scientific reports

Check for updates

OPEN

Variational quantum support vector machine based on Γ matrix expansion and variational universal-quantum-state generator

Motohiko Ezawa

We analyze a binary classification problem by using a support vector machine based on variational quantum-circuit model. We propose to solve a linear equation of the support vector machine by using a Γ matrix expansion. In addition, it is shown that an arbitrary quantum state is prepared by optimizing a universal quantum circuit representing an arbitrary $U(2^N)$ based on the steepest descent method. It may be a quantum generalization of Field-Programmable-Gate Array (FPGA).

Quantum computation is a hottest topic in contemporary physics[1–3]. An efficient application of quantum computations is machine learning, which is called quantum machine learning[4–17] . A support vector machine is one of the most fundamental algorithms for machine learning[18,22,23], which classifies data into two classes by a hyperplane. A support vector machine (SVM) is a computer algorithm that learns by examples to assign labels to objects. It is a typical method to solve a binary-classification problem[18]. The optimal hyperplane is determined by an associated linear equation $F|\psi_{\rm in}\rangle = |\psi_{\rm out}\rangle$, where F and $|\psi_{\rm out}\rangle$ are given. A quantum support vector machine solves this linear equation by a quantum computer[10,13,24]. Usually, the linear equation is solved by the Harrow-Hassidim-Lloyd (HHL) algorithm[25]. However, this algorithm requires many quantum gates. Thus, the HHL algorithm is hard to be executed by using a near-term quantum computer. Actually, this algorithm has experimentally been verified only for two and three qubits[26–28]. In addition, it requires a unitary operator to execute e^{iFt}, which is quite hard to be implemented. The Kernel based SVM implementation based on the quantum is reported[19–21].

The number of qubits in current quantum computers is restricted. Variational quantum algorithms are appropriate for these small-qubit quantum computers, which use both quantum computers and classical computers. Various methods have been proposed such as Quantum Approximate Optimization Algorithm (QAOA)[29], variational eigenvalue solver[30], quantum circuit learning[31] and quantum linear solver[32,33]. We use wave functions with variational parameters in QAOA, which are optimized by minimizing the expectation value of the Hamiltonian. A quantum circuit has variational parameters in quantum circuit learning[31], which are optimized by minimizing a certain cost function. A quantum linear solver solves a linear equation by variational ansatz[32,33]. The simplest method of the optimization is a steepest-descent method.

In this paper, we present a variational method for a quantum support vector machine by solving an associated linear equation based on variational quantum circuit learning. We propose a method to expand the matrix F by the Γ matrices, which gives simple quantum circuits. We also propose a variational method to construct an arbitrary state by using a universal quantum circuit to represent an arbitrary unitary matrix $U(2^N)$. We prepare various internal parameters for a universal quantum circuit, which we optimize by minimizing a certain cost function. Our circuit is capable to determine the unitary transformation U satisfying $U|\psi_{\rm initial}\rangle = |\psi_{\rm final}\rangle$ with arbitrary given states $|\psi_{\rm initial}\rangle$ and $|\psi_{\rm final}\rangle$. It will be a quantum generalization of field-programmable-gate array (FPGA), which may execute arbitrary outputs with arbitrary inputs.

Results

Support vector machine. A simplest example of the SVM reads as follows. Suppose that there are red and blue points whose distributions are almost separated into two dimensions. We classify these data points into two classes by a line, as illustrated in Fig. 1a.

In general, M data points are spattered in D dimensions, which we denote $\boldsymbol{x}_j$, where $1 \leq j \leq M$. The problem is to determine a hyperplane,

Department of Applied Physics, University of Tokyo, Hongo 7-3-1, Tokyo 113-8656, Japan. email: ezawa@ap.t.u-tokyo.ac.jp

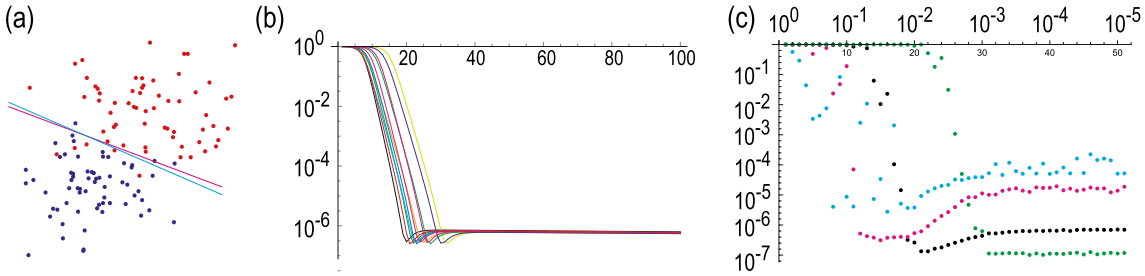

Figure 1. (**a**) Binary classification of red and blue points based on a quantum support vector machine with soft margin. A magenta (cyan) line obtained by an exact solution (variational method). (**b**) Evolution of the cost function. The vertical axis is the $\log_{10} E_{\rm cost}$. The horizontal axis is the variational step number. We have used $r = 2, \xi_1 = 0.001$ and $\xi_2 = 0.0005$ and $\gamma = 1$. We have runed simulations ten times, where each simulation is plotted in different color. (**c**) The saturated value of the cost function $\log_{10} E_{\rm opt}$ as a function of ξ_2 ranging $10^{-1} \le \xi_2 \le 10^{-5}$ for various ξ_1. The green dots indicates $\xi_1 = 0.0001$, black dots indicates $\xi_1 = 0.001$, magenta dots indicates $\xi_1 = 0.01$ and cyan dots indicates $\xi_1 = 0.1$.

$$\boldsymbol{\omega} \cdot \boldsymbol{x} + \omega_0 = 0, \tag{1}$$

separating data into two classes with the use of a support vector machine. We set

$$\boldsymbol{\omega} \cdot \boldsymbol{x} + \omega_0 > 0 \tag{2}$$

for red points and

$$\boldsymbol{\omega} \cdot \boldsymbol{x} + \omega_0 < 0 \tag{3}$$

for blue points. These conditions are implemented by introducing a function

$$f(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{\omega} \cdot \boldsymbol{x} + \omega_0), \tag{4}$$

which assigns $f(\boldsymbol{x}) = 1$ to red points and $f(\boldsymbol{x}) = -1$ to blue points. In order to determine ω_0 and $\boldsymbol{\omega}$ for a given set of data $\boldsymbol{x}_j$, we introduce real numbers α_j by

$$\boldsymbol{\omega} = \sum_{j=1}^{M} \alpha_j \boldsymbol{x}_j. \tag{5}$$

A support vector machine enables us to determine ω_0 and α_j by solving the linear equation

$$F \begin{pmatrix} \omega_0 \\ \alpha_1 \\ \vdots \\ \alpha_M \end{pmatrix} = \begin{pmatrix} 0 \\ y_1 \\ \vdots \\ y_M \end{pmatrix}, \tag{6}$$

where $y_i = f(x_i) = \pm 1$, and F is a $(M+1) \times (M+1)$ matrix given by

$$F = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & & & \\ \vdots & & K + I_M/\gamma & \\ 1 & & & \end{pmatrix}. \tag{7}$$

Here,

$$K_{ij} = \boldsymbol{x}_i \cdot \boldsymbol{x}_j, \tag{8}$$

is a Kernel matrix, and γ is a certain fixed constant which assures the existence of the solution of the linear equation (6) even when the red and blue points are slightly inseparable. Note that $\gamma \to \infty$ corresponds to the hard margin condition. Details of the derivation of Eq. (6) are given in Method A.

