Proceedings of the 5th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/100/2025.17860

Research on Quantum Computing Acceleration of Support Vector Machines in Multi-dimensional Nonlinear Feature Spaces

Haozhe Liu[1,a,*]

[1]College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China
a. 20221522@mail.sheip.edu.cn
**corresponding author*

Abstract: The current development of Support Vector Machines (SVM) has reached a bottleneck, with issues such as long training times and weak interpretability when dealing with large-scale, multi-dimensional data. This paper introduces the concept of Quantum Support Vector Machines (QSVM) and achieves efficient solutions through quantum algorithms such as the HHL algorithm. The research designs a computational architecture that combines classical and quantum computing, utilizing the Pauli decomposition of Hermitian matrices to simulate the quantum simulation of Hamiltonian quantities and implementing the quantum simulation of unitary operators. Based on the conclusions, a complete quantum linear solver circuit is designed, achieving exponential growth in computational complexity. Experiments using the Iris dataset demonstrate the excellent classification performance of QSVM, which outperform classical SVM in classification accuracy, computational time complexity, and memory requirements. More efficient quantum mapping algorithms and quantum circuit optimization methods are implemented, providing new ideas and methods for the application of SVM to large-scale datasets.

Keywords: Quantum Linear Solver, Support Vector Machine, Hamiltonian Simulation, Quantum Circuit.

1. Introduction

The Support Vector Machine (SVM) is a supervised learning model initially developed based on the research by Soviet scholars Vladimir N. Vapnik and Alexander Y. Lerner in 1963. It is commonly used for pattern recognition, classification, and regression analysis, serving as a widely applied discriminative method in the field of machine learning. In 1992, Bernhard E. Boser, Isabelle M. Guyon, and Vapnik introduced kernel methods, enabling SVMs to handle non-linearly separable data [1].

The SVM algorithm essentially solves a quadratic programming problem. When dealing with large-scale sample data, the number of matrix elements grows quadratically with the size of the training set. In 2014, P. Rebentrost et al. proposed the Quantum Support Vector Machine (QSVM) [2]. The QSVM transforms the quadratic programming problem of SVMs into a least squares problem and leverages quantum algorithms to efficiently solve key steps such as vector inner products, thereby significantly reducing computational complexity. Compared to classical SVMs, QSVMs have significant advantages in computational efficiency, especially when dealing with large-scale datasets, where the complexity is reduced exponentially.

Since 1995, when Professor Kak of Louisiana State University proposed the concept of quantum neural computing, the theoretical and applied research on quantum computing acceleration has gradually gained popularity. In 2001, Horn et al. introduced a new clustering algorithm based on quantum mechanics for unsupervised clustering. In 2009, Harrow et al. proposed the famous HHL algorithm, which achieved exponential speedup compared to the best-known classical algorithm with a time complexity of $O(n)$[3].

Although many scholars have conducted research on the time complexity and practical applications of quantum support vector machines and made outstanding contributions to the direction of quantum computing, there are still issues of low accuracy and large approximation values in unitary matrix simulations, and the feature dimension of the training dataset is only two-dimensional.

To apply quantum support vector machines in multi-feature dimensional non-linear data spaces, this paper designs a computational architecture combining classical and quantum computing. It utilizes Pauli decomposition of Hermitian matrices to simulate the quantum simulation of Hamiltonians. Based on the HHL algorithm, quantum phase estimation, quantum Fourier transform, and conditional rotation transformations of quantum states are designed to construct a quantum circuit for solving linear equations and realize the quantum simulation of unitary operators[4].

2. Quantum Algorithm Design

To solve the equation $\boldsymbol{Ax} = \boldsymbol{b}$ for a given Hermitian matrix $\boldsymbol{A} \in \boldsymbol{RN} \times \boldsymbol{N}$ and a vector $\vec{\boldsymbol{b}} \in \boldsymbol{RN}$ on a quantum computer using the HHL algorithm, the first step is to perform quantum state encoding. Assuming the dimension of the Hermitian matrix $\boldsymbol{A}$ is $\boldsymbol{N} = \boldsymbol{2n}$, where $\boldsymbol{n}$ is the number of quantum bits used for quantum state encoding. For vectors $\vec{\boldsymbol{b}}$ and $\vec{\boldsymbol{x}}$, during quantum state encoding, the i-th element of b (or x) is encoded as the i-th basis state of the quantum superposition state $|b\rangle$ (or $|x\rangle$). Through a quantum circuit, the solution satisfying A$|x\rangle = |b\rangle$ is obtained[5].

