New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
25 January 2016

REVISED
7 June 2016

ACCEPTED FOR PUBLICATION
10 June 2016

PUBLISHED
6 July 2016

PAPER

Quantum discriminant analysis for dimensionality reduction and classification

Iris Cong[1,2,4] **and Luming Duan**[2,3]

[1] Department of Computer Science, University of California, Los Angeles, CA 90095, USA
[2] Center for Quantum Information, IIIS, Tsinghua University, Beijing 100084, People's Republic of China
[3] Department of Physics, University of Michigan, Ann Arbor, Michigan 48109, USA
[4] Author to whom any correspondence should be addressed.

E-mail: iriscong23@ucla.edu **and** lmduan@umich.edu

Keywords: quantum information, machine learning, discriminant analysis, quantum computation, dimensionality reduction, quantum algorithms

Abstract

We present quantum algorithms to efficiently perform discriminant analysis for dimensionality reduction and classification over an exponentially large input data set. Compared with the best-known classical algorithms, the quantum algorithms show an exponential speedup in both the number of training vectors M and the feature space dimension N. We generalize the previous quantum algorithm for solving systems of linear equations (2009 *Phys. Rev. Lett.* **103** 150502) to efficiently implement a Hermitian chain product of k trace-normalized $N \times N$ Hermitian positive-semidefinite matrices with time complexity of $O(\log(N))$. Using this result, we perform linear as well as nonlinear Fisher discriminant analysis for dimensionality reduction over M vectors, each in an N-dimensional feature space, in time $O(p\ \mathrm{polylog}(MN)/\epsilon^3)$, where ϵ denotes the tolerance error, and p is the number of principal projection directions desired. We also present a quantum discriminant analysis algorithm for data classification with time complexity $O(\log(MN)/\epsilon^3)$.

1. Introduction

With the rise in the fields of big data analysis and machine learning in the modern era, techniques such as dimensionality reduction and classification have gained significant importance in the information sciences. In machine learning and statistical analysis problems, when input vectors are given in an extremely large feature space, it is often necessary to reduce the data to a more manageable dimension/size before manipulation or classification. One classical example is in the problem of face recognition [8, 9], where the size of the feature space is determined by a huge number of pixels representing each face. More recent applications are also seen in fields of medical imaging. For instance, [11] shows the necessity for dimensionality reduction in diagnosing cases of liver cirrhosis. Also, [12] shows the importance of dimensionality reduction for early Alzheimer's disease detection.

One widely used technique for dimensionality reduction is principal components analysis (PCA), where the data is projected onto the directions of maximal variance. However, a significant disadvantage of PCA is that it looks only at the overall data variance, and does not consider the class data. The extreme example of this would occur if the overall data variance is in exactly the same direction as the maximal within-class data variance, but orthogonal to the direction of maximal between-class data variance. In such a case, it is possible for a PCA projection to completely overlap the data from different classes, making it impossible to use the projected data to perform future discriminations. Fisher's linear discriminant analysis (LDA) is a technique developed to overcome this problem by instead projecting the data onto directions that maximize the between-class variance, while minimizing the within-class variance of the training data. It is hence not surprising that LDA is shown to be more effective than PCA in machine learning problems involving dimensionality reduction before classification [8, 9].

Another common application of discriminant analysis is to use it as a classifier itself, where labeled training vectors are presented as input and new cases must be efficiently assigned to their respective classes. The discriminant analysis classifier has recently been used in medical analysis, such as in analyzing electromyography (EMG) signals [13, 14], lung cancer classification [15], and breast cancer diagnosis [16]. While other algorithms such as the support vector machine (SVM) reach similar accuracy rates as the discriminant analysis classifier [17, 18], studies [18] show that discriminant analysis is a significantly more stable model. This is because the separating hyperplane chosen by the SVM can depend only on a few support vectors, subjecting it to high variance in the case of training vector errors. On the other hand, since discriminant analysis bases its classification on the entire class means and variances, it tends to show less fluctuation and is potentially more robust in the presence of error.

A significant drawback of discriminant analysis in both dimensionality reduction and classification is the time complexity. Even the best existing classical algorithms for LDA dimensionality reduction require time $O(Ms)$ [7], where M is the number of training vectors given, and s is the sparseness (maximum number of nonzero components) of each feature vector. For large data sets in high dimensions, this can scale rather poorly, especially since it is often hard to guarantee the sparseness of training vectors. While quantum algorithms have been designed to exponentially speed up PCA to be polylogarithmic in the number of input vectors and their dimension [5], no such work has been done yet to speed up LDA. In section 3, we provide a quantum algorithm for LDA polylogarithmic in both M, the number of training vectors, and N, the initial feature space dimension, regardless of the sparseness of the training vectors.

Similarly, while a quantum algorithm has been presented to provide exponential speedup in SVM classification [3], no work has been done yet for the discriminant analysis classifier. The best known algorithms for the classical discriminant analysis classifier also require time polynomial in M and N (see section 2 below), which again can scale rather poorly. In section 4, we provide a quantum algorithm for discriminant analysis classification logarithmic in both the number of input vectors M and the dimension N.

This paper is arranged as follows: in section 2 we briefly review the classical discriminant analysis algorithms for dimensionality reduction and data classification. In sections 3 and 4, we present our major results of quantum discriminant analysis algorithms for dimensionality reduction as well as classification. The detailed proof of theorem 1 in section 3, which is about efficient quantum implementation of a Hermitian chain product of k trace-normalized $N \times N$ Hermitian positive-semidefinite matrices, is included in the appendix.

