

Quantum algorithm for marginal fisher analysis

Jing Li(李静), Yanqi Song(宋燕琪), Sujuan Qin(秦素娟), Wenmin Li(李文敏), and Fei Gao(高飞)
Citation: Chin. Phys. B, 2025, 34 (**12**): 120302. DOI: 10.1088/1674-1056/addeb8
http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Domain adaptation method inspired by quantum convolutional neural network

Chunhui Wu(武春辉), Junhao Pei(裴骏豪), Yihua Wu(吴逸华), Anqi Zhang(张安琪), and Shengmei Zhao(赵生妹)
Chin. Phys. B, 2025, 34 (**7**): 070302. DOI: 10.1088/1674-1056/adc7ed

Variational data encoding and correlations in quantum-enhanced machine learning

Ming-Hao Wang(王明浩) and Hua Lü(吕桦)
Chin. Phys. B, 2024, 33 (**9**): 090307. DOI: 10.1088/1674-1056/ad5c3b

Analysis of learnability of a novel hybrid quantum—classical convolutional neural network in image classification

Tao Cheng(程涛), Run-Sheng Zhao(赵润盛), Shuang Wang(王爽), Rui Wang(王睿), and Hong-Yang Ma(马鸿洋)
Chin. Phys. B, 2024, 33 (**4**): 040303. DOI: 10.1088/1674-1056/ad1926

Variational quantum semi-supervised classifier based on label propagation

Yan-Yan Hou(侯艳艳), Jian Li(李剑), Xiu-Bo Chen(陈秀波), and Chong-Qiang Ye(叶崇强)
Chin. Phys. B, 2023, 32 (**7**): 070309. DOI: 10.1088/1674-1056/acb9fb

A new method of constructing adversarial examples for quantum variational circuits

Jinge Yan(颜金歌), Lili Yan(闫丽丽), and Shibin Zhang(张仕斌)
Chin. Phys. B, 2023, 32 (**7**): 070304. DOI: 10.1088/1674-1056/ac9b32

Quantum algorithm for marginal Fisher analysis

Jing Li(李静)[1,2,3], Yanqi Song(宋燕琪)[4], Sujuan Qin(秦素娟)[1], Wenmin Li(李文敏)[1], and Fei Gao(高飞)[1,†]

[1] *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*
[2] *National Engineering Research Center of Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*
[3] *School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*
[4] *China Academy of Information and Communications Technology, Beijing* 100191, *China*

(Received 11 April 2025; revised manuscript received 21 May 2025; accepted manuscript online 30 May 2025)

Marginal Fisher analysis (MFA) stands out as a prominent dimensionality reduction algorithm, striving to minimize within-class scatter while maximizing the separability between marginal data points. However, MFA and its variants require substantial computational resources when dealing with large-scale data. To address this, we propose quantum algorithms for MFA (called QMFA). QMFA is composed of two core processes: the first is the efficient construction of the weight matrices for the intrinsic and penalty graphs, and the second is solving the generalized eigenvalue problem (GEP) using the block-encoding technique. Compared to classical MFA, the proposed QMFA achieves a polynomial acceleration in the number of samples and exponential acceleration in the dimensionality. Additionally, we investigate quantum algorithms for different variants of MFA. Specifically, for enhanced MFA and multiple MFA, we address the construction of the related weight matrix, which differs from that in standard MFA. For kernel MFA, we solve the GEP associated with the corresponding kernel matrix. The proposed quantum algorithms achieve a speedup equivalent to that of QMFA.

Keywords: quantum machine learning, block-encoding, dimensionality reduction, marginal Fisher analysis, graph construction

PACS: 03.67.Ac, 03.67.Lx

DOI: 10.1088/1674-1056/addeb8 **CSTR:** 32038.14.CPB.addeb8

1. Introduction

Over the past decades, machine learning has played a crucial role in various fields.[1] However, with the exponential growth of data, the development of machine learning has been constrained by limited computational resources. It is essential to explore more efficient computational approaches. Quantum computation stands out as one of the most promising computing paradigms. It has demonstrated the ability to accelerate solving for certain problems that challenge classical computing.[2–4] Currently, combining quantum computation with machine learning, quantum machine learning (QML) has achieved significant advancements in supervised learning,[5–7] unsupervised learning,[8–13] and reinforcement learning.[14–16]

Dimensionality reduction (DR) is a powerful tool in machine learning. The goal of DR is to reduce the number of variables in a dataset through a projection while preserving the intrinsic structure.[17–19] Representatively, linear discriminant analysis (LDA)[20] aims to find a linear projection by minimizing the ratio of within-class scatter to between-class scatter. Locality preserving projection (LPP)[21] is developed to map the data while preserving its manifold structure. However, LDA was proposed under a strong assumption that the data of each class is approximately Gaussian distributed. In contrast, LPP disregards the class information, resulting in diminished performance for classification tasks. To overcome the limitations of LDA and LPP, marginal Fisher analysis (MFA)[22,23] was proposed.

MFA integrates the class information with the manifold structure of the data. In MFA, the within-class scatter is characterized by an intrinsic graph where each sample is connected to its nearest neighbors of the same class, and the between-class scatter is characterized by a penalty graph where the marginal point pairs of different classes are connected. Then, the linear projection is obtained by solving the generalized eigenvalue problem (GEP) of $X^{\mathrm{T}}LX$ and $X^{\mathrm{T}}L^{\mathrm{P}}X+\mu I$. Here, $X\in\mathbb{R}^{N\times M}$ denotes the data matrix, μ is a regularization parameter used to prevent trivial solutions, and L (L^{P}) denotes the Laplacian matrix of the intrinsic (penalty) graph which is derived from the weight matrix. Furthermore, several variants have been proposed to improve the performance of MFA. Prominently, enhanced MFA (EMFA)[24] and multiple MFA (MMFA)[25] were proposed in accordance with different rules for constructing the penalty graph. In addition, kernel MFA (KMFA)[26,27] extends MFA from linear projections to nonlinear cases by using kernel trick. However, the time complexities of MFA and its variants are $\mathcal{O}(N^2M+M^3)$. When dealing with large-scale data, they are extremely time-consuming. Although quite a few quantum algorithms[28–32] for DR have

†Corresponding author. E-mail: gaof@bupt.edu.cn

© 2025 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

been proposed, there is no quantum algorithm that efficiently realizes MFA and its variants.

In this paper, we propose a quantum algorithm for MFA, called QMFA. To accelerate the two time-consuming stages of MFA, QMFA consists of two core processes. The first is the efficient construction of the weight matrices of the intrinsic and penalty graphs by employing quantum distance estimation[33,34] and quantum minimum search algorithm.[35,36] The second is solving the GEP using the block-encoding technique.[37,38] Specifically, we obtain the block-encodings of weight matrices and further construct the efficient implementations of $X^{\mathrm{T}}LX$ and $X^{\mathrm{T}}L^{\mathrm{P}}X+\mu I$. Then, using the quantum approach for solving the GEP based on block-encoding, the projection vectors are obtained and utilized for the final data compression. As a result, the complexity of QMFA is $\mathcal{O}(N\sqrt{N}\,\mathrm{poly}\log(NM)/\varepsilon))$ for a given error ε. In addition, we present quantum algorithms for the variants of MFA. Specifically, for EMFA and MMFA, we address the construction of the related weight matrix, which differs from that in standard MFA. For KMFA, we obtain the block-encoding of the corresponding kernel matrix which is utilized in the subsequent process of solving the GEP. All of the proposed quantum algorithms achieve polynomial acceleration in the number of samples N and exponential acceleration in the dimensionality M compared to their classical counterparts.

The rest of this paper is organized as follows. In Section 2, we review the classical MFA and its variants in Subsection 2.1, and introduce the definitions and lemmas related to block-encoding in Subsection 2.2. Section 3 presents quantum algorithms for MFA and its variants in detail. In Section 4, we analyze the time complexity of each quantum algorithm. Finally, conclusions are given in Section 5.

2. Preliminaries

2.1. Classical marginal Fisher analysis and its variants

Given a dataset $\{\boldsymbol{x}_i\}_{i=1}^N\,(\boldsymbol{x}_i\in\mathbb{R}^M)$ with the classes $\mathcal{C}(\boldsymbol{x}_i)\in\{c_1,\ldots,c_{n_{\mathrm{c}}}\}$, MFA extracts low-dimensional features $\{\boldsymbol{y}_i\}_{i=1}^N\,(\boldsymbol{y}_i\in\mathbb{R}^m)$, with the goal of minimizing the within-class compactness S_{w} while maximizing the between-class separability S_{b}. S_{w} is defined as the sum of distances between each sample and its neighbors within the same class. S_{b} is characterized by the sum of distances between margin points and their neighboring points from different classes. Considering the linear projection that $\boldsymbol{y}_i=U^{\mathrm{T}}\boldsymbol{x}_i$, S_{w} and S_{b} are expressed by an intrinsic graph and a penalty graph as follows:

$$\begin{aligned}S_{\mathrm{w}}&=\sum_i\sum_{i\in\mathcal{N}_{k_1}(j)\text{ or }j\in\mathcal{N}_{k_1}(i)}\left\|U^{\mathrm{T}}\boldsymbol{x}_i-U^{\mathrm{T}}\boldsymbol{x}_j\right\|^2\\&=2\mathrm{Tr}[U^{\mathrm{T}}X^{\mathrm{T}}LXU],\end{aligned}\tag{1}$$

$$\begin{aligned}S_{\mathrm{b}}&=\sum_i\sum_{(i,j)\in\mathcal{P}_{k_2}(\mathcal{C}(\boldsymbol{x}_i))\text{ or }(i,j)\in\mathcal{P}_{k_2}(\mathcal{C}(\boldsymbol{x}_j))}\left\|U^{\mathrm{T}}\boldsymbol{x}_i-U^{\mathrm{T}}\boldsymbol{x}_j\right\|^2\\&=2\mathrm{Tr}[U^{\mathrm{T}}X^{\mathrm{T}}L^{\mathrm{P}}XU],\end{aligned}\tag{2}$$

where $X=(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N)^{\mathrm{T}}$. $\mathcal{N}_{k_1}(i)$ represents the index set of the k_1 nearest neighbors of sample $\boldsymbol{x}_i$ that are in the same class and $\mathcal{P}_{k_2}(c)$ is a set of k_2 data pairs that are the nearest pairs among the set $\{(i,j)\,|\mathcal{C}(\boldsymbol{x}_i)=c,\mathcal{C}(\boldsymbol{x}_j)\neq c\}$. $L=D-W$ and $L^{\mathrm{P}}=D^{\mathrm{P}}-W^{\mathrm{P}}$ are the Laplacian matrices of the intrinsic and penalty graphs, respectively. Here, to obtain L and L^{P}, it is necessary first to construct the weight matrices W and W^{P} which take the form of

$$[W]_{ij}=\begin{cases}1, & \text{if } i\in\mathcal{N}_{k_1}(j)\text{ or }j\in\mathcal{N}_{k_1}(i),\\0, & \text{else,}\end{cases}\tag{3}$$

$$[W^{\mathrm{P}}]_{ij}=\begin{cases}1, & \text{if } (i,j)\in\mathcal{P}_{k_2}(\mathcal{C}(\boldsymbol{x}_i))\text{ or}\\ & \quad(i,j)\in\mathcal{P}_{k_2}(\mathcal{C}(\boldsymbol{x}_j)),\\0, & \text{else.}\end{cases}\tag{4}$$

Here D and D^{P} are degree matrices where the diagonal elements are $D_{ii}=\sum_{j=1}^N W_{ij}$ and $D_{ii}^{\mathrm{P}}=\sum_{j=1}^N W_{ij}^{\mathrm{P}}$ with all off-diagonal elements being equal to 0. An example is given in Fig. 1 to show the construction of the intrinsic graph and the penalty graph in MFA with 3 classes.

The objective of MFA is

$$\arg\min_U\frac{\mathrm{Tr}[U^{\mathrm{T}}X^{\mathrm{T}}LXU]}{\mathrm{Tr}[U^{\mathrm{T}}X^{\mathrm{T}}L^{\mathrm{P}}XU]}.\tag{5}$$

An approximation to the projection matrix U can be computed by solving the GEP

$$\left(X^{\mathrm{T}}LX\right)\boldsymbol{u}=\lambda\left(X^{\mathrm{T}}L^{\mathrm{P}}X+\mu I\right)\boldsymbol{u},\tag{6}$$

where $\mu\in(0,1)$ denotes the regularization parameter. Here, the regularization parameter ensures the matrix is positive definite, thereby preventing trivial solutions. Then, generalized eigenvectors $\{\boldsymbol{u}_i\}_{i=1}^m$ corresponding to m smallest generalized eigenvalues form the projection matrix U. In particular, the algorithm of MFA is shown in Algorithm 1.

