cost function. D denotes the depth of the circuit.

$$[P_j, \rho] = i\left[U_j\left(\frac{\pi}{2}\right)\rho U_j^\dagger\left(\frac{\pi}{2}\right) - U_j\left(-\frac{\pi}{2}\right)\rho U_j^\dagger\left(-\frac{\pi}{2}\right)\right]. \tag{2}$$

The gradient can be evaluated by

$$\frac{\partial\langle B\rangle}{\partial\theta_j} = \frac{1}{2}\text{Tr}\left[BU_{l:j+1}U_j\left(\frac{\pi}{2}\right)\rho_j U_j^\dagger\left(\frac{\pi}{2}\right)U_{l:j+1}^\dagger\right] - \frac{1}{2}\text{Tr}\left[BU_{l:j+1}U_j\left(-\frac{\pi}{2}\right)\rho_j U_j^\dagger\left(-\frac{\pi}{2}\right)U_{l:j+1}^\dagger\right], \tag{3}$$

where $\rho_j = U_{j:1}\rho_{\text{in}}U_{j:1}^\dagger$. Just by inserting $\pm\pi/2$ rotation generated by P_j and measuring the respective expectation values $\langle B\rangle_j^\pm$, we can evaluate the exact gradient of an observable $\langle B\rangle$, via $\frac{\partial\langle B\rangle}{\partial\theta_j} = \frac{\langle B\rangle_j^+ - \langle B\rangle_j^-}{2}$. A similar method is used by Li *et al.* [24] in their research on control pulse optimization with a target quantum system.

III. NUMERICAL SIMULATIONS

We demonstrate the performance of the QCL framework on several prototypical machine learning tasks by numerically simulating a quantum circuit in the form of Fig. 2 with $N = 6$ and $D = 6$. $U(\theta_j^{(i)})$ in Fig. 2 is an arbitrary rotation of a single qubit. We use the decomposition $U(\theta_j^{(i)}) = R_j^X(\theta_{j1}^{(i)})R_j^Z(\theta_{j2}^{(i)})R_j^X(\theta_{j3}^{(i)})$. H is the Hamiltonian of a fully connected transverse Ising model:

$$H = \sum_{j=1}^{N} a_j X_j + \sum_{j=1}^{N}\sum_{k=1}^{j-1} J_{jk} Z_j Z_k. \tag{4}$$

The coefficients a_j and J_{jk} are taken randomly from a uniform distribution on $[-1, 1]$. The evolution time T is fixed to 10. The results shown throughout this section are generated by the Hamiltonian with the same coefficients. Here we note that we have checked that similar results can be achieved with different Hamiltonians. The dynamics under this form of Hamiltonian can generate a highly entangled state and is, in general for a large number of qubits, not efficiently simulatable on a classical computer. Equation (4) is the basic form of interaction in trapped ions or superconducting qubits,

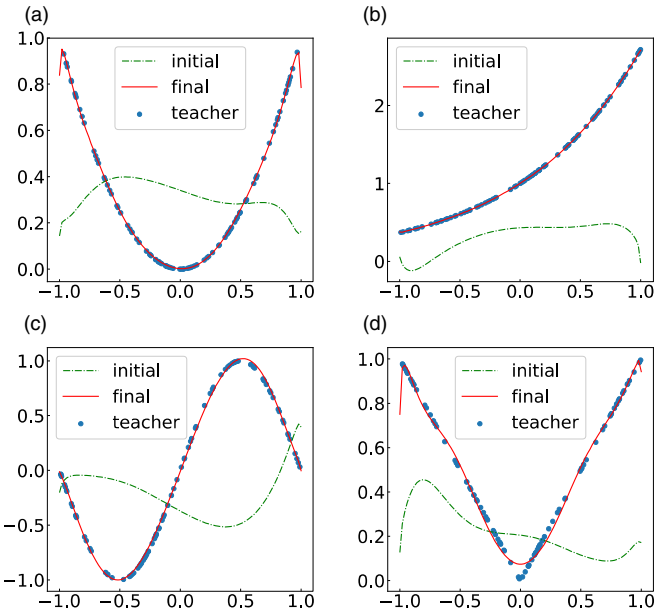

FIG. 3. Demonstration of QCL performance to represent functions. "Initial" indicates the output of the quantum circuit with randomly chosen θ; "final," the output from the optimized quantum circuit. Fitting of (a) x^2, (b) e^x, (c) $\sin x$, and (d) $|x|$.