2. Review of classical discriminant analysis

2.1. Dimensionality reduction

The classical LDA dimensionality reduction algorithm is designed to return the directions of projection that maximize the between-class variance (for class discrimination), but minimize the within-class variance. With this result, in big data problems as listed in section 1, the vectors in a high-dimensional feature space may be projected onto a lower-dimensional subspace (spanned by the returned optimal unit vectors), so that less resources may be used to store the same amount of information. Suppose we are given M (real-valued) input data vectors $\{x_i \in \mathbb{R}^N: 1 \leqslant i \leqslant M\}$ each belonging to one of k classes. Let μ_c denote the within-class mean (centroid) of class c, and $\overline{x}$ denotes the mean/centroid of all data points x. Furthermore, let

$$S_B = \sum_{c=1}^{k} (\mu_c - \overline{x})(\mu_c - \overline{x})^{\mathrm{T}} \tag{1}$$

denote the between-class scatter matrix of the dataset, and let

$$S_W = \sum_{c=1}^{k} \sum_{x \in c} (x - \mu_c)(x - \mu_c)^{\mathrm{T}} \tag{2}$$

denote the within-class scatter matrix. The goal is then to find a direction of projection $w \in \mathbb{R}^N$ that maximizes the between-class variance $w^{\mathrm{T}} S_B w$ relative to the within-class variance $w^{\mathrm{T}} S_W w$. Mathematically, assuming that the classes have approximately multivariate Gaussian distribution with similar covariances, this is the problem of maximizing the objective function (commonly known as *Fisher*'s discriminant)

$$J(w) = \frac{w^{\mathrm{T}} S_B w}{w^{\mathrm{T}} S_W w}. \tag{3}$$

Since the expression for $J(w)$ is invariant under constant rescaling of w, it is clear that the maximization problem given in (3) is equivalent to the optimization problem

$$\min_{w} \; - w^{\mathrm{T}} S_B w, \tag{4}$$

$$\text{subject to } w^{\mathrm{T}} S_W w = 1. \tag{5}$$

We are thus minimizing the Lagrangian [1]

$$\mathcal{L}_P = -w^{\mathrm{T}} S_B w + \lambda (w^{\mathrm{T}} S_W w - 1), \tag{6}$$

where λ is the desired Lagrange multiplier. By the Karush–Kuhn–Tucker conditions [27], this means that

$$S_W^{-1} S_B w = \lambda w. \tag{7}$$

It follows that w is an eigenvector of $S_W^{-1} S_B$. By plugging (7) back into the objective function $J(w)$, we get $J(w) = \lambda$. Hence, we choose w to be the principal eigenvector.

The above procedure generalizes easily to higher-dimensional projection subspaces. In this case, we seek p vectors which form a basis for our projection subspace; this corresponds to maximizing the discriminant

$$J(W) = \frac{W^{\mathrm{T}} S_B W}{W^{\mathrm{T}} S_W W},$$

where W is the $N \times p$ matrix whose columns are the basis vectors. Using the same analysis as above, one can show that the columns of W will be the eigenvectors corresponding to the p largest eigenvalues of $S_W^{-1} S_B$, as in the case of PCA.

2.2. Classification

Although its most widely used application is probably in dimensionality reduction, discriminant analysis is also commonly used to directly perform data classification. For classification, one constructs the *discriminant functions* for each class c

$$\delta_c(x) = x^{\mathrm{T}} \Sigma_c^{-1} \mu_c - \frac{1}{2} \mu_c^{\mathrm{T}} \Sigma_c^{-1}, \mu_c + \log \pi_c, \tag{8}$$

where Σ_c is the covariance matrix for class c, μ_c is the class mean for c as before, and π_c is the prior probability for classifying into class c [19]. Given a vector x, it is then classified into the class $c = \mathrm{argmax}_c \delta_c(x)$. From the training vector data, if M_c is the number of training vectors belonging to class c, we can approximate $\pi_c = M_c/M$ for simplicity, i.e. the probability of classifying to a certain class c is directly proportional to the fraction of training vectors belonging to c [19]. Assuming multivariate Gaussian distributions for each class, we also estimate

$$\Sigma_c = \frac{1}{M_c - 1} \sum_{x \in c} (x - \mu_c)(x - \mu_c)^{\mathrm{T}}. \tag{9}$$

Note that in the special case where the covariance matrices are all approximately equal (i.e. $\Sigma_c \approx \Sigma \; \forall \, c$), Σ is proportional to the scatter matrix S_W given by equation (2). In this special case, the functions δ_c are known as *linear discriminant functions*. In our paper, we present a quantum algorithm to solve the more general case, known as *quadratic* discriminant analysis (QDA), in time logarithmic in both the number of input vectors M and their dimension N. Our algorithm will be easily applicable to the special case of LDA classification. This provides exponential speedup over the fastest existing algorithms, since the classical construction/inversion of Σ_c to evaluate the discriminant functions must require time polynomial in both M and N.

3. Quantum LDA algorithm: dimensionality reduction

3.1. Assumptions and initialization

The quantum PCA algorithm of [5] presents methods for processing input vector data if the covariance matrix of the data is efficiently obtainable as a density matrix, under specific assumptions about the vectors given in quantum mechanical form. While our major contributions are also in the processing mechanisms of the within- and between-class covariance matrices, we will describe how to obtain this density matrix under certain assumptions about the input data, like in [2, 4, 5].