Algorithm 1 Classical marginal Fisher analysis

Input: Dataset $\{\boldsymbol{x}_i\}_{i=1}^N(\boldsymbol{x}_i\in\mathbb{R}^M)$,
label information $\mathcal{C}(\boldsymbol{x}_i)\in\{c_1,\ldots,c_{n_{\mathrm{c}}}\}$, k_1, k_2, and m.

Output: $\{\boldsymbol{y}_i\}_{i=1}^N(\boldsymbol{y}_i\in\mathbb{R}^m)$. 1. Search for $\{\mathcal{N}_{k_1}(i)\}_{i=1}^N$, construct weight matrices W according to Eq. (3), and obtain graph Laplace matrices L;
2. Search for $\{\mathcal{P}_{k_2}(c_p)\}_{p=1}^{n_{\mathrm{c}}}$, construct weight matrices W^{P} according to Eq. (4), and obtain graph Laplace matrices L^{P};
3. Solve the generalized eigenvalue problem as shown in Eq. (6);
4. Compute the low-dimensional embedding for each sample point, $\boldsymbol{y}_i=U^{\mathrm{T}}\boldsymbol{x}_i$;
5. **Return**: $\{\boldsymbol{y}_i\}_{i=1}^N$.

EMFA[24] and MMFA[25] improve the MFA by employing different penalty graph constructions. Specifically, EMFA searches for margin points in the closest classes to simplify the construction of the penalty graph, and the set of k_2 nearest data pairs is $\mathcal{P}_{k_2}^{\mathrm{E}}(c)$ in $\{(i,j)|\mathcal{C}(\boldsymbol{x}_i)=c,\mathcal{C}(\boldsymbol{x}_j)\in\mathcal{N}_c\}$ where $\mathcal{N}_c$ denotes the set of nearest classes of c. MMFA is developed to address the class-isolation issue, where some classes are isolated from others leading to the neglect of separability between the isolated classes and the remaining classes. MMFA identifies the margin points between each pair of classes to ensure the margin points over all classes, and $\mathcal{P}_{k_2}^{\mathrm{M}}(c_p)=\bigcup_{q\neq p}\mathcal{S}_{k_2}^{\mathrm{M}}(c_p,c_q)$ where $\mathcal{S}_{k_2}^{\mathrm{M}}(c_p,c_q)$ is the set of k_2 nearest data pairs in $\{(i,j)|\mathcal{C}(\boldsymbol{x}_i)=c_p,\mathcal{C}(\boldsymbol{x}_j)=c_q\}$. The illustration of the constructions of penalty graphs in EMFA and MMFA are shown in Fig. 2.

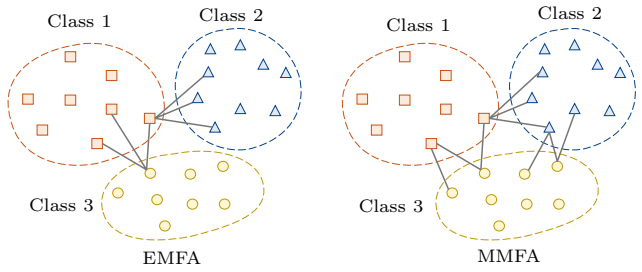


Fig. 1. An example of the intrinsic and penalty graphs in MFA with 3 classes where $k_1=3$ and $k_2=3$.

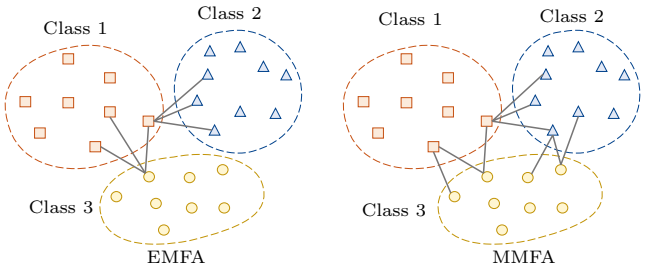


Fig. 2. An example of penalty graphs in EMFA and MMFA with 3 classes where $k_1=3$ and $k_2=3$. Especially, we suppose that $\mathcal{N}_{c_1}=\{c_2\}$, $\mathcal{N}_{c_2}=\{c_1\}$ and $\mathcal{N}_{c_3}=\{c_1\}$ in EMFA.

The KMFA[26,27] uses the kernel method to handle the nonlinear manifolds. A kernel is defined as a function $\kappa(\boldsymbol{x},\boldsymbol{x}')$ in the original input space $\mathcal{X}$, which is associated with an inner product implicitly in a higher-dimensional feature space $\mathcal{F}$, that is

$$\kappa(\boldsymbol{x},\boldsymbol{x}')=\langle\boldsymbol{\Phi}(\boldsymbol{x}),\boldsymbol{\Phi}(\boldsymbol{x}')\rangle, \tag{7}$$

where the feature map $\boldsymbol{\Phi}:\mathcal{X}\to\mathcal{F}$. Given the input $\{\boldsymbol{x}_i\}_{i=1}^N\in\mathcal{X}$, entries of the kernel matrix are $K_{ij}=\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)$. By introducing the kernel method, the problem that KMFA solves is formed as

$$\left(K^{\mathrm{T}}LK\right)\boldsymbol{u}'=\lambda\left(K^{\mathrm{T}}L^{\mathrm{P}}K+\mu I\right)\boldsymbol{u}', \tag{8}$$

where K denotes the kernel matrix. The projection direction is $\boldsymbol{u}_j=\sum_{i=1}^N u'_j\boldsymbol{\Phi}(\boldsymbol{x}_i)$. For a new data point $\boldsymbol{x}$, its projection to the derived optimal direction is obtained as $y_j=\sum_{i=1}^N u'_{ji}\kappa(\boldsymbol{x},\boldsymbol{x}_i)$.

2.2. Block-encoding

Block-encoding constitutes a fundamental framework in quantum algorithms for implementing advanced matrix operations.[39,40] In this subsection, we review its definitions and related lemmas.

Definition 1 (block-encoding[38,41]) Suppose that A is a s-qubit operator, $\alpha,\varepsilon\in\mathbb{R}_+$ and $a\in\mathbb{N}$, then the $(s+a)$-qubit unitary U is a (α,a,ε)-block-encoding of A, if

$$\left\|A-\alpha\left(\langle 0|^{\otimes a}\otimes I\right)U\left(|0\rangle^{\otimes a}\otimes I\right)\right\|\leq\varepsilon. \tag{9}$$

Definition 2 (state preparation pair[41]) Let $\boldsymbol{y}\in\mathbb{C}^n$ and $\|\boldsymbol{y}\|_1\leq\beta$. The pair of unitaries $(P_{\mathrm{L}},P_{\mathrm{R}})$ is called a (β,b,ε_y)-state-preparation-pair if $P_{\mathrm{L}}|0\rangle^{\otimes b}=\sum_{j=1}^n c_j|j\rangle$, $P_{\mathrm{R}}|0\rangle^{\otimes b}=\sum_{j=1}^{2^b}d_j|j\rangle$, such that $\sum_{j=1}^n\left|\beta\left(c_j^*d_j\right)-y_j\right|\leq\varepsilon_y$, and for all $j\in\{m+1,\ldots,2^b\}$, we have $c_j^*d_j=0$.

Lemma 1 (linear combination of block-encoded matrices[41]) Let $A=\sum_{j=1}^m y_jA_j$ be an s-qubits operator and $\varepsilon_A\in\mathbb{R}_+$. Assume that $(P_{\mathrm{L}},P_{\mathrm{R}})$ is a (β,b,ε_y)-state-preparation-pair for $\boldsymbol{y}\in\mathbb{C}^m$ and U_j is an (α,a,ε_A)-block-encoding of A_j, then

$$\begin{aligned}\mathcal{U}=&\left(P_{\mathrm{L}}^{\dagger}\otimes I_a\otimes I_s\right)\left(\sum_{j=1}^m|j\rangle\langle j|\otimes U_j\right.\\&\left.+\left(\left(I-\sum_{j=1}^m|j\rangle\langle j|\right)\otimes I_a\otimes I_s\right)\right)\left(P_{\mathrm{R}}\otimes I_a\otimes I_s\right),\end{aligned} \tag{10}$$

is an $(\alpha\beta,a+b,\alpha\varepsilon_y+\beta\varepsilon_A)$-block-encoding of A.

Here, Lemma 1 holds under the assumption that the scaling factors α of block-encodings of A_j are equal. Considering the case of inconsistent scaling factors that U_j is an $(\alpha_j,a_j,\varepsilon_j)$-block-encoding of A_j, we generalize Lemma 1 to the following Lemma 2, and the proof is shown in Appendix A.

Lemma 2 (generalized linear combination of block-encoded matrices) Let $A=\sum_{j=1}^m y_jA_j$ be an s-qubits operator and $\varepsilon_A\in\mathbb{R}_+$. Assume that $(P_{\mathrm{L}},P_{\mathrm{R}})$ is a (β,b,ε_y)-state-preparation-pair for $\boldsymbol{y}\in\mathbb{C}^m$ and U_j is an $(\alpha_j,a_j,\varepsilon_j)$-block-encoding of A_j, then

$$\begin{aligned}\tilde{\mathcal{U}}=&\left(P_{\mathrm{L}}^{\dagger}\otimes I_a\otimes I_s\right)\left(\sum_{j=1}^m|j\rangle\langle j|\otimes I_{a-a_j}\otimes U_j\right.\\&\left.+\left(I-\sum_{j=1}^m|j\rangle\langle j|\right)\otimes I_a\otimes I_s\right)\left(P_{\mathrm{R}}\otimes I_a\otimes I_s\right),\end{aligned} \tag{11}$$

is an $\left(\alpha\beta,a+b,\alpha\varepsilon_y+\sum_j y_j\varepsilon_j\right)$-block-encoding of A, where $\alpha=\max_j\{\alpha_j\}$ and $a=\max_j\{a_j\}$.

Lemma 3 (product of block-encoded matrices[41]) If U is an $(\alpha;a;\delta)$-block-encoding of a s-qubit operator A that can be implemented in time T_1, and V is a $(\beta;b;\varepsilon)$-block-encoding of an s-qubit operator B that can be implemented in time T_2, then $(I_b\otimes U)(I_a\otimes V)$ is an $(\alpha\beta;a+b;\alpha\varepsilon+\beta\delta)$-block-encoding of AB that can be implemented in time $\mathcal{O}(T_1+T_2)$.

Definition 3 (quantum access to data[8,42]) We say that we have quantum access to a matrix $A \in \mathbb{R}^{N\times M}$ if there exists a procedure to perform the following mapping, for $i \in [N]$, in time T_A:

$$O_A : |i\rangle|0\rangle \mapsto |i\rangle|\|A_{i\cdot}\|\rangle, \tag{12}$$

$$\tilde{O}_A : |i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle|A_{ij}\rangle, \tag{13}$$

$$U_A : |i\rangle|0\rangle \mapsto |i\rangle|A_{i\cdot}\rangle = |i\rangle\frac{1}{\|A_{i\cdot}\|}\sum_{j=1}^{M} A_{ij}|j\rangle, \tag{14}$$

$$V_A : |0\rangle|j\rangle \mapsto \frac{1}{\|A\|_{\mathrm{F}}}\sum_{i=1}^{N}\|A_{i\cdot}\||i\rangle|j\rangle, \tag{15}$$

where $A_{i\cdot}$ denotes the i-th row of A and $\|A\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}A_{ij}^2}$ is the Frobenius norm of A. If the data matrix A is stored in a quantum random access memory (QRAM) data structure, the time to access A is $T_A = \mathcal{O}(\mathrm{poly}\log(NM/\varepsilon))$ for $\varepsilon \in (0,1)$.

