Cost-function embedding and dataset encoding for machine learning with parametrized quantum circuits

Shuxiang Cao,[1,2,*] Leonard Wossnig,[3,2] Brian Vlastakis,[1,4] Peter Leek,[1,4] and Edward Grant[3,2]
[1]*Department of Physics, Clarendon Laboratory, University of Oxford, OX1 3PU, United Kingdom*
[2]*Rahko Limited, London N4 3JP, United Kingdom*
[3]*Department of Computer Science, University College London, London WC1E 6BT, United Kingdom*
[4]*Oxford Quantum Circuits Limited, Oxford OX2 6HT, United Kingdom*

(Received 13 December 2019; revised manuscript received 24 February 2020; accepted 11 March 2020; published 5 May 2020)

Machine learning is seen as a promising application of quantum computation. For near-term noisy intermediate-scale quantum devices, parametrized quantum circuits have been proposed as machine learning models due to their robustness and ease of implementation. However, the cost function is normally calculated classically from repeated measurement outcomes, such that it is no longer encoded in a quantum state. This prevents the value from being directly manipulated by a quantum computer. To solve this problem, we give a routine to embed the cost function for machine learning into a quantum circuit, which accepts a training dataset encoded in superposition or an easily preparable mixed state. We also demonstrate the ability to evaluate the gradient of the encoded cost function in a quantum state.

DOI: 10.1103/PhysRevA.101.052309

I. INTRODUCTION

Machine learning (ML) is one of the most successful research areas of the past decade and has a wide range of applications [1–3]. Meanwhile, quantum computing is the area of research that aims to design more efficient or generally more powerful methods using a new model of computation based on quantum mechanics [4]. Different methods to implement ML techniques on quantum computers have been proposed [5–10]. While the first stream of quantum ML algorithms exploited fast linear algebra subroutines to obtain a quantum speedup [11,12], recently, classical-quantum hybrid approaches to ML, so-called parametrized quantum circuits (PQCs) have experienced a surge of interest [13–18]. The main reason for their popularity is their suitability for noisy intermediate-scale quantum (NISQ) devices, hence their appeal to become a candidate for the first applications in quantum ML [10,13,19,20].

One exciting aspect of PQC-based ML is the equivalence to tensor-network-based ML algorithms [19,21]. Promising results indeed indicate that certain types of these algorithms can be executed efficiently on a quantum computer but cannot be efficiently evaluated classically, which implies that they have stronger expressive power [22] and a more flexible feature space [23,24]. Based on these results, there is hope that such quantum models will yield a practical advantage in ML.

Just as with classical deep learning, parameter training is the most time-consuming process of building a PQC model. Finding efficient ways to obtain optimal or near-optimal parameters is a major objective of current research. Several interesting methods have been proposed to optimize parameters. First, there are gradient-based optimization methods [25]; the partial derivative of the expectation value can be directly evaluated with a Hadamard test [26,27], or the *shift rule* can be used to do a Hadamard Test with indirect measurement [15,28]. Alternatively, a gradient-free optimization method called *rotosolve* can be used [29,30].

The target to be optimized (in ML, the *cost function*) is typically calculated from measured expectation values. Therefore, the value of the cost function is no longer encoded in the quantum state. This brings us some inconvenience. For example, the Hadamard Test and shift-rule methods can be used together with the chain rule to calculate the cost-function gradient [31], but this requires several function evaluations.[1] In addition, the gradient-free method *rotosolve*, which was developed for the specific value landscape of Hermitian expectation values, cannot be applied to classical nonlinear cost functions [29,30].

Therefore, an open question is whether we can find a method which embeds the cost function into a quantum circuit. Embedding the cost function into the quantum circuit directly will not only allow PQC-based ML models to be trained with the optimization techniques we mentioned above but also opens the possibility for performing further quantum operations after the cost function is evaluated, such

*shuxiang.cao@physics.ox.ac.uk

[1]Suppose our cost function is given by $f(x)$ and the measurement expectation value is given by $g(x)$, the derivative is then given by $f(g(x))' = f'(g(x))g'(x)$. To calculate this derivative, both the intermediate gradient $g(x)'$ and the expectation value $g(x)$ need to be evaluated.

as quantum enhanced measurement [32] and reduction of measurement times by phase estimation [33]. In this paper, we present a method which achieves the cost function embedding for PQC-based ML models.

We propose two cost functions that can be encoded directly through a quantum circuit, i.e., the outcome of the quantum circuit corresponds to the evaluation of the cost function for a given input. We then propose a corresponding data encoding method which allows the calculation of averaged cost function values among the encoded training dataset. Our approach could be used in combination with existing algorithms [13,14,17,18] by replacing the part of data encoding and cost function evaluating in these existing frameworks.

II. CLASSIFICATION WITH PARAMETRIZED QUANTUM CIRCUITS

Classification belongs to the category of supervised learning. In supervised learning we are given a data set $\mathcal{S} := \{x_i, y_i\}_{i=1}^m$ of m points, where $x_i \in \mathcal{X}$ are independent and identically distributed samples drawn from a fixed but unknown probability distribution and $y_i \in \mathcal{Y}$ are the corresponding labels, i.e., the ideal output for a given input x_i. In a classification task, the labels are obtained from a finite set of possible outcomes, for example, a binary outcome, i.e., $\mathcal{Y} = \{0, 1\}$. The goal of the classification task is then to find a function, the so-called model f, such that $f(x_i)$ returns a result that matches the label y_i for all $i \in [m]$ with the highest possible accuracy on a subset of $\mathcal{S}$, called the training set $\mathcal{S}_t$. Additionally we require that the function f also performs well on samples $x_j \notin \mathcal{S}_t$ which we have not used in order to train, i.e., find the model f, the so-called unseen data or test set $\mathcal{S}_p = \mathcal{S} \setminus \mathcal{S}_t$. A learning algorithm, or training algorithm $\mathcal{A}$, then takes as input the data set $\mathcal{S}$, and possibly additional hyperparameters, and returns a model $f = \mathcal{A}(\mathcal{S}, \cdot)$ which minimizes some specified measure of the discrepancy between the desired output and the model output. The measure of this discrepancy is called the cost function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, also called the objective of the learning problem, and is an integral part of the solution.

