Variational quantum support vector machine based on Hadamard test

© 2022 Institute of Theoretical Physics CAS, Chinese Physical Society and IOP Publishing Printed in China and the UK Communications in Theoretical Physics

Commun. Theor. Phys. **74** (2022) 055106 (9pp) https://doi.org/10.1088/1572-9494/ac6358

Variational quantum support vector machine based on Hadamard test

Li Xu[1], Xiao-Yu Zhang[1], Jin-Min Liang[2], Jing Wang[1], Ming Li[1], Ling Jian[3] and Shu-qian Shen[1]

[1] College of Science, China University of Petroleum, Qingdao 266580, China
[2] School of Mathematical Sciences, Capital Normal University, Beijing 100048, China
[3] School of Economics and Management, China University of Petroleum, Qingdao 266580, China

E-mail: liming@upc.edu.cn

Received 15 January 2022, revised 2 April 2022
Accepted for publication 2 April 2022
Published 18 May 2022

Abstract

Classical machine learning algorithms seem to be totally incapable of processing tremendous amounts of data, while quantum machine learning algorithms could deal with big data with ease and provide exponential acceleration over classical counterparts. Meanwhile, variational quantum algorithms are widely proposed to solve relevant computational problems on noisy, intermediate-scale quantum devices. In this paper, we apply variational quantum algorithms to quantum support vector machines and demonstrate a proof-of-principle numerical experiment of this algorithm. In addition, in the classification stage, fewer qubits, shorter circuit depth, and simpler measurement requirements show its superiority over the former algorithms.

Keywords: quantum support vector machine, Hadamard test, variational quantum algorithm

(Some figures may appear in colour only in the online journal)

1. Introduction

Machine learning [1] is a multi-field interdisciplinary subject, which is based on studying how computers simulate or realize human learning behavior to acquire new knowledge or skills, then reorganize the existing knowledge structure and improve its performance. There are two types of machine learning, supervised learning and unsupervised learning, the difference is whether the label is stamped in advance. A support vector machine (SVM) [1, 2] for classification is a frequently-used supervised learning algorithm. Like all machine learning algorithms, SVM is weak in facing billions and trillions of training data. Due to the exponential acceleration of quantum algorithms over classical algorithms [3, 4], a quantum support vector machine (QSVM) [5] emerges as the times require, and the experiment also proves its feasibility [6]. In the last decade, research and application of SVM has always been a hot topic, such as least squares SVM (LS-SVM) [7, 8], multi-class SVM [9–11], quantum multi-class SVM [12, 13], and twin support vector machine [14–16] etc.

In recent years, many classical algorithms that can replace quantum algorithms have been proposed, and they also achieve the exponential acceleration that quantum algorithms can achieve. Due to Tang's [17] query and challenge to quantum machine learning, many quantized versions of classical machine learning are eclipsed. Since quantum random access memory [18, 19] may be replaced by 'sample and query access' and quantum machine learning algorithm is not accelerated in the process, hitting a number of scholars who study how to quantize classical algorithms. Before the advent of quantum computers, we could not get an accurate result from this debate. At this time, the practicability of the variational quantum algorithm shows its value. Up to now, variational quantum algorithms have been widely used and solved some problems that are beyond the power of classical computers [20–24].

The manuscript is organized as follows. We first review the SVM and QSVM algorithms and give the quantum circuit diagrams. We then propose a variational QSVM algorithm that uses the most fundamental measurement method—Hadamard Test—to complete the classification stage. Finally, the numerical experiment shows the feasibility of the algorithm.

2. Support vector machine

An SVM is used for binary classification of data, which classifies $\{(\vec{x}_i, y_i): \vec{x}_i \in \mathbb{R}^N, y_i = \pm 1\}(i = 1, 2,\ldots,M)$ into two classes, where y_i is the label of $\vec{x}_i$. Whether y_i equals $+1$ or -1 depends on which class $\vec{x}_i$ belongs to. For classification, two hyperplanes are found to separate the data, with no data inside. The maximum distance between hyperplanes is $2/\|\vec{\omega}\|$, where $\vec{\omega}$ is the normal vector. Generally speaking, $\vec{\omega}\cdot\vec{x}_i + b \geqslant 1$ constructs $+1$ class, correspondingly, $\vec{\omega}\cdot\vec{x}_i + b \leqslant -1$ constructs -1 class. Based on the above conditions, the expression of the SVM can be written as

$$\begin{aligned}&\min_{\omega,b}\frac{1}{2}\|\omega\|^2\\ &s.t. y_i(\vec{\omega}\cdot\vec{x}_i + b) \geqslant 1,\ i = 1, 2,\ldots,M. \qquad (1)\end{aligned}$$

By adding the Lagrange multiplier $\vec{\alpha} = (\alpha_1; \alpha_2;\ldots;\alpha_N)$ to equation (1) and taking partial derivatives, the dual problem is obtained

$$\begin{aligned}&\max_{\vec{\alpha}}\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j K_{ij}\\ &s.t.\sum_{i=1}^{N}\alpha_i y_i = 0,\\ &\alpha_i \geqslant 0,\ i = 1, 2,\ldots,M, \qquad (2)\end{aligned}$$

where the kernel function is defined as $K_{ij} = k(x_i, x_j) = \phi(x_i)^{\mathrm{T}}\phi(x_j)$. The Karush–Kuhn–Tucker [1] condition should be satisfied in equation (2). A binary classifier for new data $\vec{x}$ can be derived as

$$y(\vec{x}) = \mathrm{sgn}(\sum_{i=1}^{M}\alpha_i k(\vec{x}_i, \vec{x}) + b).$$

3. Quantum least squares support vector machine

In this section, we give a short review of [5, 6]. Suppose there are oracles that encode training data to quantum states

$$|\vec{x}_i\rangle = \frac{1}{|\vec{x}_i|}\sum_{k=1}^{N}(\vec{x}_i)_k|k\rangle.$$