Quantum linear solver based on Γ matrix expansion. We solve the linear equation (6) by a quantum computer. In general, we solve a linear equation

$$F|\psi_{\rm in}\rangle = c|\psi_{\rm out}\rangle, \tag{9}$$

for an arbitrary given non-unitary matrix F and an arbitrary given state $|\psi_{\rm out}\rangle$. Here, the coefficient c is introduced to preserve the norm of the state, and it is given by

$$c = \sqrt{\langle\psi_{\rm in}|F^\dagger F|\psi_{\rm in}\rangle}. \tag{10}$$

The HHL algorithm[25] is a most famous algorithm to solve this linear equation by a quantum computer. We first construct a Hermitian matrix by

$$H = \begin{pmatrix} 0 & F \\ F^\dagger & 0 \end{pmatrix}. \tag{11}$$

Then, a unitary matrix associated with F is uniquely obtained by e^{iHt} . Nevertheless, it requires many quantum gates. In addition, it is a nontrivial problem to implement e^{iHt}.

Recently, variational methods have been proposed[32] to solve the linear equation (9). In one of the methods, the matrix F is expanded in terms of some unitary matrices U_j as

$$F = \sum_{j=0}^{2^N-1} c_j U_j. \tag{12}$$

In general, a complicated quantum circuit is necessary to determine the coefficient c_j.

We start with a trial state $|\tilde{\psi}_{\rm in}\rangle$ to determine the state $|\psi_{\rm in}\rangle$. Application of each unitary matrix to this state is efficiently done by a quantum computer, $U_j|\tilde{\psi}_{\rm in}\rangle = |\tilde{\psi}^{(j)}_{\rm out}\rangle$, and we obtain

$$F|\tilde{\psi}_{\rm in}\rangle = \sum_{j=0}^{2^N-1} c_j U_j|\tilde{\psi}_{\rm in}\rangle = \sum_{j=0}^{2^N-1} c_j|\tilde{\psi}^{(j)}_{\rm out}\rangle \equiv c|\tilde{\psi}_{\rm out}\rangle, \tag{13}$$

where $|\tilde{\psi}_{\rm out}\rangle$ is an approximation of the given state $|\psi_{\rm out}\rangle$. We tune a trial state $|\tilde{\psi}_{\rm in}\rangle$ by a variational method so as to minimize the cost function[32]

$$E_{\rm cost} \equiv 1 - \left|\langle\tilde{\psi}_{\rm out}|\psi_{\rm out}\rangle\right|^2, \tag{14}$$

which measures the similarity between the approximate state $|\tilde{\psi}_{\rm out}\rangle$ and the state $|\psi_{\rm out}\rangle$ in (9). We have $0 \le E_{\rm cost} \le 1$, where $E_{\rm cost} = 0$ for the exact solution. The merit of this cost function is that the inner product is naturally calculated by a quantum computer.

Let the dimension of the matrix F be 2^N. It is enough to use N satisfying $2^{N-1} < D \le 2^N$ without loss of generality by adding trivial $2^N - D$ components to the linear equation. We propose to expand the matrix F by the gamma matrices Γ_j as

$$F = \sum_{j=0}^{2^N-1} c_j \Gamma_j, \tag{15}$$

with

$$\Gamma_j = \bigotimes_{\beta=1}^{N} \sigma_\alpha^{(\beta)}, \tag{16}$$

where $\alpha = 0, x, y$ and z.

The merit of our method is that it is straightforward to determine c_j by the well-known formula

$$c_j = {\rm Tr}\left[\Gamma_j F\right]. \tag{17}$$

In order to construct a quantum circuit to calculate c_j, we express the matrix F by column vectors as

$$F = \left\{\left|f_0\right\rangle, \cdots, \left|f_{2^N-1}\right\rangle\right\}. \tag{18}$$

We have $\left(\left|f_{q-1}\right\rangle\right)_p = F_{pq}$, where subscript p denotes the p-th component of $\left|f_{q-1}\right\rangle$. Then c_j is given by

$$c_j = \sum_{q=0}^{2^N-1} \left(\Gamma_j\left|f_q\right\rangle\right)_q = \sum_{q=0}^{2^N-1} \left\langle\left\langle q\right|\Gamma_j\left|f_q\right\rangle, \right. \tag{19}$$

where the subscript q denotes the $(q+1)$-th component of $\Gamma_j\left|f_q\right\rangle$. We have introduced a notation $\left|q\right\rangle\!\rangle \equiv |n_1 n_2 \cdots n_N\rangle$ with $n_i = 0, 1$, where q is the decimal representation of the binary number $n_1 n_2 \cdots n_N$. See explicit examples for one and two qubits in Method B.

The state $\left|q\right\rangle\!\rangle \equiv |n_1 n_2 \cdots n_N\rangle$ is generated as follows. We prepare the NOT gates $\sigma_x^{(i)}$ for the i-th qubit if $n_i = 1$. Using all these NOT gates we define

$$U_X^{(q)} = \bigotimes_{n_i=1} \sigma_x^{(i)}. \tag{20}$$

We act it on the initial state $|0\rangle\!\rangle$ and obtain

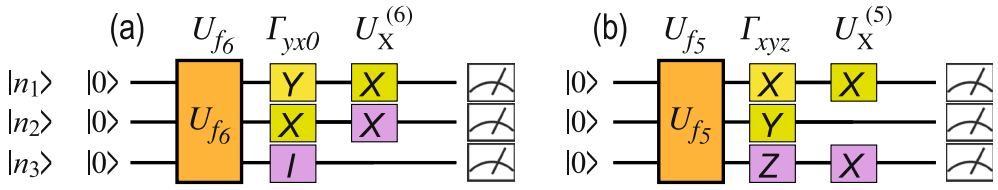

Figure 2. Quantum circuits determining c_j. We show an example with (**a**) $\Gamma_{yx0} = \sigma_y \otimes \sigma_x \otimes \sigma_0$. $U_X^{(6)}|0\rangle\rangle = \sigma_x^{(1)}\sigma_x^{(2)}|000\rangle = |110\rangle = |6\rangle\rangle$ and (**b**) $\Gamma_{xyz} = \sigma_x \otimes \sigma_y \otimes \sigma_z$. $U_X^{(5)}|0\rangle\rangle = \sigma_x^{(1)}\sigma_x^{(3)}|000\rangle = |101\rangle = |5\rangle\rangle$.

$$U_X^{(q)}|0\rangle\rangle = |q\rangle\rangle. \tag{21}$$

Next, we construct a unitary gate U_{f_q} generating $|f_q\rangle$,

$$U_{f_q}|0\rangle\rangle = |f_q\rangle. \tag{22}$$

We will discuss how to prepare U_{f_q} by a quantum circuit soon later; See Eq. (33). By using these operators, c_j is expressed as

$$c_j = \sum_{q=0}^{2^N-1} \langle\langle 0|\, U_X^{(q)} \Gamma_j U_{f_q}|0\rangle\rangle, \tag{23}$$

which can be executed by a quantum computer. We show explicit examples in Fig. 2.

