Analog Quantum Variational Embedding Classifier

Analog Quantum Variational Embedding Classifier

Rui Yang,[1,*] Samuel Bosch,[2] Bobak Kiani,[2] Seth Lloyd,[2] and Adrian Lupascu[1]

[1] *Institute for Quantum Computing, and Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada*

[2] *Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139, USA*

(Received 4 November 2022; revised 15 February 2023; accepted 12 April 2023; published 5 May 2023)

Quantum machine learning has the potential to provide powerful algorithms for artificial intelligence. The pursuit of quantum advantage in quantum machine learning is an active area of research. For current noisy intermediate-scale quantum computers, various quantum-classical hybrid algorithms have been proposed. One such previously proposed hybrid algorithm is a gate-based variational embedding classifier, which is composed of a classical neural network and a parameterized gate-based quantum circuit. We propose a quantum variational embedding classifier based on an analog quantum computer, where control signals vary continuously in time: our particular focus is an implementation using quantum annealers. In our algorithm, the classical data are transformed into the parameters of the time-varying Hamiltonian of the analog quantum computer by a linear transformation. The nonlinearity needed for a nonlinear classification problem is purely provided by the analog quantum computer through the nonlinear dependence of the final quantum state on the control parameters of the Hamiltonian. We perform numerical simulations that demonstrate the effectiveness of our algorithm for performing binary and multiclass classification on linearly inseparable datasets such as concentric circles and MNIST digits. Our classifier can reach accuracy comparable with that of the best classical classifiers. We find that the performance of our classifier can be increased by increasing the number of qubits, until the performance saturates and fluctuates. Moreover, the number of optimization parameters of our classifier scales linearly with the number of qubits. The increase of the number of training parameters when the size of our model increases is therefore not as fast as that of a neural network. Our algorithm presents the possibility of using current quantum annealers for solving practical machine-learning problems, and it could also be useful to explore quantum advantage in quantum machine learning.

DOI: 10.1103/PhysRevApplied.19.054023

I. INTRODUCTION

Recent progress in the field of quantum computation has led to the attainment of several important milestones. Quantum supremacy has been demonstrated in a few platforms [1–3]. Quantum processors with tens of qubits have been implemented and become available for operation via clouds [4]. Quantum error correction using the surface code [5] has been demonstrated in several experiments [6–8].

Quantum computers have the potential to solve certain problems significantly more efficiently than classical computers. The original motivation for using quantum computers is to efficiently simulate quantum mechanical systems such as molecules [9]. Besides this quantum advantage foreseeable in quantum chemistry, a quantum advantage could also be found in other computation problems. For example, Shor's quantum algorithm can factorize large numbers efficiently [10], thereby posing a threat to the widely used RSA encryption system, upon which the modern financial system is built. Grover's algorithm can speed up an unstructured-database search [11].

A promising area of application for quantum computers is machine learning. The field of machine learning has developed at a fast pace in recent years. Since 2011, the progress made in neural-network algorithms and large-scale classical computing hardware, such as graphics processing units, has led to increased performance of machine learning for many applications. A notable example is image classification, where such algorithms perform better than humans. Machine-learning algorithms have a wide range of profitable industry applications, such as recommendation systems [12,13]. Whether quantum computers can contribute to the machine-learning community by providing quantum advantage is an intriguing question [14].

To explore the potential advantage brought about by quantum machine learning, many quantum machine learning algorithms have been proposed [14–18]. Some

*r249yang@uwaterloo.ca

of the quantum machine learning algorithms have been shown to have advantages over their classical counterparts [15–17]. Many proposed algorithms fall into two categories: classification tasks [17,19] or generative tasks [18]. For classification tasks, several algorithms based on variational embedding [19], quantum kernels [19], or quantum support vector machines [17] have been proposed. The mechanism behind many quantum classification algorithms is based on finding a simplified separation boundary between different classes after nonlinear mapping of the original data into a new space that usually has a much higher dimension [20,21].

We now discuss recent progress on variational embedding classifiers. Havlíček *et al.* [19] proposed and demonstrated a variational embedding algorithm for binary-classification tasks that embeds classical data into quantum states of a high-dimensional Hilbert space through parameters of quantum gates, and uses quantum measurements to classify the embedded quantum states. This approach focuses on finding an operator whose expectation value with respect to the states corresponding to embedded data is different for the two classes. Specifically, the two classes lead to positive and negative expectation values, respectively. This decision operator effectively defines a hypersurface in the multiqubit Hilbert space used to separate embedded quantum states corresponding to two classes. Lloyd *et al.* [20] proposed a more-general version of the variational method that uses a general decision operator defined with density matrices from collections of embedded data. This algorithm has a training procedure that optimizes the embedding to maximize the distance between embedded quantum states with different labels. In this approach, a neural network is used to transform the classical data into a dataset with different dimensionality. A parameterized quantum circuit is introduced where the rotation angles of the gates are the circuit parameters. Each datum can be mapped into a final evolved quantum state of a multiqubit Hilbert space by the filling of some of the rotation angles of the quantum circuit with the transformed data values. The rest of the quantum circuit parameters are knobs used to maximize the distance between the density matrices formed by states of each class. After maximization of the distances, the classification can be done with a simple decision-operator expectation-value metric to find the label. One open question regarding this algorithm concerns the role of the quantum part in the operation of the algorithm since the nonlinearity in the neural network is sufficient by itself for performing classification.