The core idea is the least square method, which transforms the quadratic process into solving linear equations. The introduction of slack variables replaces the inequality constraint with the equality constraint

$$\begin{aligned}&\min_{\omega,b,e}\frac{1}{2}\|\omega\|^2 + \frac{\gamma}{2}\sum_{i=1}^{M}e_i^2\\ &s.t.\quad y_i = \omega^{\mathrm{T}}\phi(x_i)\\ &+b + e_i,\ i = 1, 2,\ldots,M. \qquad (3)\end{aligned}$$

The slack variables e_i represent the degree to which the sample does not satisfy the constraint, where user-specified γ determines the relative weight of the training error and the SVM objective. The following equation can be obtained by calculating the partial derivative of the dual Lagrange problem

$$F\begin{pmatrix}b\\ \vec{\alpha}\end{pmatrix} = \begin{pmatrix}0 & \vec{1}^{\mathrm{T}}\\ \vec{1} & K_{ij} + \gamma^{-1}I\end{pmatrix}\begin{pmatrix}b\\ \vec{\alpha}\end{pmatrix} = \begin{pmatrix}0\\ \vec{y}\end{pmatrix}. \qquad (4)$$

Here, the matrix F is $(M+1)\times(M+1)$ dimensional, $K_{ij} = \vec{x}_i^{\mathrm{T}}\cdot\vec{x}_j$ is a symmetric kernel matrix, $\vec{y} = (y_1; y_2;\ldots;y_M)$, and $\vec{1} = (1; 1;\ldots;1)$. The hyperplane coefficients b and $\vec{\alpha} = (\alpha_1; \alpha_2;\ldots;\alpha_M)$ can be optimized by solving $(b; \vec{\alpha}) = F^{-1}(0; \vec{y})$. Quantum exponential speed-up algorithms for solving linear equations have been proposed in [25].

QSVM algorithms are divided into two parts: training and classification, as shown in figure 1.

The quantum states are defined as

$$\begin{aligned}|0, \vec{y}\rangle &= \frac{1}{\sqrt{N_{0,y}}}(|0\rangle + \sum_{i=1}^{M}y_i|i\rangle),\\ |b, \vec{\alpha}\rangle &= \frac{1}{\sqrt{N_{b,\alpha}}}(b|0\rangle + \sum_{i=1}^{M}\alpha_i|i\rangle),\end{aligned}$$

where $N_{0,y} = 1 + M^2$ and $N_{b,\alpha} = b^2 + \sum_{k=1}^{M}\alpha_k^2$ are normalization factors. Take notice that if and only if the auxiliary qubit measurement result is on $|1\rangle$, the matrix inversion operation is successful. Under this condition, the training-data oracle transforms $|b, \vec{\alpha}\rangle$ to $|u\rangle$,

$$|u\rangle = \frac{1}{\sqrt{N_u}}(b|0\rangle|0\rangle + \sum_{i=1}^{M}\alpha_i|\vec{x}_i||i\rangle|\vec{x}_i\rangle),$$

with $N_u = b^2 + \sum_{k=1}^{M}\alpha_k^2|\vec{x}_k|^2$.

The whole process of the training section is as follows:

$$\begin{aligned}&|0\rangle|0, \vec{y}\rangle|0\rangle \to |0\rangle|0, \vec{y}\rangle\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\\ &\to\frac{1}{\sqrt{2}}|0\rangle|0, \vec{y}\rangle|1\rangle + \frac{1}{\sqrt{2}}|00\rangle|0\rangle\\ &\to\frac{1}{\sqrt{2}}|0\rangle|b, \vec{\alpha}\rangle|1\rangle + \frac{1}{\sqrt{2}}|00\rangle|0\rangle\\ &\to\frac{1}{\sqrt{2}}|u\rangle|1\rangle + \frac{1}{\sqrt{2}}|00\rangle|0\rangle.\end{aligned}$$

In the classification section, the decision function $y(\vec{x}_0) = \mathrm{sgn}(\langle x_0|u\rangle)$ is constructed first. The query state

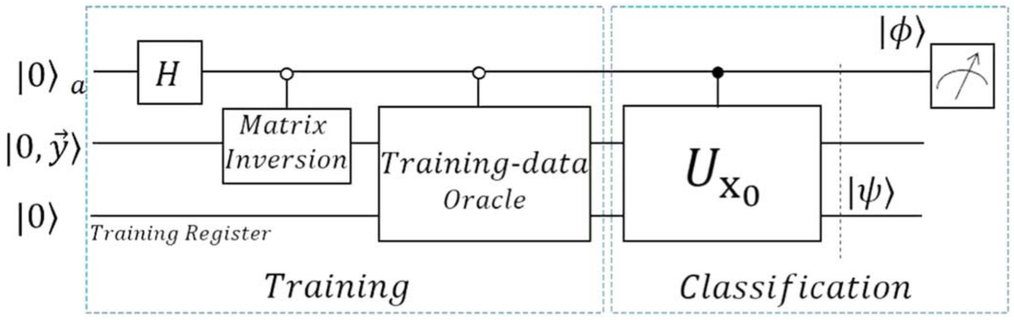
$$|x_0\rangle = \frac{1}{\sqrt{N_{x_0}}}(|0\rangle|0\rangle + \sum_{i=1}^{M}|\vec{x}_0||i\rangle|\vec{x}_0\rangle),$$

where $N_{x_0} = M|\vec{x}|^2 + 1$. When constructing the query state $|x_0\rangle$, U_{x_0} needs to be introduced which transfers $|x_0\rangle$ to $|00\rangle$, i.e., $U_{x_0}|x_0\rangle = |00\rangle$.