Lemma 4 (implementing block-encodings from quantum data structures[41,43,44]) If the data matrix $A \in \mathbb{R}^{N\times M}$ is stored in a QRAM data structure, then there exist two unitary operators U_A and V_A that can be implemented in time $\mathcal{O}(\mathrm{poly}\log(NM/\varepsilon))$ such that $U_A^{\dagger}V_A$ is an $(\|A\|_{\mathrm{F}}, \mathrm{poly}\log(N), \varepsilon)$-block-encoding of A.

Theorem 1 (solving the generalized eigenvalue problem (GEP) of $(A;B)$ based on block-encoding[45]) Assume that A, B are $N\times N$ Hermitian matrices with B being positive-definite. Let U_A be an $(\alpha_A; q_A; \varepsilon_A)$-block-encoding of A that is implemented in time T_A, and U_B an $(\alpha_B; q_B; \varepsilon_B)$-block-encoding of B that is implemented in time T_B. Denote κ_B as the condition number of B, the generalized eigen-pairs of $A\boldsymbol{u} = \lambda B\boldsymbol{u}$ as $\{(\lambda_j, |E_j\rangle)\,|\,j = 1,\ldots,N\}$. Given access to copies of the state $\sum_j \beta_j|E_j\rangle$ that is prepared in time $\mathcal{O}(T_{\mathrm{in}})$, there is a quantum algorithm that returns a state proportional to $\sum_j \beta_j|\tilde{\lambda}_j\rangle|E_j\rangle$ in time

$$\mathcal{O}\left(\kappa_B^{0.5}T_{\mathrm{in}} + \frac{\alpha_A\kappa_B^{1.5}}{\|B\|\varepsilon}\left(T_A + \frac{\alpha_B\kappa_B}{\|B\|}\right)\right), \tag{16}$$

where $|\tilde{\lambda}_j - \lambda_j| \le \varepsilon$, κ_B denotes the condition number of B, and $\|B\|$ denotes the operator norm equal to the maximal singular value of B.

3. Quantum algorithm

3.1. Quantum marginal Fisher analysis

Focusing on accelerating the graph construction and solving the GEP, we present QMFA and its basic flow is shown in Fig. 3, detailing the following steps.

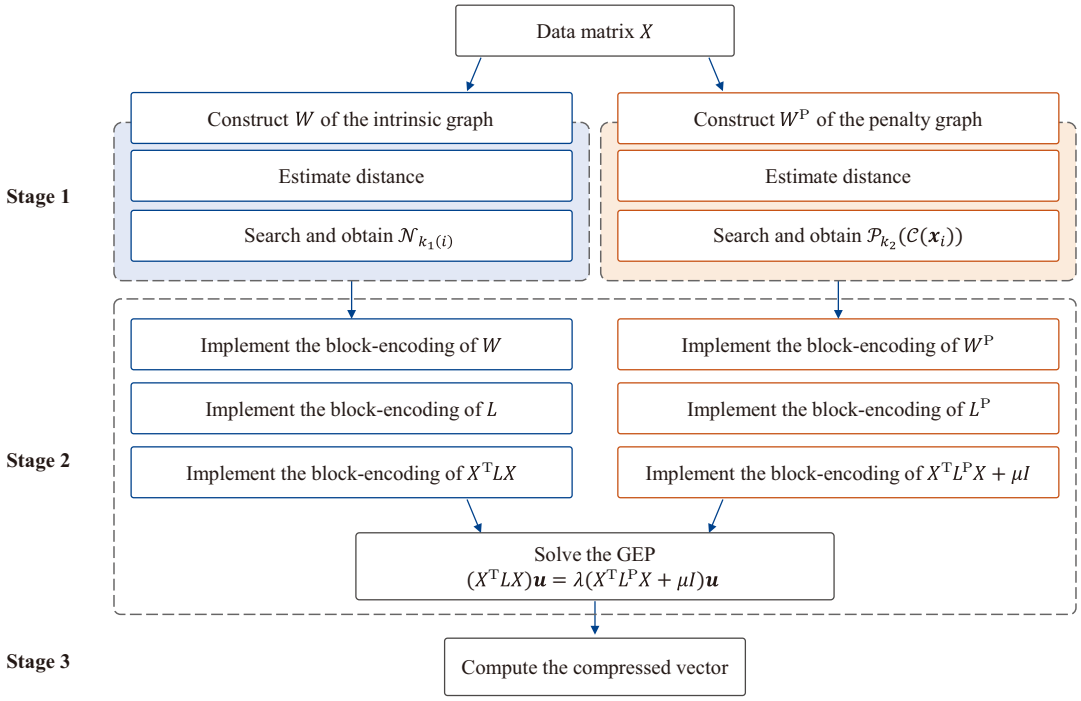


Fig. 3. Schematic workflow of the QMFA.

Stage 1: Constructing the weight matrices

The construction of the weight matrix involves two essential steps: calculating distances and finding nearest neighbors. In this stage, we utilize the next Theorem 2 to estimate the distances between samples and employ the quantum minimum search algorithm[35,36] to accelerate the search process.

Theorem 2 (quantum distance estimation[33,34]). Given quantum access in time T to two matrices U and V with rows $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ of dimension M, there is a quantum algorithm that, for any pair (i,j), performs the following map-

ping: $|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle|\overline{d^2(\boldsymbol{u}_i,\boldsymbol{v}_j)}\rangle$, estimating the Euclidean distance between $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ with precision $|d^2(\boldsymbol{u}_i,\boldsymbol{v}_j) - \overline{d^2(\boldsymbol{u}_i,\boldsymbol{v}_j)}| \leq \varepsilon$ for any $\varepsilon > 0$. The algorithm has a running time given by $\widetilde{O}(T\eta/\varepsilon)$, where $\eta = \max_{i,j}(\|\boldsymbol{u}_i\|\|\boldsymbol{v}_j\|)$, assuming that $\min_i(\|\boldsymbol{u}_i\|) = \min_i(\|\boldsymbol{v}_i\|) = 1$.

(i) Constructing the weight matrix W of the intrinsic graph

Assuming that we have quantum access to X which is stored in a QRAM data structure, we can access the vectors in time $\mathcal{O}(\mathrm{poly}\log(NM/\varepsilon_x))$. For each point $\boldsymbol{x}_i$, we prepare the quantum state $|\phi_i^0\rangle$ initially, where

$$|\phi_i^0\rangle = |i\rangle \frac{1}{\sqrt{|\mathcal{C}(x_i)|}} \sum_{j\in\mathcal{C}(x_i)} |j\rangle|0\rangle, \tag{17}$$

and $\mathcal{C}(x_i)$ denotes the set of indices of samples in the same class with $\boldsymbol{x}_i$ with the scale of $|\mathcal{C}(x_i)|$. Using Theorem 2, we obtain

$$|\phi_i^1\rangle = |i\rangle \frac{1}{\sqrt{|\mathcal{C}(x_i)|}} \sum_{j\in\mathcal{C}(x_i)} |j\rangle|\overline{d^2(\boldsymbol{x}_i,\boldsymbol{x}_j)}\rangle. \tag{18}$$

Then, with the utilization of the quantum minimum search algorithm[35,36] to search the k_1 shortest distances, $\mathcal{N}_{k_1}(i)$ is obtained in time $T_1 = \mathcal{O}(\sqrt{k_1|\mathcal{C}(x_i)|})$. Iterate N rounds until acquiring $\{\mathcal{N}_{k_1}(i)\}_{i=1}^N$, after which the weight matrix W is constructed according to Eq. (3).

(ii) Constructing the weight matrix W^{P} of the penalty graph

For each class c_p in $\{c_1,\ldots,c_{n_c}\}$, we prepare the initial quantum state

$$|\phi_p^0\rangle = \frac{1}{\sqrt{|c_p|}} \sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p} |i\rangle \frac{1}{\sqrt{N-|c_p|}} \sum_{\mathcal{C}(\boldsymbol{x}_j)\neq c_p} |j\rangle, \tag{19}$$

where $|c_p|$ denotes the number of samples of class c_p. Using Theorem 2, we obtain

$$|\phi_p^1\rangle = \frac{1}{\sqrt{|c_p|}} \sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p} |i\rangle \otimes \frac{1}{\sqrt{N-|c_p|}} \sum_{\mathcal{C}(\boldsymbol{x}_j)\neq c_p} |j\rangle|\overline{d^2(\boldsymbol{x}_i,\boldsymbol{x}_j)}\rangle. \tag{20}$$

Then, with the utilization of the quantum minimum search algorithm[35,36] to search the k_2 shortest distances, $\mathcal{P}_{k_2}(c_p)$ is obtained in time $T_2 = \mathcal{O}(\sqrt{k_2|c_p||N-c_p|})$. Similar to the construction of W of the intrinsic graph, after obtaining all $\{\mathcal{P}_{k_2}(c_p)\}_{p=1}^{n_c}$, we construct W^{P} according to Eq. (4).

Stage 2: Solving the GEP of $(X^{\mathrm{T}}LX; X^{\mathrm{T}}L^{\mathrm{P}}X + \mu I)$

Using Theorem 1, the GEP can be solved once the block-encodings of $X^{\mathrm{T}}LX$ and $X^{\mathrm{T}}L^{\mathrm{P}}X + \mu I$ can be efficiently implemented. Next, we introduce the construction of the required block-encodings in detail.

(i) Implementing the block-encodings of L and L^{P}

Assume that we have quantum access to W and W^{P} obtained in Stage 1. Using Lemma 4, we can implement a $(\|W\|_{\mathrm{F}}, \mathrm{poly}\log(N), \varepsilon_w)$-block-encoding of W in time $\mathcal{O}(\mathrm{poly}\log(N/\varepsilon_w))$. Then, the block-encodings of D and L can be implemented efficiently using the following Lemma 5, which is proven in Appendix B.

Lemma 5 (implementing block-encoding of D and L) Assume we have a weight matrix W with entries in $\{0,1\}$ stored in a QRAM data structure, then there exists a $(\max_i(\|W_{i\cdot}\|^2), \mathrm{poly}\log(N), \varepsilon_w)$-block-encoding of the degree matrix D and a $(2\|W\|_{\mathrm{F}}, \mathrm{poly}\log(N), \varepsilon_w)$-block-encoding of L, both of which can be implemented in time $\mathcal{O}(\mathrm{poly}\log(N/\varepsilon_w))$.

Similarly, using Lemma 5, a $(2\|W^{\mathrm{P}}\|_{\mathrm{F}}, \mathrm{poly}\log(N), \varepsilon_w)$-block-encoding of L^{P} can be implemented in time $\mathcal{O}(\mathrm{poly}\log(N/\varepsilon_w))$.

(ii) Implementing the block-encoding of $X^{\mathrm{T}}LX$ and $X^{\mathrm{T}}L^{\mathrm{P}}X + \mu I$

By leveraging Lemmas 2, 3, and 4, we derive the following Lemma 6, which enables the construction of a block-encoding for the matrix $X^{\mathrm{T}}ZX$, where Z is a given matrix.

Lemma 6 (implementing block-encoding of $X^{\mathrm{T}}ZX$) Given the data matrix $X \in \mathbb{R}^{N\times M}$ stored in QRAM and a matrix $Z \in \mathbb{R}^{N\times N}$ with a $(\alpha_z; q_z; \varepsilon_z)$-block-encoding U_Z that can be implemented in time T_Z, we can implement a $\left(\|X\|_{\mathrm{F}}^2\alpha_z, 2\mathrm{poly}\log(N) + q_z, 2\|X\|_{\mathrm{F}}\alpha_z\varepsilon_x + \|X\|_{\mathrm{F}}^2\varepsilon_z\right)$-block-encoding of $X^{\mathrm{T}}ZX$ in time $\mathcal{O}(\mathrm{poly}\log(NM/\varepsilon_x) + T_Z)$.

Furthermore, given that the identity operator I serves as a $(1,0,0)$-block-encoding of itself, it is straightforward to derive a $\left(\|X\|_{\mathrm{F}}^2\alpha_z + \mu, 2\mathrm{poly}\log(N) + q_z, 2\|X\|_{\mathrm{F}}\alpha_z\varepsilon_x + \|X\|_{\mathrm{F}}^2\varepsilon_z\right)$-block-encoding of $X^{\mathrm{T}}ZX + \mu I$. This construction can be executed in time $\mathcal{O}(\mathrm{poly}\log(NM/\varepsilon_x) + T_Z)$ according to Lemma 2.