The HHL algorithm circuit design mainly consists of three modules: quantum phase estimation, controlled rotation of quantum states, and quantum state inversion and calculation.

2.1. Principle of the HHL Algorithm

Given a Hermitian matrix A and a vector $\vec{b}$, suppose the spectral decomposition of A is as follows:

$$A = \sum_{j=1}^{N} \lambda_j |u_j\rangle\langle u_j| \tag{1}$$

Express $|b\rangle$ in terms of the basis $\{|u_j\rangle\}$: $|b\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle$. Using quantum phase estimation (QPE) with $U = e^{iA\Delta t}$, the result is as follows:

$$|b\rangle \overset{\text{QPE}}{\rightarrow} \sum_{j=1}^{N} \beta_j |u_j\rangle \left|\tilde{\lambda_j}\right\rangle \tag{2}$$

For equation (2), applying operator U yields $e^{iA\Delta t} = \sum_{j=1}^{N} e^{i\lambda_j \Delta t} |u_j\rangle\langle u_j|$, where $|\tilde{\lambda_j}\rangle$ represents an estimate of the eigenvalue λ_j (the wavy line indicates an estimate, the same applies to the following). The solution to the equation is:

$$|x\rangle = A^{-1}|b\rangle = \sum_j \beta_j(\lambda_j)^{-1}|u_j\rangle \tag{3}$$

The key of the algorithm is to simulate the operator $U = e^{iA\Delta t}$ on $|\mathrm{b}\rangle$, which will be described in detail later.

2.2. Data preprocessing and quantum state encoding

In the HHL algorithm, matrix A is required to be Hermitian, that is, matrix A needs to satisfy $A^\dagger = A$, this requirement can be relaxed in the concrete implementation, if A is not Hermitian, construct $\tilde{A}$ as follows:

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} \tag{4}$$

Then solving the equation $\tilde{A}\vec{y} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}$, it is verified that the solution $\vec{y}$ of the obtained equation must have the form $\vec{y} = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix}$, where $A\vec{x} = \vec{b}$. Because the HHL algorithm is a quantum algorithm, the vector $\vec{b}$ needs to be encoded into the quantum state and input in the quantum state $|b\rangle$.

One of the keys of the HHL algorithm is the quantum state encoding of vectors and matrices, and the target register is set to store the information of quantum state $|b\rangle$. The initial state of the quantum state is $|0\rangle^{\otimes n}$, that is, all qubits are in the $|0\rangle$ state, and the value of the vector $\vec{b}$ is encoded to $|\mathrm{b}\rangle$ to satisfy

$$|b\rangle = \sum_{i=1}^{N} b_i\,|i\rangle \tag{5}$$

Where b_i is the i-th component of vector b and $\sum_i |b_i|^2 = 1$.

In the concrete implementation, the rotation gate is used to encode the components of the vector into the amplitude of the qubit. As shown in Figure. 1, using a rotation gate, the quantum state is rotated from $|0\rangle$ to the target state related to the Angle θ. The amplitude is encoded by adjusting the rotation Angle. The rotation Angle θ is calculated based on the inverse trigonometric function of the amplitude:

$$\theta = 2 \times arcsin(b_i) \tag{6}$$

After rotation, the amplitude of each qubit has been associated with the corresponding component of vector $\vec{b}$, and the quantized representation $|\mathrm{b}\rangle$ of vector $\vec{b}$ is obtained[6].