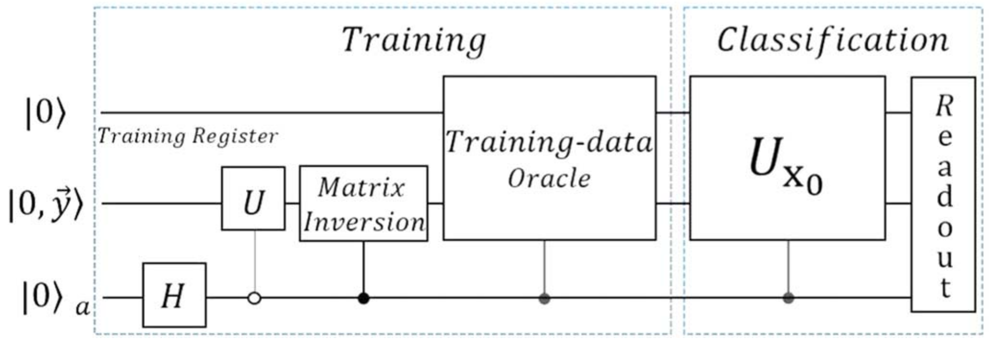
Obviously, the final state should be $|\psi\rangle = |\phi\rangle|1\rangle_a + |00\rangle|0\rangle_a$, with $|\phi\rangle = U_{x_0}|u\rangle$, $|0\rangle_a$ and $|1\rangle_a$ are ancillary qubits. The classification result is then obtained as

$$y(\vec{x}_0) = \mathrm{sgn}(\langle 00|\phi\rangle) = \mathrm{sgn}(\langle\psi|O|\psi\rangle), \qquad (5)$$

where $|O\rangle = |00\rangle\langle 00|\otimes(|1\rangle\langle 0|)_a$. If the result is greater than 0, x_0 belongs to the $+1$ class. Otherwise, x_0 belongs to the -1 class.



(a) QSVM circuit in [5].



(b) QSVM circuit in [6].

Figure 1. Brief circuit diagram of QSVM. The Matrix Inverse is employed to acquire the hyperplane parameters. The Training-data Oracle prepares $|u\rangle$. The information of the query state is stored in U_{x_0}. The auxiliary qubit used to solve the equation is omitted. (a) $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|u\rangle + |1\rangle|x_0\rangle)$ and $|\phi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. (b) U resets $|0, \vec{y}\rangle$ to $|00\rangle$. The QSVM algorithm in this section is based on (b).

4. Variational quantum support vector machine

In this section, we propose a variational QSVM algorithm that divides training and classification into two parts and perform a numerical simulation to verify the practicability.

4.1. Training part

The matrix F in (4) can be decomposed in the form of

$$F = \sum_{i=1}^{m} \delta_i P_i, \tag{6}$$

where

$$P_i = \bigotimes_{\beta=1}^{n} \sigma_t^{\beta},$$

$t = 0, x, y, z$, that is, σ_t is the Pauli matrix. In order to get each δ_i, we need to use the following formula

$$\mathrm{Tr}[P_i F] = \mathrm{Tr}[P_i \sum_{l=0}^{m} c_l P_l] = \mathrm{Tr}[\delta_i P_i P_i] = \delta_i 2^n.$$

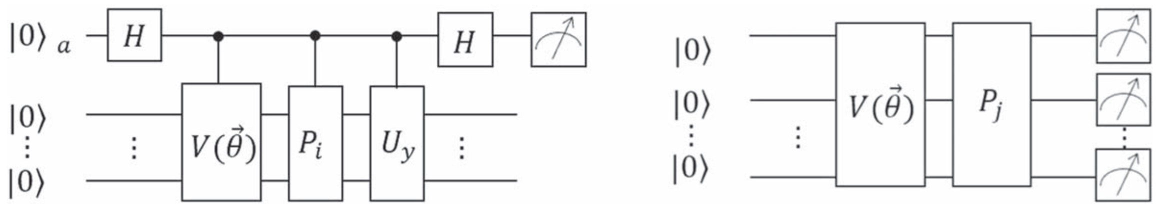
Thus, we have

$$\delta_i = \frac{\mathrm{Tr}[P_i F]}{2^n}.$$

Of course, applying Pauli decomposition to the kernel function of a QSVM is the simplest, but it may be inefficient. Thus, we will try to find some more effective decomposition methods [26–28] to improve the feasibility of the algorithm. After the decomposition of equation (6) is finished, the next step is training the QSVM, and the circuit is shown in figure 2(a). Due to our choice of the Hadamard test measurement method, the loss function can be defined as follows

$$\begin{aligned} E_{\mathrm{cost}} &= 1 - \frac{(0;\, \vec{y})^{\mathrm{T}} F \vec{b}^{*}}{|(0;\, \vec{y})| \cdot |F\vec{b}^{*}|} \\ &= 1 - \sum_{i=1}^{m} \frac{\delta_i \langle 0 \ldots 0 | U_y P_i V(\vec{\theta}) | 0 \ldots 0\rangle}{|F\vec{b}^{*}|} \in [0, 2], \end{aligned}$$

where $\langle 0 \ldots 0 | U_y | 0, \vec{y}\rangle = 1$, $\vec{b}^{*} = |\rho_{\vec{\theta}}\rangle = V(\vec{\theta})|0 \ldots 0\rangle$. The total measurement number of getting E_{cost} with an error ϵ is roughly scaling as m/ϵ^{-2} [29–31]. The gradient descent



(a) Training circuit. (b) Estimate expectation values of operators P_j.

Figure 2. Brief circuit diagrams of training part of variational QSVM. (a) Calculating the molecular part of the loss function. $V(\vec{\theta})$ can be expressed as a product of L unitaries $V_l(\theta_l)$ sequentially acting on the input state $|0...0\rangle$. As indicated, each unitary $V_l(\theta_l)$ can in turn be decomposed into a sequence of parametrized and unparametrized gates. U_y is a unitary matrix that satisfies $\langle 0\ldots 0|U_y = \langle 0,\, \vec{y}| = \frac{(0;\vec{y})^{\mathrm{T}}}{|(0;\vec{y})|}$. (b) Calculating the denominator of the loss function. The $\vec{\theta}$ is consistent with (a), P_j is the Pauli decomposition of $F^{\dagger}F$. Efficient derandomization Pauli measurements are employed at the end of the circuit, rather than random single-qubit measurements.

method can be selected to solve the above equation, and the descending direction is $\frac{\partial E_{\mathrm{cost}}}{\partial \vec{\theta}}$.