which makes the time evolution easily implementable experimentally. $\boldsymbol{\theta}$ is initialized with random numbers uniformly distributed on $[0, 2\pi]$. In all numerical simulations, outputs are taken from Z expectation values. To emulate a sampling, we added small Gaussian noise with standard deviation σ determined by $\sigma = \sqrt{2/N_s}(\langle Z\rangle^2 - 1)/4$, where N_s and $\langle Z\rangle$ are the number of samples and a calculated expectation value, to $\langle Z\rangle$ [25].

First, we perform the fitting of $f(x) = x^2, e^x, \sin x, |x|$ as a demonstration of the representability of nonlinear functions [18]. We use the normal quadratic loss for the cost function. The number of teacher samples is 100. The output is taken from the Z expectation value of the first qubit as shown in Fig. 2. In this simulation, we allow the output to be multiplied by a constant a which is initialized to unity. This constant a and $\boldsymbol{\theta}$ are simultaneously optimized. The input state $\rho_{\text{in}}(x)$ is prepared by applying $U_{\text{in}}(x) = \prod_j R_j^Z(\cos^{-1} x^2) R_j^Y(\sin^{-1} x)$ to initialized qubits $|0\rangle$. This unitary creates a state similar to Eq. (1).

Results are shown in Fig. 3. All of the functions are well approximated by a quantum circuit driven by the presented QCL framework. To approximate highly nonlinear functions such as $\sin x$ or a nonanalytical function $|x|$, QCL has brought out the high-order terms which are initially hidden in nonlocal operators. The result of fitting $|x|$ [Fig. 3(d)] is relatively poor because of its nonanalytical characteristics. A possible solution for this is to employ different functions as the input functions, such as Legendre polynomials. Although the choice of input functions affects the performance of QCL, the result shows that QCL with simple input has the ability to output a wide variety of functions.

As the second demonstration, the classification problem, which is an important family of tasks in machine learning,

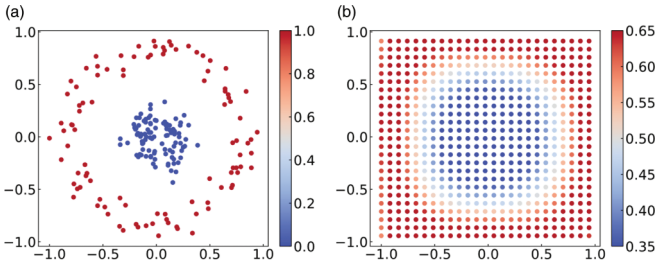

FIG. 4. Demonstration of a simple nonlinear classification task. (a) Teacher data. Blue dots indicate class 0; red dots, class 1. (b) Optimized output from the first qubit (after softmax transformation). The threshold for classification is 0.5; less than 0.5 and greater than 0.5 indicate that the point is classified as class 0 and class 1, respectively.