In our algorithm, similar to the assumptions made in [2, 4, 5], we will assume we have quantum access to the training vector data in a quantum RAM (as described in [10]). We will assume that each training vector is stored in the quantum RAM in terms of its difference from the class means. That is, if a training vector x_j belongs to class c_j with centroid μ_{c_j}, we have the Euclidean norm and complex-valued components of the difference vector $d_j = x_j - \mu_{c_j}$ stored as floating-point numbers in quantum RAM in polar form (alternatively, if the input is presented directly as the training vectors x_j and the class means μ_c, we may first perform a component-wise subtraction of the given floating-point numbers, by [26]). Following the methodology of [3–5], we will assume the following oracle:

$$\mathcal{O}_1(|j\rangle|0\rangle|0\rangle|0\rangle) \rightarrow |j\rangle|\|x_j - \mu_{c_j}\|\rangle|x_j - \mu_{c_j}\rangle|c_j\rangle$$

to get the jth training vector and its class c_j, where $|x_j - \mu_{c_j}\rangle$ has already been normalized to one. Similarly, we also assume that we are given the quantum representations of the class centroids $|\mu_c\rangle$, in terms of their differences from the overall training vector mean $|\overline{x}\rangle$. That is, if $D_c = \mu_c - \overline{x}$, we assume the oracle

$$\mathcal{O}_2(|c\rangle|0\rangle|0\rangle) \rightarrow |c\rangle|\|\mu_c - \overline{x}\|\rangle|\mu_c - \overline{x}\rangle,$$

where we similarly assume that $|\mu_c - \overline{x}\rangle$ has been normalized to one. These oracles could, as an example, be realizable if the input data is presented in this form as the output of a preceding quantum system, or if the vector components are presented as floating-point numbers in the quantum RAM, and the sub-norms of the vectors can be estimated efficiently [4, 28–30]. The oracles $\mathcal{O}_1$ and $\mathcal{O}_2$ allow us to construct density matrices proportional to S_B and S_W as follows: let

$$|\Psi_1\rangle = \mathcal{O}_2\left(\frac{1}{\sqrt{k}}\sum_{c=1}^{k}|c\rangle|0\rangle|0\rangle\right) = \frac{1}{\sqrt{k}}\sum_{c=1}^{k}|c\rangle|\|\mu_c - \overline{x}\|\rangle|\mu_c - \overline{x}\rangle, \tag{10}$$

$$|\Phi_1\rangle = \mathcal{O}_1\left(\frac{1}{\sqrt{M}}\sum_{j=1}^{M}|j\rangle|0\rangle|0\rangle|0\rangle\right) = \frac{1}{\sqrt{M}}\sum_{j=1}^{M}|j\rangle|\|x_j - \mu_{c_j}\|\rangle|x_j - \mu_{c_j}\rangle. \tag{11}$$

By [4, 28–30], if the norms of the vectors form an efficiently integrable distribution, we can obtain the states

$$|\Psi_2\rangle = \frac{1}{\sqrt{A}}\sum_{c=1}^{k}\|\mu_c - \overline{x}\||c\rangle|\|\mu_c - \overline{x}\|\rangle|\mu_c - \overline{x}\rangle, \tag{12}$$

$$|\Phi_2\rangle = \frac{1}{\sqrt{B}}\sum_{j=1}^{M}\|x_j - \mu_{c_j}\||j\rangle|\|x_j - \mu_{c_j}\|\rangle|x_j - \mu_{c_j}\rangle|c_j\rangle, \tag{13}$$

where $A = \sum_{c=1}^{k}\|\mu_c - \overline{x}\|^2$, $B = \sum_{j=1}^{M}\|x_j - \mu_{c_j}\|^2$.

In both cases, we now take the partial trace over the first register. Then, for the state of equation (12), the density matrix of the second register [5] is given by

$$S_B = \frac{1}{A}\sum_{c=1}^{k}\|\mu_c - \overline{x}\|^2|\mu_c - \overline{x}\rangle\langle\mu_c - \overline{x}| \tag{14}$$

and for the state of equation (13), the density matrix of the second register is given by

$$S_W = \frac{1}{B}\sum_{c=1}^{k}\sum_{i\in c}\|x_i - \mu_c\|^2|x_i - \mu_c\rangle\langle x_i - \mu_c|. \tag{15}$$

Assuming our oracles $\mathcal{O}_1$ and $\mathcal{O}_2$, we can hence construct the Hermitian operators S_B, S_W in time $O(\log(MN))$.

3.2. LDA approach

Having initialized the means and operators S_B, S_W, our main task will be to solve the eigenvector problem of (7). This problem would be simpler if only $S_W^{-1}S_B$ were Hermitian positive semidefinite. However, we can reduce this problem to an eigenvalue problem for a Hermitian density matrix: specifically, since S_B is Hermitian positive semidefinite, letting $w = S_B^{-1/2}v$ reduces (7) to the eigenvalue problem [1]

$$S_B^{1/2}S_W^{-1}S_B^{1/2}v = \lambda v. \tag{16}$$

To apply the quantum phase estimation algorithm to solve (16), we must first be able to construct the density matrix $S_B^{1/2}S_W^{-1}S_B^{1/2}$. In the following section, we present a more general theorem that can be applied to construct this density matrix.

3.3. Implementing the Hermitian chain product

In this section, we state a theorem to construct the density matrix corresponding to the Hermitian chain product

$$[f_k(A_k)...f_1(A_1)][f_k(A_k)...f_1(A_1)]^{\dagger} \tag{17}$$

to error ϵ, for arbitrary normalized $N \times N$ Hermitian positive semidefinite matrices $A_1,...A_k$, and functions $f_1,...f_k$ with convergent Taylor series. The derivation of this theorem follows the method presented in [2], and is presented in appendix A.