Obviously, we have

$$|F\vec{b}^{*}| = |FV(\theta)|0\ldots 0\rangle| \\ =\sqrt{\langle 0\ldots 0|V(\vec{\theta})F^{\dagger}FV(\theta)|0\ldots 0\rangle}, \\ F^{\dagger}F = \sum_{j=1}^{m'} \eta_j P_j.$$

The method of calculating $|F\vec{b}^{*}|$ is to estimate expectation values for the set of operators P_j in the quantum state $|\rho_{\vec{\theta}}\rangle$ that can be prepared repeatedly using the programmable quantum system called ansatz (shown in figure 2(b)). If all P_j have relatively low weight (the number of nonidentity tensor factors in P_j), the randomized protocol can achieve our goal quite efficiently. Because, even for sparse QSVM, the decomposition P_j of matrix $F^{\dagger}F$ always has some high weight components. Thus, the derandomized Pauli measurements [32] show tremendous efficiency over randomized protocol. The derandomized protocol builds deterministic measurements one Pauli label at a time by comparing three conditional expectation values $E_p[\mathrm{Conf}(O;P)|P^{\#},\ P(k,\ m)=W]$ $(W=\sigma_{x,y,z})$ (equation (6) in [32]) and then assigning the Pauli label that achieves the smallest score. Finally, the derandomized measurement $P^{\#}$ is obtained.

Implementing the circuit in figure 2(a), we measure the probabilities of obtaining $|0\rangle$ and $|1\rangle$ are

$$\frac{1}{2}(1 + \langle 0\ldots 0|U_y P_i V(\vec{\theta})|0\ldots 0\rangle), \\ \frac{1}{2}(1 - \langle 0\ldots 0|U_y P_i V(\vec{\theta})|0\ldots 0\rangle),$$

respectively. After M repetitions for certain i, one obtains m_0 zeros and m_1 ones from the measurement outcomes of the ancilla qubit. The quantities $(m_0-m_1)/M$ are an estimate of $\langle 0\ldots 0|U_y P_i V(\vec{\theta})|0\ldots 0\rangle$. The algorithm of the training part is shown in algorithm 1. Once the algorithm terminates, it means that the training ends, then we can start the classification stage.

Algorithm 1. Training variational QSVM

```
Input. An ancilla qubit initialized in $|0\rangle$, a register initialized in
    $|0\ldots 0\rangle$, a string of variable parameters $\vec{\theta}$, and set a suitable
    threshold $E_{\mathrm{cost}}'$.
Step 1. Choose $i = 1$, repeat the circuit 2(a) $M$ times, and
    record $(m_0 - m_1)/M$.
Step 2. Choose $j = 1$, running circuit 2(b) and calculate the mean.
Step 3. After traversing all $i$ and $j$, calculate the loss function $E_{\mathrm{cost}}$.
Step 4. If $E_{\mathrm{cost}} > E_{\mathrm{cost}}'$, update parameters $\vec{\theta}$, and then turn to Step 1;
    else terminate.
```

4.2. Classification part

We represent the classification stage as shown in figure 3 step by step.

First, the state is initialized in

$$|0\rangle_a|0\rangle^{\otimes n}|0\ldots 0\rangle.$$

After the first Hadamard gate, we get

$$\frac{1}{\sqrt{2}}|0\rangle_a|0\rangle^{\otimes n}|0\ldots 0\rangle + \frac{1}{\sqrt{2}}|1\rangle_a|0\rangle^{\otimes n}|0\ldots 0\rangle.$$

Then the ansatz $V(\vec{\theta})$ transforms the state to

$$\frac{1}{\sqrt{2}}|0\rangle_a|0\rangle^{\otimes n}|0\ldots 0\rangle + \frac{1}{\sqrt{2}}|1\rangle_a|b,\,\vec{\alpha}\rangle|0\ldots 0\rangle.$$

One then applies the training-data oracle to the two registers. The output state is

$$\frac{1}{\sqrt{2}}|0\rangle_a|0\rangle^{\otimes n}|0\ldots 0\rangle + \frac{1}{\sqrt{2}}|1\rangle_a|u\rangle,$$

where $|u\rangle = \frac{1}{\sqrt{N_u}}(b|0\rangle|0\rangle + \sum_{i=1}^{M}\alpha_i|\vec{x}_i||i\rangle|\vec{x}_i\rangle)$, with $N_u = b^2 + \sum_{k=1}^{M}\alpha_k^2|\vec{x}_k|^2$.

The next step is to perform U_{x_0} construction according to query state $|x_0\rangle$ on the above state, the output is

$$\frac{1}{\sqrt{2}}|0\rangle_a|0\rangle^{\otimes n}|0\ldots 0\rangle + \frac{1}{\sqrt{2}}|1\rangle_a U_{x_0}|u\rangle.$$

After the last Hadamard gate operating in the above state, we have the terminal state

$$\frac{1}{2}|0\rangle_a|0\rangle^{\otimes n}|0\ldots 0\rangle + \frac{1}{2}|1\rangle_a|0\rangle^{\otimes n}|0\ldots 0\rangle + \frac{1}{2}|0\rangle_a U_{x_0}|u\rangle - \frac{1}{2}|1\rangle_a U_{x_0}|u\rangle.$$

Finally, the probabilities of measuring $|0\rangle$ and $|1\rangle$ are

$$\frac{1}{2}(1 + \langle 0\ldots 0|U_{x_0}|u\rangle) = \frac{1}{2}(1 + \langle x_0|u\rangle),$$
$$\frac{1}{2}(1 - \langle 0\ldots 0|U_{x_0}|u\rangle) = \frac{1}{2}(1 - \langle x_0|u\rangle),$$

respectively. Similarly, we can repeat this line T times and get t_0 zeros and t_1 ones. If $t_0 > t_1$, the classification result will be positive; otherwise, it will be negative. At this point, the algorithm ends.