In the PQC setting the model $f(x_i)$ is given by a large unitary matrix $U \in \mathbb{C}^{2^n \times 2^n}$, $U^\dagger U = \mathbb{I}$ acting on an n-qubit system, and the input x_i is given by a state $|\psi_i\rangle$. A prediction of the model is then given by the expectation value for a measurement of one qubit after U is applied to some input state $|\psi_i\rangle \in \mathbb{C}^{2^n}$, see Fig. 1(a). Here we denote the qubits that store $|\psi_i\rangle$ as *data qubits*. U can be decomposed in a number of different ways into elementary logical elements, which are called quantum gates. One such decomposition then results in a sequence of single-qubit rotation gates and CNOT gates, which can then be parameterized by the rotation angles θ of the single-qubit gates [34,35]. After obtaining the gate sequence we arbitrarily choose a subset of the qubits as the *output qubits*. The state of the output qubits is given by density matrix ρ_{output}. Then we consider their measurement output as the prediction of the ansatz. For binary classification problems we choose only one of the qubits as the output and ignore the information stored in all others. Once we have the output of the measurement of $|\psi_i\rangle$, i.e., the prediction of our model, the cost function can be evaluated by measurement and applying some postprocessing in order to evaluate the discrepancy $\ell(f(|\psi_i\rangle), y_i)$. The training task then consists of adjusting the parameters θ of the model $U(\theta)$ in order to match the expectation value of a measurement of the output qubit to the corresponding label in the data set. Throughout this paper, we let this measurement be in the Z basis. We now move on to describe our proposal.

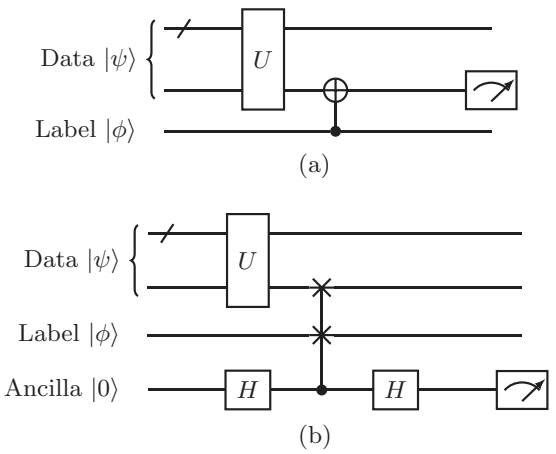

FIG. 1. Two ways to embeds cost functions into quantum circuits. The data qubits store the input data and the label qubit stores the label (expected output). U denotes an arbitrary ansatz, while we consider that the output from the ansatz is encoded in one qubit (here it is the qubit represented as the second line of the circuit). In panel (a) we apply a CNOT gate on the output qubit, and flip the state based on the label qubit. This will give us the CNOT cost function and encode the output into the Z axis expectation value of the output qubit. In panel (b) we add an ancillary qubit and apply a Fredkin gate between the ansatz output and the label, controlled by the ancillary qubit. This will encode the overlap between ansatz output and the label into the ancillary qubit.

III. METHODS

A. Cost function embedding

For the concrete implementation, we add another qubit $|\phi\rangle$, which holds the desired output for the supervised learning task, which we call the *label qubit*. For the binary classification problem, the label $\beta \in \{0, 1\}$ can be encoded as states $|0\rangle$ or $|1\rangle$ of the label qubit. A simple way to implement a cost function for binary classification is to use a CNOT gate to flip the output state based on the input training label; see Fig. 1(a). Notably, under this operation, the output qubit remains $|1\rangle$ if the prediction and the expectation value are unequal, and will be $|0\rangle$ if they are the same.

Suppose the expectation value of the measurement $\langle M \rangle = \alpha$, and recall that the label is $\beta \in \{0, 1\}$, then the CNOT cost function can be shown to be (see Appendix)

$$E_{\text{cost}(\psi)} = (1 - 2\beta)\alpha + \beta. \tag{1}$$

Note that this can be transformed into a well-studied loss function in ML, the so-called hinge loss [36].

In general, we want to treat continuous as well as binary outcomes, i.e., $\mathcal{Y} = \mathbb{R}$. In the quantum setting, this means that the ancillary qubit would no longer be in one of the states

$\{|0\rangle, |1\rangle\}$, and the above CNOT trick can no longer be applied. To handle this scenario, we can fall back to a well-known method that has been proposed for quantum classification: the swap test [37,38]. The swap test is a method to encode the overlap between two unknown states into an ancillary qubit. A swap test consists of a Hadamard gate followed by a controlled swap gate between two states and a final Hadamard gate. If the two states that we aim to compare, let them be $|\psi\rangle$ and $|\phi\rangle$, are the same, i.e., $\langle\psi|\phi\rangle = 1$, then the swap test acts as the identity. This means that the output state remains the same. However, if the two states are different, i.e., $\langle\psi|\phi\rangle \neq 1$, then the difference will be reflected in the phase of the control qubit. A swap test between the label state and the prediction state can hence be used to estimate the overlap. We can then use the overlap as the output of our cost function; see Fig. 1(b).

Suppose the input state is $|\psi\rangle$, then the Z axis expectation value of the measurement of the ancillary qubits can be written as [37,38]

$$E_{\mathrm{cost}(\psi)} = \frac{1 - \langle\phi|\,\rho_{\mathrm{output}}\,|\phi\rangle}{2}. \tag{2}$$

The controlled swap gate is also referred to as the quantum Fredkin gate, and several physical realizations using photonics or superconducting systems have been realized [39–41]. We hence have a single-qubit encoding of the cost function.