To elucidate the source of nonlinearities in a hybrid classifier and establish whether the quantum part alone can do the "heavy lifting" for realization of a nonlinear classifier, we propose and investigate a quantum variational embedding classifier based on an analog quantum computer, whose control fields vary continuously in time. We focus on an implementation corresponding to the transverse-field Ising model used in quantum annealing; alternatively, this approach can also be explored with other types of Hamiltonian. In our algorithm, the neural network is replaced with a simple linear transformation, and the variational quantum circuit composed of gates is replaced with an analog quantum computer with direct control of the continuously varying Hamiltonian parameters. The nonlinear mapping of the classical data to a high-dimensional density matrix is now realized with the analog quantum computer. The mapping can be regarded as a point in a $(2^n \times 2^n)$-dimensional space defined on a complex-number domain, where n is the number of qubits. Efficient classification is achieved when the data corresponding to distinct labels form separate clusters in Hilbert space. Separability is analyzed in terms of the distinguishability of density matrices averaged over the states corresponding to different labels. To form separate clusters, the L_2 (Hilbert-Schmidt) distance [20] between the averaged density matrices is maximized in the training stage by adjustment of the parameters in the linear transformation that converts classical data into the parameters of the analog quantum computer. In the classification stage, a simple classifier based on a distance metric is used, yielding the predicted label for a new dataset as the label of the closest averaged density matrix obtained at the training stage. This distance-based strategy is also known as "nearest-centroid classification" [22]. This distance metric could be obtained purely with a quantum circuit [20]. We find that our algorithm can classify linearly inseparable datasets [23] with high accuracy. The dependence of the performance on the number of qubits shows that increasing the number of qubits can boost performance. Our algorithm opens up the possibility to use quantum annealers [24] for solving practical machine-learning problems.

II. ALGORITHM

In this section, we present our algorithm, which realizes a quantum variational embedding classifier on multiclass datasets. An important distinguishing feature of our algorithm, when compared with previous work, is that it uses control of the quantum system that is done via control of its Hamiltonian, in contrast with previous work, where control was done on the basis of quantum gates, as developed in the context of a gate-based quantum computation model.

In this work, we focus on the implementation of an analog variational embedding using a quantum annealer. Quantum annealing is an example of an analog quantum computation [23]. In quantum annealing, the initial system Hamiltonian is a transverse-field Hamiltonian, and the initial state is the ground state of this Hamiltonian. The Hamiltonian is continuously deformed, reaching at the end of the evolution an Ising form. This approach has been

developed and explored in connection with prospects for solving hard computational problems.

While we retain the key elements of quantum annealing, such as initial-ground-state preparation and continuous transverse to Ising Hamiltonian interpolation, our work focuses on optimizing evolution for classification. Perhaps most importantly, we do not assume that the quantum annealer remains in its ground state throughout the analog computation: that is, the quantum annealer performs *diabatic* quantum annealing, in which the continuous-time control fields drive it to a nonequilibrium final state. We emphasize that our algorithm can be implemented with other types of Hamiltonian, such as Hamiltonians used in gate-based quantum computers: our method applies to any quantum information processor that is controlled by continuously-time-varying fields.

The Hamiltonian of the quantum annealer used in our algorithm takes the form

$$H(t) = (1-s)H_0(s) + sH_1(s) + H_{\text{add}}(s), \tag{1}$$

where $s = t/t_{\text{max}}$, with t the time and t_{max} the total evolution time.

The initial Hamiltonian H_0 and the final Hamiltonian H_1 are given by

$$H_0 = \sum_i h_{x_i}\sigma_{x_i} \tag{2}$$

and

$$H_1 = \sum_i h_{z_i}\sigma_{z_i} + \sum_{i,j} J_{ij}\sigma_{z_i}\sigma_{z_j}, \tag{3}$$

where σ_{x_i} and σ_{z_i} are Pauli matrices.

The additional Hamiltonian H_{add} vanishes at the beginning and the end of the evolution [$H_{\text{add}}(0) = H_{\text{add}}(1) = 0$], and is given by

$$H_{\text{add}}(s) = \sum_i P_{z_i}(s)\sigma_{z_i} + \sum_i R_{x_i}(s)\sigma_{x_i} + \sum_{i,j} V_{ij}(s)\sigma_{z_i}\sigma_{z_j}. \tag{4}$$

The time-dependent coefficients in front of the Pauli matrices (schedules) take the following form:

$$P_{z_i}(s) = \sum_k c_{z_i,k}\sin((k+1)\pi s), \tag{5}$$

$$R_{x_i}(s) = \sum_k c_{x_i,k}\sin((k+1)\pi s), \tag{6}$$

$$V_{ij}(s) = \sum_k c_{ij,k}\sin((k+1)\pi s). \tag{7}$$

These forms are complete Fourier-series expansions consistent with the cancellation condition at $s = 0$ and $s = 1$. The Fourier-expansion waveform ansatz of the control fields used for quantum optimal control is an example of the chopped random basis (CRAB) [25,26]. Previous studies on CRAB show the optimization landscape of the CRAB ansatz is good for trainability [25,26], and as few as three Fourier terms in the Fourier ansatz are enough for a good performance [25,26].

We emphasize that our Hamiltonian is assumed to have full independent control of the initial transverse-field Hamiltonian and the final Ising Hamiltonian, as well as of the time dependence of the additional Hamiltonian. This model is consistent with a recently developed platform for coherent quantum annealing [27].

The connection between the data in the classification problem and the Hamiltonian is made as a linear transformation from the data to the system parameters, such as the Fourier coefficients of the additional Hamiltonian, as illustrated in Fig. 1. The configuration we use for this Hamiltonian is a simple one-dimensional configuration with nearest-neighbor ZZ coupling. Among all possible configurations, this is the simplest configuration for building quantum computers, and therefore is a good start for testing quantum algorithms.

In our algorithm, classical data are represented by a vector of dimension d, which is mapped into the parameters of an annealing Hamiltonian. The Hamiltonian is used to evolve a multiqubit (n-qubit) quantum system; see Fig. 1 and Algorithm 1 for the illustration. For a d-dimensional datum $\hat{X} = [x_1, \ldots, x_d]$, which represents a point in a d-dimensional space, it can be transformed to the schedule parameters with the following linear transformation:

$$\hat{\boldsymbol{V}}_{\boldsymbol{c}} = \hat{\boldsymbol{W}}\hat{\boldsymbol{X}}^T, \tag{8}$$

where

$$\hat{\boldsymbol{V}}_{\boldsymbol{c}} = \begin{bmatrix} \vdots \\ c_{zi_k} \\ \vdots \\ c_{xi_k} \\ \vdots \\ \vdots \\ c_{ij_k} \\ \vdots \end{bmatrix} \tag{9}$$

and $\hat{\boldsymbol{W}}$ is a matrix with a size of $(n_s \times m, d)$, where n_s is the number of sine terms in the ansatz, $m = 2n + (n-1)$ is the number of Pauli terms in the Hamiltonian (n is the number of qubits in a chain), and d is the dimension of the data. $n_s \times m \times d$ is hence the number of optimization parameters in this case.