In the classification stage, we do not need to calculate the exact value of $\langle x_0|u\rangle$, we just need to know whether it is positive or negative. In other words, we just need to compare the size of t_0 and t_1. As mentioned above, estimating the value of $\langle x_0|u\rangle$ with an error ϵ demands measurements ϵ^{-2} times

$$|\langle x_0|u\rangle - \frac{t_0 - t_1}{T}| < \epsilon.$$

The fault tolerance classification result will be revealed by

$$\mathrm{sgn}\left(\frac{t_0 - t_1}{T}, - , \epsilon\right) = 1, \tag{7}$$

$$\mathrm{sgn}\left(\frac{t_0 - t_1}{T}, + , \epsilon\right) = -1. \tag{8}$$

In order to improve the classification efficiency, we set two thresholds $\epsilon = 0.1$, 0.01, and the detailed process of the algorithm is shown in algorithm 2.

Algorithm 2. Classification variational QSVM

```
Input. An ancilla qubit initialized in $|0\rangle$, a register initialized in $|0\rangle^{\otimes n}$,
    a training register initialized in $|0\ldots 0\rangle$, construct $U_{x_0}$ according to
    query qubit $|x_0\rangle$.
Step 1. Set $\epsilon = 0.1$, repeat the circuit $\epsilon^{-2}$ times. If the calculation
    result satisfies equations (7)
  or (8), terminate & output. Else, turn to Step 2.
Step 2. Set $\epsilon = 0.01$, repeat the circuit $\epsilon^{-2}$ times. If the calculation
    result satisfies equations (7)
  or (8), terminate & output. Else, turn to Step 3.
Step 3. Erase $\epsilon$ in equations (7) and (8), terminate & output.
```

4.3. Numerical experiment simulation

In order to certificate the feasibility of the algorithm and its advantageous aspects over the conventional QSVM, we devise numerical experiments to test and verify it. For convenience, we quote the experimental data of [6]. Besides, to keep consistent with this, we discard the offset term b and take $F' = K/\mathrm{tr}K$. According to the data in figure 4(a), one educes

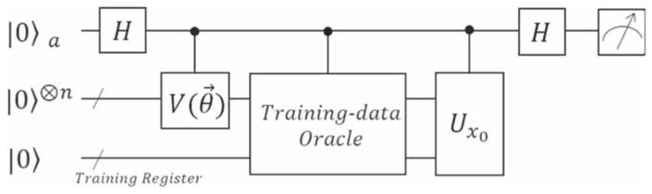


Figure 3. Brief circuit diagram of classification part of variational QSVM. The $\vec{\theta}$ of $V(\vec{\theta})$ is the one in the last cycle of algorithm 1. The Training-data Oracle prepares state $|u\rangle$. U_{x_0} contains the information of query state $|x_0\rangle$, and $\langle 0\ldots 0|U_{x_0} = \langle x_0|$.

$$F'\vec{x} = \begin{pmatrix} 0.50 & 0.25 \\ 0.25 & 0.50 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \left(\frac{1}{2}I + \frac{1}{4}X\right)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} = \vec{y}. \tag{9}$$

The unitary matrix U_y in figure 2(a) encoding $\vec{y}$ can be written as

$$U_y = R_y\left(-\frac{\pi}{2}\right) = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}.$$

To get the mean of $|F'b^*|$, we have

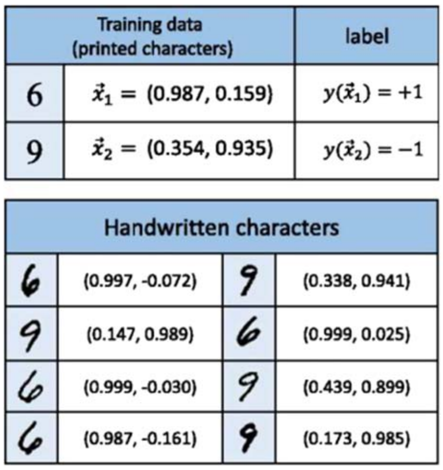
$$F'^{\dagger}F' = \begin{pmatrix} 0.31 & 0.25 \\ 0.25 & 0.31 \end{pmatrix} = \frac{5}{16}I + \frac{1}{4}X.$$

The correspondingly derandomized Pauli measurements are Z and X. Here, the Pauli Z measurement could substitute for Pauli X measurements by adding a Hadamard gate to the final states. In addition, the gradient of θ is

$$\cos(\theta/2) + \sin(\theta/2).$$

After all the preparations are ready, we conduct numerical simulation, and the iterative line chart is shown in figure 4(b). Then we construct the classification circuit shown in figure 5, where the hardware efficient ansatz with $\theta = -1.57$ will be the subsequent classification operation as a subroutine.

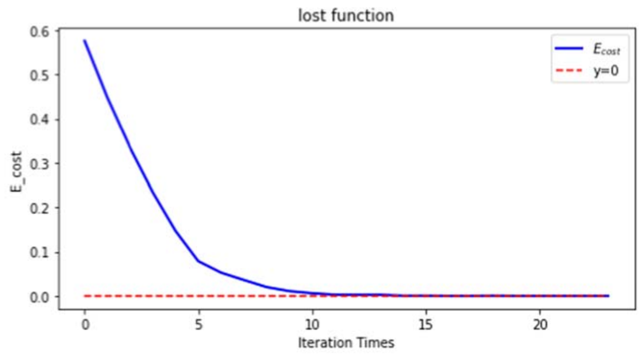
We implement this task on today's current state-of-the-art quantum computer using the IBM quantum experience platform where the experimental circuit diagram and experimental results are shown in figures A1 and A2 in the appendix. For comparison, but due to the limitation of qubits, we only give the experimental circuit diagram of applying the HHL algorithm to the QSVM (shown in appendix figure A3). The results of classification results are shown in table 1. It can be seen that by adding the fault-tolerant classification formula, we can predict the results 100% accurately when we measure only 100 times.