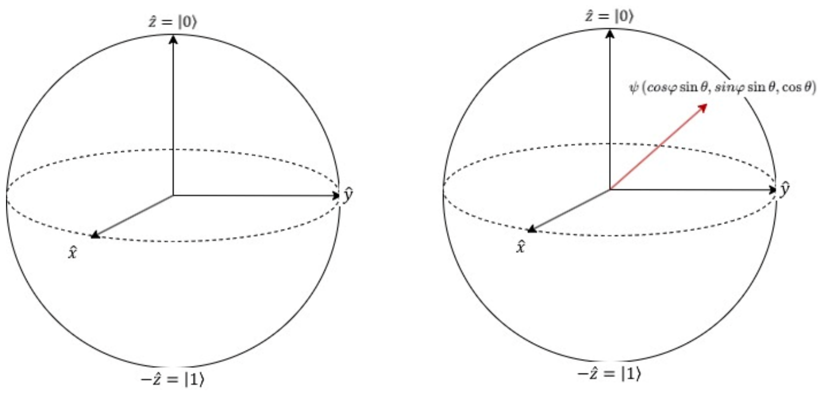

Figure 1: Quantum states are encoded on the Bloch sphere

2.3. Quantum phase estimation

For a unitary matrix U, which has a complex eigenvalue $e^{i\varphi}$ and an eigenvector $|\mathrm{u}\rangle$ with a modulus of 1, the purpose of the quantum phase estimation algorithm is to estimate the value of the phase φ within a certain error range[7].

In this paper, the unitary operator $U = e^{i2\pi A}$ corresponding to matrix A has an eigenvalue $e^{i2\pi\lambda}$ and an eigenvector $|\mathrm{u}\rangle$, where $|\mathrm{u}\rangle$ is the eigenvector of the real eigenvalue λ of A.

The steps of phase estimation can be divided into three steps. Firstly, Hadamard gate operation is performed on all qubits of the phase register, and controlled U gate operation is performed continuously on the control register, where the power of U gate is $2^0, 2^1, \dots, 2^{t-1}$, the control bits are $q_{t-1}, q_{t-2}, \dots, q_1, q_0$, then the state in the first register is going to be

$$|\psi_1\rangle = \frac{1}{2^{\frac{t}{2}}}(|0\rangle + e^{i2\pi 2^{t-1}\varphi}|1\rangle)(|0\rangle + e^{i2\pi 2^{t-2}\varphi}|1\rangle) \dots (|0\rangle + e^{i2\pi 2^{0}\varphi}|1\rangle)$$

That is

$$|\psi_1\rangle = \frac{1}{2^{\frac{t}{2}}}\sum_{k=0}^{2^t-1} e^{i2\pi\varphi k}|k\rangle \tag{7}$$

Where k is the decimal representation of the tensor product state.

Subsequently, an inverse quantum Fourier transform is applied to the phase register, denoted as $QFT^{\dagger}$ in the circuit. Applying the inverse quantum Fourier transform to $|\psi_1\rangle$ yields $|\psi_2\rangle$[8].

$$|\psi_2\rangle = QFT^{\dagger}|\psi_1\rangle = \frac{1}{2^t}\sum_{x=0}^{2^t-1} a_x|x\rangle \tag{8}$$

Which $a_x = \sum_{k=0}^{2^t-1} e^{2\pi ik(\varphi - x/2^t)}$ for eigen vector $|x\rangle (x = 0.1, \dots, 2^t)$. According to the above equation, when $2^t\varphi$ is an integer and $x = 2^t\varphi$ is satisfied, the probability amplitude is the maximum value 1, and the final state of the first register can accurately reflect φ. When $2^t\varphi$ is not an integer, x is an estimate of φ, and the larger t is, the more accurate the estimate will be.

Finally, the quantum bit of the first register was measured, and the final state of the first register $f = \sum_x^{2^t-1} a_x|x\rangle$, $x = 0,1, \dots, 2^t$, from which the largest amplitude a_{max} is found, and the corresponding intrinsic basis vector $|x\rangle$ in x divided by 2^t is the estimated value of the phase.