B. Data encoding

Now let us consider the data encoding for the cost functions discussed above. In classical deep learning, the cost function is evaluated individually for each sampled data point from the training data set. Then the cost function value is averaged across all calculated values. This trivially costs $O(N)$ repetitions of the cost function evaluation for N data points. Since we are only interested in the averaged cost function value instead of the individual value, it is natural to ask whether quantum parallelism can speed-up its evaluation. As we will now show in detail, we find interestingly that there is in fact no quantum speed-up that can be gained.

Consider a single data point. The goal of this procedure is to transfer each data point into a quantum state. For fully digitized input data, i.e., if all the input values are binary (for example, 0 or 1), the encoding step can be done by mapping the classical "1" to the $|1\rangle$ state and classical "0" to the $|0\rangle$ state. Alternatively, if we would like to prepare nondigital input for training, we first normalize it to the range $(0, \frac{\pi}{2})$, and then encode the data in the amplitude of the input qubits. This is done via the rotation angles, and we obtain for a single data point j the angle set $\gamma^j := \{\gamma_0^j, \gamma_1^j, \ldots, \gamma_n^j\}$ such that

$$\left|\psi_i^j\right\rangle = \cos\left(\gamma_i^j\right)|0\rangle + \sin\left(\gamma_i^j\right)|1\rangle, \tag{3}$$

where i is the dimension index of the data point. Taking a data point in N dimensions, the entire state is then given by the tensor product,

$$|\psi^j\rangle = \bigotimes_{i=1}^{N} \left|\psi_i^j\right\rangle. \tag{4}$$

This method is often referred to as *qubit encoding* [42]. The qubit encoding method stores each dimension of the data in the amplitude on an individual qubit. It hence maps the data to an exponentially large feature space. This allows the use of simpler models in the classifier task, such as linear classifiers. Additionally, this method is well suited for NISQ devices because it requires only single-qubit gates to implement.

Next, we need to consider how to represent our entire dataset and let the quantum circuit work out the averaged cost function naturally.

Suppose the circuit giving the cost function is represented by the unitary C, the measurement by the operator O, and the initial state for each data point be $|\chi_i\rangle = |\psi_i\rangle \otimes |\phi_i\rangle$. The cost function with respect to the ith data point x_i (encoded in ψ_i) can then be represented by

$$[E_{\mathrm{cost}}]_{x_i} = \langle\chi_i|\,C^\dagger O C\,|\chi_i\rangle = \langle\chi_i|\,D\,|\chi_i\rangle, \tag{5}$$

where $D = C^\dagger O C$. The average of the cost function is then given by

$$\overline{E_{\mathrm{cost}}} = \frac{1}{N}\sum_{i=0}^{N} \langle\chi_i|\,D\,|\chi_i\rangle, \tag{6}$$

where N is the number of data points in the dataset.

Here we can use a mixed state, which is the classical average of all possible states of our datasets. To see this, recall that the desired cost function value is the average of the cost function values across the entire data set. If we hence input a mixed state,

$$\rho_{\mathrm{mix}} = \frac{1}{n}\sum_{n} |\phi_i\rangle\,|\psi_i\rangle\,\langle\psi_i|\,\langle\phi_i|, \tag{7}$$

which is the average of all states in our data set, the outcome of the calculation is similarly the average over the data set. This mixed state can be constructed by randomly selecting one of the samples in every single run of the algorithm, and then averaging the outcome over all runs. Since all the qubits are separable, we need single-qubit rotations for each qubit to prepare each state. Notably, the accuracy of this will depend on the concrete distribution and the number of repetitions.

Rather than using the mixed state described above, it is natural to think if we can prepare data in a pure superposition state. We find that it is possible, but that there is no speed-up compared with the mixed-state preparation method. Consider a uniform superposition state over the entire dataset. After evaluating the circuit, we then obtain the expectation value as

$$\begin{aligned}\overline{E_{\mathrm{cost}}} &= \sqrt{\frac{1}{N}}\sum_{i=0}^{N}\langle\chi_i|\,D\sqrt{\frac{1}{N}}\sum_{i=0}^{N}|\chi_i\rangle \\ &= \frac{1}{N}\sum_{i=0}^{N}\langle\chi_i|\,D\,|\chi_i\rangle + \frac{1}{N}\sum_{i=0}^{N}\sum_{j\neq i}^{N}\langle\chi_i|\,D\,|\chi_j\rangle.\end{aligned} \tag{8}$$

The first term of Eq. (8) would give us then exactly what we desire, if $i = j$. However, it also includes the terms for $i \neq j$. Although the input states $|\chi_i\rangle$ are always orthogonal to each other for fully digitized input data, we cannot guarantee that $\langle\chi_j|\,D\,|\chi_i\rangle = 0$ holds. To fix this problem, we can introduce an additional register $|\epsilon_i\rangle$ which we call the *index qubits*. The new initial state is then given by $|\chi_i\rangle = |\psi_i\rangle \otimes |\phi_i\rangle \otimes |\epsilon_i\rangle$. We put the index of each data point into these qubits and do not apply