Theorem 1. *Let $A_1, \ldots A_k$ be k normalized Hermitian positive semidefinite matrices whose quantum forms can be constructed in time $O(\log(N))$ (e.g., by visits to a quantum RAM) and let $f_1, \ldots f_k$ be k functions with convergent Taylor series. Let $\{\lambda_{jl}\}_{l=1}^{N}$ denote the eigenvalues of matrix A_j. Then the Hermitian operator in equation (17) can be*

implemented in time

$$O\left(\frac{\log(N)}{\epsilon^3}\cdot\sum_{j=1}^{k}\kappa_j^2\cdot\left(\frac{\max_l|f_1(\lambda_{1l})|}{\min_l|f_1(\lambda_{1l})|}\right)\prod_{j=2}^{k}\left(\frac{\max_l|f_j(\lambda_{jl})|}{\min_l|f_j(\lambda_{jl})|}\right)^2\right), \tag{18}$$

where κ_j is the condition number for matrix A_j, i.e. the ratio of the largest to smallest eigenvalue of A_j. More generally, if construction of each matrix A_j takes time $O(X)$, the operator can be implemented in time

$$O\left(\frac{X}{\epsilon^3}\cdot\sum_{j=1}^{k}\kappa_j^2\cdot\left(\frac{\max_l|f_1(\lambda_{1l})|}{\min_l|f_1(\lambda_{1l})|}\right)\prod_{j=2}^{k}\left(\frac{\max_l|f_j(\lambda_{jl})|}{\min_l|f_j(\lambda_{jl})|}\right)^2\right). \tag{19}$$

We note that this provides exponential speedup over classical algorithms, as the optimal classical algorithm for multiplication of non-sparse $N \times N$ matrices requires time $O(N^{2.3737})$ [31].

3.4. Finding the principal eigenvectors

For LDA, we apply the theorem presented in the previous section to obtain the matrix product $S_B^{1/2}S_W^{-1}S_B^{1/2}$. Specifically, we use $A_1 = S_W$, $A_2 = S_B$, $f_1(X) = X^{-1/2}$, and $f_2(X) = X^{1/2}$. To avoid exponential complexity in the case of exponentially small eigenvalues, we adopt a technique used in [3] by pre-defining an effective condition number κ_{eff} and taking into account only eigenvalues in the range $[1/\kappa_{\text{eff}},\ 1]$ for phase estimation. (typically, one may take $\kappa_{\text{eff}} = O(1/\epsilon)$, because κ_{eff} is a limit to the amount of eigenvalues considered in phase estimation, which should be proportional to the error tolerance). By our initialization procedures, preparation of S_B and S_W take time $O(\log(MN))$, and by definition of f_1, f_2, and κ_{eff}

$$\frac{\max_l|f_1(\lambda_{1l})|}{\min_l|f_1(\lambda_{1l})|} = \frac{\max_l|f_2(\lambda_{2l})|}{\min_l|f_2(\lambda_{2l})|} = \kappa_{\text{eff}}^{1/2}.$$

Hence, by (19) we can obtain $S_B^{1/2}S_W^{-1}S_B^{1/2}$ in time $O(\log(MN)\kappa_{\text{eff}}^{3.5}/\epsilon^3)$. Using the matrix exponentiation technique presented in [5], we can then apply quantum phase estimation to obtain an approximation to the state

$$\rho = \sum_i \lambda_i|v_i\rangle\langle v_i| \otimes |\lambda_i\rangle\langle\lambda_i|,$$

where λ_i and v_i are the eigenvalues and eigenvectors of $S_B^{1/2}S_W^{-1}S_B^{1/2}$. If the p principal (largest) eigenvalues are polynomially small, sampling produces the corresponding p principal eigenvectors v_r of $S_B^{1/2}S_W^{-1}S_B^{1/2}$ in time $O(p\ \text{polylog}(MN)/\epsilon^3)$ [22]. (If all eigenvalues are indeed super-polynomially small, there are typically no suitable directions for discriminant analysis: all directions would be essentially the same in preserving between-class versus within-class data). Finally, having solved the eigenvalue problem of (16), we again use the technique of the previous section to obtain the eigenvectors

$$w_r = S_B^{-1/2}v_r \tag{20}$$

of $S_W^{-1}S_B$. After obtaining these principal eigenvectors, the data can be projected onto the dimensions of maximal between-class variance and minimal within-class variance. At this stage, one has effectively performed dimensionality reduction, and can now easily manipulate the data with existing tools, e.g. a classifier (see [3] or section 4 below).

Algorithm 1. Quantum LDA dimensionality reduction.

Step 1: Initialization. By querying the quantum RAM/oracles, construct the Hermitian positive semidefinite operators S_B and S_W as given by equations (14) and (15) in time $O(\log(MN))$.

Step 2: Since S_B and S_W are Hermitian positive semidefinite, use the method of [5] to exponentiate these operators. Apply the generalized matrix chain algorithm of theorem 1 to implement $S_B^{1/2}S_W^{-1}S_B^{1/2}$ in time $O(\log(MN)\kappa_{\text{eff}}^{3.5}/\epsilon^3)$.

Step 3: Since $S_B^{1/2}S_W^{-1}S_B^{1/2}$ is Hermitian positive semidefinite, use the method of [5] to exponentiate this operator. Use quantum phase estimation methods and sample from the resulting probabilistic mixture to then obtain the p principal eigenvalues/eigenvectors v_r, in time $O(p\ \text{polylog}(MN)/\epsilon^3)$.

Step 4: Apply $S_B^{-1/2}$ to the v_r's (by the algorithm of theorem 1) to get desired directions $w_r = S_B^{-1/2}v_r$, in time $O(p\log(MN)\kappa_{\text{eff}}^3/\epsilon^3)$.