Overall, when using t auxiliary qubits, the effect of QPE can be written as follows.

$$|b\rangle|0\rangle^{\otimes t}|0\rangle = \sum_{j=1}^{N} \beta_j|u_j\rangle|0\rangle^{\otimes t}|0\rangle \overset{\text{QPE}}{\rightarrow} \sum_{j=1}^{N} \beta_j|u_j\rangle|\tilde{\varphi}_j\rangle|0\rangle \tag{9}$$

2.4. Conditional rotation

After QPE, available quantum state $\sum_{j=1}^{N} \beta_j|u_j\rangle|\tilde{\varphi}_j\rangle|0\rangle$, To extract the information of λ_j from the quantum state $|\tilde{\varphi}_j\rangle$, the specific implementation using the controlled rotation gate $CR(k)$ is as follows:

$$CR(k)|\tilde{\varphi}\rangle|b\rangle = \{\begin{matrix} |\tilde{\varphi}\rangle|b\rangle & ,k \neq \tilde{\varphi} \\ |\tilde{\varphi}\rangle R_y\left(2\arcsin\frac{C}{\lambda}\right)|b\rangle & ,k = \tilde{\varphi} \end{matrix} \tag{10}$$

When k chooses the correct $\tilde{\varphi}$, the selection operation will be applied to the subsequent qubits. Since the correct $\tilde{\varphi}$ is not yet known, a brute force enumeration of all possible $CR(k)$ is used, and the specific effect is as follows:

$$\prod_{k=1}^{2^t-1} I \otimes CR(k) \sum_{j=1}^{N} \beta_j |u_j\rangle |\tilde{\varphi}_j\rangle |0\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle |\tilde{\varphi}_j\rangle \left(\sqrt{1 - \left(\frac{C}{\lambda_j}\right)^2} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right) \tag{11}$$

Applying the inverse QPE once more, the overall quantum state is as follows:

$$|\psi\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle |0\rangle^{\otimes t} \left(\sqrt{1 - \left(\frac{C}{\lambda_j}\right)^2} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right) \tag{12}$$

2.5. Framework of quantum circuits

The overall framework of HHL is shown in Figure 2. qc defines the qubits used for QPE, qb defines the qubits representing vector $\vec{b}$. In order to make the quantum circuit structure clear, QPE and controlled selection module are encapsulated here (quantum phase estimation is a unitary operation in general, and quantum phase inversion part can be obtained by QPE module mirror).

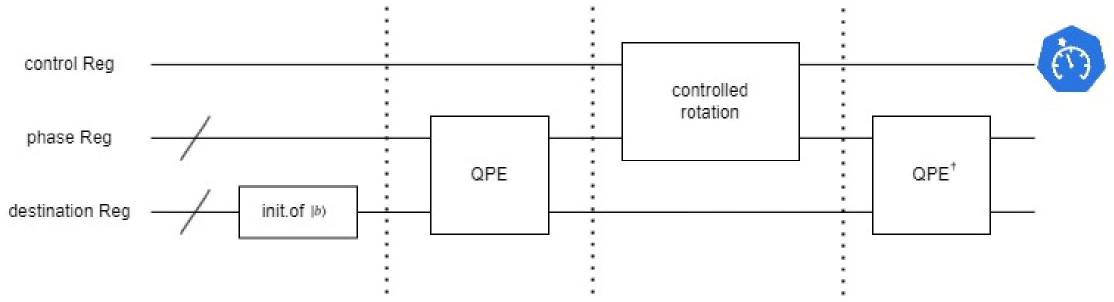

Figure 2: Quantum linear Computing Circuit

The framework of quantum phase estimation is shown in Figure. 3, QFT is the encapsulation of inverse quantum Fourier transform, and U^j represents the controlled U^j operation of the target register with the quantum bits in the phase register as the control bits one by one.

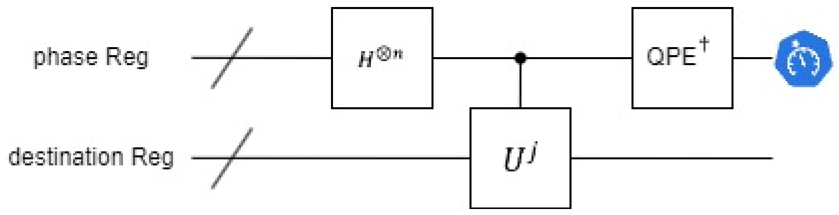

Figure 3: Quantum phase Estimation circuit

In the HHL algorithm, the goal of controlled unitary operation is to realize the time evolution e^{-iAt} of matrix A, and extract its eigenvalues through quantum phase estimation. This process depends on the controlled relationship between the control qubit and the target quantum state. In this paper, the time evolution of matrix is simulated by controlled Z rotation gate (CRZ). For matrix A, the quantum state is evolved in time as follows:

$$U(t) = e^{-iAt} \tag{13}$$

This is the evolution of A quantum state under the action of a matrix A, where t is a time parameter and i is an imaginary unit. To extract the eigenvalues from the matrix A, different powers of this time evolution operator are applied in the quantum phase estimation step: $U^j = e^{-iAt_j}$, where t_j is the time step corresponding to the phase estimation qubit. In a quantum circuit, the controlled unitary operation needs to apply different times of U^j according to the state of the control qubit. CRZ is used to approximately realize this operation. The function of $CRZ(\theta)$ is to perform a Z-axis rotation of the target quantum state with a rotation Angle of θ, which can be expressed as follows:

$$CRZ(\theta) = \begin{pmatrix} 1 & 0 \\ 1 & e^{i\theta} \end{pmatrix} \tag{14}$$

Where the rotation Angle $\theta = -2dt$ is related to the time step dt, and the phase information associated with the matrix A is gradually accumulated through 2^i controlled rotations. This phase information is subsequently decoded via an inverse quantum Fourier transform to extract matrix eigenvalues, which are ultimately used to solve linear systems of equations.

3. Multi-dimensional nonlinear feature space processing

When dealing with multi-dimensional nonlinear feature data, to avoid the high memory occupation and computational complexity caused by the explicit calculation of feature vectors in high-dimensional space. In this paper, the kernel method is used to map the data from the original space to a high-dimensional feature space, which is achieved by computing the kernel matrix. Each element of the kernel matrix represents the inner product between two samples in high-dimensional space. In this paper, Using the linear kernel function $K(x_i, x_j) = x_i \cdot x_j$ and radial basis function (RBF kernel) $K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$ process of linear and nonlinear can be divided into data respectively. Where σ is a parameter that controls the width of the RBF kernel.

The kernel matrix K is derived by calculating the similarity between samples, and each element K_{ij} represents the inner product of the training samples x_i and x_j in a high-dimensional feature space, without explicitly calculating the coordinates of the whole high-dimensional space. Suppose the samples in the training set are $\{x_1, x_2, x_3 \dots x_n\}$, each x_i is a d-dimensional vector, and the kernel matrix K is an $\mathrm{N} \times \mathrm{N}$ symmetric matrix whose entries K_{ij} are defined by

$$K_{ij} = \phi(x_i)^T \phi(x_j) \tag{15}$$

Where ϕ is a mapping function from the original input space to a high-dimensional feature space, and the kernel $K(x_i, x_j)$ is defined as follows:

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{16}$$

The kernel function can compute the inner product of two samples in the high-dimensional feature space without explicitly constructing the mapping ϕ. For each pair of samples (x_i, x_j), $K(x_i, x_j)$ is calculated through the kernel function, and the corresponding position K_{ij} of the kernel matrix is filled to complete the calculation of the kernel matrix K [9].

4. Implementation of support vector machine calculation acceleration

After kernel method mapping, the data that cannot be linearly classified in the original space can be found in a linearly separable hyperplane in the high-dimensional feature space. However, with the growth of the data set, the computational complexity of the kernel matrix increases rapidly in the face of large sample size and high-dimensional features, which brings significant computational overhead. In order to cope with this problem, this paper proposes to introduce quantum computing to accelerate the calculation of kernel matrix, and use the HHL algorithm to significantly accelerate the linear system solution.

Using the property of quantum coherent superposition, for solving the linear equation system of kernel matrix K: $K\alpha = b$, where K is the kernel matrix, α is the Lagrange multiplier, and b is the polarization term. Compared with the classical solution, the time complexity grows polynomial-ly with the increase of the number of samples, which is usually $O(n^3)$ level, the quantum computing acceleration reduces the complexity to the exponential level, namely $\mathrm{O}(\log n)$.