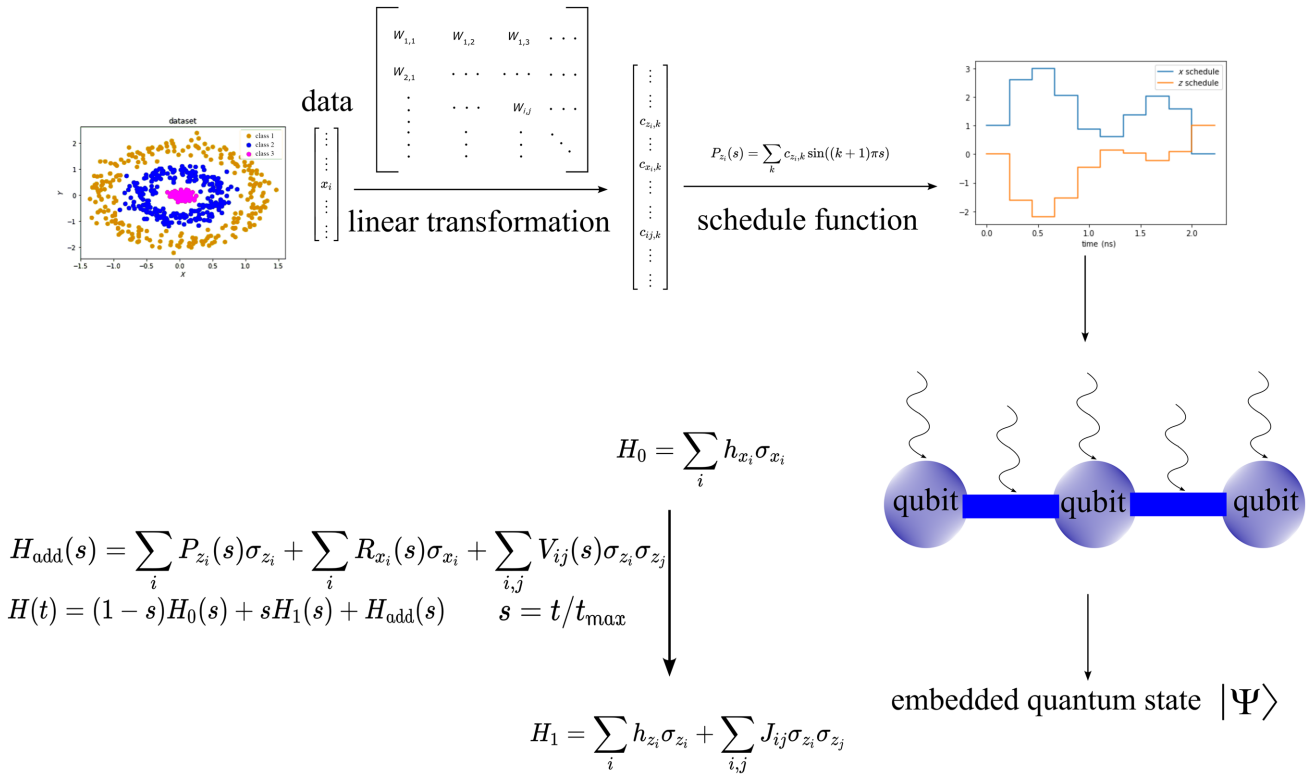

FIG. 1. Our analog quantum variational embedding classifier. The balls placed in a line represent a multiqubit quantum annealer. The connection lines represent couplings. The wavy arrows represent time-dependent driving on the qubits and couplings. The Hamiltonian of the annealer starts with H_0 and gradually changes to H_1, driven by a time-dependent H_{add} [see Eqs. (1)–(7)]. The time-dependent coefficients (the schedule) of the Pauli terms of H_{add} control the evolution [see Eqs. (4)–(7)]. The schedule is expressed with a summation of sine terms to meet a vanishing boundary condition at the beginning and end of the evolution [see Eqs. (5)–(7)]. The data are transformed into the parameters $c_{z_i,k}$, $c_{x_i,k}$, and $c_{ij,k}$ by a linear-transformation matrix to define the schedule of the quantum annealer [see Eqs. (5)–(9)].

For each classical data point, after the relevant Hamiltonian parameters are obtained through a linear transformation, a corresponding state is calculated on the basis of the evolution under the Hamiltonian. This process is repeated for a set of classical data points drawn randomly from the data. Density matrices are formed by combination of the states corresponding to each point within a certain class. The L_2 distance is maximized in the training stage by adjustment of the linear transformation $\hat{\boldsymbol{W}}$, which converts the raw data into schedule parameters, as outlined in Algorithm 1. The absolute loss for the optimization or training is therefore based on the L_2 distance. The entries ($\hat{W}_{ij}$) of the linear transformation are therefore taken to be the parameters to be adjusted to maximize the distances (minimize the loss). In the classification stage, the predicted label for a new dataset is the label of the closest average density matrix.

The parameters h_{z_i}, h_{x_i}, and J_{ij} in Eqs. (2) and (3) determine the initial and final energy levels of the Hamiltonian. They can be treated on the same footing as the Fourier coefficients—the parameters h_{x_i}, h_{z_i}, and J_{ij} of H_0 and H_1 can be transformed from the data by a linear-transformation matrix, i.e., h_{x_i}, h_{z_i}, and J_{ij} can be appended to $\hat{V}_c$. These parameters could also be fixed; for example, taking on values to make the initial and final Hamiltonians of the annealer nondegenerate. $t_{\max}$ is a fixed parameter, representing the total evolution time.