In this step, using Lemma 6, we leverage the obtained block-encodings of L and L^{P} to implement a $(2\|X\|_{\mathrm{F}}^2\|W\|_{\mathrm{F}}, 3\mathrm{poly}\log(N), 2\|X\|_{\mathrm{F}}\|W\|_{\mathrm{F}}\varepsilon_x + \|X\|_{\mathrm{F}}^2\varepsilon_w)$-block-encoding of $X^{\mathrm{T}}LX$ and a $(2\|X\|_{\mathrm{F}}^2\|W^{\mathrm{P}}\|_{\mathrm{F}} + \mu, 3\mathrm{poly}\log(N), 2\|X\|_{\mathrm{F}}\|W^{\mathrm{P}}\|_{\mathrm{F}}\varepsilon_x + \|X\|_{\mathrm{F}}^2\varepsilon_w)$-block-encoding of $X^{\mathrm{T}}L^{\mathrm{P}}X + \mu I$ in time $\mathcal{O}(\mathrm{poly}\log(NM/(\varepsilon_x\varepsilon_w)))$.

(iii) Solving the GEP of $(X^{\mathrm{T}}LX; X^{\mathrm{T}}L^{\mathrm{P}}X + \mu I)$

By applying Theorem 1 with the initialized quantum state $\frac{1}{\sqrt{M}}\sum_{i=1}^M |i\rangle$ that can be decomposed into the eigenbasis $\{|\boldsymbol{u}_1\rangle, |\boldsymbol{u}_2\rangle, \ldots\}$, we obtain the state $\sum_i \beta_i|\bar{\lambda}_i\rangle|\boldsymbol{u}_i\rangle$. Subsequently, by executing quantum search algorithm, m smallest eigenvalues $\{|\bar{\lambda}_i\rangle\}_{i=1}^m$ and the corresponding eigenvectors $\{|\boldsymbol{u}_i\rangle\}_{i=1}^m$ are obtained.

Stage 3: Computing the compressed vector $\boldsymbol{y}_i$

For each $\boldsymbol{x}_i$, the compressed vector $\boldsymbol{y}_i$ is

$$\boldsymbol{y}_i = U^{\mathrm{T}}\boldsymbol{x}_i = (\langle\boldsymbol{u}_1|\boldsymbol{x}_i\rangle, \ldots, \langle\boldsymbol{u}_m|\boldsymbol{x}_i\rangle)^{\mathrm{T}}. \tag{21}$$

One efficient approach to obtain $\boldsymbol{y}_i$ is computing each

entry by Hadamard test[5,46] with time complexity being $\mathcal{O}(m\log(M))/\varepsilon_y^2)$.

3.2. Quantum enhanced marginal Fisher analysis

EMFA selects between-class marginal points for each point from the nearest classes rather than from all other classes. To reduce its complexity, we propose quantum EMFA (called QEMFA). Different from standard QMFA, QEMFA involves a two-step process for constructing the weight matrix of the penalty graph: searching for the nearest classes of each class and finding the marginal points. The steps to construct $\mathcal{P}_{k_2}^{\mathrm{E}}(c)$ are as follows.

(i) Searching for k_{c} nearest classes of each class

For the class c_p and c_q, the mean vectors of the samples of c_p and c_q are

$$\boldsymbol{\nu}_p = \frac{1}{|c_p|}\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p}\boldsymbol{x}_i, \quad \boldsymbol{\nu}_q = \frac{1}{|c_q|}\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_q}\boldsymbol{x}_i. \tag{22}$$

Then, the similarity between c_p and c_q can be measured by the Euclidean distance between $\boldsymbol{\nu}_p$ and $\boldsymbol{\nu}_q$. To compute the mean vector of each class efficiently, we use the following Theorem 3.

Theorem 3 (quantum mean estimation[47]) Assume that the matrix $A\in\mathbb{R}^{N\times M}$ is stored in a QRAM. For any $\Delta>0$ and $\varepsilon>0$, there exists a quantum algorithm that acts as

$$U_{\mathrm{mean}}:|i\rangle|0\rangle\mapsto|i\rangle|\overline{A_{i,\cdot}}\rangle, \tag{23}$$

in time $\mathcal{O}\big(\frac{\max_{i,j}|A_{ij}|\log(1/\Delta)}{\varepsilon}\big)$ with probability at least $1-2\Delta$, where $\overline{A_{i,\cdot}}$ is the mean of all elements in the i-th row of matrix A, and ε is the error of $\overline{A_{i,\cdot}}$.

Assume that X and its corresponding class vector $(\mathcal{C}(\boldsymbol{x}_1),\ldots,\mathcal{C}(\boldsymbol{x}_N))^{\mathrm{T}}$ are stored in a QRAM. Then, for each class c_p, there exists the oracle O_{XC} that can be applied to prepare the state

$$\begin{aligned}|\tilde{\phi}_p^0\rangle &= \frac{1}{\sqrt{|c_p|}}\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p}|i\rangle|\boldsymbol{x}_i\rangle|c_p\rangle\\ &\quad\otimes\frac{1}{\sqrt{N-|c_p|}}\sum_{\mathcal{C}(\boldsymbol{x}_j)\neq c_p}|j\rangle|\boldsymbol{x}_j\rangle|\mathcal{C}(\boldsymbol{x}_j)\rangle\\ &= \frac{1}{\sqrt{|c_p|}}\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p}|i\rangle|\boldsymbol{x}_i\rangle|c_p\rangle\\ &\quad\otimes\frac{1}{\sqrt{N-|c_p|}}\sum_{c_q\neq c_p}\sum_{\mathcal{C}(\boldsymbol{x}_j)=c_q}|j\rangle|\boldsymbol{x}_j\rangle|c_q\rangle.\end{aligned} \tag{24}$$

Using Theorem 3, we obtain the state

$$|\tilde{\phi}_p^1\rangle = \frac{1}{\sqrt{|c_p|}}\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p}|\boldsymbol{\nu}_p\rangle|c_p\rangle\frac{1}{\sqrt{N-|c_p|}}\sum_{c_q\neq c_p}|\boldsymbol{\nu}_q\rangle|c_q\rangle. \tag{25}$$

Then, using Theorem 2, the distance between between $\boldsymbol{\nu}_p$ and each $\boldsymbol{\nu}_q$ is calculated. We have

$$\begin{aligned}|\tilde{\phi}_p^2\rangle &= \frac{1}{\sqrt{|c_p|}}\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p}|\boldsymbol{\nu}_p\rangle|c_p\rangle\\ &\quad\otimes\frac{1}{\sqrt{N-|c_p|}}\sum_{c_q\neq c_p}|\boldsymbol{\nu}_q\rangle|c_q\rangle|\overline{d^2(\boldsymbol{\nu}_p,\boldsymbol{\nu}_q)}\rangle.\end{aligned} \tag{26}$$

Subsequently, with the use of the quantum search algorithm, $\mathcal{N}_{c_p}$, the set of k_{c} nearest classes of c_p, is obtained.

(ii) Searching for the marginal points

Preparing the state

$$|\tilde{\phi}_p^3\rangle = \frac{1}{\sqrt{|c_p|}}\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p}|i\rangle\frac{1}{\sqrt{\sum_{c_q\in\mathcal{N}_{c_p}}|c_q|}}\sum_{\mathcal{C}(\boldsymbol{x}_j)\in\mathcal{N}_{c_p}}|j\rangle, \tag{27}$$

and using Theorem 2 to estimate the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we have

$$\begin{aligned}|\tilde{\phi}_p^4\rangle &= \frac{1}{\sqrt{|c_p|}}\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p}|i\rangle\\ &\quad\otimes\frac{1}{\sqrt{\sum_{c_q\in\mathcal{N}_{c_p}}|c_q|}}\sum_{\mathcal{C}(\boldsymbol{x}_j)\in\mathcal{N}_{c_p}}|j\rangle|\overline{d^2(\boldsymbol{x}_i,\boldsymbol{x}_j)}\rangle.\end{aligned} \tag{28}$$

Then, search the k_2 shortest distances through quantum search algorithm to form $\mathcal{P}_{k_2}^{\mathrm{E}}(c_p)$.

Iterating the two steps n_{c} rounds, we obtain $\{\mathcal{P}_{k_2}^{\mathrm{E}}(c_p)\}_{p=1}^{n_{\mathrm{c}}}$. $W_{\mathrm{E}}^{\mathrm{P}}$ is constructed with entries

$$[W_{\mathrm{E}}^{\mathrm{P}}]_{ij} = \begin{cases}1, & (i,j)\in\mathcal{P}_{k_2}^{\mathrm{E}}(c(\boldsymbol{x}_i))\,\mathrm{or}\,(i,j)\in\mathcal{P}_{k_2}^{\mathrm{E}}(c(\boldsymbol{x}_j)),\\ 0, & \text{else.}\end{cases} \tag{29}$$

After that, $W_{\mathrm{E}}^{\mathrm{P}}$ serves as the input in Stage 2, along with W. Specifically, storing $W_{\mathrm{E}}^{\mathrm{P}}$ in a QRAM, we implement a $\big(2\|W_{\mathrm{E}}^{\mathrm{P}}\|_{\mathrm{F}},\mathrm{poly}\log(N),\varepsilon_w\big)$-block-encoding of $L_{\mathrm{E}}^{\mathrm{P}}$ in time $\mathcal{O}(\mathrm{poly}\log(N/\varepsilon_w))$ according to Lemma 5. Using Lemma 6, the block-encoding of $X^{\mathrm{T}}L_{\mathrm{E}}^{\mathrm{P}}X+\mu I$ can be obtained and used for the subsequent steps of solving the GEP of $(X^{\mathrm{T}}LX;X^{\mathrm{T}}L_{\mathrm{E}}^{\mathrm{P}}X+\mu I)$ and computing the compressed vectors.

3.3. Quantum multiple marginal Fisher analysis

Different from EMFA and standard MFA, in MMFA, the construction of the penalty graph starts with searching the set of marginal points between each pair of classes (c_p,c_q), denoted as $\mathcal{S}_{k_2}^{\mathrm{M}}(c_p,c_q)$. With a focus on accelerating this process, we propose quantum MMFA (called QMMFA).

Specifically, for (c_p,c_q), we initially prepare the state $|\phi_{p,q}^0\rangle$, where

$$|\phi_{p,q}^0\rangle = \frac{1}{\sqrt{|c_p|}}\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_p}|i\rangle\frac{1}{\sqrt{|c_q|}}\sum_{\mathcal{C}(\boldsymbol{x}_j)=c_q}|j\rangle. \tag{30}$$

Applying Theorem 2 to estimate the distances and quantum search algorithm to search the k_2 nearest data pairs, we obtain $\mathcal{S}_{k_2}^{\mathrm{M}}(c_p,c_q)$. After iterating over all pairs of classes, the sets $\{\mathcal{P}_{k_2}^{\mathrm{M}}(c_p)\}_{p=1}^{n_{\mathrm{c}}}$ are formed. Subsequently, similar to

QEMFA and standard QMFA, we construct the weight matrix $W_{\mathrm{M}}^{\mathrm{P}}$ formed as

$$[W_{\mathrm{M}}^{\mathrm{P}}]_{ij}=\begin{cases}1, & (i,j)\in\mathcal{P}_{k_2}^{\mathrm{M}}(c(\boldsymbol{x}_i)) \text{ or } (i,j)\in\mathcal{P}_{k_2}^{\mathrm{M}}(c(\boldsymbol{x}_j)),\\ 0, & \text{else.}\end{cases} \tag{31}$$

Then, similar to QEMFA and standard QMMFA, we execute Stage 2. After storing $W_{\mathrm{M}}^{\mathrm{P}}$ in a QRAM, the block-encoding of $X^{\mathrm{T}}L_{\mathrm{M}}^{\mathrm{P}}X$ is obtained using Lemma 5 and Lemma 6. This block-encoding, in conjunction with the block-encoding of $X^{\mathrm{T}}LX$ obtained in Subsection 3.1, serves as the input for the subsequent steps.

3.4. Quantum kernel marginal Fisher analysis

KMFA[26,27] extends MFA algorithms to nonlinear cases. According to Eq. (8), the core of designing a quantum algorithm for KMFA is implementing the block-encoding of the kernel matrix K. To address this issue, we present quantum KMFA, namely, QKMFA.