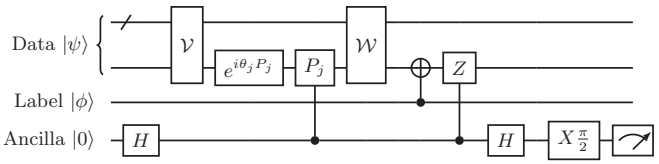

FIG. 2. Circuit with a CNOT cost function and single-circuit gradient evaluation. The input data are encoded into the data register ψ represented as the first two lines, and the expected output (label) is encoded in the label register $|\phi\rangle$. When we optimize the parameter θ_j we decompose the ansatz U into $U = V(e^{i\theta_j P_j} \otimes \mathbb{I}_k^{(n-1)})W$ where $\mathbb{I}_k^{(n-1)}$ is the identity operation on all the other qubits except the kth qubit (to which we applied the single-qubit rotation), and $e^{i\theta_j P_j}$ is the single-qubit rotation gate parametrized by θ_j, where P_j is a Pauli operator. We first insert a controlled-P_j gate right after the single-qubit rotation, then add a controlled-Z gate after the output of the cost function. Both of these gates are controlled by an ancillary qubit prepared in state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. In the end we rotate the ancillary qubit back to the $|0\rangle, |1\rangle$ basis, then do a half π rotation on the X axis. The partial derivative of the cost function with respect to θ_j is stored in the ancillary qubit.

any operation on them throughout the circuit. Since $\{|\epsilon_i\rangle\}$ is an orthogonal set, the entire state after running the circuit will also be orthogonal, and we will obtain $\langle\chi_i| D |\chi_j\rangle = 0$ for any $i \neq j$.

It is well known that such a superposition state can be prepared using variants of Grover's state preparation [43,44]. However, these approaches require relatively deep circuits and are therefore not immediately applicable for NISQ devices. Moreover, the index qubits we introduced here do not participate in any computation of the PQC. We can measure these index qubits at the very beginning, resulting in a state collapse into a specific data point. This superposition-state encoding is therefore equivalent to the mixed-state encoding we proposed above, which randomly selects a data point at the beginning and averages the outcome in the end. This encoding will have the same outcome as long as the difference between the random number generator and the quantum randomness is negligible.

By simply preparing the input data into a superposition, the expected value of the quantum circuit is not the same as evaluating each data point individually and taking the average. This is because of the quantum interference between each data point. Once the interference is removed by adding index qubits, there is no quantum speed-up from encoding the data in superposition.

IV. RESULTS

In this section we present numerical simulations of the proposed methods. We use the gradient-finding method using the Hadamard test [26,27] and the Adam optimizer [45]. First we give a brief review of the Hadamard test gradient estimation, and then demonstrate how to use Hadamard test gradient estimation together with the proposed method in this paper. Here we show two tasks, to train a classifier for a XOR gate and to train a classifier for a more realistic problem, the canonical Iris flower dataset.

A. Gradient estimation with the Hadamard test

Gradient estimation with the Hadamard test has previously been used to calculate the partial derivative of the eigenenergy of a molecule [26,27]. The Hamiltonian Pauli terms used to approximate molecular energies are typically multiqubit terms. For this reason, they can require a separate control operation on a large number of qubits, which in NISQ devices has the potential to introduce a prohibitive level of noise [4]. Here, by encoding the cost function valuation into a single qubit, the control operation is reduced to a simple controlled-Z gate which is acceptable for NISQ applications.

A circuit that implements this Hadamard test gradient estimation together with cost function embedding is shown in Fig. 2. Using this method, it is possible to evaluate the partial derivative of the expectation value of the output qubit with respect to the rotation angle θ. Here θ parametrizes the rotation generated by some Pauli operator P. For example, for P being the Pauli X operator, we have a single-qubit rotation gate with angle θ about the X axis. Typically there will be multiple single-qubit gates inside the ansatz. We can perform this method on each one of them to get the partial derivative of our cost function with respect to each rotation angle. The whole process is given by the following: We begin by preparing an ancillary qubit in the $|0\rangle$ state, and then apply a Hadamard gate to obtain the plus state, $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. Next, we apply a controlled-P gate right after the single-qubit rotation gate $e^{i\theta P}$, where the control is on the ancillary qubit. Finally, because we encoded our cost function value in the Z

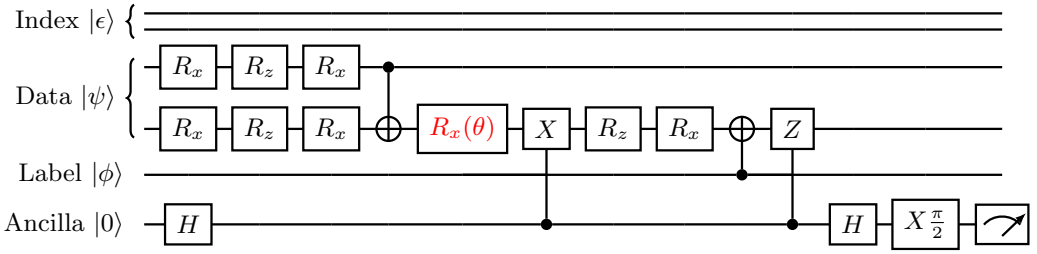

FIG. 3. Evaluation of the partial derivative of theta for training the XOR circuit with the CNOT cost function. This circuit is used to find the partial derivative of the CNOT cost function with respect to the rotation angle θ. Here we choose an ansatz with arbitrary single-qubit rotation applied on each input qubit first, then use a CNOT gate to exchange the information between two qubits, and again apply an arbitrary single-qubit rotation on the output qubit. We use three parametrized single-qubit rotations, *Rx*, *Rz*, *Rx*, to construct an arbitrary single-qubit rotation. The input state is encoded with Eq. (9).