For multiclass classification, to simultaneously maximize the pairwise distances, we define the absolute loss as the product of the pairwise distance between an arbitrary pair of density matrices from collections of embedded data (see Algorithm 1). The definition of loss is not unique to multiclass classification. Essentially, any meaningful loss is acceptable; for example, we can also define the absolute loss as a summation of the pairwise distance between any pair of density matrices from collections of embedded data (which gives an algorithm with similar performance). Compared with previous work on variational embedding classifiers, here we extend the definition of loss to handle multiclass-classification situations.

Given that neural networks are a common way to implement classification, we compare our algorithm with a neural network. This comparison is illustrated in Fig. 2, which shows a comparison of the nonlinearity in quantum

Initial: Initialize parameters in the linear transformation matrix $\hat{\boldsymbol{W}}$; Initialize system state;
for iterations $j = 1, E$ **do**
  for label iteration $i = 1, L_{classes}$ **do**
    for sample in class i **do**
      Convert the sample to parameters of the system Hamiltonian by equation (8), then, define the system Hamiltonian (equation (1)); Evolving the initial state of the system to get a density matrix for the sample
      Add the density matrices together
    end for
    Averaging: taking the summation of the density matrix divided by the number of samples in class i as an average density matrix M_i for label i
  end for
  Calculate the L_2 distance $D_{ij} = Tr((M_i - M_j)^2)$ between the average density matrices M_i and M_j
  Perform a gradient descent on the loss function $L = -\prod_{ij} D_{ij}$ (for a binary classification task, $L = -D_{ij}$; $-\prod_{ij} D_{ij}$ is an extended loss for a multi-class classification task, other loss definition will also work) with respect to the parameters in the linear transformation $\hat{\boldsymbol{W}}$.
  Update the parameters in the linear transformation $\hat{\boldsymbol{W}}$
end for
Return: Parameters in the linear transformation $\hat{\boldsymbol{W}}$

Algorithm 1 Analog quantum variational embedding classifier algorithm.

variational circuits and a neural network. In a classical neural-network classifier, there are two stages: the first stage is a linear transformation that maps a point in the original data space into another point in another space. The next stage is a nonlinear transformation, which consists of a nonlinear distortion of the data points. It is this nonlinear distortion that makes the distorted mapped data points separate, and a simple boundary can be drawn between different classes. In our algorithm, the first stage is still linearly mapping the raw data into points in another space. The second stage is a nonlinear mapping into a $(2^n \times 2^n)$-dimensional space, taking advantage of the nonlinear dependence of the final state of a quantum system on the schedule parameters. In a quantum system, the mapping between the initial quantum state and the final quantum state is linear since the evolution is from a unitary evolution. However, the mapping between the schedule parameters and the final state is nonlinear. This is the source of nonlinearity in our algorithm. The approach we use is similar to that of a classical neural network in that the raw data in the original data space are mapped into another space (here the Hilbert space for our classifier) after being linearly transformed and nonlinearly distorted. The classification is performed on transformed data in the new space.

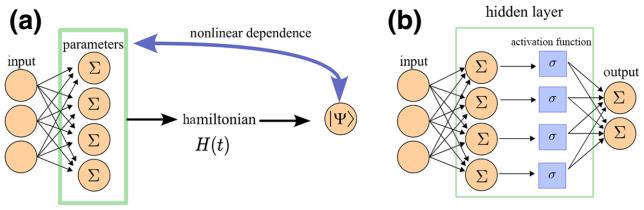

FIG. 2. Comparison of nonlinearity in our classifier and a neural network. (a) The nonlinearity in our classifier is from the nonlinear dependence between the quantum state and the parameters of the quantum system. (b) The nonlinearity in a neural network is from the activation function (such as a sigmoid function).

In the characterization of our algorithm, we use numerical simulations to calculate the evolved quantum states. In the numerical simulations, discrete time steps are used (see Fig. 1). In each time step, a constant Hamiltonian evolves the system, and the unitary evolution operator is calculated by direct matrix exponentiation: $U(t_n) = \exp[(-i/\hbar)H(t_n)\delta t]$ ($H(t_n)$ is the Hamiltonian at time t_n, and δt is the time step). The total evolution is from the product of these unitary evolution operators: $U(t_{\max}) = \prod_n \exp[(-i/\hbar)H(t_n)\delta t]$. In the simulated classifier, to maximize the distance (with the linear-transformation parameters as the control knobs), we use PyTorch's autograd [28] feature to perform an efficient gradient-descent optimization. Autograd [28] keeps a record of tensors and all executed operations, and the resulting new tensors in a computational graph whose leaves are the input tensors and roots are the output tensors. By tracing this graph from roots to leaves, we can automatically compute the gradients using the chain rule. The whole simulation in this study is written with PyTorch objects with the autograd features, which is essential to propagate the gradient backward toward tens or even hundreds of control-knob parameters.

III. SIMULATION RESULTS FOR THE ANALOG QUANTUM VARIATIONAL EMBEDDING CLASSIFIER

In this section, we discuss the characterization of the performance of our algorithm for performing classification on linearly inseparable datasets. We test our algorithm on three typical linearly inseparable datasets used for benchmarking machine-learning classifiers: concentric circles, spirals, and MNIST images (see Fig. 3). Our algorithm can perform classification for all of these datasets. Here the performance characterization is discussed in detail for concentric circles and MNIST digits, which are standard examples for machine-learning benchmarking.

A. Concentric circles

The concentric circles dataset is composed of labeled points distributed in several concentric circular areas, separated by gaps [23]. The classification task consists in training with these labeled points and using it to predict

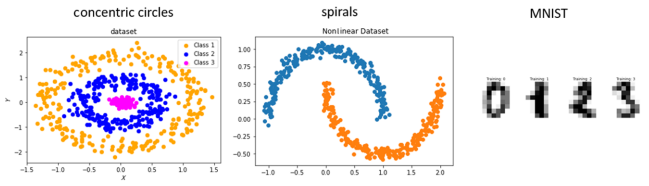

FIG. 3. Classical datasets for testing the performance of machine-learning algorithms. The first two datasets are the classical linearly inseparable datasets. The last one is the MNIST digits dataset.

labels of unseen points. We test our classifier on two-circle and three-circle cases. In our testing, we use the `datasets` module from the `scikit-learn` PYTHON package to generate our dataset. We randomly sample 500 (600) points as a training dataset and randomly sample 100 (120) data points as a test dataset for a two-circle (three-circle) case. We train our classifier on the training dataset, using gradient-descent optimization to maximize the separation between the average density matrices corresponding to different classes. We use three sine terms in the Fourier expansion, and ten time steps, and $t_{\max}$ is 2.

