Quantum classifiers with a trainable kernel

Li Xu, Xiao-yu Zhang, Ming Li,[*] and Shu-qian Shen
College of Science, China University of Petroleum, Qingdao 266580, People's Republic of China

(Received 21 January 2024; revised 4 April 2024; accepted 30 April 2024; published 28 May 2024)

Kernel function plays a crucial role in machine learning algorithms such as classifiers. In this paper, we aim to improve the classification performance and reduce the reading out burden of quantum classifiers. We devise a universally trainable quantum feature mapping layout to broaden the scope of feature states and avoid the inefficiently straight preparation of quantum superposition states. We also propose an improved quantum support vector machine that employs partially evenly weighted trial states. In addition, we analyze its error sources and superiority. As a promotion, we propose a quantum iterative multiclassifier framework for one-versus-one and one-versus-rest approaches. Finally, we conduct corresponding numerical demonstrations in the *qiskit* package. The simulation result of trainable quantum feature mapping shows considerable clustering performance, and the subsequent classification performance is superior to the existing quantum classifiers in terms of accuracy and distinguishability.

DOI: 10.1103/PhysRevApplied.21.054056

I. INTRODUCTION

Quantum algorithms have been widely studied in the past few decades, with the hope of overcoming some problems that are intractable to classical computers. Notable examples include Shor's quantum factoring algorithm [1], Grover's search algorithm [2], and the quantum algorithm for linear systems of equations proposed by Harrow, Hassidim, and Lloyd (the HHL algorithm) [3]. At the same time, machine learning has exploded into a hot research topic. The intersection between machine learning and quantum computation has burst into an interdisciplinary field, quantum machine learning, attracting massive academic research and application during the last decade [4–7].

A support vector machine (SVM) is a representative supervised machine-learning algorithm aiming to return an optimal hyperplane that separates two classes of samples with distinct labels and then assigns a label for an alternative datum determined by which side it lies [8,9]. Drawing support from the parallelism of quantum algorithms, quantum machine-learning algorithms could surpass their classical counterparts, especially in processing big data. Specifically, the least-squares quantum support vector machine (LS QSVM) inherits a logarithmic level complexity by means of superpositions and the HHL algorithm [10]. Meanwhile, the experiment verifies a proof-of-principle task [11]. However, as the dataset size multiplicatively increases, the distinguishability of LS QSVM becomes poor, which goes against "big data"(refer to Theorem 1). To overcome this challenge, we propose the support-vector-based quantum support vector machine (SV QSVM). Of note, if the data are fed classically, it will take $O(N)$ operations to load classical data into superposition, where N represents the data dimension [12–14]. This scaling can predominate the complexity of a quantum algorithm and, thereby, impair potential quantum advantages.

Recently, Schuld *et al.* and Havlíček *et al.* have proposed an artful path that could circumvent the direct preparation of coherent superpositions by mapping the data to an exponential Hilbert feature space [15,16]. This kernel-based method stimulates the widespread applications of noisy intermediate-scale quantum (NISQ) computers [17–21]. Here we collectively refer to the approaches only invoking classical optimizers in Refs. [15,16] as an *explicit approach*, and the approaches that utilize classical machine-learning algorithms are referred to as an *implicit approach*. We briefly retrospectively investigate these two approaches and analyze their discrepancies and drawbacks. Both approaches demand the use of quantum feature mapping (QFM) to load classical data into quantum feature states. The explicit approach applies a parameterized quantum circuit (PQC) $W(\theta)$ optimized during the training process to the feature state, then measures the final states in the Z basis. The classification result is revealed by some Boolean function $f : \{0,1\}^n \rightarrow \{1,-1\}$ through the output bit string, and the optimal parameters are acquired by minimizing the empirical risk function. However, the efficiency of the optimization phase is unsatisfactory, and what is more, it cannot achieve a satisfactory classification success rate. In contrast, the *implicit approach* utilizes quantum devices only for evaluating the kernel matrices and leaves the other assignments to classical computers. Hence, the *implicit approach* inherits the accurate calculation from

[*]Corresponding author: liming@upc.edu.cn, liming3737@163.com

the classical computers and weakness—intractable with handling big data.

To summarize, both approaches utilize QFM for loading classical information into quantum feature states, and then processing them in the Hilbert space. With an appropriate QFM, data can be repositioned ideally, i.e., data of the same category can be clustered, while data of different categories can be separated from each other. When given an ill-conditioned QFM, for the *explicit approach*, it will increase the burden of PQC to reposition the locations of data linearly. Even sometimes PQC cannot fully complete the tasks due to linearity. In contrast, the *implicit approach* will directly return an ill-conditioned kernel matrix, since it thoroughly relies on QFM to construct a kernel matrix. As a consequence, the follow-up classical machine-learning algorithm will show poor performance. Accordingly, finding a suitable QFM will bring significant benefits to both approaches. More generally, the properties of kernel matrices are crucial for machine-learning algorithms that rely on kernel functions.

Inspired by data reuploading strategy [22], we combine the *explicit approach* with the *implicit approach* to propose a trainable quantum feature mapping (TQFM) layout that owns multiple subsequent usability. On the one hand, a well-trained QFM circuit serves as the *explicit approach*, and it owns a better data-fitting effect because of nonlinear data reuploading. Although the reuploading model can be mapped to a linear explicit model, the overhead is high—$O(D\log D)$ additional qubits and gates, with D the number of encoding gates used by the data reuploading model [23]. On the other hand, the quantum feature states can be used to prepare kernel matrices, and then applied as subroutines to kernel-based machine-learning algorithms. Compared to the *explicit approach*, which relies only on training parameters and no longer uses samples for classification. The latter requires both the parameters and samples, that is, to fully utilize the information of samples. Therefore, this method is particularly useful for small sample learning.

The structure of the paper is as follows: Sec. II presents the construction of trainable quantum feature mapping. The three subsequent uses of TQFM is discussed simply in Sec. III. The algorithm flow of SV QSVM and error analysis is shown in Sec. IV. In Sec. V, we discuss the essence of multiclassification algorithms and propose quantum iterative multiclassifiers. The numerical simulations of Secs. II–V are exhibited in the four subsections of Sec. VI, respectively. Section VII concludes the primary work and future outlook.

II. TRAINABLE QUANTUM FEATURE MAPPING

The quantum feature mappings are superior to classical counterparts in respect of expressiveness and get benefit from the exponential Hilbert space, hence machine-learning algorithms with quantum kernels have the potential to achieve better performance than those with classical kernels. In this section, we propose the layout and loss function of trainable quantum feature mapping, and explore the optimal way to optimize the loss function.

A. Layout and loss function

The core idea of quantum kernel tricks that leads to its performance surpassing classical is the quantum kernel estimation, which maps n-dimensional classical data $x \in \mathbb{R}^n$ nonlinearly to a quantum state $|\psi(x)\rangle = U(x)|0\dots0\rangle$ with the general dimension 2^n. Here $U(x)$ is a PQC usually compiled by the feature of x that decides the unique quantum feature mapping, and its layout affects the behavior of QFM.

The central insight of quantum kernel is utilizing quantum circuits estimating the kernel functions $k^p(x_i,x_j) = |\langle\psi(x_i)|\psi(x_j)\rangle|^p$ in the Hilbert space, where p is a positive integer. Why it is not possible to directly apply the PQC $W(\theta)$ of *explicit approach* after $|\psi(x)\rangle$ and then estimate the kernel function is that $\langle\psi(x_i)|W^\dagger(\theta)W(\theta)|\psi(x_j)\rangle = \langle\psi(x_i)|\psi(x_j)\rangle$. Even if $W(\theta)$ possesses the capacity to reposition data, it seems to do some useless work when calculating kernel functions. A common option to circumvent this situation is to reupload data and integrate the layout of QFM with PQC, giving the formula of TQFM

$$|\psi(x,\theta)\rangle = U(x,\theta)|0\rangle^n,$$
$$U(x,\theta) = U_l(x,\theta_l)U_{\text{ent}}\dots U_2(x,\theta_2)U_{\text{ent}}U_1(x,\theta_1), \quad (1)$$

where θ is a family of parameters, and the functional relationship between x and θ is user specified, just like the activation function in neural networks. It should be pointed out that the construction of this function has different requirements depending on its subsequent use. If TQFM is directly used for classification (Sec. III A), more sophisticated functions are required, such as in Ref. [22]. If TQFM is used for preparing ensembles or kernel matrices, a linear function is sufficient. The layout of $U(x,\theta)$ is shown in Fig. 1(a).