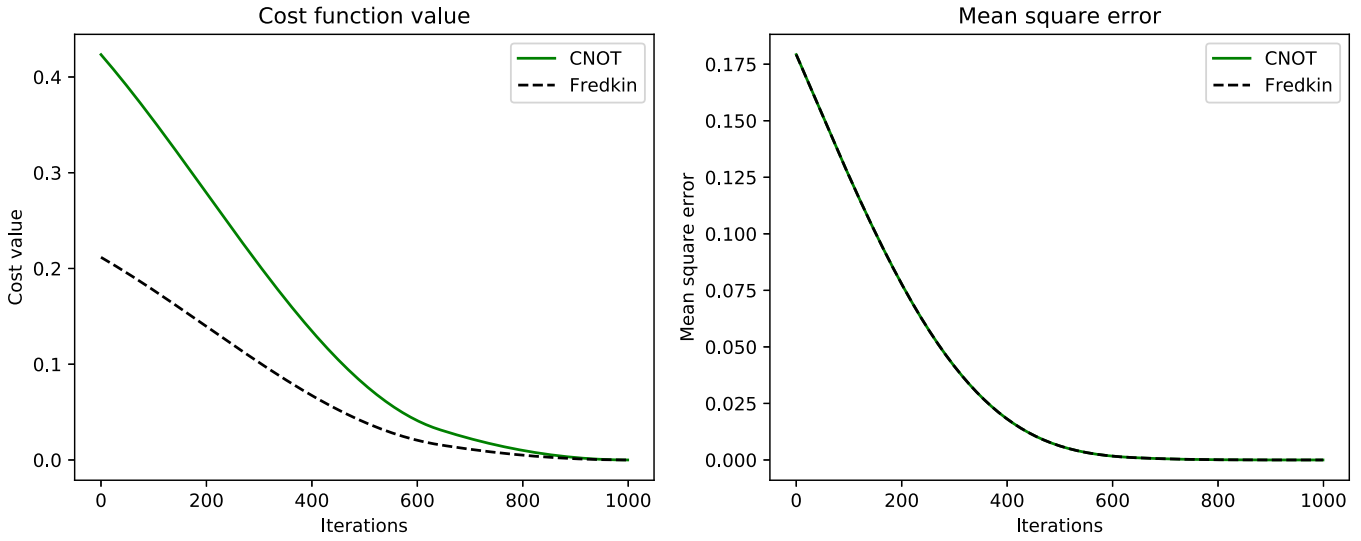

FIG. 4. Numerical simulation result for circuit showed in Fig. 3. Here we tested the cost function value and the mean square error for the same training dataset prepared as Eq. (9). During the training process, the circuits for evaluating partial derivatives of each of the parameters θ_i are generated. For each iteration, all these circuits are evaluated, which gives the derivative of each parameter, which is the gradient of the cost function. Then the gradient value is passed to the Adam optimizer. Here the learning rate of the optimizer is 10^{-3}. At the end of each iteration, the parameters are updated by the Adam optimizer. The result here shows that both the CNOT and Fredkin cost function can be used for training, and they give similar convergence speed.

axis of the second qubit, we add a controlled-Z gate between the ancillary qubit and the second qubit, followed by another Hadamard gate on the ancillary qubit which rotates the state back to the $\{0, 1\}$ basis. The resulting ancillary qubit now contains the gradient encoded in its phase. Finally, we apply a $\frac{\pi}{2}$ rotation about the X axis on the ancillary qubit so that the gradient is encoded in the Z axis of the ancilla and can be determined by measuring in the $\{|0\rangle, |1\rangle\}$ basis.

We can therefore use the circuit in Fig. 2 to estimate the gradient by simply measuring the single ancillary qubit.

B. XOR experiment

We can now proceed to test the above methods for training PQCs.

The first example task consists of training the circuit to perform the classical exclusive or (XOR) operation. Here we require the circuit to implement the truth table of XOR, i.e., yield true if and only if the input bits are different, and false otherwise. We use $|0\rangle$ to denote the FALSE value, and $|1\rangle$ to denote TRUE. Here we demonstrate the method of encoding data in superposition. The circuit ansatz is shown in Fig. 3.

The input bits are $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ [middle digits in the kets in Eq. (9) below], the corresponding labels, i.e., evaluations of XOR on the respective input, are (fifth digit) $|0\rangle$, $|1\rangle$, $|1\rangle$, $|0\rangle$, and the indices are given by $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ (first two digits). Including an extra qubit as the ancillary qubit, the final input state is given by

$$|\chi\rangle = \tfrac{1}{2}(|000000\rangle + |010110\rangle + |101010\rangle + |111100\rangle) \quad (9)$$

After the gradients are determined by evaluating the circuit they are used to update the parameters by using the Adam optimizer [45].

The simulation result is shown in Fig. 4. During the experiment, we were able to see that circuits with CNOT or Fredkin cost functions can successfully give the correct gradient direction. We observed that training converges for both fully digitized data encoding or the mixed state encoding, and the convergence rates are the same.

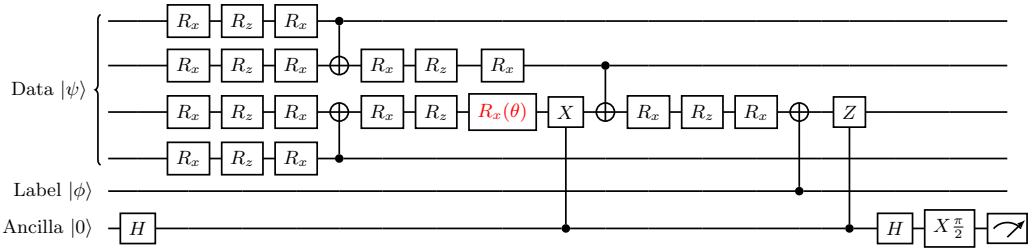

FIG. 5. Training circuit for quantum hierarchical classifiers [13]. The input state is encoded with Eq. (7). Here we demonstrate that the circuit gives the partial derivative of the cost function against the parameter θ which parametrizes one of the single-qubit rotation gates (highlighted in red). A controlled-X gate (CNOT) is applied right after the parametrized gate.