For a simple binary-classification or two-circle-classification problem, the absolute loss is defined as the L_2 distance between the two average density matrices corresponding to the two classes. We further extend it to a multiclass case by defining a loss to maximize pairwise distances between any two density matrices (see Algorithm 1). The absolute loss can be defined as the product of the pairwise distances (see Algorithm 1).

After training by using a gradient-descent optimizer (Adam optimizer of PyTorch, `torch.optim.Adam`) to adjust the weight matrix to minimize the loss (see Sec. II), we test our trained classifier on the test dataset. The predicted label for each datum in the test dataset is chosen to be the label of the nearest average density matrix.

During training, it is observed that at the starting stages, the embedded quantum states are not clustered. Only after training do the labeled embedded quantum states start to cluster according to their labels. The overlap (squared inner product) [20] between samples belonging to the same label is much stronger than the overlap between samples from different labels. The training results in the embedded states from the same class cluster together. The clustering of the training dataset can be visualized with the overlap matrix. Figure 4 shows the separation of different classes for the binary-classification (two-circle) case. Here the parameters h and J of H_0 and H_1 are fixed, taking on values to make the initial and final Hamiltonians nondegenerate. A trained classifier is used here. The training dataset is fed into the annealer with the trained parameters. In a classification task, the final evolved quantum states of $s = 1$ are used as the embedded states for the data. We record the evolution of the quantum states at each time step. The image sequence is from $s = 0$ (the top-left image) to $s = 1$ (the bottom-right image).

Next we discuss the characterization of the performance as a function of the number of qubits. To make the dependence on the number of qubits more universal, the parameters h_{x_i}, h_{z_i}, and J_{ij} of H_0 and H_1 are treated on the same footing as the Fourier coefficients here—they are also transformed from the data by a linear-transformation matrix. In Fig. 5, we show the classification accuracy versus the number of qubits. Figure 5(a) is the result of the training accuracy after optimization of the analog quantum computer parameters to maximize the separation of the density matrices. Figure 5(b) shows the results for the

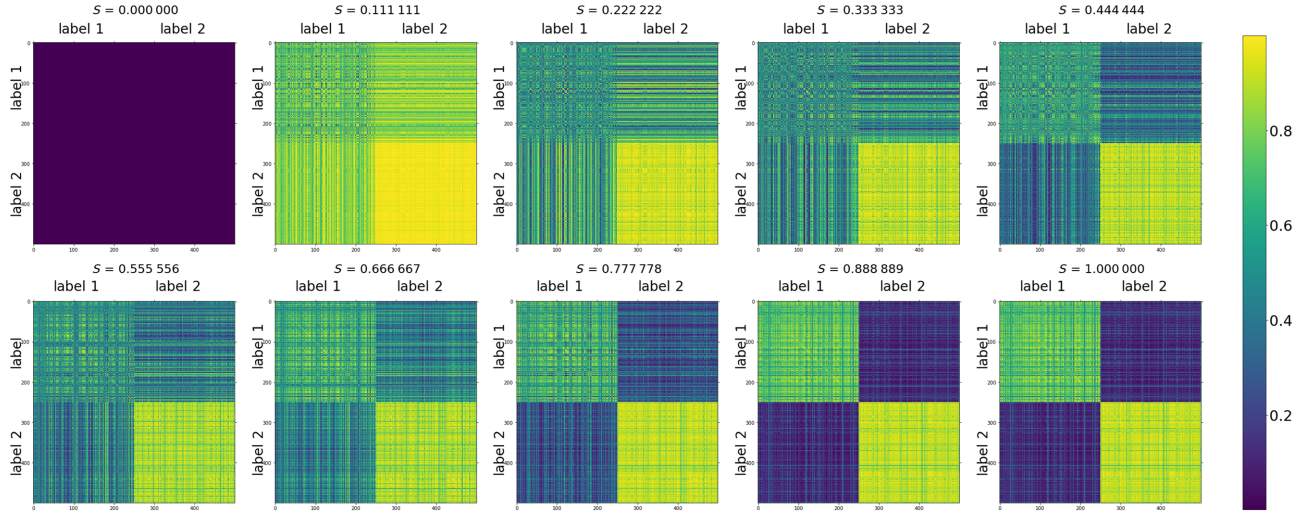

FIG. 4. Time evolution of the overlap matrix of training data for binary classification on circles. The data-to-Hamiltonian mapping corresponds to a fully trained classifier. For each time step, the evolved quantum states are calculated, and the overlap matrix is calculated. The image sequence is from $s = 0$ (top-left image) to $s = 1$ (bottom-right image). The brightness indicates the overlap between the embedded quantum states. Labels 1 and 2 represent the outer and inner circles, respectively.

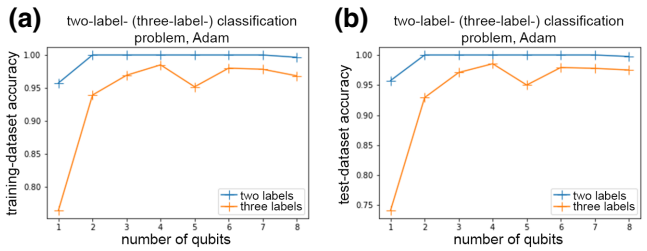

FIG. 5. The classification accuracy (with a trained classifier) versus the number of qubits for multiclass-classification cases (two examples—two-label and three-label cases—are shown here). (a) Classification accuracy versus the number of qubits for the training dataset (dataset used for training our classifier). (b) Classification accuracy versus the number of qubits for the test dataset (unseen dataset), directly using the classifier trained on the training dataset.

test accuracy obtained with the same trained analog quantum computer (obtained after optimization of the analog quantum computer parameters to maximize the separation of the density matrices). For each data point, four independent random initializations (starting from four random initial guesses of the linear-transformation matrix $\hat{\boldsymbol{W}}$) of the training stage are performed to obtain sufficient statistics. The mean and standard deviations of the accuracies are recorded.