Once we have c_j, the final state is obtained by applying Γ_j to $|\tilde{\psi}_{\rm in}\rangle$ and taking sum over j, which leads to

$$|\tilde{\psi}_{\rm out}\rangle = F|\tilde{\psi}_{\rm in}\rangle = \sum_{j=0}^{2^N-1} c_j \Gamma_j |\tilde{\psi}_{\rm in}\rangle. \tag{24}$$

The implementation of the Γ matrix is straightforward in quantum circuit, because the Γ matrix is composed of the Pauli sigma matrices, as shown in Fig. 2.

Steepest-descent method. One of the most common approaches to optimization is the steepest-descent method, where we make iterative steps in directions indicated by the gradient[34]. We may use this method to find an optimal trial state $|\tilde{\psi}_{\rm in}\rangle$ closest to the state $|\psi_{\rm in}\rangle$. To determine the gradient, we calculate the difference of the cost function $\Delta E_{\rm cost}$ when we slightly change the trial state $|\tilde{\psi}_{\rm in}(t)\rangle$ at step t by the amount of $\Delta|\tilde{\psi}_{\rm in}(t)\rangle$ as

$$\Delta E_{\rm cost} \equiv E_{\rm cost}\left(|\tilde{\psi}_{\rm in}(t)\rangle + \Delta|\tilde{\psi}_{\rm in}(t)\rangle\right) - E_{\rm cost}\left(|\tilde{\psi}_{\rm in}(t)\rangle\right) \simeq \frac{\Delta E_{\rm cost}}{\Delta|\tilde{\psi}_{\rm in}(t)\rangle}\Delta|\tilde{\psi}_{\rm in}(t)\rangle. \tag{25}$$

We explain how to construct $|\tilde{\psi}_{\rm in}(t)\rangle$ by a quantum circuit soon later; See Eq. (33). Then, we renew the state as

$$|\tilde{\psi}_{\rm in}(t)\rangle \to |\tilde{\psi}_{\rm in}(t)\rangle - \eta(t)\frac{\Delta E_{\rm cost}}{\Delta|\tilde{\psi}_{\rm in}(t)\rangle}\Delta|\tilde{\psi}_{\rm in}(t)\rangle, \tag{26}$$

where we use an exponential function for η_t,

$$\eta(t) = \xi_1 e^{-\xi_2 t}. \tag{27}$$

We choose appropriate constants ξ_1 and ξ_2 for an efficient search of the optimal solution, whose explicit examples are given in the caption of Fig. 1b. We stop the renewal of the variational step when the difference $\Delta|\tilde{\psi}_{\rm in}(t)\rangle$ becomes sufficiently small, which gives the optimal state of the linear equation (9).

In the numerical simulation, we discretize the time step

$$t = n\Delta t, \tag{28}$$

with a fixed Δt. We add a small value $\eta(n\Delta t)$ in the p-th component of the trial state $|\tilde{\psi}_{\rm in}^{(p)}(t)\rangle$ at the n step

$$|\tilde{\psi}_{\rm in}^{(p)}((n+1)\Delta t)\rangle = \tilde{\psi}_{\rm in}^{(p)}(n\Delta t)\rangle_p + \eta(t)\delta^{(p)}, \tag{29}$$

where $\delta^{(p)}$ denotes a unit vector where only the p component is 1 and the other components are zero. Then, we calculate the costfunction

$$E_{\rm cost}^{(p)}((n+1)\Delta t) \equiv 1 - \left|\langle\tilde{\psi}_{\rm in}^{(p)}((n+1)\Delta t)|\psi_{\rm out}\rangle\right|^2. \tag{30}$$

By running p from 1 to 2^N, we obtain a vector $E_{\rm cost}^{(p)}((n+1)\Delta t)$, whose p-th component is $E_{\rm cost}^{(p)}((n+1)\Delta t)$. Then, the gradient is numerically obtained as

$$\Delta E_{\rm cost}(n+1) \equiv \left(E^{(p)}_{\rm cost}((n+1)\Delta t) - E^{(p)}_{\rm cost}(n\Delta t)\right), \tag{31}$$

and we set the trial state at the $n+1$ step.

$$|\tilde{\psi}^{(p)}_{\rm in}((n+1)\Delta t)\rangle = |\tilde{\psi}^{(p)}_{\rm in}(n\Delta t)\rangle + \Delta E_{\rm cost}(n+1). \tag{32}$$

We iterate this process so that $\Delta E_{\rm cost}(n+1)$ becomes sufficiently small.

We denote the saturated cost function $E_{\rm opt}$. It depends on the choice of ξ_1 and ξ_2 in Eq. (27). We show $\log_{10} E_{\rm opt}$ as a function of ξ_2 for various ξ_1 in Fig. 1c. There are some features. First, $E_{\rm opt}$ is small for small ξ_1. Namely, we need to choose small ξ_1 in order to obtain a good solution. On the other hand, the required step increases for small ξ_1. It is natural that small ξ_1 means that the step size is small. The required step number is antiproportional to ξ_1. Second, there is a critical value to obtain a good solution as a function of ξ_2 for a fixed value of ξ_1. We find that it is necessary to set $\xi_2 < 10^{-3}$.

A comment is in order. The cost function does not become zero although it becomes very small. It means that the solution is trapped by a local minimum and does not reach the exact solution. It is a general feature of variational algorithms, where we cannot obtain the exact solution. However, the exact solution is unnecessary in many cases including machine learnings. Actually, the classification shown in Fig. 1a is well done.

Variational universal-quantum-state generator. In order to construct the trial state $|\tilde{\psi}_{\rm in}(t)\rangle$, it is necessary to prepare an arbitrary state $|\psi\rangle$ by a quantum circuit. Alternatively, we need such a unitary transformation U that

$$U|0\rangle\rangle = |\psi\rangle. \tag{33}$$

It is known that any unitary transformation is done by a sequential application of the Hadamard, the $\pi/4$ phase-shift and the CNOT gates[35,36]. Indeed, an arbitrary unitary matrix is decomposable into a sequential application of quantum gates[35,36], each of which is constructed as a universal quantum circuit systematically[37–42]. Universal quantum circuits have so far been demonstrated experimentally for two and three qubits[43–46].

We may use a variational method to construct U satisfying Eq. (33). Quantum circuit learning is a variational method[31], where angle variables θ_i are used as variational parameters in a quantum circuit U, and the cost function is optimized by tuning θ_i. We propose to use a quantum circuit learning for a universal quantum circuit. We show that an arbitrary state $|\psi(\theta_i)\rangle$ can be generated by tuning $U(\theta_i)$ starting from the initial state $|0\rangle\rangle$ as

$$U(\theta_i)|0\rangle\rangle = |\psi(\theta_i)\rangle. \tag{34}$$

We adjust θ_i by minimizing the cost function

$$E_{\rm cost}(\theta_i) \equiv 1 - |\langle\psi(\theta_i)|\psi\rangle|^2, \tag{35}$$

which is the same as that of the variational quantum support vector machine. We present explicit examples of universal quantum circuits for one, two and three qubits in Method C.