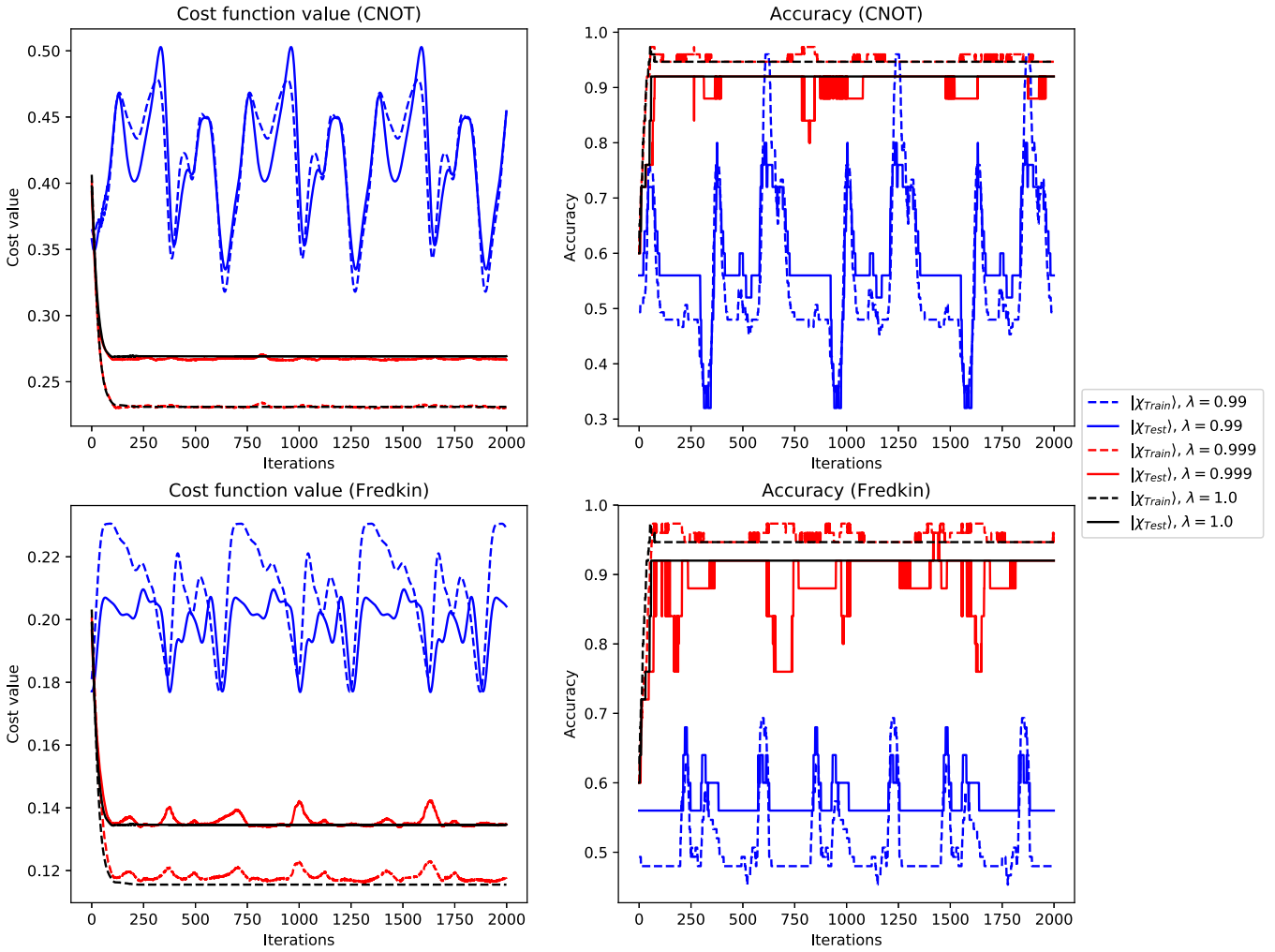

FIG. 6. Simulation of training the hierarchical classifier for the Iris dataset. Here we choose two flowers (labeled 1 and 2) from the Iris dataset. The training process is the same as in Fig. 4. Here the learning rate is 10^{-2}. We consider that the prediction of the flower label is 2 if the probability to measure and get state $|1\rangle$ is greater than 0.5, otherwise the prediction will be considered to be label 1. The accuracy is the ratio of the correct prediction among the entire training dataset $|\chi_{Train}\rangle$ (dashed lines) or the test dataset $|\chi_{Test}\rangle$ (solid lines). Then we add the depolarization channel $\Delta_\lambda(\rho) = \lambda\rho + (1-\lambda)/2^n\mathbb{I}$, where n is the number of qubits to the circuit, and compare the simulation between results with and without noise.

C. Iris experiment

Next we investigate a more realistic problem. As a second example we implement a classifier for the Iris dataset [46]. The Iris dataset contains 150 labeled examples in total, with three different types of Iris flowers. Each example is described by four features. We prepare a mixed state with the method from Eq. (7), and the training circuit is shown in Fig. 5.

Several interesting results were found during the training process; see Fig. 6. First, we report that by using the Adam optimizer, the PQCs can converge. The CNOT cost function and the Fredkin cost function were able to achieve similar training performance and convergence rates.

To investigate the robustness of the circuit, we applied the depolarization channel to the system. The channel is described as $\Delta_\lambda(\rho) = \lambda\rho + (1-\lambda)/2^n\mathbb{I}$ where n is the number of qubits. We observe that the algorithm still converges to a region close to the optimal point when $\lambda = 0.999$ is applied. When $\lambda = 0.99$, the cost function no longer converges.

We can see for the circuit in Fig. 5 that we can tolerate depolarization noise when $\lambda = 0.999$, which is quite close to gate fidelities of state-of-art NISQ devices [47,48]. This method could also be combined with error mitigation [49], which can suppress some errors and make this algorithm even more robust against the noise on a NISQ machine.