is performed. Figure 4(a) shows the training data set; blue and red points indicate classes 0 and 1, respectively. Here we train the quantum circuit to classify based on each training input data points $\boldsymbol{x}_i = (x_{i,0}, x_{i,1})$. We define the teacher $f(\boldsymbol{x}_i)$ for each input $\boldsymbol{x}_i$ to be the two-dimensional vector $(1, 0)$ for class 0 and $(0, 1)$ for class 1. The number of teacher samples is 200 (100 for class 0 and 100 for class 1). The output is taken from the expectation value of the Pauli Z operator of the first two qubits, and they are transformed by the softmax function $\boldsymbol{F}$. For the d-dimensional vector $\boldsymbol{q}$, the softmax function returns the d-dimensional vector $\boldsymbol{F}(\boldsymbol{q})$, with its kth element being $F_k(\boldsymbol{q}) = e^{q_k}/\sum_i e^{q_i}$. Thus the output $\boldsymbol{y}_i = (y_{i,0}, y_{i,1})$ is defined by $\boldsymbol{y}_i = \boldsymbol{F}(\langle Z_1(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle, \langle Z_2(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle)$. For the cost function, we use the cross-entropy $L = \sum_i \sum_{k\in\{0,1\}} (f(\boldsymbol{x}_i))_k \log y_{ik}$. The input state is prepared by applying $U_{\text{in}}(x) = \prod_j R_j^Z(\cos^{-1} x_{i,j \bmod 2}^2) R_j^Y(\sin^{-1} x_{i,j \bmod 2})$ to initialized qubits $|0\rangle$. $j \bmod 2$ is the remainder of j divided by 2. In this task, the multiplication constant a is fixed to unity.

Learned output is shown in Fig. 4(b). We see that QCL works as well for the nonlinear classification task. The same task can be classically performed using, for example, the kernel-trick support vector machine. The kernel-trick approach discards the direct use of a large number of basis functions with respect to the number of qubits, as opposed to the QCL approach, which utilizes an exponentially large number of basis functions under certain constraints. In this sense, QCL can benefit from the use of a quantum computer.

Finally, we demonstrate the ability of QCL to perform a fitting task of quantum many-body dynamics. Simulation of the dynamics of a 10-spin system under the fully connected transverse Ising Hamiltonian, Eq. (4), is performed in advance to generate teacher data. Coefficients a_j and J_{jk} are taken from a uniform distribution on $[-1, 1]$, independently of the coefficients of the Hamiltonian in the circuit. The dynamics started from the initialized state $|0\rangle^{\otimes 10}$. The transient at the beginning of evolution is discarded for the duration $T_{\text{transient}} = 300$. For practical use, one can employ the dynamics obtained experimentally from a quantum system with an unknown Hamiltonian as teacher data. The learned dynamics is of Z expectation values of three spins during $t \in [T_{\text{transient}}, T_{\text{transient}} + 8]$. This span of t is mapped on $x \in [-1, 1]$ uniformly by $t = 4(x + 1) + T_{\text{transient}}$ to be properly

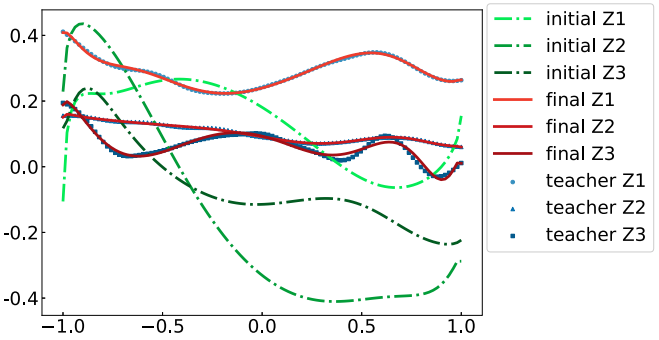

FIG. 5. Demonstration of fitting quantum many-body dynamics. The partial dynamics of the 10-spin system can be well approximated by a six-qubit circuit.

introduced to the input gate. Outputs are taken from the Z expectation values of the first, second, and third qubits in the circuit. The quadratic cost function is employed. The number of teacher samples is 100 for each. The multiplication constant a is fixed to unity.

The result is shown in Fig. 5. It is notable that the three observables of a complex 10-spin system can be well fitted, simultaneously, using the three observables of a tuned six-qubit circuit. Although the task performed here is not what is commonly referred to as a quantum simulation, we believe that we provide an alternative way to learn a quantum many-body dynamics with a near-term quantum computer. It may also be possible to extract partial information from the system Hamiltonian by taking the derivative of the output with respect to x, which can readily be performed using the same method for calculating a gradient.