B. Training stage

After introducing θ, it is the turn to optimize these parameters to cluster homogeneous data to improve the performance of quantum kernel functions [Fig. 1(b)]. Let $|y_j\rangle_{j=1}^L$ be a real label vector, and $\langle y_i|y_j\rangle = \delta_{ij}$, where δ_{ij} is the Kronecker δ function. Assuming that each y_j corresponds to M_j training samples. We minimize the following

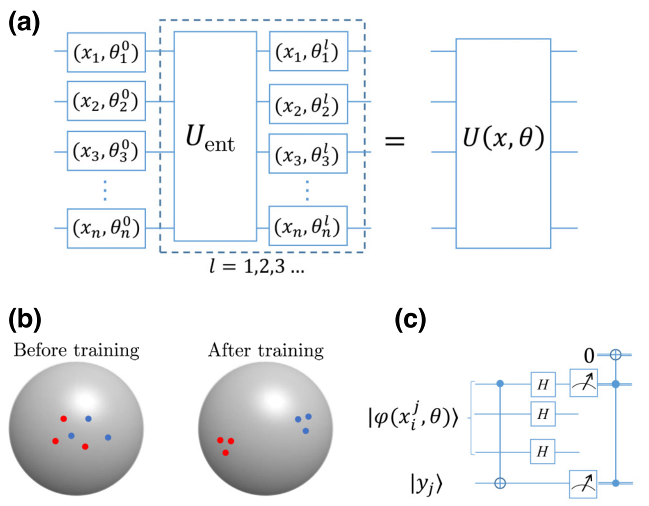


FIG. 1. (a) Circuits representation of $U(x,\theta)$. The single-qubit gates (x,θ) are Pauli rotations. U_{ent} are entangled quantum gates. (b) Illustration of trainable quantum feature mapping. After training, samples of the same category are clustered together, while samples of different categories are orthogonal to each other. (c) Optimal circuit diagram for optimizing loss function. The inner product is given by the probability difference of the classical bit.

loss function:

$$E(\theta) = 1 - \frac{1}{L}\sum_{j=1}^{L}\frac{1}{M_j}\sum_{i=1}^{M_j}|\langle\psi(x_i^j,\theta)|y_j\rangle|^2, \tag{2}$$

where x_i^j is the i th data in class j.

Take binary classification as an example, any binary classifier can be transformed into a multiclass classifier using the one-versus-one or one-versus-rest strategy. Minimizing Eq. (2) is equivalent to maximizing the value of $|\langle\psi(x_i,\theta)|\psi(x_j,\theta)\rangle|^2$ when x_i and x_j belong to the same category and minimizing the value when not. To calculate such an inner product using a quantum computer, one simply applies $U(x_i,\theta)$ and $U(x_j,\theta)$ to the initial state $|0\rangle^n$ individually and then perform a SWAP test. Each SWAP operator needs $2n+1$ qubits, l-layer PQC, and one set of measuring devices. As shown in Ref. [16], if the embedded circuit $U(x,\theta)$ can be inverted, one implements a circuit $U^\dagger(x_i,\theta)U(x_j,\theta)|0\rangle^n$, and measures the overlap to the $|0\rangle^n$ state. This inversion test needs only n qubits, $2l$-layer PQC, and n sets of measuring devices overall.

The authors of Ref. [24] have proposed multiple ways of optimizing the conventional SWAP test. After employing the protocol [Fig. 1(c)], the overhead of minimizing Eq. (2) can be decreased to $n+1$ qubits, l-layer PQC, and two sets of measuring devices in total. Of note, the reduced local loss function can be trained more effectively without suffering from barren plateaus. Supposing the labels of two categories are 1 and -1 we can rewrite the label vector $|y_a\rangle = |(1+a)/2\rangle|\cdot\rangle^{n-1}$, with $a=\pm1$. Removing the last $n-1$ qubits of label vector does not affect the orthogonality, it only has a certain impact on the compactness of clustering. Therefore, to save resources, swapping only the first qubit of $|\psi(x^a,\theta)\rangle$ with $|(1+a)/2\rangle$ is sufficient.

III. USABILITY OF TRAINABLE QUANTUM FEATURE MAPPING

Compared to QFM, the trainability (the number of trainable parameters) and nonlinearity of TQFM give it more powerful modes and more optional subsequent uses. Generally speaking, a TQFM with more parameters means stronger trainability. In this section, we present three subsequent options of TQFM in obtaining the optimal parameter θ^*.

A. Explicit approach

Explicit approach, also known as *quantum neural network* (QNN), consists of PQCs and gradient-based optimizers. TQFM can serve as a QNN since one just needs to compile datum x and θ^* into the TQFM framework, and then reveals the result by measuring the frequency of the label qubit. This is just one of the follow-up uses of TQFM, and we will not delve deeper into it here, as quantum neural networks have been widely studied [25–29].

B. Ensemble model

As the training data has already been clustered, one can prepare the ensembles for the two categories [30,31]

$$\rho^a = \sum_{i=1}^{M_a} p_i^a\rho_i^a = \sum_{i=1}^{M_a} p_i^a|\psi(x_i^a,\theta^*)\rangle\langle\psi(x_i^a,\theta^*)|,$$

where $p_i^a \geqslant 0$ and $\sum_i p_i^a = 1$, $a=\pm1$. If the l_1 distance is chosen, the label for the trial datum is given by

$$y = \text{sgn}\left(\text{Tr}(\rho^1\rho) - \text{Tr}(\rho^{-1}\rho)\right),$$

where $\rho = |\psi(x,\theta^*)\rangle\langle\psi(x,\theta^*)|$ embedded trial datum.

C. Quantum kernel matrix

After undergoing the TQFM stage, one can prepare a well-conditioned kernel matrix and then train an SVM for classification. For brevity, the inner product of $|\psi(x_i,\theta^*)\rangle$ and $|\psi(x,\theta^*)\rangle$ is represented by $k(x_i,x)$, and $M = M_1 + M_{-1}$. The classification expression of a general quantum SVM is as follows:

$$y = \text{sgn}(f(x)) = \text{sgn}\left(\sum_{i=1}^{M}\frac{\alpha_i y_i k(x_i,x)}{\sqrt{M}}\right), \tag{3}$$

where α_i are hyperparameters of the classification hyperplane and meet the requirements $\alpha_i \geqslant 0$ and $\sum_i \alpha_i^2 = 1$.

When facing diverse big data, Eq. (3) faces the dilemma of poor distinguishability.

Theorem 1. Assuming that the number of samples M is a large number and trial datum x is randomly chosen, then $f(x) = \sum_{i=1}^{M}((\alpha_i y_i k(x_i, x))/\sqrt{M})$ is scaling to $O(1/\sqrt{M})$.

Proof. Due to the randomness of x and M being a large number, we can deduce that $k(x_i, x)$ is independent and identically distributed with respect to x_i. Assuming $\alpha_i \propto 1/\sqrt{M}$, we obtain that $(\alpha_i y_i k(x_i, x))/\sqrt{M}$ is a distribution on the interval $[-(1/M), (1/M)]$. From the *central limit theorem*, it can be concluded that $f(x)$ roughly follows a normal distribution $N(0, (1/M))$. ■

Performing SWAP for k times allows one to estimate the value to an accuracy $\pm\sqrt{|f(x)|(1-|f(x)|)k}$ [30]. As the absolute value of $f(x)$ decreases with M increasing, maintaining the same accuracy of $f(x)$ requires more measurements. Unfortunately, the sign function is sensitive to values near 0. References [10,18] always encounter this challenge when processing big data, we will provide a solution to tackle this problem in the next section.

IV. QUANTUM SUPPORT VECTOR MACHINE

In the following, we analyze the advantages and disadvantages of SVM and LS QSVM, and provide the detailed process of SV QSVM algorithm.

SVM is an efficient binary classification algorithm with high robustness, which can process high-dimensional data with the help of kernel functions, although it requires a complex training process for seeking out support vectors. In return, SVM has extremely high classification efficiency because it relies only on support vectors to classify other data. Equipped with a well-trained quantum kernel function, the classification error rate of SVM will be significantly reduced. Compared with quantum support vector machines, classical SVM has always been at a disadvantage in computing big data due to the lack of quantum parallelism. LS QSVM overcomes the problem of computational scale, causing a decrease in distinguishability as the multiplication of samples due to treating all of them as support vectors.

We pick a compromise—support-vector-based variational quantum support vector machine—which can alleviate the issues of lower computational scale of classical SVM and the poor distinguishability of quantum classifiers. The fundamental reason why SV QSVM owns better distinguishability is that, like SVM, it uses only support vectors for classification, transforming the uniform superposition trial state related to sample size into support-vector-dependent ones. This is equivalent to amplifying the amplitude of the trial state, that is, increasing the value of $f(x)$.