V. CONCLUSION

We now summarize the advantages of our proposed method. First, the encoding of the value of the cost function into a measurement expectation value enables the ability to do further quantum information processing. For example, the direct use of advanced optimization methods such as the Hadamard test, *shift rule*, and *rotosolve*. Since the cost function value is encoded in the state of a single qubit, the implementation of a full optimization algorithm making use of this method can be straightforward.

Second, the proposed data-encoding method allows for efficiently estimating expectation values across the training

dataset. It is classically inefficient to calculate the cost function for each example and then average them across the entire data set. Instead, the proposed method indicates that the expectation value can be obtained via random sampling from the training dataset.

We show that simply encoding the dataset into a superposition state will not give us the average of the cost function values. To fix this issue, we introduced the index qubit. However, we showed that the index qubit would make it equivalent to doing a mixed-state preparation and that such superposition-state preparation has no speed-up compared with mixed-state preparation.

In conclusion, we proposed a method to encode cost functions into quantum circuits and a corresponding method for preparing the input data. This method therefore enables quantum information processing on the cost function value. The averaged cost value can be calculated across the entire dataset with both mixed-state data preparation and superposition data preparation. We demonstrated gradient evaluation of the cost function with the Hadamard test and investigated its performance under a depolarization noise channel.

ACKNOWLEDGMENTS

E.G. is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) [EP/P510270/1]. L.W. acknowledges the support through the Google PhD Fellowship in Quantum Computing. B.V. acknowledges support from an EU Marie Skåodowska-Curie fellowship. P.L. acknowledges support from the EPSRC [EP/M013243/1] and Oxford Quantum Circuits Limited. We thank M. Benedetti, X. Yuan, P. Spring, and T. Tsunoda for insightful discussions. We acknowledge the use of the University of Oxford Advanced Research Computing facility.

APPENDIX: CNOT COST FUNCTION DERIVATION

Suppose the expectation value of the measurement $\langle M\rangle = \alpha$, the state of the output qubit is given by

$$\rho_0 = \begin{pmatrix} 1-\alpha & \epsilon^* \\ \epsilon & \alpha \end{pmatrix}, \tag{A1}$$

where ϵ is the off-diagonal term for the reduced density matrix. Because the output qubit could be in a mixed state, the exact definition of ϵ depends on the choice of the ansatz. For each data point, the label qubit can be in state either $|0\rangle$ or $|1\rangle$, which we introduce $\beta = 0$ when state $|0\rangle$ and $\beta = 1$ when state $|1\rangle$. The state of the label qubit is given by

$$\rho_\phi = \begin{pmatrix} 1-\beta & 0 \\ 0 & \beta \end{pmatrix}, \tag{A2}$$

and the state of the system of labeling qubit and output would be

$$\rho_1 = \rho_0 \otimes \rho_\phi = \begin{pmatrix} (1-\alpha)(1-\beta) & 0 & (1-\beta)\epsilon^* & 0 \\ 0 & (1-\alpha)\beta & 0 & \beta\epsilon^* \\ (1-\beta)\epsilon & 0 & \alpha(1-\beta) & 0 \\ 0 & \beta\epsilon & 0 & \alpha\beta \end{pmatrix}. \tag{A3}$$

After applying the CNOT gate, the state would be

$$\rho_2 = U_{\text{CNOT}}^\dagger \rho_1 U_{\text{CNOT}} = \begin{pmatrix} (1-\alpha)(1-\beta) & 0 & (1-\beta)\epsilon^* & 0 \\ 0 & \alpha\beta & 0 & \beta\epsilon \\ (1-\beta)\epsilon & 0 & \alpha(1-\beta) & 0 \\ 0 & \beta\epsilon^* & 0 & (1-\alpha)\beta \end{pmatrix}. \tag{A4}$$

Trace away the labeling qubit, the state of the output qubit would be

$$\rho_{\text{output}} = Tr_\phi(\rho_2) = \begin{pmatrix} (1-\alpha)(1-\beta)+\alpha\beta & (1-\beta)\epsilon^* + \beta\epsilon \\ (1-\beta)\epsilon + \beta\epsilon^* & \alpha(1-\beta)+(1-\alpha)\beta \end{pmatrix}. \tag{A5}$$

The expectation value of the output qubit would be the bottom right element of ρ_{output}, therefore,

$$E_{\text{cost}(\psi)} = P_{|1\rangle} = \alpha(1-\beta)+(1-\alpha)\beta = \alpha+\beta-2\alpha(1-2\beta)\alpha+\beta. \tag{A6}$$

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

[2] Y. Guo, Yu Liu, T. Georgiou, and M. S. Lew, A review of semantic segmentation using deep neural networks, Int. J. Multimed. Inf. Retr. **7**, 87 (2018).

[3] C. C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, Bo Li, J. Chorowski, and M. Bacchiani, State-of-the-Art speech recognition with sequence-to-sequence models, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP* (IEEE, Piscataway, NJ, 2018), Vol. 2018, p. 4774.

[4] M. A. Nielsen and I. L. Chuang, in *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000), p. 676.

[5] N. Wiebe, A. Kapoor, and K. Svore, Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning, Quantum Inf. Comput. **15**(3&4), 0318 (2014).