Firstly, the kernel matrix K is calculated by the kernel method, and the matrix K and vector $\vec{b}$ are encoded into quantum states $|k\rangle$ and $|b\rangle$. Through quantum phase estimation, the eigenvalue information of matrix K is obtained by the HHL algorithm, so as to invert the matrix. The algorithm constructs the inverse of the matrix by reverse operation, and finally obtains the quantum state representation of α

$$|\alpha\rangle = K^{-1}|b\rangle \tag{17}$$

By decoding the quantum states, the corresponding solution can be obtained. It greatly improves the efficiency of solving the kernel matrix in the classical SVM, especially in the processing of large-scale high-dimensional data.

5. Experimental results and analysis

5.1. Error analysis

In the HHL algorithm, the time evolution e^{-iAt} is approximately realized by the controlled revolving gate CRZ, which can only be approximated infinitely. In this study, the approximation error is specified as follows:

In terms of the approximation error of the discrete time evolution, since the time evolution of the matrix cannot be directly and accurately realized, this study approximates the simulation by a series of discrete controlled rotation gates. Each time the CRZ gate is applied, the actual rotation performed by the system is a discrete time step dt. The step size of the time evolution operator is as follows:

$$U(t) = e^{-iAt} \approx \prod_{j=1}^{N} e^{-iA\Delta t} \tag{18}$$

Approximation error analysis For dt, the Taylor series expansion of Equation (18) is carried out to evolve in n time steps, and the overall evolution operator is as follows: $U(t) = I - iAt + \frac{(-iAdt)^2}{2} n + O((dt)^3)$, where $n = \frac{t}{dt}$, the approximation error mainly comes from the higher order terms, then the approximation error of the time evolution operator can be obtained as follows:

$$\epsilon_U = \frac{t^2}{2} \|A\|^2 \cdot \frac{1}{(dt)^2} \tag{19}$$

Where $\Delta t = \frac{t}{N}$ is the time step is the number of steps. This discretization introduces an error proportional to the step size Δt, typically $O(\Delta t^2)$. If the number of time steps N is large enough, the error can become small, but at the same time it will increase the circuit depth and complexity.

The approximation of the time evolution depends on the range of eigenvalues of the matrix A. If the eigenvalue λ_i of matrix A has a large span, the error will be amplified. The larger the eigenvalue is, the faster the change of the phase factor $e^{-i\lambda_i t}$ in the time evolution operator, and the discrete-time evolution approximation of these phases will produce large errors. The relationship between the estimated error and the eigenvalue size is as follows:

$$\epsilon_\lambda = \frac{\Delta t}{\lambda_{max}} \tag{20}$$

Where λ_{max} is the largest eigenvalue of matrix A. When the eigenvalues are large, a smaller time step Δt is required to keep the error small.

The accuracy of quantum phase estimation is limited by the number of QPE qubits. QPE performs phase estimation through finite number of qubits, and the estimation accuracy is proportional to the number of qubits t. If t qubits are used, the error in phase estimation is as follows:

$$\epsilon_{QPE} = \frac{2\pi}{2^t} \tag{21}$$

In the concrete implementation, the CRZ controlled rotation gate itself may be affected by hardware implementation errors and noise. On practical quantum hardware, the precision of gate operation may lead to additional errors due to noise and imprecision of gate operation.

Considering the above errors, the total error of the HHL algorithm can be approximately expressed as follows:

$$\epsilon_{total} = \epsilon_U + \epsilon_\lambda + \epsilon_{QPE} \tag{22}$$

The total error depends on the number of time steps N, the number of qubits t, the range of matrix eigenvalues, and the specific hardware noise. If the number of qubits t used is small or the eigenvalue λ_{max} is large, a finer time step or more QPE qubits are needed to reduce the error.