Quantum field-programmable-gate array. We next consider a problem to find a unitary transformation $U_{\text{ini-fin}}$ which maps an arbitrary initial state $|\psi_{\rm initial}\rangle$ to an arbitrary final state $|\psi_{\rm final}\rangle$,

$$U_{\text{ini-fin}}|\psi_{\rm initial}\rangle = |\psi_{\rm final}\rangle. \tag{36}$$

Since we can generate an arbitrary unitary matrix as in Eq. (33), it is possible to generate such matrices $U_{\rm ini}$ and $U_{\rm fin}$ that

$$U_{\rm ini}|0\rangle\rangle = |\psi_{\rm initial}\rangle, \qquad U_{\rm fin}|0\rangle\rangle = |\psi_{\rm final}\rangle. \tag{37}$$

Then, Eq. (36) is solved as

$$U_{\rm fin} = U_{\text{ini-fin}}U_{\rm ini}, \tag{38}$$

since $U_{\text{ini-fin}}|\psi_{\rm initial}\rangle = U_{\text{ini-fin}}U_{\rm ini}|0\rangle\rangle = |\psi_{\rm final}\rangle = U_{\rm fin}|0\rangle\rangle$.

An FPGA is a classical integrated circuit[47–50], which can be programmable by a customer or a designer after manufacturing in a factory. An FPGA executes any classical algorithms. On the other hand, our variational universal quantum-state generator creates an arbitrary quantum state. We program by using the variational parameters θ_i. In this sense, the above quantum circuit may be considered as a quantum generalization of FPGA, which is a quantum FPGA (q-FPGA).

We show explicitly how the cost function is renewed for each variational step in the case of two- and three-qubit universal quantum circuits in Fig. 3, where we have generated the initial and the final states randomly. We optimize 15 parameters θ_i for two-qubit universal quantum circuits and 82 parameters θ_i for three-qubit universal quantum circuits. We find that $U_{\text{ini-fin}}$ is well determined by variational method as in Fig.3.

Variational quantum support vector machine. We demonstrate a binary classification problem in two dimensions based on the support vector machine. We prepare a data set, where red points have a distribution around $(r\cos\Theta, r\sin\Theta)$ with variance r, while blue points have a distribution around $(-r\cos\Theta, -r\sin\Theta)$ with

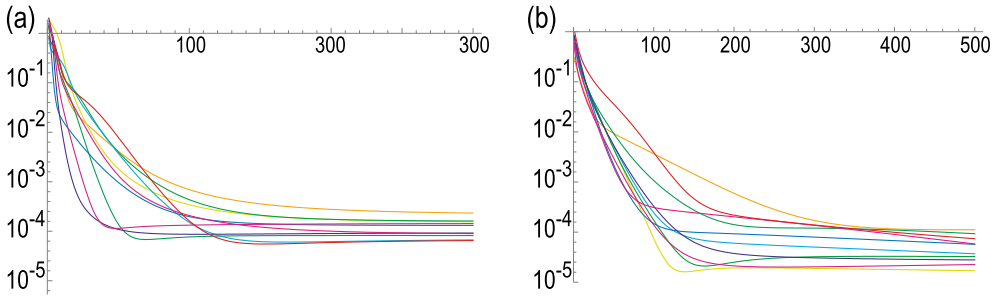

Figure 3. Evolution of the cost function for (**a**) two qubits and (**b**) three qubits. The vertical axis is the $\log_{10} E_{\rm cost}$. The horizontal axis is the number of variational steps. We use $c_1 = 0.005$ and $c_2 = 0.005$ for both the two- and three-qubit universal quantum circuits. We prepare random initial and final states, where we have runed simulations ten times. Each simulation is plotted in different color.

variance r. We assume the Gaussian normal distribution. We choose Θ randomly. We note that there are some overlaps between the red and blue points, which is the soft margin model.

As an example, we show the distribution of red and blue points and the lines obtained by the variational method marked in cyan and by the direct solution of (6) marked in magenta in Fig. 1a. They agrees well with one another, where both of the lines well separate red and blue points. We have prepared 31 red points and 32 blue points, and used six qubits.

Discussion

Efficiency. The original proposal[10] requires $O\big(\log(N_{\rm D}M)\big)$ runtime, where $N_{\rm D}$ is the dimension of the feature space and M is the number of training data points. It has an advantage over the classical protocol which requires $O\big(\text{polynomial}(N_{\rm D}, M)\big)$. There exisits also a quantum-inspired classical SVM[51], which requires polynomial runtime as a function of the number of data points M and dimension of the feature space $N_{\rm D}$.

N qubit can represent $2^{N-1} < N_{\rm D}M \le 2^N$. Hence, the required number of qubits is $N = \log(N_{\rm D}M)$. We need $4^N - 1$ quantum gates for an exact preparation of a universal quantum state. On the other hand, a hardware-efficient universal quantum circuit prepares an approximate universal quantum state by using the order of $4N$ quantum gates[52–54]. We need N quantum gates for the execution of $U_X^{(q)}$ and Γ_j, separately. We need $4^N + 2N - 1$ quantum gates for exact preparation and $6N$ for approximate preparation. In machine learning, the exact solution is unnecessary. Thus, $6N$ quantum gates are enough.

On the other hand, the accuracy is independent of the number of required quantum gates. It is determined by ξ_1 as shown in new Fig.1c.

Radial basis function. In this paper, we have used the linear Kernel function (8), which is efficient to classify data points linearly. However, it is not sufficient to classify data points which are not separated by the linear function. The radial basis function[55,56] is given by

$$K_{ij} = \exp\left[-\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|^2 / 2\sigma^2\right], \tag{39}$$

with a free parameter σ. It is used for a nonlinear classification[57]. It is known[58,59] that the depth of a quantum is linear to the dimension of the feature space N.

Conclusion

We have proposed that the matrix F is efficiently inputted into a quantum computer by using the Γ-matrix expansion method. There are many ways to use a matrix in a quantum computer such as linear regression and principal component analysis. Our method will be applicable to these cases.

Although it is possible to obtain the exact solution for the linear equation by the HHL algorithm, it requires many gates. On the other hand, it is often hard to obtain the exact solution by variational methods since trial functions may be trapped to a local minimum. However, this problem is not serious for the machine learning problem because it is more important to obtain an approximate solution efficiently rather than an exact solution by using many gates. Indeed, our optimized hyperplane also well separates red and blue points as shown in Fig. 1a.

In order to classify M data, we need to prepare $\log_2 M$ qubits. It is hard to execute a large number of data points by current quantum computers. Recently, it is shown that electric circuits may simulate universal quantum gates[60–62] based on the fact that the Kirchhoff law is rewritten in the form of the Schrödinger equation[63]. Our variational algorithm will be simulated by using them.

Methods

Support vector machine. A support vector machine is an algorithm for supervised learning[18,22,23]. We first prepare a set of training data, where each point is marked either in red or blue. Then, we determine a hyperplane separating red and blue points. After learning, input data are classified into red or blue by comparing the input

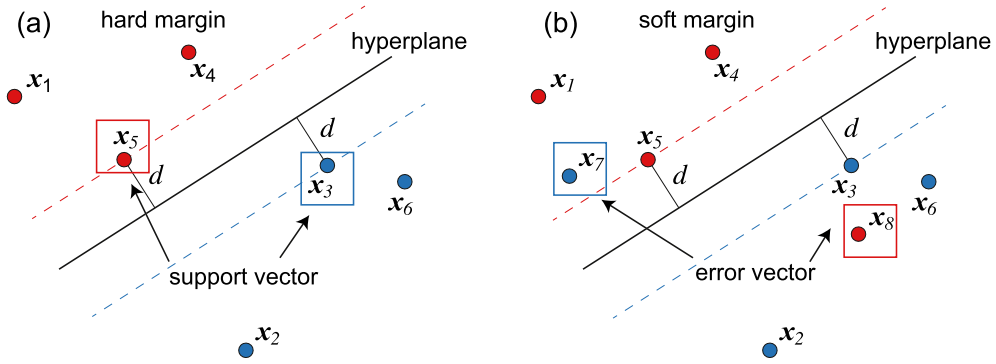

Figure 4. Illustration of the hyperplane and the support vector. Two support vectors are marked by red and blue squares. (**a**) Hard margin where red and blue points are separated perfectly, and (**b**) soft margin where they are separated imperfectly.

data with the hyperplane. The support vector machine maximizes a margin, which is a distance between the hyperplane and data points. If red and blue points are perfectly separated by the hyperplane, it is called a hard margin problem (Fig.4a). Otherwise, it is called a soft margin problem (Fig.4b).