Figure 5 shows that as the number of qubits increases, the performance increases until it saturates and fluctuates. The saturation accuracy is more than 95% (see Sec. IV for a comparison with classical classifiers). The origin of the fluctuation may be the increased optimization complexity as the number of qubits increases. These results show that the classification accuracies on both the training dataset and the test dataset can be generally increased by increasing the number of qubits. The similar scaling for both the training dataset and test dataset means that our classifier has good generalization. The performance boost as the number of qubits increases indicates that the expressivity of our classifier can be improved as the number of qubits increases. Therefore, a more-complicated classification task could be handled once we add more qubits to our classifier. Whether the performance after a certain number of iterations of training will always increase as a function of the number of qubits is a complex issue. Indeed, previous studies showed that a system with too many qubits might have barren plateaus with a vanishing gradient, and training or optimization of performance will become difficult in this case [22,29,30]. Some cures for the barren plateau have been proposed [22,30].

B. MNIST digits

We also test our classifier on the MNIST digits dataset. We test both binary classification (on digits 3 and 5) and multiclass classification (on digits 1, 3, and 5). Just as for the concentric circle case, the MNIST dataset is also generated by the `datasets` module of the `scikit-learn` package. The dimension of raw MNIST digit data from `scikit-learn` is an 8×8 array. The 8×8 array is reshaped into a one-dimensional array of dimension 64, which is transformed into parameters of the quantum annealer in our classifier, as described in our algorithm (see Fig. 1). There are three sine terms in the Fourier expansion, the number of time steps is 10, and the maximum time $t_{\max}$ used is 0.91 for the binary classification (on digits 3 and 5) and 0.91 for the multiclass classification (on digits 1, 3, and 5). The parameters h and J are fixed here, taking on values to make the initial and final Hamiltonians nondegenerate. Only the Fourier coefficients in the schedule are transformed from the data by a linear-transformation matrix. The loss used in the training stage for binary classification is $-D_{\min}/r_{\max}$, where $D_{\min}$ is the smallest distance between centroids of different classes and $r_{\max}$ is the largest spread of distance of a collection of embedded samples with respect to their corresponding centroid.

For the binary-classification task on digits 3 and 5, all the digits 3 and 5 are collected, and then a random 90%-to-10% split is performed to get training and test datasets (329 samples in the collection are picked out as the training dataset and 36 samples are chosen as the test dataset). For the three-class-classification task on digits 1, 3, and 5, all the digits 1, 3, and 5 are collected, and then a random 90%-to-10% split is performed to get training and test datasets (492 samples of the collection are picked out as the training dataset and 55 samples are chosen as the test dataset).

After training, the embedded quantum states from different labels show a clustering behavior. As usual, we visualize the clustering of embedded datasets from the same labels with the overlap matrix. Figure 6 shows the overlap matrix for digit classification. A clear separation is observed between different classes. A trained classifier is used here. The training dataset is fed into the annealer with the trained parameters. In a classification task, the final evolved quantum states of $s = 1$ are used as the embedded states for the data. We record the evolution of the quantum states at each time step. For each time step, the evolved quantum states are calculated and the overlap matrix is calculated. The image sequence is from $s = 0$ (the top-left image) to $s = 1$ (the bottom-right image). The brightness indicates the overlap between the embedded quantum states.

Next we discuss the influence of the number of qubits on performance. For binary classification on digits 3 and 5, increasing the number of qubits, in general, can improve performance. This is shown in Table I, where the training and test accuracies for classification on MNIST digits 3 and 5 are recorded for various numbers of qubits. Each result is averaged over eight random trials, starting from eight random initial guesses of the linear transformation

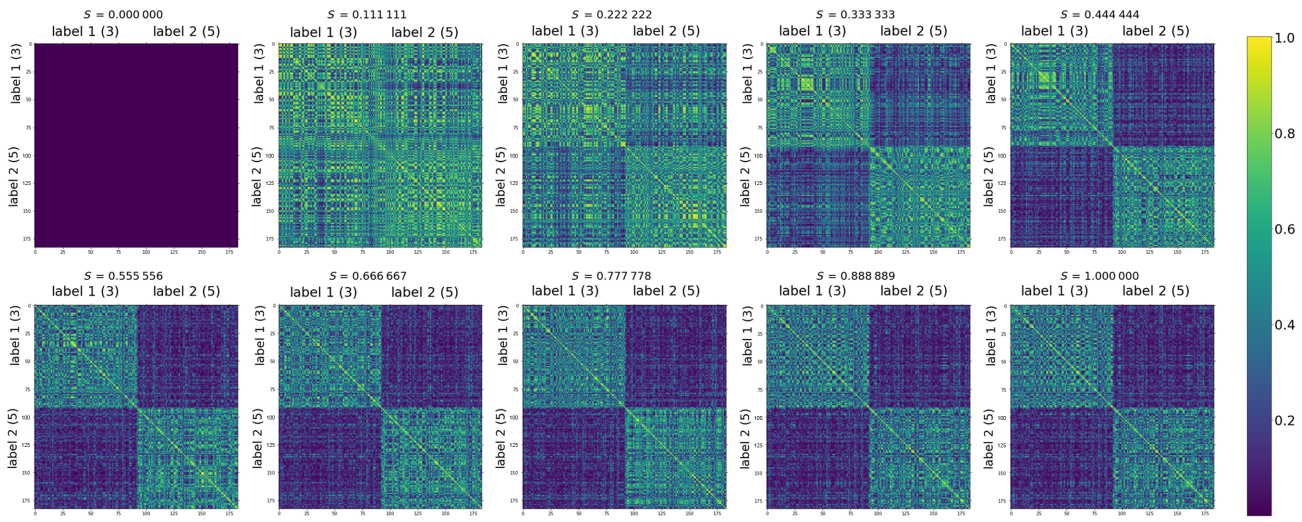

FIG. 6. Time evolution of overlap matrix of training data for the binary classification on MNIST digits 3 and 5. A trained classifier is used here. The image sequence is from $s = 0$ (top-left image) to $s = 1$ (bottom-right image). The brightness indicates the overlap between the embedded quantum states.

matrix $\hat{\boldsymbol{W}}$. For three-class classification on digits 1, 3, and 5, increasing the number of qubits can improve the performance prominently, as shown in Table II. Here each result is also averaged over eight random trials, starting from eight random initial guesses of the linear-transformation matrix $\hat{\boldsymbol{W}}$.