	Training data (printed characters)	label
6	$\vec{x}_1 = (0.987, 0.159)$	$y(\vec{x}_1) = +1$
9	$\vec{x}_2 = (0.354, 0.935)$	$y(\vec{x}_2) = -1$

Handwritten characters			
6	(0.997, -0.072)	9	(0.338, 0.941)
9	(0.147, 0.989)	6	(0.999, 0.025)
6	(0.999, -0.030)	9	(0.439, 0.899)
6	(0.987, -0.161)	9	(0.173, 0.985)

(a) Training and classification data.



(b) E_{cost} iteration.

Figure 4. (a) Experimental data of [6], where the feature values are chosen as the vertical ratio and the horizontal ratio. The training data is in a standard font (Times New Roman). The arbitrary chosen handwritten samples for classification and their feature vectors. (b) Loss function image optimized by gradient descent penalty. The calculation result is simulated by Python.

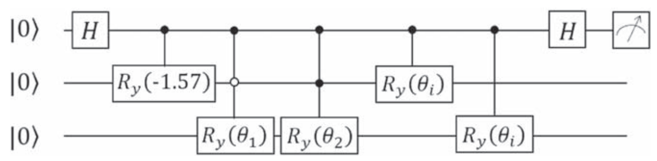


Figure 5. The classification circuit. The rotation angles $\theta_1 = 2\arctan\frac{(x_1)_2}{(x_1)_1}$, $\theta_2 = 2\arctan\frac{(x_2)_2}{(x_2)_1}$. The last two controlled $R_y(\theta_i)$ gates form U_{x_0}, with $\theta_i = -2\arctan\frac{(x_i)_2}{(x_i)_1}$, $i = 3, 4,\ldots,10$.

Table 1. The classification results of samples. In the two columns below the number of measurements, the left one is the number of times the auxiliary qubit measures 0 and the right is obtaining 1.

Samples\Times&Results	100		R	10000		R
$x_3 = (0.997, -0.072)$	73	27	6	6405	3695	6
$x_4 = (0.147, 0.989)$	22	78	9	3636	6364	9
$x_5 = (0.999, -0.030)$	82	18	6	7466	2534	6
$x_6 = (0.987, -0.161)$	68	32	6	7540	2460	6
$x_7 = (0.338, 0.941)$	37	63	9	4048	5952	9
$x_8 = (0.999, 0.025)$	62	38	6	6924	3076	6
$x_9 = (0.439, 0.899)$	36	64	9	4562	5438	9
$x_{10} = (0.173, 0.985)$	30	70	9	4524	5476	9

4.4. Advantage analysis

The QSVMs shown in figure 1 have put training and classification in a circuit which means that they run the whole route every time one classifies a new data. Once the classification data is much larger than the training data, the algorithms will lead to a lot of redundant operations. In practice, the long line of figure 1(a) and the complex measurement mode of figure 1(b) will be an obstacle to the successful implementation of the algorithm. Since the variational QSVM divides training and classification into two parts, when the training stage is completed, and it will turn to the classification circuit. When classifying, our algorithm only requires Hadamard test measurements and owns a shorter line depth, which is more amenable to execution on near-term devices.

Here, we analyze the circuit depth and qubits numbers of our algorithm compared with the algorithm in [5]. Our algorithm needs 2, 1 and 3 qubits, respectively, in solving equations part, calculating modules part and classifying parts. In contrast, if one adopts the HHL algorithm directly to the QSVM, there must be 4 qubits—1 ancilla qubit(abbreviated a_1), 2 register qubits, and 1 qubit storing the input message and output result—in the stage of solving equations. Meanwhile, in constructing the state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|u\rangle + |1\rangle|x_0\rangle)$, there must be 1 more ancilla qubit(abbreviated a_2) and 1 register qubit for building $|u\rangle$ according to the output of the HHL algorithm. The total qubits number of applying the HHL algorithm to the QSVM is 6.

When applying the HHL algorithm to equation (9), we have to use 2 qubits for the register, despite the fact that the more registered qubits the more accurate result, and the fewer registered qubits the easier the quantum circuit. In the phase estimation stage, if we only use 1 registered qubit, the estimated eigenvalue of F' is either 0 or $\frac{1}{2}$ which leads to a great error. Although 3 registered qubits could estimate the eigenvalue among $\frac{i}{8}$, $i = 0, 1,\ldots,7$, it will bring a great burden on the construction of quantum circuits. Besides, the addition of a_2 produces a number of $C^2(U)$ gates which can be decomposed into 5 universal quantum gates. Thus, the circuit is much deeper than the variational QSVM circuit. Furthermore, the quantum circuit is implemented successfully under the circumstances a_1 measures 1 which means that the run times exceed the variational QSVM circuit.

In a more general sense, suppose the dimension of F' is $N \times N$, one applies the HHL algorithm to the QSVM demands at least $3 + 3\log N$ qubits overall. The corresponding part of the variational QSVM needs $1 + \log N$, $\log N$ and $1 + 2\log N$ qubits, gathered $2 + 4\log N$. It can be seen that with the increase of N, the number of qubits required by applying the HHL algorithm to the QSVM is less than that of the variational QSVM. However, the number of qubits used in the phase estimation stage $1 + \log N$ is only the minimum number we assume, and the accuracy cannot be guaranteed. Meanwhile, our algorithm shows great advantages in the depth of quantum circuits.

5. Conclusion and outlook

We propose a quantum–classical variational algorithm to solve the QSVM based on the Hadamard test and construct a loss function that is only suitable for vectors' own equal module length. That is, it applies to quantum states. Besides, our algorithm adopts the derandomization Pauli measurements protocol which shows strong advantages over random Pauli measurements due to the high weight decomposition. We have carried out simulated quantum experiments to prove the feasibility of the algorithm. In the future, we will also consider applying this algorithm to a multi-classification QSVM.

Acknowledgments

This work is supported by the Shandong Provincial Natural Science Foundation for Quantum Science No. ZR2020LLZ003, ZR2021LLZ002.