We minimize the distance d_j between a data point $\boldsymbol{x}_j$ and the hyperplane given by

$$d_j = \frac{|\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0|}{|\boldsymbol{\omega}|}. \tag{40}$$

We define support vectors $\boldsymbol{x}$ as the closest points to the hyperplane. There is such a vector in each side of the hyperplane, as shown in Fig. 4a. This is the origin of the name of the support vector machine. Without loss of generality, we set

$$|\boldsymbol{\omega} \cdot \boldsymbol{x} + \omega_0| = 1 \tag{41}$$

for the support vectors, because the hyperplane is present at the equidistance of two closest data points and because it is possible to set the magnitude of $|\boldsymbol{\omega} \cdot \boldsymbol{x} + \omega_0|$ to be 1 by scaling $\boldsymbol{\omega}$ and ω_0. Then, we maximize the distance

$$d = \frac{|\boldsymbol{\omega} \cdot \boldsymbol{x} + \omega_0|}{|\boldsymbol{\omega}|} = \frac{1}{|\boldsymbol{\omega}|}, \tag{42}$$

which is identical to minimize $|\boldsymbol{\omega}|$.

First, we consider the hard margin problem, where red and blue points are perfectly separable. All red points satisfy $\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0 > 1$ and all blue points satisfy $\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0 < -1$. We introduce variables y_j, where $y_j = 1$ for red points and $y_j = -1$ for blue points. Using them, the condition is rewritten as

$$\big(\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0\big) y_j \geq 1 \tag{43}$$

for each j. The problem is reduced to find the minimum of $|\boldsymbol{\omega}|^2$ under the above inequalities. The optimization under inequality conditions is done by the Lagrange multiplier method with the Karush-Kuhn-Tucker condition[64]. It is expressed in terms of the Lagrangian as

$$L(\boldsymbol{\omega}, \omega_0, \boldsymbol{\alpha}) = \frac{1}{2}|\boldsymbol{\omega}|^2 - \sum_j \beta_j [\big(\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0\big) y_j - 1], \tag{44}$$

where β_j are Lagrange multipliers to ensure the constraints.

For the soft margin case, we cannot separate two classes exactly. In order to treat this case, we introduce slack variables ξ_j satisfying

$$\big(\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0\big) y_j \geq 1 - \xi_j, \qquad \xi_j \geq 0 \tag{45}$$

and redefine the cost function as

$$E_{\text{cost}} = \frac{1}{2}|\boldsymbol{\omega}|^2 + \gamma \sum_{j=1}^{M} \xi_j^2. \tag{46}$$

Here, $\gamma = \infty$ corresponds to the hard margin. The second term represents the penalty for some of data points to have crossed over the hyperplane. The Lagrangian is modified as

$$L(\boldsymbol{\omega}, \omega_0, \xi_i, \boldsymbol{\beta}) = \frac{1}{2}|\boldsymbol{\omega}|^2 + \gamma \sum_{j=1}^{M} \xi_j^2 - \sum_{j=1}^{M} \left[\left(\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0 \right) \beta_j y_j - (1 - \xi_i) \right]. \tag{47}$$

The stationary points are determined by

$$\frac{\partial L}{\partial \boldsymbol{\omega}} = \boldsymbol{\omega} - \sum_{j=1}^{M} \beta_j y_j \boldsymbol{x}_j = 0, \tag{48}$$

$$\frac{\partial L}{\partial \omega_0} = -\sum_{j=1}^{M} \beta_j y_j = 0, \tag{49}$$

$$\frac{\partial L}{\partial \xi_j} = \gamma \xi_j - \beta_j = 0, \tag{50}$$

$$\frac{\partial L}{\partial \beta_j} = \left(\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0 \right) y_j - (1 - \xi_i) = 0. \tag{51}$$

We may solve these equations to determine $\boldsymbol{\omega}$ and ν_j as

$$\boldsymbol{\omega} = \sum_{j=1}^{M} \beta_j y_j \boldsymbol{x}_j, \tag{52}$$

from (48), and

$$\xi_j = \beta_j / \gamma \tag{53}$$

from (50). Inserting them into (51), we find

$$y_j \sum_{i=1}^{M} \left(\beta_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x}_j + \omega_0 \right) - \left(1 - \beta_j / \gamma \right) = 0. \tag{54}$$

Since $y_j^2 = 1$, it is rewritten as

$$\omega_0 + \sum_{i=1}^{M} \left(\boldsymbol{x}_i \cdot \boldsymbol{x}_j + \delta_{ij} / \gamma \right) \beta_i y_i = y_j. \tag{55}$$

Since β_j appears always in a pair with y_j, we introduce a new variable defined by

$$\alpha_j = \beta_j y_j, \tag{56}$$

and we define the Kernel matrix K_{ij} as

$$K_{ij} = \boldsymbol{x}_i \cdot \boldsymbol{x}_j. \tag{57}$$

Then, ω_0 and α_j are obtained by solving linear equations

$$\sum_{i=1}^{M} \alpha_j = 0, \tag{58}$$

$$\omega_0 + \sum_{i=1}^{M} \left(\boldsymbol{x}_i \cdot \boldsymbol{x}_j + \delta_{ij} / \gamma \right) \alpha_i = y_j, \tag{59}$$

which are summarized as

$$\begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & & & \\ \vdots & & K + I_M / \gamma & \\ 1 & & & \end{pmatrix} \begin{pmatrix} \omega_0 \\ \alpha_1 \\ \vdots \\ \alpha_M \end{pmatrix} = \begin{pmatrix} 0 \\ y_1 \\ \vdots \\ y_M \end{pmatrix}, \tag{60}$$

which is Eq. (6) in the main text. Finally, $\boldsymbol{\omega}$ is determined by

$$\boldsymbol{\omega} = \sum_{j=1}^{M} \alpha_j \boldsymbol{x}_j. \tag{61}$$

Once the hyperplane is determined, we can classify new input data into red if

$$\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0 > 0 \tag{62}$$

and blue if

$$\boldsymbol{\omega} \cdot \boldsymbol{x}_j + \omega_0 < 0. \tag{63}$$

Thus, we obtain the hyperplane for binary classification.

Γ matrix expansion. We explicitly show how to calculate c_j in (17) based on the Γ matrix expansion for the one and two qubits.