Step 5: Project data onto the w_r's for dimensionality reduction, or otherwise work in the directions of maximal class discrimination.

3.5. Algorithmic complexity for dimensionality reduction

Algorithm 1 above shows the pseudocode for our LDA algorithm. Step 1 (initialization) takes time $O(\log(MN))$ with our quantum oracles. Implementing the operator $S_B^{1/2}S_W^{-1}S_B^{1/2}$ takes time $O(\log(MN)\kappa_{\text{eff}}^{3.5}/\epsilon^3)$, and

finding its principal eigenvectors then takes $O(\log(MN)/\epsilon^3)$. Finally, Step 4 takes time $O(\log(MN)\kappa_{\text{eff}}^3/\epsilon^3)$ to apply $S_B^{-1/2}$ to the v_r's and obtain the eigenvectors of $S_W^{-1}S_B$. Hence, we can add these to get the total runtime:

Theorem 2. *The quantum LDA algorithm presented in this paper (with pseudocode given by algorithm 1) can be implemented in time polylogarithmic in both the number of input vectors M and the input vector dimension N. Specifically, it has a runtime of*

$$O(p\ \text{polylog}(MN)\kappa_{\text{eff}}^{3.5}/\epsilon^3), \tag{21}$$

where κ_{eff} is a pre-defined condition number restricting the range of eigenvalues considered for phase estimation, typically taken to be $O(1/\epsilon)$, and p is the number of principal eigenvectors.

Note that the complexity presented here in (21) is polylogarithmic in both the number of input vectors M and their dimension N, regardless of training vector sparseness.

3.6. Nonlinear/kernel FDA

Certainly, in many real-world cases, a straightforward linear discriminant may not be sufficient. Classically, a simple generalization is known as kernel FDA, where the input vectors are first mapped (nonlinearly) by a function $\phi : x_j \to \phi(x_j)$ into a higher-dimensional feature space $\mathcal{F}$. The linear discriminant corresponding to $J(w)$ in the feature space then becomes nonlinear in the original space. In the classical case, if the dimension of $\mathcal{F}$ is too large, it becomes computationally infeasible to perform operations such as matrix multiplication or exponentiation on the resulting large covariance matrices S_B^ϕ and S_W^ϕ. Instead, one must find workarounds such as kernel methods, and perform reductions so that the algorithm involves only dot products in the feature space [21], but this may seriously limit the potential choices of mapping ϕ. In the quantum case, however, we can directly perform the LDA analysis in the higher-dimensional feature space. As long as the dimension of $\mathcal{F}$ scales polynomially with the original input dimension, our algorithmic performance will be affected only by a constant factor. This allows for a much wider range of mappings ϕ into the feature space.

4. QDA algorithm for classification

4.1. Algorithm

We now present an efficient quantum algorithm for the classification of an exponentially large data set by QDA, and our results can easily be applied to perform LDA classification. As before, we assume that the class means and training vector data are given with their norms and components stored as floating-point numbers in quantum RAM. We again assume the oracle $\mathcal{O}_2$ to obtain the class means of the training vectors μ_c and their norms, and in this section, we further assume that we can obtain the jth training vector of each class. As in section 3 initialization or [3–5], our oracle gives the vectors $x_{c,j}$ are given as their difference from their class means (as in section 3, this may also be obtained from the stored floating-point numbers if necessary). Specifically, we assume the oracle

$$\mathcal{O}_3(|c\rangle|j\rangle|0\rangle|0\rangle) \to |c\rangle|j\rangle|\|x_{c,j}-\mu_c\|\rangle|x_{c,j}-\mu_c\rangle.$$

As in section 3, $|x_{c,j}-\mu_c\rangle$ has been normalized to one. Finally, we now assume that for each class c, we are given the number of training vectors M_c belonging to that class.

For QDA, we use the oracles to construct for each class c the Hermitian positive semidefinite operator

$$\Sigma_c = \frac{1}{A_c}\sum_{j=1}^{M_c}\|x_{c,j}-\mu_c\|^2|x_{c,j}-\mu_c\rangle\langle x_{c,j}-\mu_c|. \tag{22}$$

where $A_c = \sum_{j=1}^{M_c}\|x_{c,j}-\mu_c\|^2$. To do this, we first call $\mathcal{O}_3$ on the state $\frac{1}{\sqrt{M_c}}\sum_{j=1}^{M_c}|c\rangle|j\rangle|0\rangle|0\rangle$ to obtain the register $\frac{1}{\sqrt{M_c}}\sum_{j=1}^{M_c}|c\rangle|j\rangle|\|x_{c,j}-\mu_c\|\rangle|x_{c,j}-\mu_c\rangle$. As in the initialization procedure of section 3, we use the methods of [4, 28–30] to obtain the state $|\chi_c\rangle = \frac{1}{\sqrt{A_c}}\sum_{j=1}^{M}\sum_{j=1}^{M_c}\|x_{c,j}-\mu_c\||c\rangle|j\rangle|\|x_{c,j}-\mu_c\|\rangle|x_{c,j}-\mu_c\rangle$. Tracing over $|j\rangle$ in the outer product $|\chi_c\rangle\langle\chi_c|$ then yields the operator Σ_c in time $O(\log(MN))$.