A. Support vector machine

To get a soft margin SVM in the Hilbert space $\mathcal{H}$, we consider the convex quadratic optimization program

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + \frac{\gamma}{2}\sum_{i=1}^{M}\xi_i^2 \\ &\text{s.t.} \quad y_i(\langle w, \psi(x_i)\rangle_{\mathcal{H}} + b) \geqslant 1 - \xi_i, \end{aligned} \tag{4}$$

where $\xi_i \geqslant 0$ are slack variables, γ is a constant, $\{(x_i, y_i)\}_{i=1}^{M} \subseteq \mathbb{R}^n \times \{\pm 1\}$ are training samples, and w is a hyperplane in $\mathcal{H}$ that classifies data by decision function $y = \operatorname{sgn}(\langle w, \psi(x)\rangle_{\mathcal{H}} + b)$. Note that the optimization program can be solved efficiently with a countable dimension of $\mathcal{H}$. However, once mapping to an exponential dimensional space, it becomes intractable to find the optimal hyperplane. The key issue that leads to the success of SVM and bypasses this obstacle is optimizing the dual program of Eq. (4)

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{M}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{M}\alpha_i\alpha_j y_i y_j k(x_i, x_j) - \frac{1}{2\gamma}\sum_{i=1}^{M}\alpha_i^2 \\ &\text{such that} \quad \alpha_i \geqslant 0, \quad \sum_{i=1}^{M}\alpha_i y_i = 0, \end{aligned} \tag{5}$$

where $k(x_i, x_j)$ is the kernel function. As long as the kernel function can be effectively estimated, we can obtain a reasonable classifier by optimizing its dual form, giving an equivalent decision function with different expressions. Namely, $y = \operatorname{sgn}\left(\sum_{i=1}^{M}\alpha_i y_i k\langle x_i, x\rangle + b\right)$, the bias b can be replaced by $\sum_{i=1}^{M}((\alpha_i y_i)/\lambda)$, and see Appendix A for the detailed derivation process.

B. Training of a support-vector-based quantum support vector machine

To make it applicable to the SV QSVM model, we redraft the objective optimization function, Eq. (5), in the following form:

$$L(\alpha) = \min_{\alpha} \frac{1}{2}\langle\alpha|K|\alpha\rangle - \||\alpha\rangle\|_1 + Cl^2(\alpha), \tag{6}$$

where K_{ij} is the entry of kernel matrix K and $K_{ij} = y_i y_j k(x_i, x_j) + \delta_{ij}/\gamma$, $\||\alpha\rangle\|_1 = \alpha_1 + \alpha_2 + \ldots + \alpha_M$, C is a penalty multiplier and $l(\alpha) = \sum_{i=1}^{M/2}\alpha_i - \sum_{M/2+1}^{M}\alpha_i$. The first constraint $\alpha_i \geqslant 0$ can be automatically satisfied by limiting the angle of parameters, and the second constraint $\sum_{i=1}^{M}\alpha_i y_i = 0$ is given as a squared penalty term.

The first term of Eq. (6) is a variational quantum eigensolver (VQE) problem, a leading application for near-term devices [32–37]. VQE evaluates the ground-state energy

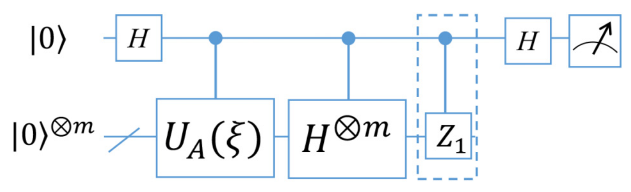


FIG. 2. Hadamard test for calculating $\||\alpha\rangle\|_1$(without C-Z gate) and $l(\alpha)$(with C-Z gate). U_A is PQC that prepares $|\alpha\rangle$, i.e., $|\alpha\rangle = U_A(\xi)|0\rangle^{\otimes m}$, with $m = \log M$. The subscript of controlled-Z gate represents the qubit applied.

for a Hamiltonian, which can be decomposed into a linear combination of unitary matrices by optimizing a PQC for generating the energy state. The most convenient representation basis is given by the Pauli operators. This is always efficient if the terms grow polynomially with qubit numbers. However, K may not admit a Pauli decomposition within polynomial terms, i.e., the summation of terms grows exponentially with the system size. In this case, the random sampling trick (Appendix B) takes over the mission since persisting in using Pauli decomposition can cause inefficiency and therefore loss of quantum advantage [38,39].

The other two terms can be calculated by a Hadamard test circuit (Fig. 2). The probability of measurement on the ancilla qubit is

$$P_a = \frac{1}{2}[1 + (-1)^a \langle 0^{\otimes m}|H^{\otimes m}|U_A|0^{\otimes m}\rangle],$$

with $a = 0$ or 1. The representation of $\||\alpha\rangle\|_1$ is given by $(P_0 - P_1)$ multiplied by the dimensional factor $\sqrt{2^m}$. The formulation of $l(\alpha)$ can be acquired in the same way by adding the *controlled-Z* gate.

C. Classification of support-vector-based quantum support vector machine

The classification stage can be applied when finishing the training stage since the feedback of optimal ξ^* is an essential prerequisite for classifying. Before building a classification circuit, we need to measure $U_A(\xi^*)$ to find out the positions of the support vectors, aiming at improving the distinguishability of trial data, as well as cutting down the circuit depth. The criterion for searching support vectors is whether the measurement frequency of the corresponding position is greater than a certain threshold, and nonzero values below this threshold are considered errors caused by noise.

The classification result will be revealed by

$$y = \mathrm{sgn}\,(f(x)) = \mathrm{sgn}\,(\langle \upsilon|\mu\rangle) = \mathrm{sgn}\left(\sum_{i=1}^{M} \frac{\alpha_i y_i k(x_i, x)}{\sqrt{m_s}}\right)$$

$$|\mu\rangle = \sum_{i=1}^{M} \alpha_i y_i c_i |i-1\rangle|\psi(x_i), \theta^*\rangle$$

$$|\upsilon\rangle = \sum_{i=1}^{M} \frac{c_i|i-1\rangle|\psi(x, \theta^*)\rangle}{\sqrt{m_s}},$$

where m_s is the number of support vectors, and $c_i = 1$ if and only if $|i-1\rangle$ corresponds to the positions of the support vectors, otherwise $c_i = 0$. Under the premise that the locations of support vectors are found, one could improve the distinguishability relatively by substituting the evenly weighted superposition $\sum_{i=1}^{M}((|i-1\rangle|\psi(x,\theta^*)\rangle)/\sqrt{M})$ with a partial evenly weighted state $|\upsilon\rangle$, since support vectors account only for a small portion. A commonly used method for this issue is amplitude amplification, which applies the Grover operator $G = H^{\otimes m} O H^{\otimes m} T$ to the evenly weighted superposition $R^G \leqslant \lceil(\pi/4)\sqrt{(M/m_s)}\rceil$ times and then returns a partially weighted state [2,40,41], where $T = \mathrm{diag}((-1)^{c_1}, (-1)^{c_2}, \ldots, (-1)^{c_M})$ and $O = \mathrm{diag}(2, 0, 0, \ldots, 0) - I_M$. Using partial superposition $|\upsilon\rangle$ for classification can improve the distinguishability of classification results, which can be simply understood as $(1/\sqrt{m_s}) > (1/\sqrt{M})$.

The SV QSVM gains rewards by constructing partial evenly weighted states that LS QSVM cannot achieve, as it applies displacement to all samples, regarding them as support vectors. Although the solution process has been accelerated, at the expense of losing the choice to utilize partial evenly weighted states, making it difficult to extract the symbol of $\langle \upsilon|\mu\rangle$. There are two commonly used methods to calculate whether the inner product is positive or negative. One is the Hadamard test, which needs to construct controlled unitary. The other one draws support from the SWAP test, but we need an ancilla to construct $|\tilde{a}\rangle = 1/\sqrt{2}(|0\rangle|0\ldots0\rangle + |1\rangle|a\rangle)$ $(a = \upsilon$ or $\mu)$ before testing.

We reintroduce the fault-tolerance classification equation

$$\mathrm{sgn}\left(\langle \upsilon|\mu\rangle \mp \frac{1}{\sqrt{k}}\right) = \pm 1.$$

to improve the robustness of the result and reduce the shot frequency [42], where k is the shot number.

D. Error analysis

In this subsection, we rigorously analyze the error source of SV QSVM and the advantages compared with LS QSVM in Ref. [10]. For a brief review of LS QSVM, see Appendix C.

Lemma 1 (Theorem 2.1 in Ref. [43]). Let x_0 be the solution to the quadratic program

$$\begin{aligned} \min \frac{1}{2}x^T K x - c^T x \\ \text{such that } Gx \geqslant g, \\ Dx = d, \end{aligned} \tag{7}$$

where K is noiseless and positive definite with the smallest eigenvalue $\lambda_{\min} > 0$. Let K' be a noisy positive definite matrix such that $\|K' - K\|_F \leqslant \varepsilon' < \lambda_{\min}$. Let x_0' be the solution to Eq. (7) with K replaced by K'. Then

$$\|x_0' - x_0\|_2 \leqslant \frac{\varepsilon'}{\lambda_{\min} - \varepsilon'}(1 + \|x_0\|_2).$$

Lemma 2. Let $h(x) = \sum_{i=1}^{m_s}((\alpha_i y_i k(x,x_i))/\sqrt{m_s})$ and $h'(x) = \sum_{i=1}^{m_s}((\alpha_i' y_i k(x,x_i))/\sqrt{m_s})$ to be the noiseless or noisy classifier,

$$|h'(x) - h(x)| \leqslant \epsilon$$

the measurement shots for kernel matrix K scaling as $O(M^4/\epsilon^2)$.