Appendix

As shown in figures A1, A2 and A3, it is obvious that the variational QSVM is much more efficient in classification.

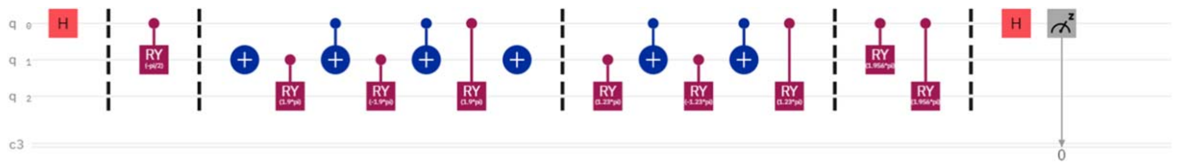


Figure A1. The experimental classification circuit of variational QSVM.

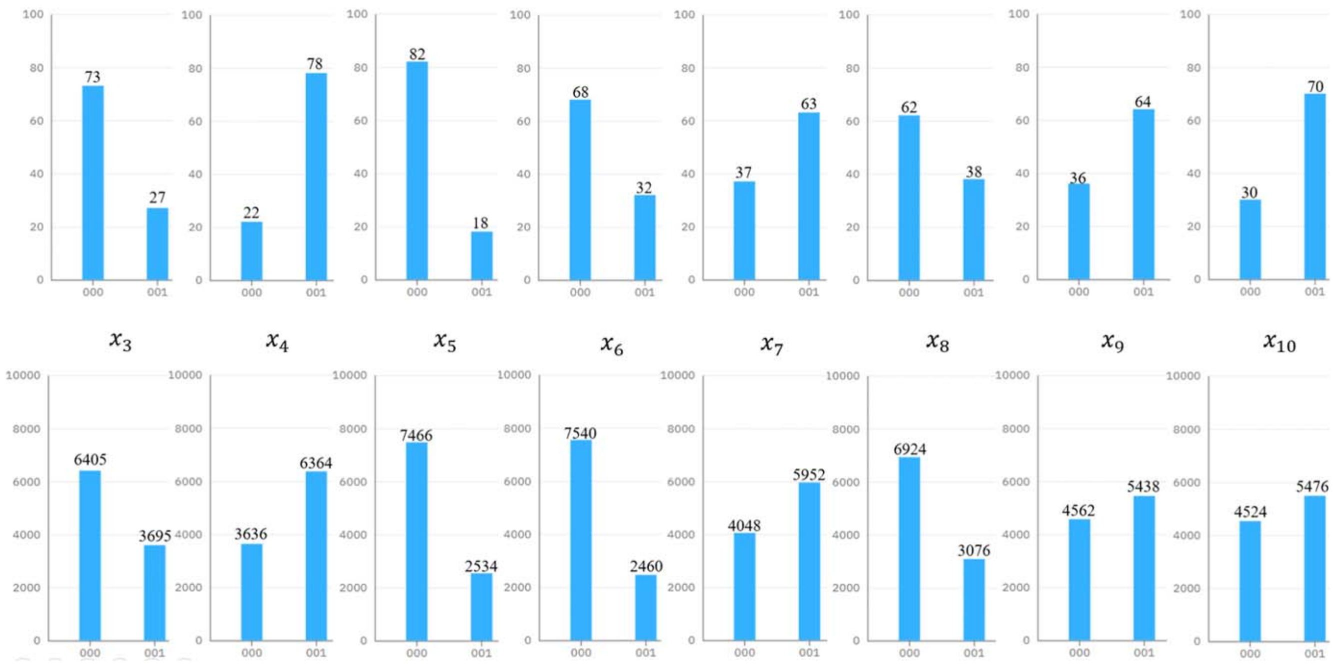


Figure A2. Experimental data.

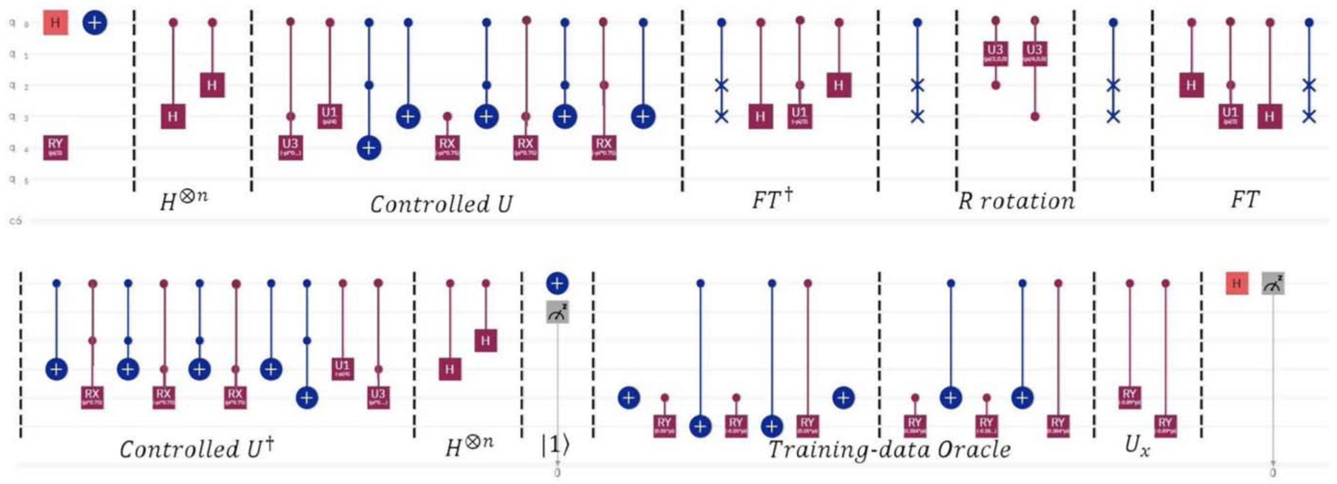


Figure A3. The experimental circuit of applying the HHL algorithm to the QSVM. Here, q_0 refers to the ancilla a_2, q_1 refers to the ancilla a_1, q_2 and q_3 are registers, q_4 stores the input and output message, and q_5 is the training register. Besides, the $C^2(U)$ gates in the stage of the HHL algorithm should be decomposed into universal quantum gates.