One qubit. We show an explicit example of the Γ -matrix expansion for one qubit. One qubit is represented by a 2×2 matrix,

$$F = \begin{pmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{pmatrix}. \tag{64}$$

The column vectors are explicitly given by

$$|f_1\rangle = \begin{pmatrix} F_{11} \\ F_{21} \end{pmatrix} = F_{11}|0\rangle + F_{21}|1\rangle, \tag{65}$$

$$|f_2\rangle = \begin{pmatrix} F_{12} \\ F_{22} \end{pmatrix} = F_{12}|0\rangle + F_{22}|1\rangle. \tag{66}$$

The coefficient c_j in (17) is calculated as

$$c_j = \mathrm{Tr}\left[\sigma_j F\right] = \langle 0|\sigma_j|f_1\rangle + \langle 1|\sigma_j|f_2\rangle = \sum_{p=0,1} \langle p|\sigma_j|f_p\rangle = \sum_{p=0,1} \langle 0|U_X^{(p)} \sigma_j U_{f_p}|0\rangle. \tag{67}$$

Two qubits. Next, we show an explicit example of the Γ-matrix expansion for two qubits. Two qubits are represented by a 4×4 matrix,

$$F = \begin{pmatrix} F_{11} & F_{12} & F_{13} & F_{14} \\ F_{21} & F_{22} & F_{23} & F_{24} \\ F_{31} & F_{32} & F_{33} & F_{34} \\ F_{41} & F_{42} & F_{43} & F_{44} \end{pmatrix}. \tag{68}$$

The column vectors are explicitly given by

$$|f_1\rangle = \begin{pmatrix} F_{11} \\ F_{21} \\ F_{31} \\ F_{41} \end{pmatrix} = F_{11}|00\rangle + F_{21}|01\rangle + F_{31}|10\rangle + F_{41}|11\rangle, \tag{69}$$

$$|f_2\rangle = \begin{pmatrix} F_{12} \\ F_{22} \\ F_{32} \\ F_{42} \end{pmatrix} = F_{12}|00\rangle + F_{22}|01\rangle + F_{32}|10\rangle + F_{42}|11\rangle, \tag{70}$$

$$|f_3\rangle = \begin{pmatrix} F_{13} \\ F_{23} \\ F_{33} \\ F_{43} \end{pmatrix} = F_{13}|00\rangle + F_{23}|01\rangle + F_{33}|10\rangle + F_{43}|11\rangle, \tag{71}$$

$$|f_4\rangle = \begin{pmatrix} F_{14} \\ F_{24} \\ F_{34} \\ F_{44} \end{pmatrix} = F_{14}|00\rangle + F_{24}|01\rangle + F_{34}|10\rangle + F_{44}|11\rangle. \tag{72}$$

The coefficient c_j in (17) is calculated as

$$c_j = \mathrm{Tr}\left[\Gamma_j F\right] = \langle 00|\Gamma_j|f_1\rangle + \langle 01|\Gamma_j|f_2\rangle + \langle 10|\Gamma_j|f_3\rangle + \langle 11|\Gamma_j|f_4\rangle = \sum_{p=0}^{3} \langle\langle p|\Gamma_j|f_p\rangle = \sum_{p=0}^{3} \langle\langle 0|U_X^{(p)} \Gamma_j U_{f_p}|0\rangle\rangle. \tag{73}$$

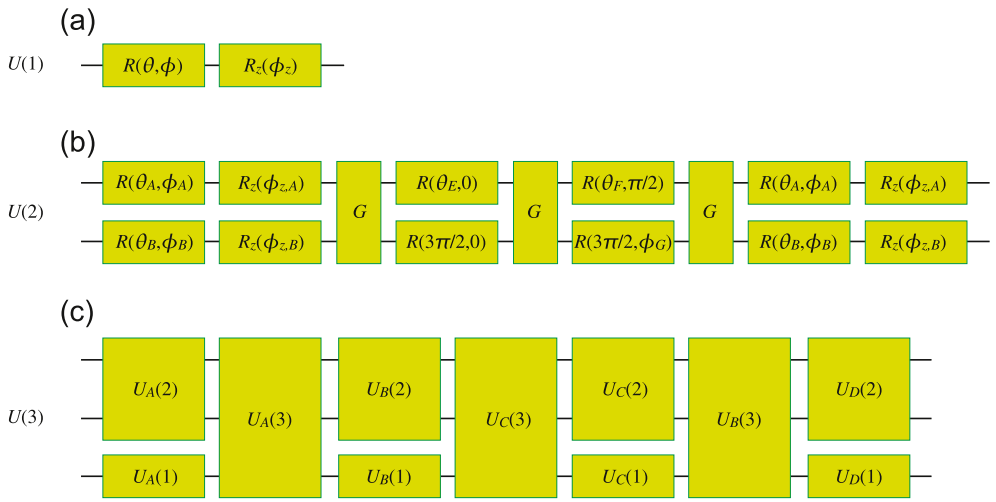

Figure 5. Universal quantum circuits for (**a**) one, (**b**) two and (**c**) three qubits.

Universal quantum circuits. Angle variables are used as variational parameters in a universal quantum circuit learning. We present examples for one, two and three qubits.

One-qubit universal quantum circuit. The single-qubit rotation gates are defined by

$$R(\theta,\phi) = \exp\left[-i\theta\left(\sigma_x\cos\phi + \sigma_y\sin\phi\right)/2\right], \tag{74}$$

$$R_z(\phi_z) = \exp\left[-i\sigma_z\phi_z/2\right]. \tag{75}$$

The one-qubit universal quantum circuit is constructed as

$$U^{(1)}(\theta,\phi,\phi_z) = R(\theta,\phi)R_z(\phi_z) = \begin{pmatrix} e^{-i\phi_z/2}\cos\frac{\theta}{2} & -ie^{i(\phi_z/2-\phi)}\sin\frac{\theta}{2} \\ -ie^{-i(\phi_z/2-\phi)}\sin\frac{\theta}{2} & e^{i\phi_z/2}\cos\frac{\theta}{2} \end{pmatrix}. \tag{76}$$

We show a quantum circuit in Fig. 5a. There are three variational parameters.

It is obvious that an arbitrary state is realized starting from the state $|0\rangle$ as

$$U(1)\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} e^{-i\phi_z/2}\cos\frac{\theta}{2} \\ -ie^{-i(\phi_z/2-\phi)}\sin\frac{\theta}{2} \end{pmatrix}. \tag{77}$$

Two-qubit universal quantum circuit. The two-qubit universal quantum circuit is constructed as[43]

$$\begin{aligned} U(2) \equiv & \left[U^{(1)}\left(\theta_A,\phi_A,\phi_{z,A}\right)\otimes U^{(1)}\left(\theta_B,\phi_B,\phi_{z,B}\right)\right]U_G\left[R(\theta_E,0)\otimes R\left(\frac{3\pi}{2},0\right)\right]U_G\left[R\left(\theta_F,\frac{\pi}{2}\right)\otimes R\left(\frac{3\pi}{2},\theta_G\right)\right] \\ & U_G\left[U^{(1)}\left(\theta_C,\phi_C,\phi_{z,C}\right)\otimes U^{(1)}\left(\theta_D,\phi_D,\phi_{z,D}\right)\right], \end{aligned} \tag{78}$$

where the entangling two-qubit gate is defined by[43]

$$U_G = e^{-i\pi/4}\exp\left[\frac{i\pi}{4}\sigma_z\otimes\sigma_z\right]. \tag{79}$$

The two-qubits universal quantum circuit contains 15 variational parameters. We show a quantum circuit in Fig. 5b.