Proof. To guarantee $\|K' - K\|_F \leqslant \varepsilon'$, each matrix entry should take $O(M^2\varepsilon'^{-2})$ shots [16]. For each new datum x,

$$\begin{aligned} |h'(x) - h(x)| &= \left| \sum_{i=1}^{m_s} \frac{\alpha_i y_i k(x,x_i)}{\sqrt{m_s}} - \sum_{i=1}^{m_s} \frac{\alpha_i' y_i k(x,x_i)}{\sqrt{m_s}} \right| \\ &\leqslant \sum_{i=1}^{m_s} \left| \frac{\delta_i k(x,x_i)}{\sqrt{m_s}} \right| \\ &\leqslant \|\delta\|_2, \end{aligned}$$

with $\delta_i = \alpha_i' - \alpha_i$. The upper bound of $\|\delta\|_2$ depends on the minimum eigenvalue of K. Due to the positive definiteness of kernel matrix K, we have $\lambda_{\min}(K + (1/\gamma)I) \geqslant (1/\gamma)$ is lower bounded by a constant [17]. Then Lemma 1 gives

$$\|\delta\|_2 \leqslant O\left(\frac{M}{\sqrt{R}}\right) \leqslant \epsilon.$$

Due to the symmetry of the kernel matrix and the trivial diagonals, $(M^2 - M)/2$ matrix entries have to be estimated. Therefore, a total of $O(M^4/\epsilon^2)$ shots is sufficient for restricting $|h'(x) - h(x)| \leqslant \epsilon$. ■

Assuming trial datum x is taken from a uniform distribution and abundant, and $p(x)$ be the probability density function of $h(x)$ that satisfies

$$\int_{-1}^{1} p(x)dx = 1,$$

and $p'(x)$ corresponds to $h'(x)$. Due to statistical errors during the quantum kernel estimation phase, the classifier $h'(x)$ we use has a maximum error ϵ compared to the ideal classifier $h(x)$, which leads to a maximum misclassification rate $\int_0^{\epsilon} p(x)dx$ [integral between two black dashed lines of Fig. 3(a)].

The aforementioned analysis is under the condition that the exact value of $h'(x)$ is possessed, however, we merely have access to an estimated approximate value $\widetilde{h}'(x)$. In general, the accuracy requirement of the training stage is higher than classification, as slightly incorrect classifiers can lead to unpleasant classification accuracy. We provide a theoretical analysis below.

In the case $\varepsilon \geqslant \epsilon$ [Fig. 3(a)],

$$1 - \int_{-\varepsilon}^{\varepsilon} p(x)dx$$

is the probability of correct classification, and $\int_{-\varepsilon}^{\varepsilon} p(x)dx$ is inconclusive. However, with the increase of measurement shots, i.e., $\varepsilon < \epsilon$ [Fig. 3(b)], this will give the correct classification rate

$$1 - \int_{-\epsilon}^{\varepsilon} p(x)dx,$$

with $\int_{-\varepsilon}^{\varepsilon} p(x)dx$ inconclusive and $\int_{-\epsilon}^{-\varepsilon} p(x)dx$ wrong. The above analysis concludes that the benefits of increasing measurement shots during the classification stage are relatively minimal since the accuracy of classification is always affected by the errors ϵ stemming from quantum kernel estimation.

As the scale of data increases, no matter what kind of quantum classifiers, the probability density function becomes increasingly sharp as long as the query state is evenly weighted [blue line in Fig. 3(c)]. Some of these QSVMs can circumvent this dilemma by constructing the partially evenly weighted trail state—only for support vectors. Since the inconclusive rate $\int_{-\varepsilon}^{\varepsilon} p(x)dx < \int_{-\varepsilon}^{\varepsilon} p_{LS}(x)dx$, building up the query state according to support vectors is necessary.

V. QUANTUM ITERATIVE MULTICLASSIFIER

In this section, we analyze the essence of multiclassification algorithms and propose a quantum iterative multiclassifier framework.

A. Principle analysis and preparation

Classification problems are of great significance, especially multiclassification. The vast majority of multiclassifiers are derived from the generalization of binary classification. One can use one-versus-one or one-versus-rest strategies to transform binary classifiers into multiclassification.

When applying one-versus-one to L classification, one needs to prepare $(L(L-1))/2$ binary classifiers for each

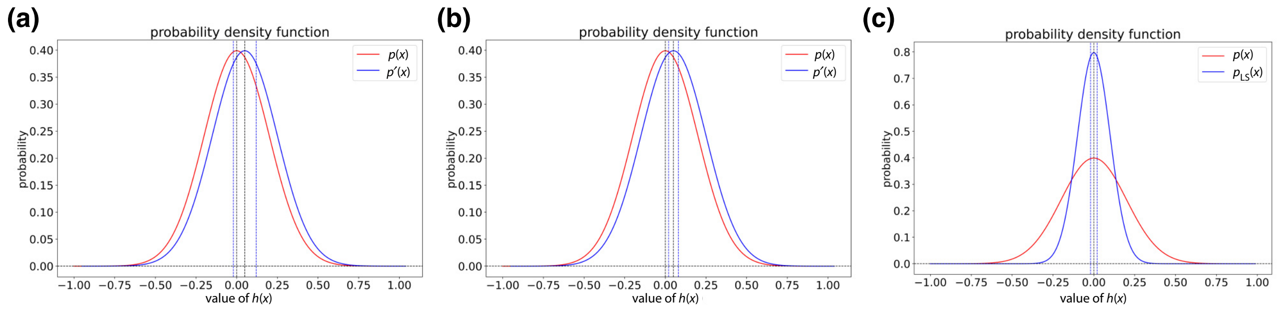


FIG. 3. Probability density functions in several different cases. The black dashed line represents the error between a noisy classifier and a noiseless classifier, while the blue dashed line represents the interval of the estimated value of the output of a noisy classifier. (a) Comparison between $p(x)$ and $p'(x)$ with $\varepsilon \geqslant \epsilon$. (b) Comparison between $p(x)$ and $p'(x)$ with $\varepsilon < \epsilon$. (c) Comparison between $p(x)$ and $p_{\mathrm{LS}}(x)$.

pair of different categories. There is a voting session after each classifier, and the eventual label is the one that earns the most votes. Adapting to the ensemble model, the above process can be simplified as finding the biggest trace distance between the L ensembles and the trial density matrix, that is

$$\arg\max_{j\in L}\left\{\mathrm{Tr}(\rho^j\rho)|\rho^j=\sum_i p_i^j|\psi(x_i^j,\theta^*)\rangle\langle\psi(x_i^j,\theta^*)|,\right.$$
$$\left.\rho=|\psi(x,\theta^*)\rangle\langle\psi(x,\theta^*)|\right\}.$$

When adopting a one-versus-rest approach, L classifiers are sufficient, and each one separates y_j from the others. The final label is given by the unique classifier that chooses y_j as positive. The above behavior can be explained as finding out $j\in L$ that satisfies

$$(\rho^j\rho)^{\mathrm{OVR}}=\mathrm{Tr}(\rho^j\rho)-\frac{1}{L-1}\sum_{k=1,k\neq j}^{L}\mathrm{Tr}(\rho^k\rho)>0.$$

The one-versus-rest model can also be adapted to SVM directly. More generally, no matter one-versus-one or one-versus-rest is chosen, the classification result can be interpreted as finding the largest element in sets $\{\mathrm{Tr}(\rho^j\rho)\}_{j=1}^L$ and $\{(\rho^j\rho)^{\mathrm{OVR}}\}_{j=1}^L$.

B. Algorithm flow

A trivial option is to calculate each element in turn, then the answer will be revealed naturally. However, this strategy is costly because calculating each element requires a large amount of measurement. Another artful way to read out the classification result is amplitude amplification as Grover search algorithm owns the capability to flip amplitude to the target qubit, then the qubit can be read out with a high probability—usually greater than 50%. To complete the above task, one needs to store all elements in the set proportionally in the amplitude of the quantum state. That is to say, U_L, U_x and their inversion need to be prepared

$$U_L|0\ldots0\rangle=\sum_{j=1}^{L}\frac{1}{\sqrt{L}}|j-1\rangle\sum_{i=1}^{M_j}\sqrt{p_i}|i-1\rangle|\psi(x_i^j,\theta^*)\rangle, \tag{8}$$

$$U_x|j-1\rangle|0\ldots0\rangle=\sum_{j=1}^{L}|j-1\rangle\sum_{i=1}^{M_j}\frac{|i-1\rangle|\psi(x,\theta^*)\rangle}{\sqrt{M_j}}, \tag{9}$$