Given an input vector $|x\rangle$ in quantum form, we now present a method to find the maximum among the k discriminant functions $\delta_c(x)$ given in equation (8). First, we apply Σ_c^{-1} to the class mean $|\mu_c\rangle$, using the matrix inversion algorithm of [2]. As before, to avoid exponential complexity with small eigenvalues, we introduce the pre-defined effective condition number κ_{eff} to limit the range of considered eigenvalues. By [2], we can thus construct the state

$$|\Sigma_c^{-1}\mu_c\rangle \tag{23}$$

for each class c in time $O(\log(MN)\kappa_{\text{eff}}^3/\epsilon^3)$. Next, recognizing the first two terms of the discriminant function of (8) as an inner product, we perform a SWAP test [32] on the states $|\Sigma_c^{-1}\mu_c\rangle$ and $|x - \frac{1}{2}\mu_c\rangle$ to obtain the value

$$x^{\mathrm{T}}\Sigma_c^{-1}\mu_c - \frac{1}{2}\mu_c^{\mathrm{T}}\Sigma_c^{-1}\mu_c. \tag{24}$$

This inner product evaluation requires time $O(\log(N))$. Finally, for each class c, we add to the value in (24) the class prior $\pi_c = M_c/M$. We hence obtain the discriminant values $\delta_c(x)$ for all of the k classes in time $O(k\log(MN)\kappa_{\text{eff}}^3/\epsilon^3)$. It is then straightforward to identify the class yielding the maximum discriminant, to which the input vector is then classified. Note that our algorithm can be easily applied to perform quantum LDA classification, by using an operator proportional to the scatter matrix S_W (see section 3, initialization) in place of Σ_c for each class.

Algorithm 2. Quantum discriminant analysis classifier.

Step 1: Initialization. By querying the quantum RAM or oracles, construct the Hermitian positive semidefinite operators Σ_c in time $O(k\log(MN))$ for all classes c.

Step 2: Since Σ_c is Hermitian positive semidefinite, use the method of [5] to exponentiate this operator. Apply the inversion algorithm of [2] on the state $|\mu_c\rangle$ for each class to construct the states given by equation (23) in time $O(k\log(MN)\kappa_{\text{eff}}^3/\epsilon^3)$.

Step 3: Take the inner product of $\Sigma_c^{-1}\mu_c$ with $x - \frac{1}{2}\mu_c$ using the SWAP test, yielding the value in equation (24). This step requires time $O(\log N)$.

Step 4: For each class c, add the class prior $\pi_c = M_c/M$ to the value in (24) to obtain the final discriminant value $\delta_c(x)$.

Step 5: Select the class c yielding the maximum discriminant value, and classify x accordingly in time $O(k)$.

4.2. Algorithmic complexity for classification

Algorithm 2 above shows the pseudocode for our quantum QDA classifier. Step 1 (initialization) takes time $O(k\log(MN)/\epsilon)$. Applying the inversion algorithm of [2] for each class then takes time $O(k\log(MN)\kappa_{\text{eff}}^3/\epsilon^3)$. Computing all of the inner products given by (24) requires time $O(k\log N)$, and adding the class priors requires time $O(k)$. Finally, selecting the class with maximum discriminant takes time $O(k)$. We add these to give the overall complexity below:

Theorem 3. *The quantum QDA classifier algorithm presented in this paper (with pseudocode given by algorithm 2) can be implemented in time logarithmic in both the number of input vectors M and the input vector dimension N. Specifically, it has a runtime of*

$$O(k\log(MN)\kappa_{\text{eff}}^3/\epsilon^3), \tag{25}$$

where k is the number of classes for classification, and κ_{eff} is a pre-defined condition number restricting the range of eigenvalues considered for phase estimation, typically taken to be $O(1/\epsilon)$.

Note that the complexity presented in equation (25) is logarithmic in both M and N regardless of training vector sparseness.

5. Discussion

In this paper, we have presented a generalized algorithm from [2] for implementing Hermitian matrix chain operators, and applied it to implement an algorithm for quantum LDA in polylogarithmic time. As demonstrated by classical works such as [8, 9], LDA is a powerful tool for dimensionality reduction in fields such as machine learning and big data analysis. Although our performance in terms of error ϵ is poorer than classical algorithms (polynomial instead of logarithmic in $1/\epsilon$), we believe that this is acceptable, since it is unlikely that someone desiring extreme levels of precision will wish to perform significant dimensionality reduction like that provided by LDA. Rather, we believe that the exponential speedup in terms of the parameters M and N should be more significant in reducing the overall algorithmic runtime.

Our work has also presented a quantum algorithm providing exponential speedup for the LDA and QDA classifiers. As classical studies [17, 18] have shown, these classifiers typically perform just as well in terms of accuracy as the SVM (for which a quantum algorithm has been developed, [3]). However, they tend to have much better model stability [18], which can make them more robust in face of training data errors. Finally, discriminant analysis methods are much simpler when generalizing to multi-class classification, whereas the SVM is more suited for binary classification [23]. In conclusion, this work has provided efficient exponential speedup for two important algorithms for dimensionality reduction and classification in big data analysis.

Acknowledgments

We thank the Institute of Interdisciplinary Information Sciences (IIIS) of Tsinghua University for hospitality to host our visit during which this work is done. LMD acknowledges in addition support from the IARPA MUSIQC program, the AFOSR and the ARO MURI program.

Appendix A. Proof of theorem 1

In this appendix, we present the derivation of equation (19) from theorem 1 in section 3. The proof of the theorem closely follows the matrix inversion algorithm presented in [2] (referred to in the following as the HHL algorithm). The HHL algorithm begins with the initial state

$$|\psi_0\rangle =: \sum_{\tau=0}^{T-1} \sin\frac{\pi(\tau+1/2)}{T}|\tau\rangle \tag{26}$$

for large T (see [2] for more details on original algorithm, we make a sketch below). Since the original algorithm was designed to apply the inverse of a matrix A on a specific vector $|b\rangle$, HHL considers the state $|\psi_0\rangle \otimes |b\rangle$. Here, we are interested instead in obtaining an operator for (17), so we use the density operator $\rho_0 = \frac{1}{\sqrt{N}}\sum_{i=1}^{N}|i\rangle\langle i|$ (proportional to the identity) in place of $|b\rangle$, and we use $|\psi_0\rangle\langle\psi_0|$ in place of $|\psi_0\rangle$.