The first stage in QKMFA is similar to QMFA where the weight matrices are constructed efficiently. In the second stage, considering the linear kernel function $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$, the kernel matrix K is equal to XX^{T}. Therefore, assuming that $X\in\mathbb{R}^{N\times M}$ is stored in a QRAM and using Lemma 4, we can implement an $\left(\|X\|_{\mathrm{F}}^2,2\mathrm{poly}\log(N),2\|X\|_{\mathrm{F}}\varepsilon\right)$-block-encoding of $K=XX^{\mathrm{T}}$ in time $\mathcal{O}(\mathrm{poly}\log(NM/\varepsilon))$. Then, given an $N\times N$ matrix Z, we obtain Lemma 7 as follows.

Lemma 7 (implementing block-encoding of $K^{\mathrm{T}}ZK$ and $K^{\mathrm{T}}ZK+\mu I$) Given the data matrix $X\in\mathbb{R}^{N\times M}$ is stored in a QRAM and a matrix $Z\in\mathbb{R}^{N\times N}$ with a $(\alpha_z;q_z;\varepsilon_z)$-block-encoding U_Z that can be implemented in time T_Z, for the linear kernel matrix $K=XX^{\mathrm{T}}$, we can implement an $(\|X\|_{\mathrm{F}}^4\alpha_z,4\mathrm{poly}\log(N)+q_z,4\|X\|_{\mathrm{F}}^3\alpha_z\varepsilon_x+\|X\|_{\mathrm{F}}^4\varepsilon_z$-block-encoding and an $(\|X\|_{\mathrm{F}}^4\alpha_z+\mu,4\mathrm{poly}\log(N)+q_z,4\|X\|_{\mathrm{F}}^3\alpha_z\varepsilon_x+\|X\|_{\mathrm{F}}^4\varepsilon_z)$-block-encoding of $K^{\mathrm{T}}ZK$ and $K^{\mathrm{T}}ZK+\mu I$ in time $\mathcal{O}(\mathrm{poly}\log(NM/\varepsilon_x)+T_Z)$.

Using Lemma 7, with the utilization of the block-encodings of L and L^{P} obtained in Subsection 3.1, we obtain a $(2\|X\|_{\mathrm{F}}^4\|W\|_{\mathrm{F}},5\mathrm{poly}\log(N),4\|X\|_{\mathrm{F}}^3\|W\|_{\mathrm{F}}\varepsilon_x+\|X\|_{\mathrm{F}}^4\varepsilon_w)$-block-encoding of $K^{\mathrm{T}}LK$ and a $(\|X\|_{\mathrm{F}}^4\|W^{\mathrm{P}}\|_{\mathrm{F}}+\mu,5\mathrm{poly}\log(N),4\|X\|_{\mathrm{F}}^3\|W^{\mathrm{P}}\|_{\mathrm{F}}\varepsilon_x+\|X\|_{\mathrm{F}}^4\varepsilon_w)$-block-encoding of $K^{\mathrm{T}}L^{\mathrm{P}}K+\mu I$ in time $\mathcal{O}(\mathrm{poly}\log(NM/(\varepsilon_x\varepsilon_w)))$. Here, Z' also can be the Laplacian matrices constructed in QEMFA and QMMFA. Subsequently, using Theorem 1, the GEP of $(K^{\mathrm{T}}LK;K^{\mathrm{T}}L^{\mathrm{P}}K+\mu I)$ is solved with the initial input quantum state being $\frac{1}{\sqrt{N}}\sum_{i=1}^{N}|i\rangle$. Then, search the m smallest eigenvalues $\{|\bar{\lambda}_i\rangle\}_{i=1}^{m}$ and corresponding eigenvectors $\{|\boldsymbol{u}_i'\rangle\}_{i=1}^{m}$.

Finally, for $\boldsymbol{x}_i$, the low-dimensional vector $\boldsymbol{y}_i$ is

$$\boldsymbol{y}_i=(\langle\boldsymbol{u}_1'|\Phi_\kappa(\boldsymbol{x}_i)\rangle,\ldots,\langle\boldsymbol{u}_m'|\Phi_\kappa(\boldsymbol{x}_i)\rangle)^{\mathrm{T}}, \tag{32}$$

where $|\Phi_\kappa(\boldsymbol{x}_i)\rangle=\frac{1}{N_\kappa^i}\sum_{j=1}^{N}\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)|j\rangle$ with the normalization factor $N_\kappa^i=\sqrt{\sum_{j=1}^{N}\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)}$. The authors in Ref. [48] provides a quantum approach to preparing $|\Phi_\kappa(\boldsymbol{x}_i)\rangle$ in time $O(\sqrt{N}\mathrm{poly}\log(NM/\varepsilon_x)$, then perform the Hadamard test to obtain compressed vectors $\{\boldsymbol{y}_i\}_{i=1}^{N}$.

4. Complexity analysis

Before analyzing the time complexity of individual algorithmic steps, we first take the initial state preparation into account. Assume that the dataset X is balanced in the sense that the number of samples of class $c_p(p=\{1,\ldots,n_{\mathrm{c}}\})$ satisfies $|c_p|=\mathcal{O}(N/n_{\mathrm{c}})$ and assume $n_{\mathrm{c}}\ll N$, then the initial quantum states (including $|\phi_i^0\rangle$, $|\phi_p^0\rangle$, $|\tilde{\phi}_p^0\rangle$, $|\tilde{\phi}_p^3\rangle$ and $|\phi_{p,q}^0\rangle$) can be prepared in time $\mathcal{O}(\mathrm{poly}\log(NM))$ if we have quantum access to $\{\boldsymbol{x}_i\}_{i=1}^{N}$ and their classes $\mathcal{C}(\boldsymbol{x}_i)$. The proof is shown in Appendix C.

The subsequent discussion provides a comprehensive computational complexity analysis for each quantum algorithm, as outlined below.

(i) Complexity of QMFA

During Stage 1, for the constructions of W, the first step is to compute the distances between a sample $\boldsymbol{x}_i$ and other samples within the same class. Theorem 2 is executed with time complexity being $\mathcal{O}(\mathrm{poly}\log(NM/\varepsilon_x)\eta_i/\varepsilon_d)$ where $\eta_i=\max_{\mathcal{C}(\boldsymbol{x}_j)=\mathcal{C}(\boldsymbol{x}_i)}(\|\boldsymbol{x}_j\|)$. Subsequently, using the quantum search algorithm, k_1 nearest neighbors of $\boldsymbol{x}_i$ is obtained in time $\mathcal{O}(\sqrt{k_1|\mathcal{C}(\boldsymbol{x}_i)|})$. Iterating N rounds to obtain $\{\mathcal{N}_{k_1}(i)\}_{i=1}^{N}$, the total complexity is

$$\begin{aligned}T_W&=\mathcal{O}(\sum_{i=1}^{N}\sqrt{k_1|\mathcal{C}(\boldsymbol{x}_i)|}\mathrm{poly}\log(NM/\varepsilon_x)\eta_i/\varepsilon_d)\\&=\mathcal{O}(\eta N\sqrt{k_1N}\mathrm{poly}\log(NM/\varepsilon_x)/\varepsilon_d),\end{aligned} \tag{33}$$

where $\eta=\max_i(\eta_i)=\max_i(\|\boldsymbol{x}_i\|)$. ε_x and ε_d denote errors caused by accessing X and estimating $d(\boldsymbol{x}_i,\boldsymbol{x}_j)$, respectively.

Considering the construction of W^{P}, for the class c_p, the running time of constructing the set of k_2 nearest sample pairs $\mathcal{P}_{k_2}$ is $\mathcal{O}(\sqrt{k_2|c_p|(N-|c_p|)}\mathrm{poly}\log(NM/\varepsilon_x)\eta/\varepsilon_d)$. Therefore, after iterating n_{c} rounds, the time complexity is

$$\begin{aligned}T_{W^{\mathrm{P}}}&=\mathcal{O}\Big(\sum_{p=1}^{n_{\mathrm{c}}}\sqrt{k_2|c_p|(N-|c_p|)}\mathrm{poly}\log(NM/\varepsilon_x)\eta/\varepsilon_d\Big)\\&=\mathcal{O}(\eta\sqrt{k_2}N\mathrm{poly}\log(NM/\varepsilon_x)/\varepsilon_d).\end{aligned} \tag{34}$$

In Stage 2, the block-encodings are implemented to solve the GEP. In this stage, the block-encoding of $X^{\mathrm{T}}LX$ and $X^{\mathrm{T}}L^{\mathrm{P}}X+\mu I$ can be implemented in time $\mathcal{O}(\mathrm{poly}\log(NM/(\varepsilon_x\varepsilon_w)))$. Then, according to Theorem 1, the GEP is solved in time

$$T_G=\mathcal{O}\left(\kappa_B^{0.5}T_{\mathrm{in}}+\frac{\omega\kappa_{\mathrm{B}}^{1.5}}{\varepsilon_\lambda}\left(\mathrm{poly}\log(NM/\varepsilon_w)+\omega^{\mathrm{P}}\kappa_B\right)\right), \tag{35}$$

where $B=X^{\mathrm{T}}L^{\mathrm{P}}X+\mu I$, $\omega=\frac{2\|X\|_{\mathrm{F}}^2\|W\|_{\mathrm{F}}}{\|X^{\mathrm{T}}L^{\mathrm{P}}X+\mu I\|}$ and $\omega^{\mathrm{P}}=\frac{2\|X\|_{\mathrm{F}}^2\|W^{\mathrm{P}}\|_{\mathrm{F}}+\mu}{\|X^{\mathrm{T}}L^{\mathrm{P}}X+\mu I\|}$. Due to $\|W\|_{\mathrm{F}}=\mathcal{O}(\sqrt{Nk_1})$ and $\|W^{\mathrm{P}}\|_{\mathrm{F}}=$

$\mathcal{O}(\sqrt{n_c K_2}) = \mathcal{O}(1)$, $\omega = \mathcal{O}(\sqrt{N})$ and $\omega^{\mathrm{P}} = \mathcal{O}(1)$. Furthermore, the running time to prepare the initial state in Theorem 1 is $T_{\mathrm{in}} = \mathcal{O}(\log(M))$. Assume that the X is scaled such that $\|X\|_{\mathrm{F}}$ is a constant, then T_G is on the order of $\mathcal{O}(\sqrt{N}\mathrm{poly}\log(NM))$.

In the final stage, the complexity of obtaining all low-dimensional vectors is $T_C = \mathcal{O}(mN\log(M))/\varepsilon_y^2)$.

Further adding the running time of obtaining the compressed vectors, and assuming that errors satisfy $\varepsilon_y = \varepsilon_\lambda = \varepsilon_v = \varepsilon_d = \mathcal{O}(\log(\varepsilon_x)) = \mathcal{O}(\log(\varepsilon_w))$ denoted to ε and $k_1, k_2 \ll N$, we have the time complexity of QMFA as follows:

$$\begin{aligned} T_{\mathrm{QMFA}} &= T_W + T_{W^{\mathrm{P}}} + T_G + T_C \\ &= \mathcal{O}(N\sqrt{N}\mathrm{poly}\log(NM)/\varepsilon). \end{aligned} \tag{36}$$

Compared to the classical MFA with time complexity being $\mathcal{O}(N^2M + M^3)$, the proposed QMFA exhibits polynomial acceleration in the number of samples N and exponential acceleration in the dimensionality M. The comparison of the time complexity of each step is shown in Table 1.

Table 1. Comparison of time complexity for each step in QMFA.

	QMFA	Classical MFA
T_W	$\mathcal{O}(\eta N\sqrt{k_1 N}\mathrm{poly}\log(NM/\varepsilon_x)/\varepsilon_d)$	$\mathcal{O}(N^2M + k_1 N)$
$T_{W^{\mathrm{P}}}$	$\mathcal{O}(\eta\sqrt{k_2}N\mathrm{poly}\log(NM/\varepsilon_x)/\varepsilon_d)$	$\mathcal{O}(k_2 N^2)$
T_G	$\mathcal{O}(\kappa_B^{0.5}\log(M) + \frac{\omega\kappa_B^{1.5}}{\varepsilon_\lambda}(\mathrm{poly}\log(NM/\varepsilon_w) + \omega^{\mathrm{P}}\kappa_B))$	$\mathcal{O}(N^2M + M^2N + M^3)$
T_C	$\mathcal{O}(mN\log(M))/\varepsilon_y^2)$	$\mathcal{O}(mNM)$
T_{total}	$\mathcal{O}(N\sqrt{N}\mathrm{poly}\log(NM)/\varepsilon)$	$\mathcal{O}(N^2M + M^3)$

(ii) Complexities of QEMFA, QMMFA, and QKMFA

The primary distinction among QEMFA, QMMFA, and QMFA is the construction of the penalty graph, while QKMFA differs from QMFA in the block-encoding methodology used in its second stage. Therefore, this complexity analysis will focus on the graph construction complexity of QEMFA and QMMFA, as well as the computational complexity of the second stage of QKMFA.