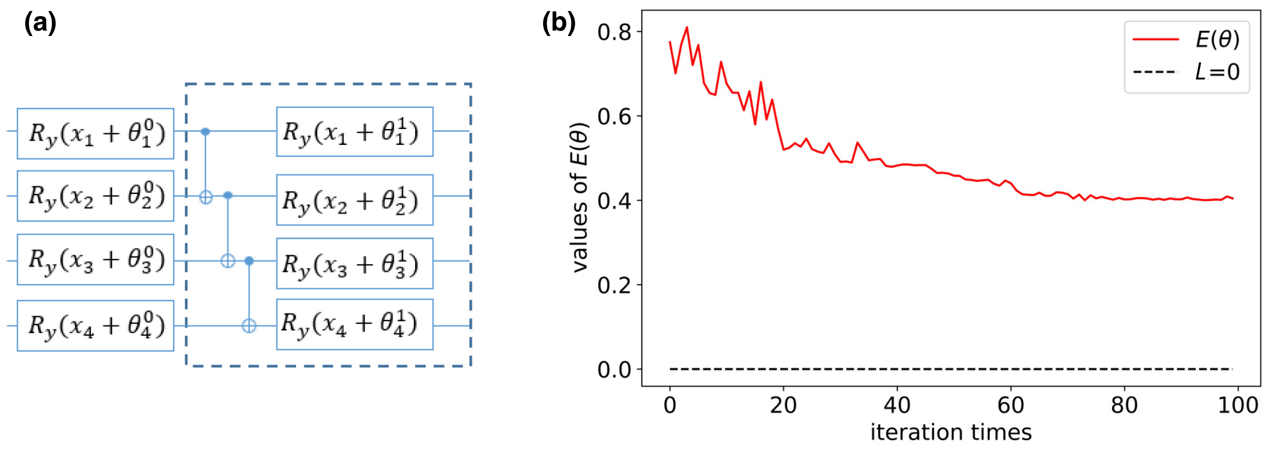
$$U_x^\dagger U_L|0\ldots0\rangle=|\phi_0\rangle=|\phi_1,\ldots,\phi_2,\ldots,\ldots,\phi_L,\ldots\rangle,$$

where the classification result $\{\mathrm{Tr}(\rho^j\rho)\}_{j=1}^L$ is stored in $\{\phi_j\}_{j=1}^L$ proportionately. For the convenience of explanation, we introduce a swap matrix S such that $S|\phi_0\rangle=|\phi\rangle=|\phi_1,\phi_2,\ldots,\phi_L,\mathrm{junk}\rangle$.

The workflow of a single Grover's search algorithm is as follows: (i) implement a phase inversion operator $T=\mathrm{diag}(-1_1,-1_2,\ldots,-1_L,1_{L+1},1,\ldots,1)$ to invert the first L phases of $|\phi\rangle$; (ii) apply $U_G^\dagger$ to $T|\phi\rangle$, $U_G=SU_x^\dagger U_L$; (iii) reverse the state by $|0\ldots0\rangle$, that is, implement an operator $O=\mathrm{diag}(1,-1,-1,\ldots,-1)$; (iv) apply U_G to step (iii). It should be noted that here $U_G|0\ldots0\rangle=|\phi\rangle$. Repeat the above operations for $R\in[1,(\sqrt{5M_j}\pi-1)/2]$ times roughly, the phases of classification results can be amplified. The proof of the number of iterations is provided in Appendix D.

VI. NUMERICAL VERIFICATION

In this section, we carry out the corresponding numerical simulations for the proposed algorithms to validate the feasibility in *Python qiskit* package. We target the IRIS dataset with $L=3$ labels and four features, and each class has 50 samples. In the classical optimization stage, taking the presence of measurement uncertainty into consideration,



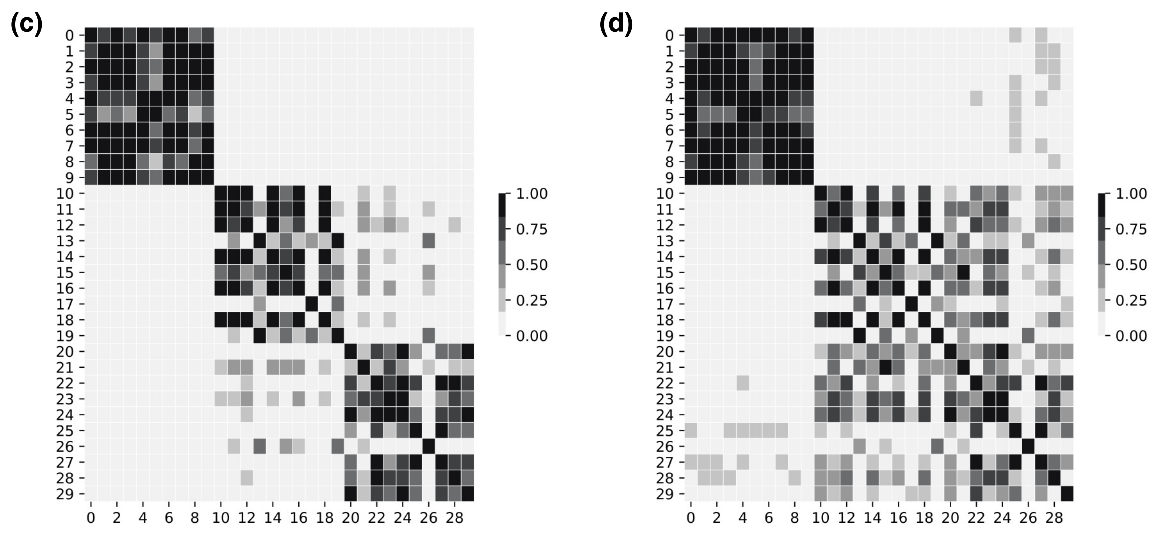


FIG. 4. Numerical verification of trainable quantum feature mapping. (a)A basic TQFM framework used in the simulations. (b) The iteration value of Eq. (10), and the starting point is $\theta = 0$. (c),(d) Visualization of overlaps between training samples. (c) is the kernel matrix of training data after training, i.e., $\theta = \theta^*$, and (d) is before training, i.e., $\theta = 0$.

we adopt *COBYLA* optimizer for convenience. Quantum gradient optimization methods can also be a good choice [44,45] as one could evaluate the derivatives of quantum expectation values from the output of variational quantum circuits.

A. Trainable quantum feature mapping

To verify the feasibility and effectiveness of trainable quantum feature mapping, we sample ten data points from each class as training data. Each feature plus an adjustable parameter is compiled into the amplitude of a parameterized quantum circuit. The layout of this simulation is shown in Fig. 4(a), the dashed box (also known as a layer) comprises entanglement gates and the reuploading gates. A better clustering effect can be achieved through training the circuit. In this simulation, the loss function becomes

$$E(\theta) = 1 - \frac{1}{3}\sum_{j=1}^{3}\frac{1}{10}\sum_{i=1}^{10}|\langle\psi(x_i^j,\theta)|y_j\rangle|^2, \qquad (10)$$

where $|y_j\rangle = |j-1\rangle|\cdot\rangle$.

The iterative process of optimizing function values is shown in Fig. 4(b). Only two layers of the repetition parts and the simple R_y rotations are used, leading the value of the loss function to reach only 0.4. We can achieve better training effects by increasing the complexity of the circuit, such as increasing the depth of the circuit and using R_z-R_y-R_z encoding. We will not delve too deeply into this matter.

The iteration of the loss function is completed with the optimal parameters θ^*. To present the training effects more directly, we visualized the overlaps between training data by drawing the hotspot matrices [Figs. 4(c) and 4(d)]. The

TABLE I. A comparison of explicit applications. SR and E mean success rate and loss function, respectively. The lowercase letters y and z represent Pauli rotation used in the validation, and r represents the number of layers.

	$r=0$		$r=1$		$r=2$	
	SR(%)	E	SR(%)	E	SR(%)	E
TQFM (y)	68	0.41	81.6	0.31	86	0.24
TQFM (zyz)	83.2	0.34	86	0.25	91.8	0.21
QFM (zyz)	63.6	0.45	84.4	0.34	89.8	0.25

training process further orthogonalized class 1 with classes 2, 3. What is more obvious is that class 2 and class 3, which were originally difficult to distinguish, have significantly improved their distinguishability after training. It can be seen that using the trained kernel matrix for machine-learning algorithms, such as SVM, will definitely outperform the untrained one in terms of performance.

B. Explicit application of training quantum feature mapping

For simplicity, we perform a binary classification validation and choose the *explicit approach* as a comparison. We choose class 2 and class 3 for the simulation because they are linearly inseparable, which better tests the performance of the classifier. We adopt the feature map $U_{\Phi(x)} = U_{\phi(x)}H^{\otimes n}U_{\phi(x)}H^{\otimes n}$ defined in Ref. [16] to generate the state and

$$U_{\phi}(x) = \bigotimes_{i=1}^{n} R_z(x^{(i)}).$$

Both the layouts of $W(\theta)$ and TQFM employ the R_z-R_y-R_z rotations, and R_y is taken for TQFM to be an object of reference. We select ten samples for training, then classify all 100 data and conduct simulations with different layers. The starting point is random, thus we conduct five simulations

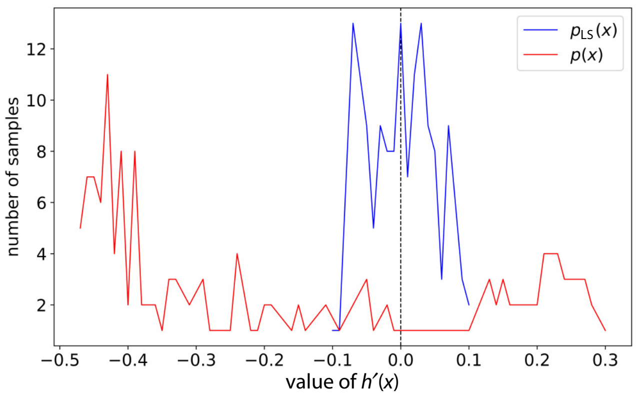


FIG. 5. Sample density distribution where the function value is rounded to two decimal places. Values greater than 0 are positive, while values less than 0 are negative.