5.2. Dataset and experimental results

In this study, the classification performance of QSVM was compared with SVM in a multidimensional nonlinear feature space. The paper used the Iris dataset, which contains 150 samples covering three species of iris (Iris mountain, Iris variatum, and Iris Virginia) with 50 samples per species[10]. Three dimensional, four dimensional, five dimensional and six dimensional feature subsets are extracted for testing. The three dimensional features include calyx length, calyx width and petal length, and the other dimensions add petal width features to evaluate the influence of different dimensions on the classification effect. Due to the bottleneck of quantum computing hardware technology, the linear solution has stability problems, so the ratio of calculation accuracy to calculation time is used as the index for performance evaluation. Using IBM qiskit development platform[11]. The model performance is shown in Figure 4:

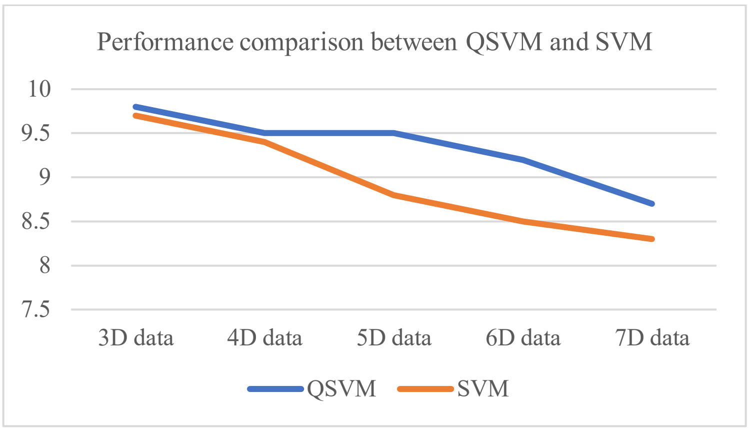

Figure 4: Performance data

The experimental results show that the classification accuracy of QSVM is generally higher than that of classical SVM on these data sets. In particular, when handling high-dimensional and nonlinear features, QSVM demonstrates superior classification capabilities. In terms of computation time and memory requirements, QSVM shows significant advantages, especially when dealing with large-scale datasets.

6. Conclusion

In this paper, the acceleration method of quantum computing for SVM in multi-dimensional nonlinear feature space is studied, and a new algorithm combining the advantages of classical and quantum computing is proposed. Experimental results show that QSVM outperforms classical SVM in terms of classification performance, computation time and memory requirements. However, the practical application of quantum computing still faces numerous challenges, including the maturity of quantum hardware and the correction of quantum errors. Future research can further explore more efficient quantum mapping algorithms and quantum circuit optimization methods to improve the performance and practicability of QSVM.

References

[1] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In COLT '92: Proceedings of the fifth annual workshop on Computational learning theory.

[2] Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum Support Vector Machine for Big Data Classification. Physical Review Letters, 113(13), 130503.

[3] Aram, W. H., Avinatan, H., Seth, L. (2008). Quantum algorithm for solving linear systems of equations. Quantum Physics.

[4] Wang, Y., Li, G., & Wang, X. (2023). A hybrid quantum-classical Hamiltonian learning algorithm. Science China (Information Sciences), 66(2), 295-296.

[5] Yue, T., Wu, C., Liu, Y., Du, Z., Zhao, N., Jiao, Y., Xu, Z., & Shi, W. (2023). HASM quantum machine learning. Science China (Earth Sciences), 66(9), 1937-1945.

[6] Wang, S., Zhang, K., & Zhang, L. (2020). Research on the decomposition of unitary operators in quantum circuits. Natural Science Journal of Heilongjiang University, 37(6), 653-660.

[7] Papadopoulos, N. J. C., Reilly, J. T., Wilson, J. D., & Holland, M. J. (2024). Reductive quantum phase estimation. Physical Review Research, 6(3), 033051.

[8] Wong, H. Y. (2023). Quantum Fourier Transform I. In Introduction to Quantum Computing (pp. 243-253).

[9] Pedrycz, W. (2002). Advances in Kernel Methods. Support Vector Learning, by B. Scholkopf, C. J. C. Burges, and A. J. Smola. Neurocomputing, 47(1), 303-304. ISBN 0-262-19416-3.

[10] Dua, D. and Graff, C. (2019). Iris Data Set. UCI Machine Learning Repository. Available at: https://archive.ics.uci.edu/ml/datasets/iris

[11] Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C. J., Lishman, J., Gacon, J., Martiel, S., Nation, P. D., Bishop, L. S., Cross, A. W., Johnson, B. R., & Gambetta, J. M. (2024). Quantum computing with Qiskit.