IV. CONCLUSION

We have presented a machine learning framework on near-term realizable quantum computers. Our method fully employs the exponentially large space of a quantum system, by mixing simply injected nonlinear functions with a low-depth circuit to approximate a complex nonlinear function. Numerical results have shown the ability to represent a function, to classify, and to fit a relatively large quantum system. Also, theoretical investigation has shown QCL's ability to provide us a means for dealing with high-dimensional regression or classification tasks, which has been unpractical on classical computers.

Note added. Recently, we have become aware of related works [26–34].

ACKNOWLEDGMENTS

K.M. and M.N. are supported by JST PRESTO JPMJPR1666. K.M., M.N., and M.K. are supported by JST CREST JPMJCR1672. K.F. is supported by KAKENHI No. 16H02211, JST PRESTO JPMJPR1668, JST ERATO JPMJER1601, and JST CREST JPMJCR1673.

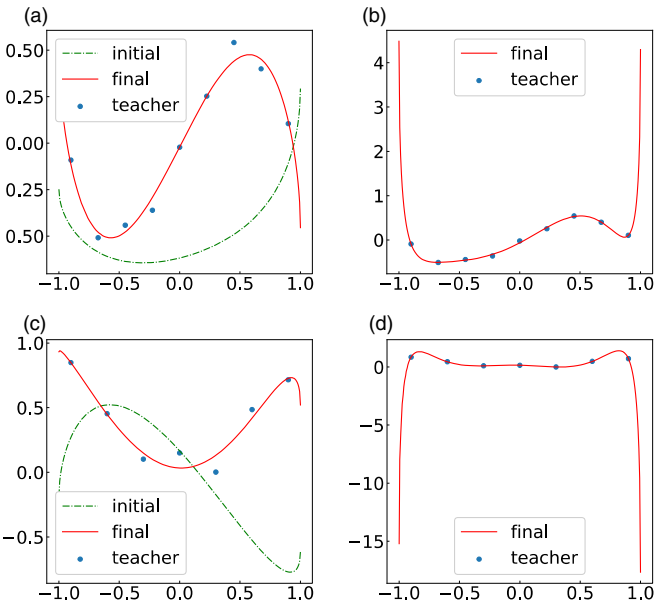

FIG. 6. A simple example of the avoidance of the overfitting resulting from unitarity. (a), (c) Fitting result of noise-added $\sin x$ and x^2 using QCL. (b), (d) Fitting result of noise-added $\sin x$ and x^2 using the classical regression with same basis functions as used in QCL.

APPENDIX: UNITARITY AVOIDS OVERFITTING

In this Appendix, we demonstrate a simple example that supports our claim in the text that the unitarity of the transformation has the effect of eliminating overfittings. We perform a one-dimensional fitting task with a small training data set to see the avoidance of overfitting. To observe the unitarity effect, we fix the multiplication constant a to unity. For simplicity, here we use a three-qubit circuit in the same form as in the text, with $D = 3$ and using $U_{\text{in}} = \prod_i R_i^Y(\sin x)$ as the input gate. In this case, the set of basis functions that QCL utilizes is $\{x, x^2, x^3, (1-x^2)^{1/2}, 1-x^2, (1-x^2)^{3/2}, x(1-x^2)^{1/2}, x^2(1-x^2)^{1/2}, x(1-x^2)\}$. Therefore for comparison, we run a simple classical linear regression program using the same basis function set.

Figures 6(a) and 6(b) show the result of the task of fitting data points of $0.5 \sin x$, with Gaussian noise of standard deviation 0.05 added, using QCL and classical regression, respectively. The result shows that, probably due to the unitarity of the transformation, QCL accepts some errors in the final output, as opposed to the classical regression, which does not accept any errors in the final output; that is, it overfits. As opposed to the $\|\boldsymbol{w}\| = 1$ constraint on QCL, the classical algorithm in this case output a weight vector with $\|\boldsymbol{w}\| \approx 134$. Figures 6(c) and 6(d) show the the result of the task to fit data points of x^2, with Gaussian noise of standard deviation 0.05 added, using QCL and classical regression, respectively. Again, the same observation can be made. The weight vector obtained by the classical algorithm exhibits $\|\boldsymbol{w}\| \approx 15\,800$ in this case.