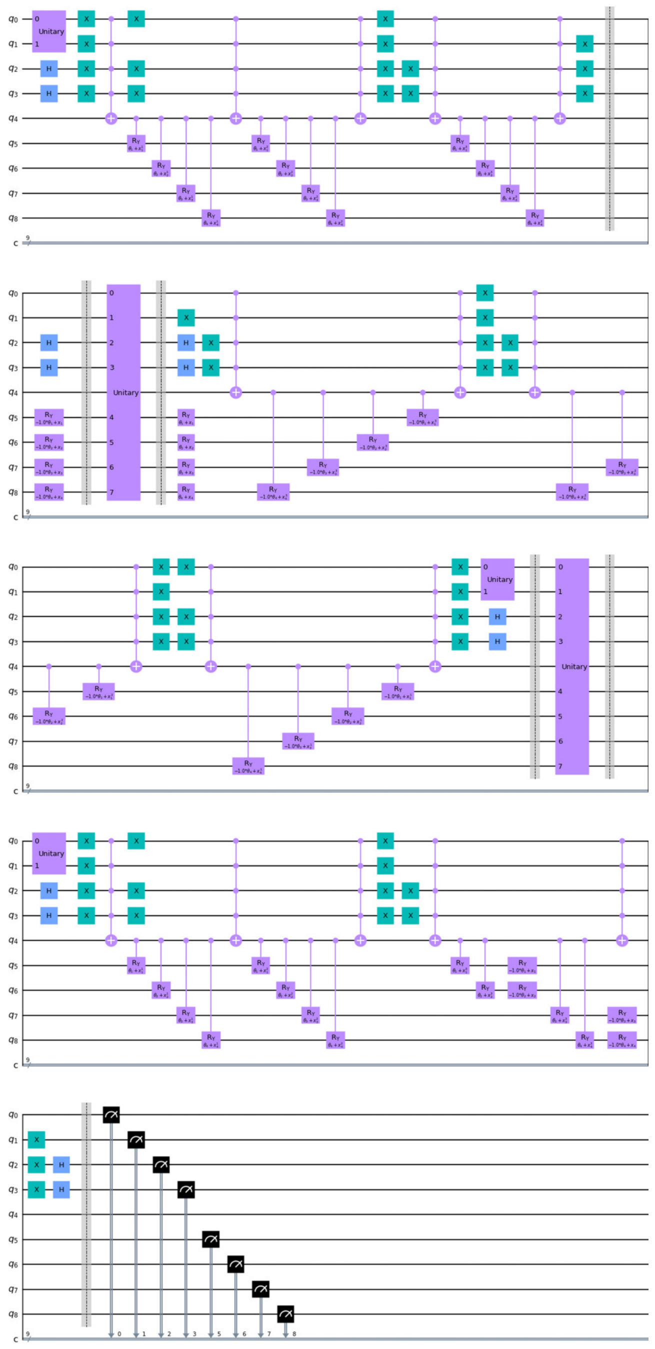


FIG. 6. Circuit diagram of quantum iterative multiclassifier with one iteration. ${}^{0}_{1}$Unitary$|00\rangle = ((\sqrt{3}/3), (\sqrt{3}/3), (\sqrt{3}/3), 0)$. The functions of the two ${}^{0,1,2,3}_{4,5,6,7}$Unitary are described by the third and fifth blocks.

to take the average value to eliminate the interference of randomness.

From the simulation results as shown in Table I, we deduce that TQFM has stronger classification performance than the *explicit approach* in Ref. [16], of course, we cannot rule out the dependence on data. But this at least reveals the advantages of TQFM in certain aspects, that is, TQFM

TABLE II. The probabilities and sample distribution of reading out IRIS dataset with 10 000 shots for each sample. r is the layers used in the TQFM stage. The column of <0.5 excludes the erroneous data.

	Before iteration			After iteration			Sample distribution		
Classes	max	min	ave	max	min	ave	>0.5	<0.5	error
$1(r=0)$	0.33	0.18	0.30	0.96	0.85	0.91	50	0	0
$2(r=0)$	0.29	0.09	0.23	0.65	0.23	0.47	20	29	1
$3(r=0)$	0.30	0.10	0.24	0.73	0.22	0.39	11	33	6
$1(r=1)$	0.33	0.12	0.28	0.96	0.66	0.81	50	0	0
$2(r=1)$	0.30	0.09	0.23	0.95	0.39	0747	44	6	0
$3(r=1)$	0.28	0.07	0.21	0.73	0.22	0.39	31	9	10
$1(r=2)$	0.32	0.12	0.26	1.00	0.71	0.94	50	0	0
$2(r=2)$	0.26	0.01	0.18	0.97	0.08	0.76	44	5	1
$3(r=2)$	0.26	0.02	0.19	1.00	0.14	0.73	42	1	7

can choose to use multiple layers to prepare the kernel matrix for classification, in order to further improve classification performance. Meanwhile, TQFM can also achieve a considerable success rate (over 80%) without introducing entanglement gates.

As a promotion, we prepare an ensemble model to test the subsequent use effect of TQFM that employs R_y rotations. We use the optimal parameters θ^* obtained from Table I, and select four samples to prepare the ensemble model, then classify 100 data. The average classification success rates for $r = 0$, 1, and 2 are 92%, 91.8%, and 94.8%, respectively. It can achieve a classification accuracy of up to 95% when using only 1/5 of the training samples with the simple use of R_y rotation.

C. Variational quantum support vector machine

For comparison, we conduct two numerical simulations to demonstrate the distinguishability of SV QSVM and choose LS QSVM [10] as a representative of quantum classifiers that employ evenly weighted superposition during classification. At present, the HHL algorithm can only be executed for several qubits, so we choose a variational quantum algorithm as an alternative [42].

Because the focus is on comparing branching, we only select eight samples (four for class 1, two for class 2, two for class 3, class 1 is the positive class, while the other two classes are negative) for training, and set the bias to be 0. When classifying, we test all 150 data points, and their classification distribution diagram is shown in Fig. 5. To approximate the value of $h(x)$ as accurately as possible, we set 4095 measurements. It is obvious that SV QSVM has much greater distinguishability than LS QSVM, which not only improves distinguishability but also reduces error rates, with actual accuracy rates of 100% and about 92%, respectively. Exporting the results for analysis, we find that the reason why the accuracy of the latter always cannot reach 100% is that the values located in the interval $[-0.05, 0.05]$ where the sign function is prone to errors accounts for a high proportion.

D. Quantum iterative multiclassifiers

In the numerical validation of multiclassifier, we conduct the one-versus-one strategy as an example. We take 12 samples (four per class) for classification and omit the training stage because assigning equal weights to these samples can also be used as a trivial training result. Equation (8) of this simulation is

$$U_L|0\dots0\rangle = \sum_{j=1}^{3}\frac{1}{\sqrt{3}}|j-1\rangle\sum_{i=1}^{4}\frac{1}{2}|i-1\rangle|\psi(x_i^j,\theta^*)\rangle.$$

The schematic diagram of the simulation is shown in Fig. 6. For the sake of simplicity, we only reserve three samples. There are a total of six barriers that divide the circuit into seven blocks. U_L is the first block, and U_x, which stores trial data, is the second block. The function of the third block is to mark the target phase, which is to flip the amplitude. The fourth block is the inverse of U_xU_L. The fifth block is a flipping operator, that is flipping all phases around $|0\dots0\rangle$. The sixth block is a repetition of blocks 1 and 2, and the last block is for measurement. There are a total of nine qubits, with q_0 and q_1 serving as labels, q_2 and q_3 storing the amplitude of sample data, q_4 being a working qubit, and q_5–q_8 being the quantum feature mapping.

In the classification validation stage, we classified all 150 data with a success rate of 95.3%, 93.3%, and 94.7%, respectively. The detailed distribution results are shown in Table II. We speculate that there may be two possible reasons why the classification success rate did not reach 100%: (i) Too few simulated samples were used, and only four samples in each class. Nonetheless, it exceeded the success rate of 92.5% in Ref. [31] (ii) The trainable quantum feature mapping structure we used is too simple and does not cluster perfectly during the training phase. From Table II, we can see that the amplitude can be significantly improved through iteration. The proportion of probability exceeding 0.5 (corresponding to an amplitude greater than 0.71) has increased from 81 to 136. This extremely reduces the burden of reading out results.