IV. DISCUSSION

These tests on MNIST digits and concentric circles show the power of our classifier on different datasets. In comparison with classical linear classifiers [23], our algorithm can significantly boost classification accuracy. The performance of our classifier is also comparable with that of the best classical classifiers. We test a few of the best classical classifiers on the same concentric circles dataset. For a three-layer (two 8-node hidden layers) neural-network classifier [12] with a rectified-linear-unit activation function (implemented with PyTorch [31]), the training accuracy and test accuracy are both 100% for the two-circle case and are 99.75% and 98.96%, respectively, for the three-circle case (for a simpler two-layer neural network, the training accuracy and test accuracy are both 100% for the two-circle case and are 99.29% and 98.9575%, respectively, for the three-circle case). For a support-vector-machine classifier [32–35] (Gaussian kernel, implemented with `scikit-learn` [36]), the training accuracy and test accuracy are both 100% for the two-circle case and are both 99.167% for the three-circle case. For a random-forest classifier [37–40] (with tree depth 2, implemented with `scikit-learn` [41]), the training accuracy and test accuracy are 100% and 99%, respectively, for the two-circle case and are 82.3% and 80.8%, respectively, for the three-circle case (for a depth-6 random-forest classifier, the training accuracy and test accuracy are both 100% for the two-circle case and are 99% and 97.5%, respectively, for the three-circle case). As a comparison, for our classifier, the training accuracy and test accuracy are both more than 99.5% for the two-circle case and are both more than 95% for the three-circle case (and both can be further increased to more than 99% when training is done with the $-D_{\min}/r_{\max}$ loss, where $D_{\min}$ is the smallest distance between centroids of different classes and $r_{\max}$ is the largest spread of distance to the centroid within a class; this loss means the separation between density matrices from collections of embedded data is larger than the spread of embedded data points with respect to their centroids).

It is worth mentioning that the number of optimization parameters in our classifier scales linearly with n (see Sec. II). The other parameters (such as h_{x_i}, h_{z_i}, and J_{ij}) will not change this scaling, since they can either be fixed as

TABLE I. Performance results for binary classification on MNIST digits 3 and 5. The training accuracy and test accuracy (average results from eight random trials) for quantum annealers with various numbers of qubits are recorded.

	One qubit	Two qubits	Three qubits	Four qubits	Five qubits
Training accuracy	0.9206	0.9850	0.9812	0.9931	0.9942
Test accuracy	0.8590	0.9359	0.9519	0.9679	0.9744

TABLE II. Performance results for a three-class classification on MNIST digits 1, 3, and 5. The training accuracy and test accuracy (average results from eight random trials) for quantum annealers with various numbers of qubits are recorded. We can see that increasing the number of qubits can greatly increase the training accuracy or test accuracy.

	Two qubits	Four qubits	Five qubits
Training accuracy	0.9098	1	1
Test accuracy	0.7864	0.9454	0.9523

constant or lead to a linear dependence on n when they are treated as variables. The linear scaling with n makes our classifier feasible in the noisy intermediate-scale quantum (NISQ) era. For a NISQ computer with approximately 100 qubits, the number of control parameters ($n_s \times m$) is approximately 1000. This is practical for controlling and measurement systems in the NISQ era.

In our tests, we intentionally do not optimize the hyperparameters (such as $t_{\max}$). Further adjustment of the hyperparameters could further boost the performance of our classifier. Use of a different loss in the training stage, such as $-D_{\min}/r_{\max}$, can also make the training better.

Regarding the quantum part of our classifier, the quantum system used in our classifier is an implementation of a quantum annealer, but this approach could be extended to other types of quantum computer run in an analog mode. As the number of qubits increases, the nonlinearity provided by the quantum computer, in general, cannot be simulated effectively with a classical computer. This could harbor a quantum advantage for quantum computation.

V. SUMMARY

We propose an analog quantum variational embedding classifier with a focus on an implementation based on a quantum annealer. The nonlinear mapping of the classical data to a high-dimensional density matrix is realized with an analog quantum computer. The classical data are transformed into the parameters of the analog quantum computer by a linear transformation, which implies that the nonlinearity needed for a nonlinear classification problem arises purely from the analog quantum computer due to the nonlinear dependence between the final quantum state and the control parameters of the Hamiltonian. By using a metric based on a density matrix from a collection of training data [20], our algorithm can handle a general classification problem. Moreover, our classifier handles both binary-classification and multiclass-classification tasks. We demonstrate the effectiveness of our algorithm for performing binary and multiclass classification on linearly inseparable datasets. Our algorithm performs much better than a classical linear classifier. The performance of our classifier is also comparable with that of the best classical classifiers. The dependence of performance on the number of qubits shows that increasing the number of qubits can improve performance until the performance saturates and fluctuates. In addition, the number of optimization parameters of our classifier scales linearly with the number of qubits. This linear scaling is an advantage of our classifier when compared with a classical neural network, whose number of training parameters scales quadratically [$O(n^2)$] with the number of nodes. Our algorithm presents the possibility of using current and near-term quantum annealers for solving practical machine-learning problems, and it could also be useful to explore quantum advantage in quantum machine learning.