[6] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum Support Vector Machine for Big Data Classification, Phys. Rev. Lett. **113**, 130503 (2014).
[7] Y. Du, T. Liu, and D. Tao, Bayesian Quantum Circuit, arXiv:1805.11089.
[8] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, Quantum machine learning: A classical perspective, Proc. R. Soc. London, Ser. A **474**, 20170551 (2017).
[9] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature (London) **549**, 195 (2017).
[10] A. Perdomo-Ortiz, M. Benedetti, J. Realpe-Gómez, and R. Biswas, Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers, Quantum Sci. Technol. **3**, 030502 (2018).
[11] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum Algorithm for Linear Systems of Equations, Phys. Rev. Lett. **103**, 150502 (2009).
[12] L. Wossnig, Z. Zhao, and A. Prakash, Quantum Linear System Algorithm for Dense Matrices, Phys. Rev. Lett. **120**, 050502 (2018).
[13] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, Hierarchical quantum classifiers, npj Quantum Inf. **4**, 65 (2018).
[14] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors, arXiv:1802.06002.
[15] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, Phys. Rev. A **98**, 032309 (2018).
[16] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, A generative modeling approach for benchmarking and training shallow quantum circuits, npj Quantum Inf. **5**, 45 (2019).
[17] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Circuit-centric quantum classifiers, Phys. Rev. A **101**, 032308 (2020).
[18] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, Variational quantum circuits for deep reinforcement learning, arXiv:1907.00397.
[19] W. Huggins, P. Patel, K. B. Whaley, E. M. Stoudenmire, P. Patil, B. Mitchell, K. B. Whaley, and E. M. Stoudenmire, Towards quantum machine learning with tensor networks, Quantum Sci. Technol. **4**, 024001 (2018).
[20] S. Adhikary, S. Dangwal, and D. Bhowmik, Supervised learning with a quantum classifier using multi-level systems, Quantum Inf. Process. **19**, 89 (2020).
[21] D. Liu, S.-J. Ran, P. Wittek, C. Peng, R. B. García, G. Su, and M. Lewenstein, Machine learning by unitary tensor network of hierarchical tree structure, New J. Phys. **21**, 073059 (2019).
[22] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, The expressive power of parameterized quantum circuits, arXiv:1810.11922.
[23] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature (London) **567**, 209 (2019).
[24] M. Schuld and N. Killoran, Quantum Machine Learning in Feature Hilbert Spaces, Phys. Rev. Lett. **122**, 040504 (2019).
[25] A. Harrow and J. Napp, Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms, arXiv:1901.05374.
[26] P.-L. Dallaire-Demers, J. Romero, L. Veis, S. Sim, and A. Aspuru-Guzik, Low-depth circuit ansatz for preparing correlated fermionic states on a quantum computer, Quantum Sci. Technol. **4**, 045005 (2019).
[27] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. J. Love, and A. Aspuru-Guzik, Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz, Quantum Sci. Technol. **4**, 014008 (2018).
[28] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).
[29] M. Ostaszewski, E. Grant, and M. Benedetti, Quantum circuit structure learning, arXiv:1905.09692.
[30] K. M. Nakanishi, K. Fujii, and S. Todo, Sequential minimal optimization for quantum-classical hybrid algorithms, arXiv:1903.12166.
[31] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, Automatic differentiation in machine learning: A survey, J. Mach. Learn. Res. **18**, 5595 (2017).
[32] V. Giovannetti, Quantum-enhanced measurements: Beating the standard quantum limit, Science **306**, 1330 (2004).
[33] D. Wang, O. Higgott, and S. Brierley, Accelerated Variational Quantum Eigensolver, Phys. Rev. Lett. **122**, 140504 (2019).
[34] F. Vatan and C. Williams, Optimal quantum circuits for general two-qubit gates, Phys. Rev. A **69**, 032315 (2004).
[35] D. P. DiVincenzo, Two-bit gates are universal for quantum computation, Phys. Rev. A **51**, 1015 (1995).
[36] K.-B. Duan and S. S. Keerthi, in *Which Is the Best Multiclass SVM Method? An Empirical Study* (Springer, Berlin, Heidelberg, 2005), p. 278.
[37] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, Quantum Fingerprinting, Phys. Rev. Lett. **87**, 167902 (2001).
[38] E. Aïmeur, G. Brassard, and S. Gambs, Machine Learning in a Quantum World, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Springer-Verlag, Berlin, 2006), Vol. 4013, p. 431.
[39] R. B. Patel, J. Ho, F. Ferreyrol, T. C. Ralph, and G. J. Pryde, A quantum Fredkin gate, Sci. Adv. **2**, e1501531 (2016).
[40] T. Ono, R. Okamoto, M. Tanida, H. F. Hofmann, and S. Takeuchi, Implementation of a quantum controlled-SWAP gate with photonic circuits, Sci. Rep. **7**, 45353 (2017).
[41] T. Liu, B.-Q. Guo, C.-S. Yu, and W.-N. Zhang, One-step implementation of a hybrid Fredkin gate with quantum memories and single superconducting qubit in circuit QED and its applications, Opt. Express **26**, 4498 (2018).
[42] E. M. Stoudenmire and D. J. Schwab, Supervised Learning with Quantum-Inspired Tensor Networks, Adv. Neural Inf. Process. Syst. **29**, 4799 (2016).
[43] Lov K. Grover, Synthesis of Quantum Superpositions by Quantum Computation, Phys. Rev. Lett. **85**, 1334 (2000).
[44] Y. R. Sanders, G. H. Low, A. Scherer, and D. W. Berry, Black-Box Quantum State Preparation without Arithmetic, Phys. Rev. Lett. **122**, 020502 (2019).
[45] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, May 7–9, 2015*,

edited by Y. Bengio and Y. LeCun (https://iclr.cc/archive/www/doku.php%3Fid=iclr2015:main.html, 2015).

[46] R. A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. **7**, 179 (1936).

[47] P. V. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, Y. Chen *et al.*, Fluctuations of Energy-Relaxation Times in Superconducting Qubits, Phys. Rev. Lett. **121**, 090502 (2018).

[48] N. M. Linke, D. Maslov, M. Roetteler, S. Debnath, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, Experimental comparison of two quantum computing architectures, Proc. Natl. Acad. Sci. USA **114**, 3305 (2017).

[49] S. Endo, S. C. Benjamin, and Y. Li, Practical Quantum Error Mitigation for Near-Future Applications, Phys. Rev. X **8**, 031027 (2018).