Following HHL, decompose ρ_0 in the eigenvector basis using phase estimation. Denote the eigenvectors of A_1 by $\{|u_{1l}\rangle\}_{l=1}^{N}$, and let $\{\lambda_{1l}\}_{l=1}^{N}$ be the corresponding eigenvalues. Then, we write

$$\rho_0 = \sum_{l,l'=1}^{N} \beta_{ll'}|u_{1l}\rangle\langle u_{1l'}|. \tag{27}$$

Quantum phase estimation is then applied on ρ_0 for time $t_0 = O(\kappa_1/\epsilon)$ to obtain the state

$$\rho_0' \approx \sum_{l,l'=1}^{N} \beta_{ll'}|\lambda_{1l}\rangle\langle\lambda_{1l'}||u_{1l}\rangle\langle u_{1l'}| \tag{28}$$

up to a tolerance error ϵ (κ_1 is the condition number of the matrix A_1, or the ratio of the largest to smallest eigenvalue). In this step, the exponentiation of the Hermitian operator A_1 is performed using the trick presented in [5]. By [5], $n = O(\kappa_1^2/\epsilon^3)$ copies of A_1 are required to perform the phase estimation to error ϵ, so if A_1 can be constructed in time $O(X)$, this step requires time $O(nX) = O(X\kappa_1^2/\epsilon^3)$.

HHL then add an ancilla and perform a unitary controlled on the eigenvalue register. Here, we generalize this step to a controlled rotation from $|0\rangle\langle 0|$ to the state $|\psi_{\lambda_{1l}}\rangle\langle\psi_{\lambda_{1l'}}|$, where

$$|\psi_{\lambda_{1l}}\rangle = \sqrt{1 - C^2 f_1(\lambda_{1l})^2}\,|0\rangle + Cf_1(\lambda_{1l})|1\rangle \tag{29}$$

and C is a constant of order $O(\min_l\{|f_1(\lambda_{1l})|^{-1}\})$ for normalization. Assuming f_1 has a convergent Taylor series, it is possible to efficiently perform this rotation using controlled gates (see appendix B). This results in the overall state

$$\rho_0'' = \sum_{l,l'=1}^{N} \beta_l\beta_{l'}|\lambda_{1l}\rangle\langle\lambda_{1l'}||u_{1l}\rangle\langle u_{1l'}||\psi_{\lambda_{1l}}\rangle\langle\psi_{\lambda_{1l'}}|. \tag{30}$$

Next, following HHL, we undo phase estimation to uncompute the eigenvalue register, resulting in the state

$$\rho_0''' = \sum_{l,l'=1}^{N} \beta_l\beta_{l'}|u_{1l}\rangle\langle u_{1l'}||\psi_{\lambda_{1l}}\rangle\langle\psi_{\lambda_{1l'}}|. \tag{31}$$

Finally, as in HHL, measure the ancilla to be $|1\rangle\langle 1|$. By choice of C, this success probability is easily seen to be[5]

$$\Omega\left(\frac{\min_l |f_1(\lambda_{1l})|^2}{\max_l |f_1(\lambda_{1l})|^2}\right). \tag{32}$$

This produces the density operator proportional to $\rho_1 = f_1(A_1)\mathbf{I}(f_1(A_1))^\dagger$ in runtime

$$O\left(\frac{X}{\epsilon^3}\cdot\kappa_1^2\cdot\frac{\max_l |f_1(\lambda_{1l})|^2}{\min_l |f_1(\lambda_{1l})|^2}\right).$$

To generalize this to the entire matrix chain, we can simply repeat this algorithm with A_2 as the matrix, f_2 as the function, and start with the state ρ_1 instead of ρ_0. More generally, at each iteration j, use A_j, f_j on ρ_{j-1}. At each

[5] As in complexity analysis, the notation $F(x) = \Omega(G(x))$ is used in place of the expression '∃a constant c such that $F(x) \geqslant cG(x)$ for all x'.

step, $n = O(\kappa_j^2/\epsilon^3)$ copies of A_j are required, and the probability of success in measuring the ancilla is given by the expression analogous to equation (32). Hence, for k matrices, the Hermitian operator in equation (17) can be implemented in time

$$O\left(\frac{X}{\epsilon^3}\cdot\sum_{j=1}^{k}\kappa_j^2\cdot\prod_{j=1}^{k}\left(\frac{\max_l|f_j(\lambda_{jl})|}{\min_l|f_j(\lambda_{jl})|}\right)^2\right). \tag{33}$$

By [2], amplitude amplification can be used on the first matrix A_1 only to increase the measurement success probability from $\Omega\left(\frac{\min_l|f_j(\lambda_{jl})|^2}{\max_l|f_j(\lambda_{jl})|^2}\right)$ to $\Omega\left(\frac{\min_l|f_j(\lambda_{jl})|}{\max_l|f_j(\lambda_{jl})|}\right)$. This reduces the complexity slightly to

$$O\left(\frac{X}{\epsilon^3}\cdot\sum_{j=1}^{k}\kappa_j^2\cdot\frac{\max_l|f_1(\lambda_{1l})|}{\min_l|f_1(\lambda_{1l})|}\cdot\prod_{j=2}^{k}\left(\frac{\max_l|f_j(\lambda_{jl})|}{\min_l|f_j(\lambda_{jl})|}\right)^2\right). \tag{34}$$

In the case where $A_1,\ldots A_k$ are density matrices presented in a quantum RAM, $X = O(\log(N))$, and we obtain equation (18).