In QEMFA, for the construction of $W_{\mathrm{E}}^{\mathrm{P}}$, the k_{c} nearest classes of the class c_p is obtained first in time $T_{p1}^{\mathrm{E}} = \mathcal{O}(\sqrt{k_{\mathrm{c}} n_{\mathrm{c}}}\max_{i,j}|X_{ij}|\log\frac{1}{\Delta}\mathrm{poly}\log(NM/\varepsilon_x)/\varepsilon_v)$ with probability at least $1-2\Delta$. Subsequently, the time complexity of forming $\mathcal{P}_{k_2}^{\mathrm{E}}(c_p)$ is

$$T_{p2}^{\mathrm{E}} = \mathcal{O}\left(\sqrt{k_2|c_p|\sum_{c_q\in\mathcal{N}_{c_p}}|c_q|}\mathrm{poly}\log(NM/\varepsilon_x)\eta/\varepsilon_d\right).$$

Thus, the total time complexity of the construction of $W_{\mathrm{E}}^{\mathrm{P}}$ is

$$\begin{aligned} T_{W_{\mathrm{E}}^{\mathrm{P}}} &= \sum_{p=1}^{n_{\mathrm{c}}} T_{p1}^{\mathrm{E}} + T_{p2}^{\mathrm{E}} \\ &= \mathcal{O}\left(\left(\eta N/\varepsilon_d + \max_{i,j}|X_{ij}|\log\frac{1}{\Delta}/\varepsilon_v\right)\right. \\ &\quad \left.\times\sqrt{k_2}\mathrm{poly}\log(NM/\varepsilon_x)\right). \end{aligned} \tag{37}$$

In QMMFA, for the construction of $W_{\mathrm{M}}^{\mathrm{P}}$, k_2 nearest sample pairs between each pair of classes (c_q, c_q) are obtained. During one iteration, the running time is $\mathcal{O}(\sqrt{k_2|c_p||c_q|}\mathrm{poly}\log(NM/\varepsilon_x)\eta_{pq}/\varepsilon_d)$ where $\eta_{pq} = \max_{\mathcal{C}(\boldsymbol{x}_i)\in\{c_p,c_q\}}(\|\boldsymbol{x}_i\|)$. After iterating n_{c}^2 rounds, the time complexity is

$$\begin{aligned} T_{W_{\mathrm{M}}^{\mathrm{P}}} &= \mathcal{O}\left(\sum_{p=1}^{n_{\mathrm{c}}}\sum_{q=1}^{n_{\mathrm{c}}}\sqrt{k_2|c_p||c_q|}\mathrm{poly}\log(NM/\varepsilon_x)\eta_{pq}/\varepsilon_d\right) \\ &= \mathcal{O}(\eta\sqrt{k_2}N\mathrm{poly}\log(NM/\varepsilon_x)/\varepsilon_d). \end{aligned} \tag{38}$$

Notably, although $T_{W^{\mathrm{P}}}$, $T_{W_{\mathrm{E}}^{\mathrm{P}}}$, and $T_{W_{\mathrm{M}}^{\mathrm{P}}}$ are all on the order of $\mathcal{O}(N\mathrm{poly}\log(NM))$, due to differences in the search spaces of sample pairs, we have

$$T_{W_{\mathrm{E}}^{\mathrm{P}}} \le T_{W^{\mathrm{P}}} \le T_{W_{\mathrm{M}}^{\mathrm{P}}}. \tag{39}$$

In classical EMFA and MMFA, the time complexities for constructing $W_{\mathrm{E}}^{\mathrm{P}}$ and $W_{\mathrm{M}}^{\mathrm{P}}$ are on the order of $\mathcal{O}(N^2)$. In contrast, quantum methods provide polynomial speedups. When considering the subsequent quantum processes, the overall time complexities for QEMFA and QMMFA are approximately on the order of $\mathcal{O}(N\sqrt{N}\mathrm{poly}\log(NM))$. Therefore, QEMFA and QMMFA achieve a speedup equivalent to that of QMFA.

In QKMFA, the GEP of $(K^{\mathrm{T}}LK; K^{\mathrm{T}}L^{\mathrm{P}}K + \mu I)$ is solved in time

$$T_K = \mathcal{O}\left(\kappa_{B_K}^{0.5}T_{\mathrm{in}}^K + \frac{\omega_K\kappa_{B_K}^{1.5}}{\varepsilon_\lambda}\left(\mathrm{poly}\log(NM/\varepsilon_w) + \omega_K^{\mathrm{P}}\kappa_{B_K}\right)\right), \tag{40}$$

where $B_K = K^{\mathrm{T}}L^{\mathrm{P}}K + \mu I$, $\omega_K = \frac{2\|X\|_{\mathrm{F}}^4\|W\|_{\mathrm{F}}}{\|K^{\mathrm{T}}L^{\mathrm{P}}K+\mu I\|}$, $\omega_K^{\mathrm{P}} = \frac{2\|X\|_{\mathrm{F}}^4\|W^{\mathrm{P}}\|_{\mathrm{F}}+\mu}{\|K^{\mathrm{T}}L^{\mathrm{P}}K+\mu I\|}$ and $T_{\mathrm{in}}^K = \log(N)$. Similar to T_G in QMFA, T_K is on the order of $\mathcal{O}(\sqrt{N}\mathrm{poly}\log(NM))$. Then, in the stage of obtaining the low-dimensional vectors, the time complexity of obtaining $\{\boldsymbol{y}_i\}_{i=1}^N$ is $\mathcal{O}(mN\sqrt{N}\mathrm{poly}\log(NM)/\varepsilon_y^2)$. Since the constructions of W and W^{P} in QKMFA are equivalent to those

in QMFA, the total complexity of QKMFA is also on the order of $\mathcal{O}(N\sqrt{N}\text{poly}\log(NM))$. Therefore, QKMFA also exhibits a polynomial acceleration in the number of samples N and exponential acceleration in the dimensionality M compared to the classical KMFA.

5. Numerical simulation

To evaluate the effectiveness of the proposed QMFA algorithm, we conduct comprehensive numerical experiments from two perspectives. First, using the Qiskit platform,[49] we simulated a simplified quantum circuit with 18 qubits to validate the feasibility and evaluate the performance of QMFA. Second, we perform classical perturbation simulations, in which random perturbations are introduced into classical algorithms to emulate quantum errors, thereby approximating realistic quantum computational environments. Given that the primary objective of dimensionality reduction algorithms is to enhance the performance of downstream tasks, we quantitatively assess embedding quality through the classification accuracy using support vector machine (SVM).[50] Specifically, we focus on the binary classification problem of distinguishing digits '2' and '3' from the handwritten digits dataset,[51] which contains 360 samples with an original feature dimension of 64. All experiments are conducted using 5-fold cross-validation, and the average classification accuracy across folds is reported as the primary performance metric.

(i) Quantum circuit simulation

Assuming that the implementation of the critical QRAM operations is feasible, the block-encoding of the target matrices requires a significant number of ancillary qubits. For instance, considering a data matrix of size $2^3 \times 2^3$, the construction of the corresponding block-encodings demands at least 30 qubits. In the subsequent phase estimation step, a 2-qubit register is employed to achieve eigenvalue precision up to 2 bits. During the Grover search process, an additional 2 qubits are allocated for eigenvalue comparison, along with 1 qubit for the Grover oracle used to flag eigenvalues below a specified threshold. Consequently, the total qubit requirement reaches 35 qubits. Moreover, the Grover amplitude amplification procedure involves repeated invocations of quantum circuits for computing $X^{\mathrm{T}}LX$ and $X^{\mathrm{T}}L^{\mathrm{P}}X + \mu I$, which imposes a considerable computational burden. This exceeds the current capabilities of our computing device.

To address the aforementioned resource constraints, we simplify the quantum algorithm by implementing a Hamiltonian simulation of the operator $(X^{\mathrm{T}}L^{\mathrm{P}}X + \mu I)^{-1}(X^{\mathrm{T}}LX)$ using Qiskit, followed by phase estimation and amplitude amplification. In this setup, phase estimation is performed with a 6-qubit register to approximate the eigenvalues, while amplitude amplification involves a pre-defined threshold to select eigenvectors associated with eigenvalues below this threshold for projection. For 64-dimensional input features, the overall quantum circuit requires 18 qubits: 5 qubits for Hamiltonian simulation, 6 for phase estimation, and 7 for amplitude amplification. Subsequently, we apply amplitude encoding to the input vector $\boldsymbol{x}_i$ and compute its inner product with the selected eigenvectors, thereby obtaining a 6-dimensional embedding vector $\boldsymbol{y}_i$ for each sample. As demonstrated in Fig. 4, the quantum algorithm achieved 5-fold cross-validation accuracies of 94.44%, 81.94%, 88.89%, 88.89%, and 95.83%, yielding a mean classification accuracy of 90.00%. In contrast, the classical algorithm attained accuracies of 91.67%, 76.39%, 93.06%, 80.56%, and 93.06%, with a mean accuracy of 86.84%. This comparative analysis reveals that the proposed QMFA outperforms its classical counterpart in classification performance.

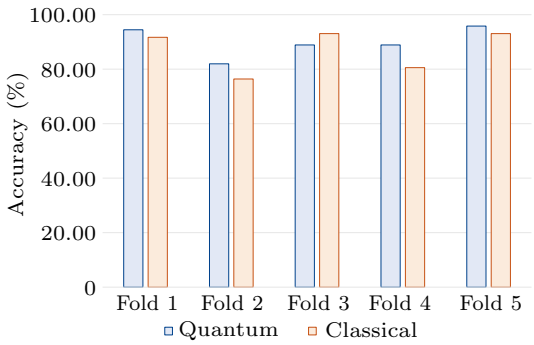


Fig. 4. Performance comparison between quantum and classical MFA under 5-fold cross-validation.

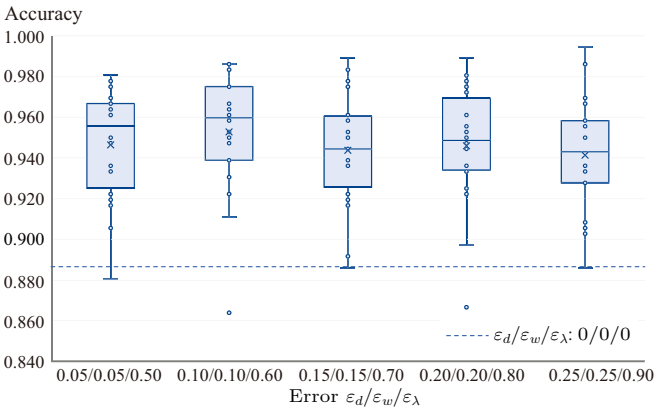


Fig. 5. Classification accuracy under different errors in classical perturbation simulation.

(ii) Classical perturbation simulation

Following the simulation approaches in Refs. [8,9], we assess the effectiveness of QMFA by introducing controlled errors and stochasticity into classical algorithms. In the classical perturbation simulation, we reduce the dimensionality to 16 and employ SVM to classify the obtained embedding vectors. Specifically, we consider three primary sources of error: ε_d (associated with distance estimation), ε_w (stemming from matrix simulation inaccuracies), and ε_λ (resulting from eigenvalue estimation imprecision). We conduct experiments across five distinct error configurations ($\varepsilon_d/\varepsilon_w/\varepsilon_\lambda$): 0.05/0.05/0.5, 0.1/0.1/0.6, 0.15/0.15/0.7, 0.2/0.2/0.8, and 0.25/0.25/0.9. For

each configuration, 20 randomized trials are performed to ensure statistical reliability. As illustrated in Fig. 5, the results show that the proposed quantum algorithm maintains competitive classification accuracy compared to its classical counterparts, and even outperforms them.