[1] G. Carleo and M. Troyer, Science **355**, 602 (2017).
[2] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, Phys. Rev. Lett. **108**, 058301 (2012).
[3] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, Sci. Rep. **7**, 8823 (2017).
[4] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, J. Chem. Theory Comput. **11**, 2087 (2015).
[5] M. August and X. Ni, Phys. Rev. A **95**, 012335 (2017).
[6] P. W. Shor, SIAM J. Comput. **26**, 1484 (1997).
[7] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, UK, 2010).
[8] I. Kerenidis and A. Prakash, arXiv:1704.04992.
[9] N. Wiebe, D. Braun, and S. Lloyd, Phys. Rev. Lett. **109**, 050505 (2012).
[10] P. Rebentrost, M. Mohseni, and S. Lloyd, Phys. Rev. Lett. **113**, 130503 (2014).
[11] Y. Cao, G. G. Guerreschi, and A. Aspuru-Guzik, arXiv:1711.11240.
[12] A. W. Harrow, A. Hassidim, and S. Lloyd, Phys. Rev. Lett. **103**, 150502 (2009).
[13] A. Peruzzo, J. McClean, P. Shadbolt, M. Yung, X. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, Nat. Commun. **5**, 4213 (2014).
[14] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Nature **549**, 242 (2017).
[15] E. Farhi, J. Goldstone, and S. Gutmann, arXiv:1411.4028.
[16] E. Farhi and A. W. Harrow, arXiv:1602.07674.
[17] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, C. A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, R. S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, B. R. Johnson, M. Reagor, M. P. da Silva, and C. Rigetti, arXiv:1712.05771.
[18] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006).
[19] M. Schuld, I. Sinayskiy, and F. Petruccione, Phys. Rev. A **94**, 022342 (2016).
[20] K. Fujii and K. Nakajima, Phys. Rev. Appl. **8**, 024030 (2017).
[21] H. Jaeger and H. Haas, Science **304**, 78 (2004).
[22] R. Raussendorf, Phys. Rev. A **72**, 022301 (2005).
[23] D. Janzing and P. Wocjan, Quantum Info. Process. **4**, 129 (2005).
[24] J. Li, X. Yang, X. Peng, and C.-P. Sun, Phys. Rev. Lett. **118**, 150503 (2017).
[25] The simulation is carried out using the Python library QuTip [35]. We use the BFGS method [36] provided in the SciPy optimization library for optimization of parameters.
[26] L. Cincio, Y. Subasi, A. T. Sornborger, and P. J. Coles, arXiv:1803.04114.
[27] E. Farhi and H. Neven, arXiv:1802.06002.
[28] M. Benedetti, D. Garcia-Pintos, Y. Nam, and A. Perdomo-Ortiz, arXiv:1801.07686.
[29] M. Schuld and N. Killoran, arXiv:1803.07128.
[30] W. Huggins, P. Patel, K. B. Whaley, and E. M. Stoudenmire, arXiv:1803.11537.
[31] J.-G. Liu and L. Wang, arXiv:1804.04168.
[32] M. Schuld, A. Bocharov, K. Svore, and N. Wiebe, arXiv:1804.00633.
[33] M. Fanizza, A. Mari, and V. Giovannetti, arXiv:1805.03477.
[34] M. Benedetti, E. Grant, L. Wossnig, and S. Severini, arXiv:1806.00463.
[35] J. Johansson, P. Nation, and F. Nori, Comput. Phys. Commun. **184**, 1234 (2013).
[36] J. Nocedal and S. Wright, *Numerical Optimization* (Springer, New York, 2006).