VII. CONCLUSION AND PROSPECT

In this paper, we propose a trainable quantum feature mapping strategy that has the power to deal with the nonlinear problems through data reuploading. A well-trained TQFM can be directly used for classification or as a subroutine in machine-learning algorithms [46,47]. We have conducted in-depth research on quantum classifiers equipped with TQFM and proposed the support-vector-based variational quantum support vector machine algorithm that shows a better distinguishability in the classification stage. We rigorously analyze the errors and advantages of SV QSVM. As a promotion, we demonstrate how to use a quantum search algorithm to iteratively read out multiclassification results with high probability.

We have numerically validated the proposed algorithms in the *qiskit* package, all the results are consistent with our analysis. We have trained the IRIS dataset with the simplest TQFM layout, and the resulting kernel matrix exhibits considerable clustering performance. We verify the preferable distinguishability by introducing LS QSVM as a comparison. The numerical simulation results not only demonstrate the better distinguishability of SV QSVM, but also surpass LS QSVM in terms of classification success rate. In respect of multiclassification, through iteration, the proportion of stored classification results with amplitude greater than 0.71 is greater than 90%. This is enough to ensure that one can accurately read out the results with a lower number of measurements.

It is necessary to mention the barren plateau phenomenon during the optimization process, as it is an open question that variational quantum algorithms need to face. As the number of qubits increases, the gradient becomes concentrated, causing the training to suffer from the curse of the barren plateau [48]. If the loss functions can be reduced to local loss functions, then their gradients vanish polynomially rather than exponentially in the number of qubits [49]. The recent advancements on the issue of barren plateaus have deeply explored this type of problem [50].

The optimization stage of SV QSVM may be undesirable for present-day NISQ devices. However, quantum annealers manufactured by *D*-Wave Systems are available with about 2000 qubits [51,52]. They automatically produce a variety of close-to-optimal solutions to an optimization problem. That is, the optimization process of SVM can be accomplished on *d*-wave annealers without haste [53]. This has broadened the path for current quantum machine learning.

ACKNOWLEDGMENTS

We thank Jin-min Liang for the helpful discussions and suggestions. This work is supported by the Shandong Provincial Natural Science Foundation for Quantum Science Grant No. ZR2021LLZ002 and the Fundamental Research Funds for the Central Universities Grant No. 22CX03005A.

APPENDIX A: SUPPORT VECTOR MACHINE

Here, we briefly review the classical support vector machine for binary classification. The mission of an SVM is assigning a label to test datum $x \in S \in \mathbb{R}^n$ depending on the decision function stems from training samples $\{x_i, y_i\}_{i=1,2,\ldots,M} \in T \times \{+1,-1\} \in \mathbb{R}^n \times R$. An optimal decision function can be generated by maximizing the distance $2/(\|w\|)$ between hyperplanes with opposite labels, equal to

$$\begin{gathered}\min \tfrac{1}{2}\|w\|^2 \\ \text{such that } y(w^T x_i + b) \geqslant 1, \\ i = 1, 2, \ldots, M.\end{gathered} \tag{A1}$$

This returns w for classification through $y = \text{sgn}\left(w^T x + b\right)$. The above assumption is under "ideal conditions," i.e., T is linearly separable. However, the most practical dataset is nonlinear separable, thus Eq. (A1) will get stuck with "bad conditions." By introducing slack variables, the above predicament can be overcome

$$\begin{gathered}\min \frac{1}{2}\|w\|^2 + \frac{\gamma}{2}\sum_{i=1}^{M} \xi_i^2 \\ \text{such that } y(w^T x_i + b) \geqslant 1 - \xi_i, \\ i = 1, 2, \ldots, M.\end{gathered} \tag{A2}$$

Although the above optimization can be directly solved, there is a more efficient choice by introducing the Lagrange multiplier, then deriving Lagrange dual form

$$\max_{\alpha} \sum_{i=1}^{M} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j K_0(x_i, x_j) - \frac{1}{2\gamma}\sum_{i=1}^{M} \alpha_i^2$$

$$\text{such that } \alpha_i \geqslant 0,\ \sum_{i=1}^{M} \alpha_i y_i = 0,\ i = 1, 2, \ldots, M.$$

This gives the optimization result

$$y = \text{sgn}\left(\sum_{i=1}^{M} \alpha_i y_i K_0(x, x_i) + b\right).$$

By adding the l_2-regularization term of bias $(\lambda/2)b^2$ to Eq. (A2) a relaxed decision function can be acquired

$$y = \text{sgn}\left(\sum_{i=1}^{M} \alpha_i y_i \left[K_0(x, x_i) + \frac{1}{\lambda}\right]\right), \tag{A3}$$

with $b = ((\sum_{i=1}^{M} \alpha_i y_i)/\lambda)$.

The primal Lagrangian is given by

$$L_P = \frac{1}{2}\|w\|^2 + \frac{\gamma}{2}\sum_i \xi_i^2 + \frac{\lambda}{2}b^2 - \sum_i \alpha_i(yw^T x + yb - 1 - \xi_i),$$

the dual optimal solutions Eq. (A3) can be connected via the Karush-Kuhn-Tucker (KKT) conditions

$$w = \sum_i \alpha_i x_i y_i,$$
$$0 = \sum_i \alpha_i y_i,$$
$$\xi_i = \frac{\alpha_i}{\gamma},$$
$$b = \sum_i \frac{\alpha_i y_i}{\lambda}.$$

APPENDIX B: RANDOM SAMPLING

The workflow of random sampling begins with refining the $M \times M$ kernel matrix K in terms of nonzero elements in the form

$$K = \sum_{(i)} K_{x^{(i)},y^{(i)}} |x^{(i)}\rangle\langle y^{(i)}| = \sum_{(i)} K_{x^{(i)},y^{(i)}} |x_1^{(i)}\rangle\langle y_1^{(i)}| \otimes |x_2^{(i)}\rangle\langle y_2^{(i)}| \otimes \ldots \otimes |x_M^{(i)}\rangle\langle y_M^{(i)}|,$$

where (i) represents the index of entries and $K_{x^{(i)},y^{(i)}}$ is the matrix entry at $(x^{(i)}, y^{(i)})$. In addition, we have

$$|0\rangle\langle 0| = (I + Z)/2,$$
$$|0\rangle\langle 1| = (X + iY)/2,$$
$$|1\rangle\langle 0| = (X - iY)/2,$$
$$|1\rangle\langle 1| = (I - Z)/2,$$

and K could be expanded as Pauli operators

$$K = \sum_{(i)} K_{x^{(i)},y^{(i)}} \sum_{j^{(i)}} \frac{1}{2^M}\sigma_{j^{(i)}},$$

with $\sigma_{j^{(i)}}$ being one of 2^M Pauli strings. At the end, to randomly measure K, we first sample (i) from $|K_{x^{(i)},y^{(i)}}|/q(i) = \sum_{(i)} |K_{x^{(i)},y^{(i)}}|$ and then toss m unbiased coins to decide $j^{(i)}$ for each $\sigma_{j^{(i)}}$. In particular, according to Hoeffding's inequality, we need $O(\log(\delta^{-1})\Lambda^2/\varepsilon^2)$ samples to an error ϵ with failure probability δ, where $\Lambda = \sum_j \lambda_j$.

APPENDIX C: QUANTUM LEAST-SQUARES SUPPORT VECTOR MACHINE

The core idea is the least-squares method, which transforms the quadratic process into solving linear equations. The introduction of slack variables replaces the inequality constraint of Eq. (A2) with the equality constraint, the restriction of $\xi_i \geqslant 0$ has been removed, this time ξ_i represents the degree to which the sample does not satisfy the constraint. After a series of algebraic operations

$$F\begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = \begin{pmatrix} 0 & \vec{1}^{\mathrm{T}} \\ \vec{1} & K_0 + \gamma^{-1}I \end{pmatrix}\begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{y} \end{pmatrix}. \tag{C1}$$

Here, the matrix F is $(M+1) \times (M+1)$ dimensional, $K_0(x_i, x_j) = \langle \psi(x_i)|\psi(x_j)\rangle$, $\vec{y} = (y_1; y_2; \ldots; y_M)$, and $\vec{1} = (1; 1; \ldots; 1)$. The hyperplane parameters can be obtained by solving Eq. (C1) in a quantum manner such as the HHL algorithm, variational quantum-classical algorithm etc.

The label for datum x will be revealed by

$$y = \mathrm{sgn}\left(\langle v|u\rangle\right),$$
$$|u\rangle = \frac{1}{\sqrt{N_u}}\left(b|0\rangle|0\rangle + \sum_{i=1}^{M} \alpha_i |i\rangle|\psi(x_i)\rangle\right),$$
$$|v\rangle = \frac{1}{\sqrt{N_x}}\left(|0\rangle|0\rangle + \sum_{i=1}^{M} |i\rangle|\psi(x)\rangle\right),$$

with the N_u and N_x normalization factor.