6. Conclusion

In this paper, we present a quantum algorithm for MFA, called QMFA. We further explore quantum algorithms for the variants of MFA, including EMFA, MMFA, and KMFA. Complexity analysis demonstrates that the proposed quantum algorithms achieve polynomial speedup in the number of samples and exponential speedup in dimensionality when compared to classical counterparts. MFA algorithm is a representative method within the family of graph embedding-based dimensionality reduction techniques. Graph embedding provides a unified framework for a wide range of dimensionality reduction algorithms by constructing an intrinsic graph to preserve local data structure and a penalty graph to suppress undesired variations, followed by solving a generalized eigenvalue problem. In this paper, we develop a quantum version of the classical MFA algorithm, thereby laying the groundwork for exploring quantum implementations of graph embedding-based dimensionality reduction methods. This quantum algorithm can serve as a reference framework and a valuable guide for extending other classical graph embedding techniques to quantum computing paradigms.

Currently, there is no efficient quantum search algorithm that can obtain the superposition state of the k nearest neighbors of multiple samples in parallel. Consequently, we iteratively search for the nearest neighbors of each sample, which makes the construction of the weight matrix time-consuming. The development of a more efficient search algorithm holds promise for significantly accelerating the proposed approach. Moreover, hybrid quantum-classical algorithms,[52] such as the quantum kernel estimator,[53] and quantum neural networks,[54] show excellent performance in various machine learning tasks. Explore their combination with MFA has the potential to yield a more diverse and effective kernel function, consequently leading to performance enhancements. We will conduct further research on these aspects in the future.

Appendix A: Proof of Lemma 3

Proof Let $V_j = I_{a-a_j} \otimes U_j$. Then, V_j is the $(\alpha_j, a, \varepsilon_j)$-block-encoding of A_j.

$$
\begin{aligned}
&\|A - \alpha\beta(\langle 0|^{\otimes b} \otimes \langle 0|^{\otimes a} \otimes I)\tilde{\mathcal{U}}(|0\rangle^{\otimes b} \otimes |0\rangle^{\otimes a} \otimes I)\| \\
=&\|A - \sum_{j=0}^{m-1} \alpha\beta c_j^*(\langle 0|^{\otimes a} \otimes I)V_j(|0\rangle^{\otimes a} \otimes I)\| \\
\leq&\alpha\varepsilon_1 + \|A - \sum_{j=0}^{m-1} \alpha y_j(\langle 0|^{\otimes a} \otimes I)V_j(|0\rangle^{\otimes a} \otimes I)\| \\
\leq&\alpha\varepsilon_1 + \|A - \sum_{j=0}^{m-1} \alpha_j y_j(\langle 0|^{\otimes a} \otimes I)V_j(|0\rangle^{\otimes a} \otimes I)\| \\
\leq&\alpha\varepsilon_1 + \sum_{j=0}^{m-1} y_j\|A_j - \alpha_j(\langle 0|^{\otimes a} \otimes I)V_j(|0\rangle^{\otimes a} \otimes I)\| \\
\leq&\alpha\varepsilon_1 + \sum_{j=0}^{m-1} y_j\varepsilon_j. \qquad \text{(A1)}
\end{aligned}
$$

Appendix B: Proof of Lemma 5

Proof

Lemma B1 (block-encoding of sparse-access matrices[41]) Let $A \in \mathbb{C}^{N\times N}$ be a matrix that is s_r-row-sparse and s_c-column-sparse. Suppose that we have access to the following sparse-access oracles acting on two $(\lceil \log(N) \rceil + 1)$-qubit registers:

$$O_r : |i\rangle|j\rangle \mapsto |i\rangle|r_{ik}\rangle, \quad \forall i \in [N]-1, k \in [s_r], \qquad \text{(B1)}$$

$$O_c : |\ell\rangle|j\rangle \mapsto |\ell\rangle|c_{\ell j}\rangle, \quad \forall \ell \in [s_c], j \in [N]-1, \qquad \text{(B2)}$$

where r_{ij} is the index for the j-th non-zero entry of the i-th row of A, or if there are fewer than i non-zero entries, then it is $j+N$, and similarly c_{ij} is the index for the i-th non-zero entry of the j-th column of A, or if there are fewer than j non-zero entries, then it is $i+N$. Additionally, assume that we have access to an oracle $\tilde{O}_A$ that returns the entries of A in a binary description:

$$\tilde{O}_A : |i\rangle|j\rangle|0\rangle^{\otimes b} \mapsto |i\rangle|j\rangle|A_{ij}\rangle, \quad \forall i, j \in [N]-1, \qquad \text{(B3)}$$

where A_{ij} is a b-bit binary description of the ij-matrix element of A. Then a $(\sqrt{s_r s_c}\max_{ij}(A_{ij}), \lceil \log(N) \rceil + 3, \varepsilon)$-block-encoding of A can be implemented with a single use of O_r, O_c, two uses of $\tilde{O}_A$, and additionally using $\mathcal{O}\left(\log(N) + \log^{2.5}\left(\frac{s_r s_c}{\varepsilon}\right)\right)$ two-qubit gates while using $\mathcal{O}\left(b, \log^{2.5}\left(\frac{s_r s_c}{\varepsilon}\right)\right)$ ancilla qubits.

Since D is a diagonal matrix with $D_{ii} = \sum_j W_{ij} = \|W_{i\cdot}\|^2$, according to Lemma B1, if there are two unitary operators U_D and V_D to access the index of non-zero entries of D and access the entries of D respectively, where

$$U_D : |i\rangle|0\rangle \mapsto |i\rangle|i\rangle, \qquad \text{(B4)}$$

$$V_D : |i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle|D_{ij}\rangle, \qquad \text{(B5)}$$

then we can construct a $(1, \text{poly}\log(N), \varepsilon)$-block-encoding of D in time $\mathcal{O}(T_U + T_D)$.

In terms of U_D, we prepare the initial quantum state $\frac{1}{\sqrt{N}}\sum_{i=1}^{N}|i\rangle|0\rangle$. Then, U_D can be implemented in time $\mathcal{O}(\text{poly}\log(N))$ by a quantum addition circuit acting as $|i\rangle|0\rangle \mapsto |i\rangle|0 \oplus i\rangle$.

For V_D, based on the assumption that we have quantum access to W, there is an oracle to access O_W the norm of the i-th row of W in time $\mathcal{O}(\mathrm{poly}\log(N))$, where

$$O_W : |i\rangle|0\rangle \mapsto |i\rangle|\|W_{i\cdot}\|\rangle. \tag{B6}$$

At first, prepare the initial quantum state $|\psi_0\rangle = \frac{1}{N}\sum_{i,j=1}^{N}|i\rangle|j\rangle|0\rangle|0\rangle_{\mathrm{c}}$. Then, checking the equality of i and j by U_{e}, the state becomes

$$|\psi_1\rangle = \frac{1}{N}\left(\sum_{i=1}^{N}|i\rangle|i\rangle|0\rangle|0\rangle_{\mathrm{c}} + \sum_{i=1}^{N}|i\rangle\sum_{j\neq i}|j|0\rangle\rangle|1\rangle_{\mathrm{c}}\right). \tag{B7}$$

Next, implementing the controlled O_W to access $\|W_{i\cdot}\|$ if the controlled qubit is in $|0\rangle$, we have

$$|\psi_2\rangle = \frac{1}{N}\left(\sum_{i=1}^{N}|i\rangle|i\rangle|\|W_{i\cdot}\|\rangle|0\rangle_{\mathrm{c}} + \sum_{i=1}^{N}|i\rangle\sum_{j\neq i}|j\rangle|0\rangle|1\rangle_{\mathrm{c}}\right). \tag{B8}$$

Implementing the inverse of U_{e}, we obtain

$$|\psi_3\rangle = \frac{1}{N}\left(\sum_{i=1}^{N}|i\rangle|i\rangle|\|W_{i\cdot}\|\rangle + \sum_{i=1}^{N}|i\rangle\sum_{j\neq i}|j\rangle|0\rangle\right)|0\rangle_{\mathrm{c}}. \tag{B9}$$

Appending a register to store $|\|D_{ii}\|\rangle = |\|W_{i\cdot}\|^2\rangle$, there exists a quantum circuit U_{pow} to realize

$$\begin{aligned}|\psi_4\rangle &= U_{\mathrm{pow}}|\psi_3\rangle \\ &= \frac{1}{N}\left(\sum_{i=1}^{N}|i\rangle|i\rangle|\|W_{i\cdot}\|\rangle|\|W_{i\cdot}\|^2\rangle + \sum_{i=1}^{N}|i\rangle\sum_{j\neq i}|j\rangle|0\rangle|0\rangle\right).\end{aligned} \tag{B10}$$

Thus, after the inverse of the controlled O_W, we have

$$|\psi_5\rangle = \frac{1}{N}\left(\sum_{i=1}^{N}|i\rangle|i\rangle|\|W_{i\cdot}\|^2\rangle + \sum_{i=1}^{N}|i\rangle\sum_{j\neq i}|j\rangle|0\rangle\right). \tag{B11}$$

Through the above operations, we access the entries of D successfully. U_{e} and U_{pow} acting as

$$U_{\mathrm{e}} : |i\rangle|j\rangle|0\rangle \mapsto\rangle|i\rangle|j\rangle|\delta(i,j)\rangle, \tag{B12}$$

$$U_{\mathrm{pow}} : |i\rangle|a_i\rangle \mapsto\rangle|i\rangle|a_i^2\rangle, \tag{B13}$$

can be implemented in time $\mathcal{O}(\mathrm{poly}\log(N))$. Here $\delta(i,j)=0$ if $i=j$, otherwise $\delta(i,j)=1$. Therefore, the time complexity of implementing V_D is $\mathcal{O}(\mathrm{poly}\log(N/\varepsilon_w))$. Overall, a $\left(\max_i(\|W_{i\cdot}\|^2), \mathrm{poly}\log(N), \varepsilon_w\right)$-block-encoding of D can be implemented in time $\mathcal{O}(\mathrm{poly}\log(N/\varepsilon_w))$.

At this moment, a $(\|W\|_{\mathrm{F}}, \mathrm{poly}\log(N), \varepsilon_w)$-block-encoding and a $\left(\max_i(\|W_{i\cdot}\|^2), \mathrm{poly}\log(N), \varepsilon_w\right)$-block-encoding of W and D can be implemented in time $\mathcal{O}(\mathrm{poly}\log(N/\varepsilon_w))$. Using Lemma 2, let $\boldsymbol{y}=(1,-1)$, $\beta = \|y\|_1 = 2$, $b=1$, and $\boldsymbol{c}=\boldsymbol{d}=(1/\sqrt{2},-1/\sqrt{2})$, then (H, HX) is a $(2,1,0)$-state-preparation-pair for $\boldsymbol{y}$. As a result, we can implement a $(2\|W\|_{\mathrm{F}}, \mathrm{poly}\log(N), \varepsilon_w)$-block-encoding of $L = D - W$ in time $\mathcal{O}(\mathrm{poly}\log(N/\varepsilon_w))$.

Appendix C: Complexity analysis of initial quantum states preparation

Assume that we have quantum access to samples with their class $\{\boldsymbol{x}_i, \mathcal{C}(\boldsymbol{x}_i)\}_{i=1}^{N}$ that are stored in a QRAM, then there exist oracles $\tilde{O}_X$ and $\tilde{O}_C$ acting as

$$\tilde{O}_X : |i\rangle|0\rangle \mapsto |i\rangle|\boldsymbol{x}_i\rangle, \tag{C1}$$

$$\tilde{O}_C : |i\rangle|0\rangle \mapsto |i\rangle|\mathcal{C}(\boldsymbol{x}_i)\rangle, \tag{C2}$$

in time $\mathcal{O}(\mathrm{poly}\log(NM))$.