Three-qubit universal quantum circuit. The three-qubit universal quantum circuit is constructed as

$$U(3) \equiv \left[U_A^{(2)}\otimes U_A^{(1)}\right]U_A(3)\left[U_B^{(2)}\otimes U_B^{(1)}\right]U_C(3)\left[U_C^{(2)}\otimes U_C^{(1)}\right]U_B(3)\left[U_D^{(2)}\otimes U_D^{(1)}\right], \tag{80}$$

where $U_A^{(1)}, U_B^{(1)}, U_C^{(1)}$, and $U_D^{(1)}$ are one-qubit universal quantum circuits, while $U_A^{(2)}, U_B^{(2)}, U_C^{(2)}$, and $U_D^{(2)}$ are two-qubit universal quantum circuit and

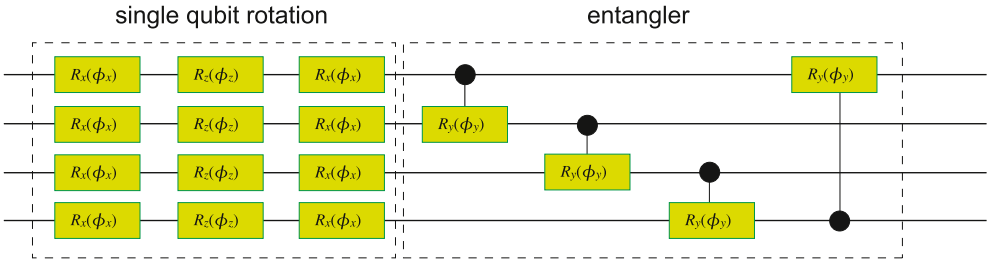

Figure 6. Hardware-efficient universal quantum circuits for four qubits[54].

$$U_A(3) = \exp\left[i\left(\theta^A_{xxz}\sigma_x\otimes\sigma_x\otimes\sigma_z + \theta^A_{yyz}\sigma_x\otimes\sigma_x\otimes\sigma_z + \theta^A_{zzz}\sigma_z\otimes\sigma_z\otimes\sigma_z\right)\right], \tag{81}$$

$$U_B(3) = \exp\left[i\left(\theta^B_{xxz}\sigma_x\otimes\sigma_x\otimes\sigma_z + \theta^B_{yyz}\sigma_x\otimes\sigma_x\otimes\sigma_z + \theta^B_{zzz}\sigma_z\otimes\sigma_z\otimes\sigma_z\right)\right], \tag{82}$$

$$U_C(3) = \exp\left[i\left(\theta^C_{xxx}\sigma_x\otimes\sigma_x\otimes\sigma_x + \theta^C_{yyx}\sigma_y\otimes\sigma_y\otimes\sigma_x + \theta^C_{zzx}\sigma_z\otimes\sigma_z\otimes\sigma_x + \theta^C_{00x}\sigma_0\otimes\sigma_0\otimes\sigma_x\right)\right]. \tag{83}$$

Eplicit quantum circuits for $U_A(3)$, $U_B(3)$ and $U_C(3)$ are shown in Ref.[42]. The three-qubits universal quantum circuit contains 82 variational parameters. We show a quantum circuit in Fig. 5c.

Multi-qubit universal quantum circuit. General multi-qubit universal quantum circuit is constructed in Ref.[39]. The minimum numbers of variational parameters are $4^N - 1$ for N-qubit unicersal quantum circuits. However, we need more variational parameters in the currently known algorithm for $N \geq 3$.

Actually, multi-qubit universal quantum circuits are well approximated by the hardware-efficient quantum circuit[52–54]. They are constructed as

$$U \equiv U_{\mathrm{CNOT}} \otimes U_{\mathrm{Rot}}, \tag{84}$$

with the use of the single qubit rotation

$$U_{\mathrm{Rot}} \equiv \bigotimes_{n=1}^{N} \exp\left[i\theta^{(n)}_1\sigma_x\right]\exp\left[i\theta^{(n)}_2\sigma_z\right]\exp\left[i\theta^{(n)}_3\sigma_x\right], \tag{85}$$

and the CNOT gates

$$U_{\mathrm{CROT}} \equiv \bigotimes_{n=1}^{N} U^{n\to n+1}_{\mathrm{CROT}}, \tag{86}$$

where $U^{n\to n+1}_{\mathrm{CROT}}$ stands for the controlled rotation gate with the controlled qubit being n and the target qubit being $n+1$. We need the order of $4N$ quantum gates for N-qubit universal quantum circuits. We show an example of the hardware-efficient quantum circuit with $N = 4$ in Fig. 6.

In addition, an ansatz based on the restricted Boltzmann machine requires N^2 quantum gates, while a unitary-coupled cluster ansatz requires N^4 quantum gates[16,34].

Simulations. All of the numerical calculations are carried out by Mathematica.

Received: 11 February 2021; Accepted: 12 April 2022
Published online: 26 April 2022

References

1. Feynman, R. P. Simulating physics with computers. *Int. J. Theor. Phys.* **21**, 467 (1982).
2. DiVincenzo, D. P. Quantum computation. *Science* **270**, 255 (1995).
3. Nielse, M. & Chuang, I. *Quantum Computation and Quantum Information* 189 (Cambridge University Press, 2016) ISBN 978-1-107-00217-3.
4. Lloyd S., Mohseni M. & Rebentrost P. Quantum algorithms for supervised and unsupervised machine learning. arXiv:1307.0411.
5. Schuld, M., Sinayskiyv, I. & Petruccione, F. An introduction to quantum machine learning. *Contemp. Phys.* **56**, 172 (2015).
6. Biamonte, J. Quantum machine learning. *Nature* **549**, 195 (2017).
7. Wittek, P. *Quantum Machine Learning: What Quantum Computing Means to Data Mining* (Academic Press, 2014).
8. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **103**, 150502 (2009).
9. Wiebe, N., Braun, D. & Lloyd, S. Quantum algorithm for data fitting. *Phys. Rev. Lett.* **109**, 050505 (2012).
10. Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum support vector machine for big data classification. *Phys. Rev. Lett.* **113**, 130503 (2014).
11. Li, Z., Liu, X., Xu, N. & Du, J. Experimental realization of a quantum support vector machine. *Phys. Rev. Lett.* **114**, 140504 (2015).