ORCID iDs

Ming Li https://orcid.org/0000-0003-0835-9635

References

[1] Zhihua Z 2016 *Machine Learning* (Beijing: Tsinghua University Press)
[2] Vapnik V 1998 *Statistical Learning Theory* vol 3 (New York, NY: Wiley) pp 401–92 Ch 10–11
[3] Shor P W 1994 Algorithms for quantum computation: discrete logarithms and factoring *Proc. of the XXXV FOCS* (New York: IEEE) pp 124–34
[4] Ladd T D, Jelezko F, Laflamme R, Nakamura Y, Monroe C and O'Brien J L 2010 Quantum computers *Nature (London)* **464** 45
[5] Rebentrost P, Mohseni M and Lloyd S 2014 Quantum support vector machine for big data classification *Phys. Rev. Lett.* **113** 130503
[6] Li Z-K, Liu X-M, Xu N-Y and Du J-F 2015 Experimental realization of a quantum support vector machine *Phys. Rev. Lett.* **114** 110504
[7] Suykens J A K and Vandewalle J 1999 Least squares support vector machine classifiers *Neural Process. Lett.* **9** 293
[8] Jian L, Shen S-Q, Li J-D, Liang X-J and Li L 2017 Budget online learning algorithm for least squares SVM *IEEE Trans Neural Netw. Learn. Syst.* **28** 2076–87
[9] Hsu C-W and Lin C-J 2002 A comparison of methods for multiclass support vector machines *IEEE Trans. Neural Networks* **13** 415–25
[10] Chen J-X, Liu W-J, Gao P-P and Wang H-B 2019 A quantum feature selection algorithm for multi-classification problem *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* pp 519–25
[11] Wang D-Y, Zhou L-J, Dai C-Q, Guo L and Liao W 2020 Insulation defect diagnostic method for OIP bushing based on multiclass ls-SVM and cuckoo search *IEEE Trans. Instrum. Meas.* **69** 163–72
[12] Bishwas A K, Mani A and Palade V 2018 An all-pair quantum SVM approach for big data multiclass classification *Quantum Inf. Process.* **17** 282
[13] Bishwas A K, Mani A and Palade V 2016 Big data classification with quantum multiclass SVM and quantum one-against-all approach *2016 II International Conference on Contemporary Computing and Informatics (IC3I)* pp 875–80
[14] Uke D, Soni K K and Rasool A 2020 Quantum based support vector machine identical to classical model *2020 XI International Conference on Computing, Communication and Networking Technologies (ICCCNT)* pp 1–6
[15] Jayadeva, Khemchandani R and Chandra S 2007 Twin support vector machines for pattern classification *IEEE Trans. Pattern Anal. Mach. Intell.* **29** 905–10
[16] Ye Z-K, Li L-Z, Situ H-Z and Wang Y-Y 2020 Quantum speedup of twin support vector machines *Sci. China Inf. Sci.* **63** 189501
[17] Tang E 2021 Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions *Phys. Rev. Lett.* **127** 060503
[18] Giovannetti V, Lloyd S and Maccone L 2008 Quantum random access memory *Phys. Rev. Lett.* **100** 160501
[19] Giovannetti V, Lloyd S and Maccone L 2008 Architectures for a quantum random access memory *Phys. Rev.* A **78** 052310
[20] Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love P J, Guzik A A and O'Brien J L 2014 A variational eigenvalue solver on a photonic quantum processor *Nat. Commun.* **5** 4213
[21] Liang J-M, Shen S-Q, Li M and Li L 2020 Variational quantum algorithms for dimensionality reduction and classification *Phys. Rev.* A **101** 032323
[22] Parrish R M, Hohenstein E G, McMahon P L and Martnez T J 2019 Quantum computation of electronic transitions using a variational quantum eigensolver *Phys. Rev. Lett.* **122** 230401
[23] Chen M-C *et al* 2020 Demonstration of adiabatic variational quantum computing with a superconducting quantum coprocessor *Phys. Rev. Lett.* **125** 180501
[24] Liang J-M, Shen S-Q, Li M and Fei S-M 2022 Quantum algorithms for the generalized eigenvalue problem *Quantum Inf. Process.* **21** 23
[25] Harrow A W, Hassidim A and Lloyd S 2009 Quantum algorithm for linear systems of equations *Phys. Rev. Lett.* **103** 150502

[26] Wan L-C, Yu C-H, Pan S-J, Qin S-J, Gao F and Wen Q-Y 2021 Block-encoding-based quantum algorithm for linear systems with displacement structures *Phys. Rev.* A **104** 062414

[27] Schlimgen A W, Marsden K H, Sager L M, Narang P and Mazziotti D A 2021 Quantum simulation of open quantum systems using a unitary decomposition of operators *Phys. Rev. Lett.* **127** 270503

[28] Liu H-L, Wu Y-S, Wan L-C, Pan S-J, Qin S-J, Gao F and Wen Q-Y 2021 Variational quantum algorithm for the Poisson equation *Phys. Rev.* A **104** 022418

[29] Kirby W M and Love P J 2021 Variational quantum eigensolvers for sparse Hamiltonians *Phys. Rev. Lett.* **127** 110503

[30] McClean J R, Romero J, Babbush R and AspuruGuzik A 2016 The theory of variational hybrid quantum-classical algorithms *New J. Phys.* **18** 023023

[31] Rubin N C, Babbush R and McClean J 2018 Application of fermionic marginal constraints to hybrid quantum algorithms *New J. Phys.* **20** 053020

[32] Huang H-Y, Kueng R and Preskill J 2021 Efficient estimation of Pauli observables by derandomization *Phys. Rev. Lett.* **127** 030503