(i) Preparation of $|\phi_i^0\rangle$

Prepare the quantum state $|i\rangle_1|0\rangle_2^{\otimes\log(n_{\mathrm{c}})}|0\rangle_3^{\otimes\log(N)}|0\rangle_4^{\otimes\log(n_{\mathrm{c}})}$ and apply Hardamard gates $H^{\otimes\log(N)}$ on the third register, then we obtain

$$|\theta_0\rangle = |i\rangle|0\rangle\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle|0\rangle. \tag{C3}$$

Next, executing the oracle $\tilde{O}_C$, we have

$$|\theta_1\rangle = |i\rangle|\mathcal{C}(\boldsymbol{x}_i)\rangle\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle|\mathcal{C}(\boldsymbol{x}_j)\rangle. \tag{C4}$$

Append an ancilla qubit and compare $|\mathcal{C}(\boldsymbol{x}_j)\rangle$ to $|\mathcal{C}(\boldsymbol{x}_i)\rangle$, then the state becomes

$$\begin{aligned}|\theta_2\rangle = |i\rangle|\mathcal{C}(\boldsymbol{x}_i)\rangle\frac{1}{\sqrt{N}}\Bigg(&\sum_{\mathcal{C}(\boldsymbol{x}_j)=\mathcal{C}(\boldsymbol{x}_i)}|j\rangle|\mathcal{C}(\boldsymbol{x}_j)\rangle|0\rangle \\ &+ \sum_{\mathcal{C}(\boldsymbol{x}_j)\neq\mathcal{C}(\boldsymbol{x}_i)}|j\rangle|\mathcal{C}(\boldsymbol{x}_j)\rangle|1\rangle\Bigg).\end{aligned} \tag{C5}$$

Measuring the ancilla qubit, $|\phi_i^0\rangle$ is obtained with the success probability of getting $|0\rangle$ being $p_0 = |\mathcal{C}(\boldsymbol{x}_i)|/N$. We need $\mathcal{O}(1/p_0)=\mathcal{O}(1)$ measurements to obtain $|\phi_i^0\rangle$ under the assumption of $|c_p| = \Omega(N/n_{\mathrm{c}})$ and $n_{\mathrm{c}} \ll N$ for $p=\{1,\ldots,n_{\mathrm{c}}\}$. Therefore, the time complexity to prepare the state $|\phi_i^0\rangle$ is $\mathcal{O}(\mathrm{poly}\log(NM))$.

(ii) Preparation of $|\phi_p^0\rangle$, $|\tilde{\phi}_p^0\rangle$, $|\tilde{\phi}_p^3\rangle$ and $|\phi_{p,q}^0\rangle$

Prepare the quantum state $|0\rangle_1^{\otimes\log(N)}|0\rangle_2^{\otimes\log(n_{\mathrm{c}})}|0\rangle_3^{\otimes\log(N)}|0\rangle_4^{\otimes\log(n_{\mathrm{c}})}$ and apply Hardamard gates $H^{\otimes\log(N)}$ on the first and third registers, then we obtain

$$|\tilde{\theta}_0\rangle = \frac{1}{\sqrt{N}}\sum_{i=1}^{N}|i\rangle|0\rangle\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle|0\rangle. \tag{C6}$$

Next, executing the oracle $\tilde{O}_C$, we have

$$|\tilde{\theta}_1\rangle = \frac{1}{\sqrt{N}}\sum_{i=1}^{N}|i\rangle|\mathcal{C}(\boldsymbol{x}_j)\rangle\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle|\mathcal{C}(\boldsymbol{x}_j)\rangle. \tag{C7}$$

Append an ancilla quantum register $|c_p\rangle$ and 2 flag qubits, and compare $|\mathcal{C}(\boldsymbol{x}_i)\rangle$ and $|\mathcal{C}(\boldsymbol{x}_2)\rangle$ to $|c_p\rangle$, then the state becomes

$$|\tilde{\theta}_2\rangle = |c_p\rangle\frac{1}{N}\Bigg(\sum_{\mathcal{C}(\boldsymbol{x}_i)=c_q}|i\rangle|\mathcal{C}(\boldsymbol{x}_i)\rangle|0\rangle_{f_1}$$

$$\otimes \sum_{\mathcal{C}(\boldsymbol{x}_j)\neq c_q} |j\rangle|\mathcal{C}(\boldsymbol{x}_j)\rangle|1\rangle_{f_2} + |\perp\rangle\Big), \tag{C8}$$

where $|\perp\rangle$ denotes the state with the flag qubits in $|0\rangle|0\rangle_{f_1,f_2}$, $|1\rangle|0\rangle_{f_1,f_2}$, and $|1\rangle|1\rangle_{f_1,f_2}$. Measuring the flag qubits to see $|0\rangle|1\rangle_{f_1,f_2}$, $|\phi_p^0\rangle$ is obtained with the success probability being $p_{01} = |c_p||N-c_p|/N^2$. Similar to the preparation of $|\phi_i^0\rangle$, $\mathcal{O}(1)$ measurements are required to obtain $|\phi_p^0\rangle$, and the total time complexity is $\mathcal{O}(\mathrm{poly}\log(NM))$.

After preparing $|\phi_p^0\rangle$, $|\tilde{\phi}_p^0\rangle$ is obtained by applying the oracle $\tilde{O}_X$ in time $\mathcal{O}(\mathrm{poly}\log(NM))$. Furthermore, $|\tilde{\phi}_p^3\rangle$ and $|\phi_{p,q}^0\rangle$ can be prepared by querying whether the forth register $|\mathcal{C}(\boldsymbol{x}_j)\rangle$ in state $|\tilde{\theta}_1\rangle$ is in $\mathcal{N}_{c_p}$ and equal to c_q with success probability being $|c_p||\sum_{q\in\mathcal{N}_{c_p}} c_q|/N^2$ and $|c_p||c_q|/N^2$ respectively. The total time complexity of preparing $|\phi_p^0\rangle$, $|\tilde{\phi}_p^0\rangle$ are $\mathcal{O}(\mathrm{poly}\log(NM))$ as well.

Acknowledgement

Project supported by the National Natural Science Foundation of China (Grant Nos. 62372048, 62371069, 62272056, and U25B2014).

References

[1] Jordan M I and Mitchell T M 2015 *Science* **349** 255
[2] Shor P W 1994 *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, Santa Fe, NM, USA, IEEE, pp. 124–134
[3] Grover L K 1997 *Phys. Rev. Lett.* **79** 325
[4] Harrow A W, Hassidim A and Lloyd S 2009 *Phys. Rev. Lett.* **103** 150502
[5] Rebentrost P, Mohseni M and Lloyd S 2014 2009 *Phys. Rev. Lett.* **113** 130503
[6] Wiebe N, Kapoor A and Svore K M 2015 *Quantum Inform. Comput.* **15** 316
[7] Song Y, Wu Y, Wu S, Li D, Wen Q, Qin S and Gao F 2024 *Sci. China-Phys. Mech. Astron.* **67** 250311
[8] Kerenidis I and Landman J 2021 *Phys. Rev. A* **103** 042415
[9] Li Y M, Liu H L, Pan S J, Qin S J, Gao F, Sun D X and Wen Q Y 2023 *Phys. Rev. A* **107** 022421
[10] Ni X H, Cai B B, Liu H L, Qin S J, Gao F and Wen Q Y 2024 *Adv. Quantum Technol.* **7** 2300419
[11] Li L, Li J, Song Y, Qin S, Wen Q and Gao F 2025 *Sci. China-Phys. Mech. Astron.* **68** 1
[12] Wu S Y, Song Y Q, Li R Z, Qin S J, Wen Q Y and Gao F 2025 *Adv. Quantum Technol.* 2400484
[13] Zhang H, Wang S, Liu X, Shen Y and Wang Y 2024 *Chin. Phys. B* **33** 020310
[14] Li J A, Dong D, Wei Z, Liu Y, Pan Y, Nori F and Zhang X 2020 *Nat. Hum. Behav.* **4** 294
[15] Kwak Y, Yun W J, Jung S, Kim J K and Kim J 2021 *International Conference on Information and Communication Technology Convergence*, 2021, Jeju Island, Korea, IEEE, pp. 416–420
[16] Jin Y X, Xu H Z, Wang Z A, Zhuang W F, Huang K X, Shi Y H, Ma W G, Li T M, Chen C T, Xu K, *et al.* 2024 *Chin. Phys. B* **33** 050301
[17] Ayesha S, Hanif M K and Talib R 2020 *Inf. Fusion* **59** 44
[18] Liang X, Lin Y E, Zhang S and Fang X 2023 *Appl. Intell.* **53** 12873
[19] Liu L, Chen J, Liu T, Chen C P and Yang B 2024 *IEEE T. Cybern.* **55** 5
[20] Tharwat A, Gaber T, Ibrahim A and Hassanien A E 2017 *AI Commun.* **30** 169
[21] He X and Niyogi P 2003 *Advances in Neural Information Processing Systems* **16** 153
[22] Yan S, Xu D, Zhang B, Zhang H J, Yang Q and Lin S 2006 *IEEE Trans. Pattern Anal. Mach. Intell.* **29** 40
[23] Xu D, Yan S, Tao D, Lin S and Zhang H J 2007 *IEEE Trans. Image Process.* **16** 2811
[24] Huang P and Chen C 2009 *International Conference on Artificial Intelligence and Computational Intelligence*, 2009, Shanghai, China, IEEE, pp. 403–407
[25] Huang Z, Zhu H, Zhou J T and Peng X 2018 *IEEE Trans. Ind. Electron.* **66** 9798
[26] Jiang L, Xuan J and Shi T 2013 *Mech. Syst. Signal Proc.* **41** 113
[27] Wang Z and Sun X 2012 *J. Comput.* **7** 2298
[28] Lloyd S, Mohseni M and Rebentrost P 2014 *Nat. Phys.* **10** 631
[29] Yu C H, Gao F, Lin S and Wang J 2019 *Quantum Inf. Process.* **18** 249
[30] Cong I and Duan L 2016 *New J. Phys.* **18** 073011
[31] He X, Zhang A and Zhao S 2022 *Quantum Inf. Process.* **21** 86
[32] Li Y M, Liu H L, Pan S J, Qin S J, Gao F and Wen Q Y 2023 *Phys. Rev. A* **108** 042418
[33] Wiebe N, Kapoor A and Svore K M 2014 arXiv: 1412.3489 [quant-ph]
[34] Kerenidis I, Landman J, Luongo A and Prakash A 2019 *33rd Conference on Neural Information Processing Systems*, 2019, Vancouver, Canada, pp. 4134–4144
[35] Dürr C and Høyer P 1996 arXiv: quant-ph/9607014
[36] Dürr C, Heiligman M, Høyer P and Mhalla M 2006 *SIAM J. Comput.* **35** 1310
[37] Chakraborty S, Gilyén A and Jeffery S 2019 *46th International Colloquium on Automata, Languages, and Programming*, 2019, Dagstuhl, Germany, pp. 33:1–33:14
[38] Low G H and Chuang I L 2019 *Quantum* **3** 163
[39] Wan L C, Yu C H, Pan S J, Qin S J, Gao F and Wen Q Y 2021 *Phys. Rev. A* **104** 062414
[40] Gao S and Yang Y G 2023 *Chin. Phys. B* **32** 100309
[41] Gilyén A, Su Y, Low G H and Wiebe N 2019 *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019, Phoenix, AZ, USA, pp. 193–204
[42] Kerenidis I and Prakash A 2016 arXiv: 1603.08675[quant-ph]
[43] Nguyen Q T, Kiani B T and Lloyd S 2022 *Quantum* **6** 876
[44] Chakraborty S, Morolia A and Peduri A 2023 *Quantum* **7** 988
[45] Shao C and Liu J P 2022 *Proc. R. Soc. A* **478** 20210797
[46] Buhrman H, Cleve R, Watrous J and De Wolf R 2001 *Phys. Rev. Lett.* **87** 167902
[47] Li Y M, Liu H L, Pan S J, Qin S J, Gao F and Wen Q Y 2023 *Quantum Inf. Process.* **22** 163
[48] Li Y, Zhou R G, Xu R, Hu W and Fan P 2020 *Quantum Sci. Technol.* **6** 014001
[49] https://www.ibm.com/quantum/qiskit
[50] Hearst M A, Dumais S T, Osuna E, Platt J and Scholkopf B 1998 *IEEE Intell. Syst. Appl.* **13** 18
[51] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits
[52] Bharti K, Cervera-Lierta A, Kyaw T H, Haug T, Alperin-Lea S, Anand A, Degroote M, Heimonen H, Kottmann J S, Menke T, *et al.* 2022 *Rev. Mod. Phys.* **94** 015004
[53] Jerbi S, Fiderer L J, Poulsen Nautrup H, Kübler J M, Briegel H J and Dunjko V 2023 *Nat. Commun.* **14** 517
[54] Schuld M, Bocharov A, Svore K M and Wiebe N 2020 *Phys. Rev. A* **101** 032308