12. Schuld, M. & Killoran, N. Quantum machine learning in feature Hilbert spaces. *Phys. Rev. Lett.* **122**, 040504 (2019).
13. Havlicek, V. *et al.* Supervised learning with quantum-enhanced feature spaces. *Nature* **567**, 209 (2019).
14. Lamata, L. Quantum machine learning and quantum biomimetics: A perspective. *Mach. Learn. Sci. Technol.* **1**, 033002 (2020).
15. Cong, I., Choi, S. & Lukin, M. D. Quantum convolutional neural networks. *Nat. Phys.* **15**, 1273 (2019).
16. Sajjan, M., Sureshbabu, S. H. & Kais, S. Quantum machine-learning for eigenstate filtration in two-dimensional materials. *Am. Chem. Soc.* **143**, 18426 (2021).
17. Xia, R. & Kais, S. Quantum machine learning for electronic structure calculations. *Nat. Commun.* **9**, 4195 (2018).
18. Vapnik, V. & Lerner, A. Pattern recognition using generalized portrait method. *Autom. Remote Control* **24**, 774 (1963).
19. Sarma A., Chatterjee R., Gili K. & Yu T. Quantum unsupervised and supervised learning on superconducting processors. arXiv:1909.04226.
20. Liu, Y., Arunachalam, S. & Temme, K. A rigorous and robust quantum speed-up in supervised machine learning. *Nat. Phys.* **17**, 1013 (2021).
21. Date, P., Arthur, D. & Pusey-Nazzaro, L. QUBO formulations for training machine learning models. *Sci. Rep.* **11**, 10029 (2021).
22. Noble, W. S. What is a support vector machine?. *Nat. Biotechnol.* **24**, 1565 (2006).
23. Suykens, J. A. K. & Vandewalle, J. Least squares support vector machine classifiers. *Neural Process. Lett.* **9**, 293 (1999).
24. Zhaokai, L., Xiaomei, L., Nanyang, X. & Jiangfeng, D. Experimental realization of a quantum support vector machine. *Phys. Rev. Lett.* **114**, 140504 (2015).
25. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **15**, 150502 (2009).
26. Cai, X. D. *et al.* Experimental quantum computing to solve systems of linear equations. *Phys. Rev. Lett.* **110**, 230501 (2013).
27. Barz, S. *et al.* A two-qubit photonic quantum processor and its application to solving systems of linear equations. *Sci. Rep.* **4**, 6115 (2014).
28. Pan, J. *et al.* Experimental realization of quantum algorithm for solving linear systems of equations. *Phys. Rev. A* **89**, 022313 (2014).
29. Farhi E., Goldstone J. & Gutmann S. *A Quantum Approximate Optimization Algorithm*. MIT-CTP/4610.
30. Peruzzo, A. *et al.* A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.* **5**, 4213 (2014).
31. Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. *Phys. Rev. A* **98**, 032309 (2018).
32. Bravo-Prieto C., LaRose R., Cerezo M., Subasi Y., Cincio L. & Coles P. J. *Variational Quantum Linear Solver*. LA-UR-19-29101.
33. Xu X., Sun J., Endo S., Li Y., Benjamin S. C. & Yuan X. *Variational Algorithms for Linear Algebra*. arXiv:1909.03898.
34. Cerezo, M. *et al.* Variational quantum algorithms. *Nat. Rev. Phys.* **3**, 625 (2021).
35. Deutsch, D. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. R. Soc. A* **400**, 97 (1985).
36. Dawson C. M. & Nielsen M. A. *The Solovay–Kitaev Algorithm*. arXiv:quant-ph/0505030.
37. Kraus, B. & Cirac, J. I. Optimal creation of entanglement using a two-qubit gate. *Phys. Rev. A* **63**, 062309 (2001).
38. Vidal, G. & Dawson, C. M. Universal quantum circuit for two-qubit transformations with three controlled-NOT gates. *Phys. Rev. A* **69**, 010301 (2004).
39. Mottonen, M., Vartiainen, J. J., Bergholm, V. & Salomaa, M. M. Quantum circuits for general multiqubit gates. *Phys. Rev. Lett.* **93**, 130502 (2004).
40. Shende, V. V., Markov, I. L. & Bullock, S. S. Minimal universal two-qubit controlled-NOT-based circuits. *Phys. Rev. A* **69**, 062321 (2004).
41. Vatan F. & Williams C. P. *Realization of a General Three-Qubit Quantum Gate*. arXiv:quant-ph/0401178.
42. Sousa P. B. M. & Ramos R. V. *Universal Quantum Circuit for n-Qubit Quantum Gate: A Programmable Quantum Gate*. arXiv:quant-ph/0602174.
43. Hanneke, D. *et al.* Realization of a programmable two-qubit quantum processor. *Nat. Phys.* **6**, 13 (2009).
44. DiCarlo, L. *et al.* Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature* **460**, 240 (2009).
45. Qiang, X. *et al.* Large-scale silicon quantum photonics implementing arbitrary two-qubit processing. *Nat. Photonics* **12**, 534 (2018).
46. Roy, T. *et al.* Programmable superconducting processor with native three-qubit gates. *Appl. Phys. Rev.* **14**, 014072 (2020).
47. Monmasson, E. & Cirstea, M. N. FPGA design methodology for industrial control systems: A review. *IEEE Trans. Ind. Electron.* **54**, 1824 (2007).
48. Naouar, M.-W., Monmasson, E., Naassani, A. A., Slama-Belkhodja, I. & Patin, N. FPGA-based current controllers for AC machine drives: A review. *IEEE Trans. Ind. Electron.* **54**, 1907 (2007).
49. Kuon, I., Tessier, R. & Rose, J. FPGA architecture: Survey and challenges. *Found. Trends Electron. Des. Autom.* **2**, 135 (2007).
50. Shawahna, A., Sait, S. M. & El-Maleh, A. FPGA-based accelerators of deep learning networks for learning and classification: A review. *IEEE Access* **7**, 7823 (2018).
51. Ding C., Bao T.-Y. & Huang H.-L. *Quantum-Inspired Support Vector Machine*. arXiv:1906.08902.
52. Kandala, A. *et al.* Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* **549**, 242 (2017).
53. Kardashin, A., Uvarov, A., Yudin, D. & Biamonte, J. Certified variational quantum algorithms for eigenstate preparation. *Phys. Rev. A* **102**, 052610 (2020).
54. Kardashin, A., Pervishko, A., Biamonte, J. & Yudin, D. Numerical hardware-efficient variational quantum simulation of a soliton solution. *Phys. Rev. A* **104**, 020402 (2021).
55. Broomhead D. S. & Lowe D. *Radial Basis Functions, Multi-variable Functional Interpolation and Adaptive Networks. Tech. Rep.* (Royal Signals and Radar Establishment Malvern (United Kingdom), 1988).
56. Broomhead, D. & Lowe, D. Multivariable functional interpolation and adaptive networks. *Complex Syst.* **2**, 321 (1988).
57. Vert J.-P., Tsuda K. & Scholkopf B. A primer on kernel methods. *Kernel Methods in Computational Biology* (2004).
58. Haug, T., Bharti, K. & Kim, M. S. Capacity and quantum geometry of parametrized quantum circuits. *Quantum Phys. Rev. X* **2**, 040309 (2021).
59. Haug T., Self C.N. & Kim M. S. *Large-Scale Quantum Machine Learning*. arXiv:2108.01039.
60. Ezawa, M. Electric circuits for universal quantum gates and quantum Fourier transformation. *Phys. Rev. Res.* **2**, 023278 (2020).
61. Ezawa, M. Dirac formulation for universal quantum gates and Shor's integer factorization in high-frequency electric circuits. *J. Phys. Soc. Jpn.* **89**, 124712 (2020).
62. Ezawa, M. Universal quantum gates, artificial neurons, and pattern recognition simulated by LC resonators. *Phys. Rev. Res.* **3**, 023051 (2021).
63. Ezawa, M. Electric-circuit simulation of the Schrodinger equation and non-Hermitian quantum walks. *Phys. Rev. B* **100**, 165419 (2019).
64. Kuhn, H. W. & Tucker, A. W. Nonlinear programming. *Proceedings of 2nd Berkeley Symposium* 481–492 (University of California Press).

Acknowledgements

The author is very much grateful to E. Saito and N. Nagaosa for helpful discussions on the subject. This work is supported by the Grants-in-Aid for Scientific Research from MEXT KAKENHI (Grants No. JP17K05490 and No. JP18H03676). This work is also supported by CREST, JST (JPMJCR16F1 and JPMJCR20T2).

Author contributions

M.E. conceived the idea, performed the analysis, and wrote the manuscript.

Competing interests

The author declares no competing interests.

Additional information

Correspondence and requests for materials should be addressed to M.E.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.