APPENDIX D: ESTIMATION OF ITERATION TIMES

In the estimation of iteration times, we proceed under the assumption of Theorem 1, and set each M_j, which is the number of category j is equal. The final state before applying amplitude amplification can be refined as

$$|\phi_0\rangle = a|a\rangle + \sqrt{1 - a^2}|b\rangle,$$

where $|a\rangle$ is the target phase, that is, the amplitude of these phases needs to be amplified. Totaling L phases, and each one stores the result. The sum of L amplitudes, i.e., the value of a, determines the number of iterations, is roughly $g(x) = \sum_{i=1}^{M_j}((\alpha_i k(x_i, x))/\sqrt{M_j})$. And $|b\rangle$ is a summary of the phase of storing junk information. The upper limit of a

is easily determined by

$$\begin{aligned} a &= \sum_{i=1}^{M_j} \frac{\alpha_i k(x_i, x)}{\sqrt{M_j}} \\ &\leqslant \|\alpha_i\| \|k(x_i, x)\| \\ &= \sqrt{\frac{k^2(x_1, x) + k^2(x_2, x) + \ldots + k^2(x_{M_j}, x)}{M_j}} \\ &= \frac{1}{\sqrt{3}}, \end{aligned}$$

where the second line uses Cauchy-Schwarz inequality and the third line uses the *central limit theorem*. Assuming a is a positive number, and the sign of a does not affect the number of iterations.

To estimate the lower bound of a, we need to use $\alpha_i \propto (1/\sqrt{M_j})$, then $g(x)$ flows a normal distribution $N(0, 1/(3M_j))$. So we have

$$\begin{aligned} &\int_0^a g(x)dx = \frac{1}{4} \\ &\int_0^a \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx = \frac{1}{4} \\ &a = \sqrt{\frac{2 \ln \frac{4}{3}}{3M_j}} \approx \frac{1}{\sqrt{5M_j}}. \end{aligned}$$

The reason we chose the interval between the median and maximum values is that the function is highly dense near the median point, and a well-trained quantum feature mapping has good distinguishability.

Then the iteration equation

$$\frac{a}{2} + Ra = \frac{\pi}{2}$$

gives $R \in [1, (\sqrt{5M_j}\pi - 1)/2]$.

[1] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. **26**, 1484 (1997).

[2] L. K. Grover, Quantum mechanics helps in searching for a needle in a haystack, Phys. Rev. Lett. **79**, 325 (1997).

[3] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. **103**, 150502 (2009).

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

[5] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: A review of recent progress, Rep. Prog. Phys. **81**, 074001 (2018).

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

[7] W. Ren, W. Li, S. Xu, K. Wang, W. Jiang, F. Jin, X. Zhu, J. Chen, Z. Song, and P. Zhang *et al.*, Experimental quantum adversarial learning with programmable superconducting qubits, Nat. Comput. Sci. **2**, 711 (2022).

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

[9] V. Vapnik, *Statistical Learning Theory* (Wiley, New York, NY, 1998), Vol. 3, Chapter 10-11, p. 401.

[10] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum support vector machine for big data classification, Phys. Rev. Lett. **113**, 130503 (2014).

[11] Z.-K. Li, X.-M. Liu, N.-Y. Xu, and J.-F. Du, Experimental realization of a quantum support vector machine, Phys. Rev. Lett. **114**, 110504 (2015).

[12] V. Giovannetti, S. Lloyd, and L. Maccone, Quantum random access memory, Phys. Rev. Lett. **100**, 160501 (2008).

[13] V. Giovannetti, S. Lloyd, and L. Maccone, Architectures for a quantum random access memory, Phys. Rev. A **78**, 052310 (2008).

[14] 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).

[15] M. Schuld and N. Killoran, Quantum machine learning in feature Hilbert spaces, Phys. Rev. Lett. **122**, 040504 (2019).

[16] 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).

[17] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, Nat. Phys. **17**, 1013 (2021).

[18] C. Blank, D. K. Park, J-K. K. Rhee, and F. Petruccione, Quantum classifier with tailored quantum kernel, npj Quantum Inf. **6**, 41 (2020).

[19] J. Jäger and R. V. Krems, Universal expressiveness of variational quantum classifiers and quantum kernels for support vector machines, Nat. Commun. **14**, 576 (2023).

[20] T. Goto, Q. H. Tran, and K. Nakajima, Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces, Phys. Rev. Lett. **127**, 090506 (2021).

[21] D. K. Park, C. Blank, and F. Petruccione, The theory of the quantum kernel-based binary classifer, Phys. Lett. A **384**, 126422 (2020).

[22] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Data re-uploading for a universal quantum classifier, Quantum **4**, 226 (2020).

[23] S. Jerbi, L. J. Fiderer, H. Poulsen Nautrup, J. M. Kübler, H. J. Briegel, and V. Dunjko, Quantum machine learning beyond kernel methods, Nat. Commun. **14**, 517 (2023).

[24] L. Cincio, Y. Subas, A. T. Sornborger, and P. J. Coles, Learning the quantum algorithm for state overlap, New J. Phys. **20**, 113022 (2018).

[25] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, Continuous-variable quantum neural networks, Phys. Rev. Res. **1**, 033063 (2019).

[26] Y. Du, M.-H. Hsieh, T. Liu, S. You, and D. Tao, Learnability of quantum neural networks, PRX Quantum **2**, 040337 (2021).

[27] Y. Du, Y. Yang, D. Tao, and M.-H. Hsieh, Problem-dependent power of quantum neural networks on multiclass classification, Phys. Rev. Lett. **131**, 140601 (2023).
[28] W. Li, Z. Lu, and D.-L. Deng, Quantum neural network classifiers: A tutorial, SciPhys. Lecture Notes 61 (2022).
[29] X. Pan, Z. Lu, W. Wang, Z. Hua, Y. Xu, W. Li, W. Cai, X. Li, H. Wang, and Y.-P. Song *et al.*, Deep quantum neural networks on a superconducting processor, Nat. Commun. **14**, 4006 (2023).
[30] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, Quantum embeddings for machine learning, ArXiv:2001.03622.
[31] N. A. Nghiem, S. Y.-C. Chen, and T.-C. Wei, Unified framework for quantum classification, Phys. Rev. Res. **3**, 033056 (2021).
[32] A. Peruzzo, J. McClean, P. Shadbolt, Man-Hong Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).
[33] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature **549**, 242 (2017).
[34] Y. Nam, J.-S. Chen, N. C. Pisenti, K. Wright, C. Delaney, D. Maslov, K. R. Brown, S. Allen, J. M. Amini, and J. Apisdorf *et al.*, Ground-state energy estimation of the water molecule on a trapped-ion quantum computer, npj Quantum Inf. **6**, 33 (2020).
[35] D. Wang, O. Higgott, and S. Brierley, Accelerated variational quantum eigensolver, Phys. Rev. Lett. **122**, 140504 (2019).
[36] R. M. Parrish, E. G. Hohenstein, P. L. McMahon, and T. J. Martínez, Quantum computation of electronic transitions using a variational quantum eigensolver, Phys. Rev. Lett. **122**, 230401 (2019).
[37] W. M. Kirby and P. J. Love, Variational quantum eigensolvers for sparse hamiltonians, Phys. Rev. Lett. **127**, 110503 (2021).
[38] X. Xu, J. Sun, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan, Variational algorithms for linear algebra, Sci. Bull. **66**, 2181 (2021).
[39] X. Wang, Z. Song, and Y. Wang, Variational quantum singular value decomposition, Quantum **5**, 483 (2021).
[40] J. Jones, M. Mosca, and R. Hansen, Implementation of a quantum search algorithm on a quantum computer, Nature **393**, 344 (1998).
[41] G. L. Long, Grover algorithm with zero theoretical failure rate, Phys. Rev. A **64**, 022307 (2001).
[42] L. Xu, X.-Y. Zhang, J.-M. Liang, J. Wang, M. Li, L. Jian, and S.-Q. Shen, Variational quantum support vector machine based on Hadamard test, Commun. Theor. Phys. **74**, 055106 (2022).
[43] J. W. Daniel, Stability of the solution of definite quadratic programs, Math. Program. **5**, 41 (1973).
[44] B. Koczor and S. C. Benjamin, Quantum analytic descent, Phys. Rev. Res. **4**, 023017 (2022).
[45] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).
[46] Y.-B. Sheng and L. Zhou, Distributed secure quantum machine learning, Sci. Bull. **62**, 1025 (2017).
[47] W. Li and D.-L. Deng, Recent advances for quantum classifiers, China Phys. Mech. Astron. **65**, 220301 (2022).
[48] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nat. Commun. **9**, 4812 (2018).
[49] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, Nat. Commun. **12**, 1791 (2021).
[50] M. Cerezo, M. Larocca, D. García-Martín, N. L. Diaz, P. Braccia, E. Fontana, M. S. Rudolph, P. Bermejo, A. Ijaz, and S. Thanasilp *et al.*, Does provable absence of barren plateaus imply classical simulability? Or, why we need to rethink variational quantum computing, ArXiv:2312.09121.
[51] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, and P. Bunyk *et al.*, Quantum annealing with manufactured spins, Nature **473**, 194 (2011).
[52] E. Gibney, *D*-Wave upgrade: How scientists are using the world's most controversial quantum computer, Nature **541**, 447 (2017).
[53] D. Willsch, M. Willsch, H. De Raedt, and K. Michielsen, Support vector machines on the *D*-wave quantum annealer, Comput. Phys. Commun. **248**, 107006 (2020).