As a prospect, in the future, topics such as the performance of our classifier with other types of Hamiltonian, the expressivity of the analog quantum computer, the universality of the nonlinearity, and the experimental realization on actual quantum computers could be investigated.

ACKNOWLEDGMENTS

This material is based on work supported by the Defense Advanced Research Projects Agency under Agreement No. HR00112109969.

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, *et al.*, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[2] Y. Wu, *et al.*, Strong Quantum Computational Advantage Using a Superconducting Quantum Processor, Phys. Rev. Lett. **127**, 180501 (2021).

[3] L. Madsen, F. Laudenbach, M. Askarani, F. Rortais, T. Vincent, J. Bulmer, F. Miatto, L. Neuhaus, L. Helt, M. Collins, A. Lita, T. Gerrits, S. Nam, V. Vaidya, M. Menotti, I. Dhand, Z. Vernon, N. Quesada, and J. Lavoie, Quantum computational advantage with a programmable photonic processor, Nature **606**, 75 (2022).

[4] Various cloud quantum computers, https://quantum-computing.ibm.com/services?services=systems, https://quantumai.google/cirq/hardware/rigetti/getting_started, https://quantumai.google/cirq/hardware/devices, https://ionq.com/, https://www.xanadu.ai/blog/beating-classical-computers-with-Borealis.

[5] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[6] Y. Zhao, *et al.*, Realization of an error-correcting surface code with superconducting qubits, e-prints ArXiv:2112.13505 (2021).

[7] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. Kraglund Andersen, M. Müller, A. Blais, C. Eichler, and A. Wallraff, Realizing repeated quantum error correction in a distance-three surface code, e-prints ArXiv:2112.03708 (2021).

[8] Google Quantum AI, *et al.*, Exponential suppression of bit or phase errors with cyclic error correction, Nature **595**, 383 (2021).

[9] R. P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. **21**, 467 (1981).

[10] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Rev. **41**, 303 (1999).

[11] L. K. Grover, Quantum Mechanics Helps in Searching for a Needle in a Haystack, Phys. Rev. Lett. **79**, 325 (1997).

[12] Y. Lecun, Y. Bengio, and G. Hinton, Deep learning, Nature **521**, 436 (2015).

[13] Q. V. Le, in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (IEEE, Vancouver, British Columbia, Canada, 2013), p. 8595.

[14] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature **549**, 195 (2017).

[15] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum principal component analysis, Nat. Phys. **10**, 631 (2014).

[16] S. Lloyd, S. Garnerone, and P. Zanardi, Quantum algorithms for topological and geometric analysis of data, Nat. Commun. **7**, 1 (2016).

[17] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum Support Vector Machine for Big Data Classification, Phys. Rev. Lett. **113**, 130503 (2014).

[18] S. Lloyd and C. Weedbrook, Quantum Generative Adversarial Learning, Phys. Rev. Lett. **121**, 040502 (2018).

[19] 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 **567**, 209 (2019).

[20] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, Quantum embeddings for machine learning, e-prints ArXiv:2001.03622 (2020).

[21] A. Little, M. Maggioni, and J. M. Murphy, Path-based spectral clustering: guarantees, robustness to outliers, and fast algorithms, e-prints ArXiv:1712.06206 (2017).

[22] W. Li and D.-L. Deng, Recent advances for quantum classifiers, Sci. China: Phys. Mech. Astron. **65**, eid220301 (2022).

[23] M. Noori, S. S. Vedaie, I. Singh, D. Crawford, J. S. Oberoi, B. C. Sanders, and E. Zahedinejad, Analog-Quantum Feature Mapping for Machine-Learning Applications, Phys. Rev. Appl. **14**, 034034 (2020).

[24] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, Perspectives of quantum annealing: Methods and implementations, Rep. Progr. Phys. **83**, eid054401 (2020).

[25] T. Caneva, T. Calarco, and S. Montangero, Chopped random-basis quantum optimization, Phys. Rev. A **84**, 022326 (2011).

[26] M. M. Müller, R. S. Said, F. Jelezko, T. Calarco, and S. Montangero, One decade of quantum optimal control in the chopped random basis, Rep. Progr. Phys. **85**, eid076001 (2022).

[27] S. Novikov, R. Hinkey, S. Disseler, J. I. Basham, T. Albash, A. Risinger, D. Ferguson, D. A. Lidar, and K. M. Zick, Exploring more-coherent quantum annealing, e-prints ArXiv:1809.04485 (2018).

[28] A gentle introduction to torch.autograd, https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html, PyTorch document.

[29] E. R. Anschuetz and B. T. Kiani, Beyond barren plateaus: Quantum variational algorithms are swamped with traps, e-prints ArXiv:2205.05786 (2022).

[30] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[31] Neural network, https://pytorch.org/docs/stable/generated/torch.nn.Module.html, PyTorch document.

[32] C. Cortes and V. Vapnik, Support-vector networks, Mach. Learn. **20**, 273 (1995).

[33] B. E. Boser, I. M. Guyon, and V. N. Vapnik, in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (Association for Computing Machinery, Pittsburgh, Pennsylvania, USA, 1992), p. 144.

[34] T. Hofmann, B. Schölkopf, and A. J. Smola, Kernel methods in machine learning, Ann. Stat. **36**, 1171 (2008).

[35] A. Aizerman, Theoretical foundations of the potential function method in pattern recognition learning, Autom. Remote Control **25**, 821 (1964).

[36] Support vector machines, https://scikit-learn.org/stable/modules/svm.html, scikit-learn document.

[37] L. Breiman, Random forests, Mach. Learn. **45**, 5 (2001).

[38] T. K. Ho, in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 1 (IEEE, Montreal, Quebec, Canada, 1995), p. 278.

[39] G. Louppe, Understanding random forests: from theory to practice, e-prints ArXiv:1407.7502 (2014).

[40] A. Géron, *Hands-On Machine Learning With Scikit-Learn, Keras, and TensorFlow* (O'Reilly Media, Inc., Sebastopol, California, USA, 2022).

[41] Random forest classifier, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, scikit-learn document.