Appendix B. Performing the controlled rotation of (29)

In this section, we show how to perform a controlled rotation in the form of (29) for an arbitrary function f with convergent Taylor series. Specifically, we are to rotate the ancilla by the angle $\theta(\lambda) = \sin^{-1}(Cf(\lambda))$. This is not trivial, even for simple cases such as $f(x) = 1/x$ in the original HHL algorithm. HHL do not provide a decomposition of this rotation in terms of controlled gates. Here, we will present such a method for arbitrary f.

The idea behind our method is to approximate $\theta(\lambda)$ by its Taylor series. We will first construct the register $|f(\lambda)\rangle$ from $|\lambda\rangle$, and then construct $|\theta(\lambda)\rangle$ from $|f(\lambda)\rangle$. This procedure is outlined below and presented in detail in algorithm 3.

Since f has a convergent Taylor series, we approximate

$$f(\lambda)\approx f(x_0)+f'(x_0)(\lambda-x_0)+\cdots.+\frac{f^{(n)}(x_0)(\lambda-x_0)^n}{n!} \tag{35}$$

for some constant x_0 near λ to order n. Now, by choice of C, $|Cf(\lambda)|<1$ lies in the radius of convergence of the Maclaurin series for $\sin^{-1}(x)$. Hence, we can substitute $Cf(\lambda)$ into this Maclaurin series and approximate

$$\theta(\lambda)=\sin^{-1}(Cf(\lambda))\approx Cf(\lambda)+\frac{(Cf(\lambda))^3}{6}+\frac{3(Cf(\lambda))^5}{40}+\frac{5(Cf(\lambda))^7}{112}+\cdots. \tag{36}$$

The multiplication in steps 1 and 2 on the quantum registers in algorithm 3 can be performed very similarly to the exponentiation a^n in Shor's algorithm [25]. Suppose the binary strings $|A\rangle$ and $|B\rangle$ are given in quantum form. Then, the classical grade-school multiplication algorithm can be carried out by performing addition operations controlled on the qubits representing $|B\rangle$: at the kth iteration, if the kth qubit of $|B\rangle$ from the right is $|1\rangle$, add $|2^kA\rangle$ (obtained by left-shifting qubits) to the result. Repeating for all k between 0 and the number of qubits in $|B\rangle$ minus one gives the desired product.

Finally, once we have the binary representation of the rotation angle in the register $|\theta(\lambda)\rangle$, we can implement the controlled rotation of the ancilla. Specifically, for each term 2^r appearing in this binary expansion, add a unitary controlled on the qubit coefficient of this term to rotate the ancilla by 2^r. Since any two-qubit controlled-U operation can be implemented with two controlled-NOT gates and single-qubit unitaries [24], the desired rotation of (29) can be implemented efficiently.

Algorithm 3. Constructing $|\theta\rangle(\lambda)$ from $|\lambda\rangle$.

Step 1: Initialization. Prepare an auxiliary register to hold the value $\lambda - x_0$. Prepare three more working registers: the first (initialized to 1) will hold the current power of $\lambda - x_0$, the second (initialized to 1) will hold the value of the first register multiplied by the Taylor coefficient $f^{(k)}(x_0)/k!$, and the third (initialized to 0) will be a running total for the right hand side of equation (35).

Step 2: Multiply the first working register by $\lambda - x_0$ from the auxiliary.

Step 3: Multiply the value in the first register by the kth Taylor coefficient $f^{(k)}/k!$ and store the result in the second working register.

Step 4: Add the value in the second working register to the third working register.

Step 5: Repeat steps 2–4 for each value $k = 1, 2,\ldots n$. After this step, we have successfully obtained the value $|f(\lambda)\rangle$ in the third working register.

Step 6: Repeat steps 1–5 now with a register containing $|Cf(\lambda)\rangle$ in place of $|\lambda\rangle$, and with the function $\sin^{-1}$ in place of f. It suffices to expand around $x_0 = 0$. This step yields the register $|\theta(\lambda)\rangle$, by the Maclaurin series approximation of (36).

References

[1] Welling M Fisher linear discriminant analysis (unpublished)
[2] Harrow A W, Hassidim A and Lloyd S 2009 *Phys. Rev. Lett.* **103** 150502
[3] Rebentrost P, Mohseni M and Lloyd S 2014 *Phys. Rev. Lett.* **113** 130503
[4] Lloyd S, Mohseni M and Rebentrost P 2013 arXiv:quant-ph/1307.0411v2
[5] Lloyd S, Mohseni M and Rebentrost P 2014 *Nat. Phys.* **10** 631–3
[6] Vasconcelos N *PCA and LDA* unpublished
[7] Cai D, He X and Han J 2008 Training linear discriminant analysis in linear time *Proc. 2008 Int. Conf. on Data Engineering (ICDE'08)*
[8] Belhumeur P N, João P H and Kriegman D J 1997 *IEEE Trans. Pattern Anal. Mach. Intell.* **19** 711
[9] Cheng Y-Q, Liu K, Yang J-Y, Zhuang Y-M and Gu N-C 1992 Human face recognition method based on the statistical model of small sample size *Intelligent Robots and Computer Vision X: Algorithms and Techniques Int. Soc. Opt. Photon.* **85** 1607
[10] Giovannetti V, Lloyd S and Maccone L 2008 Quantum random access memory *Phys. Rev. Lett.* **100** 160501
[11] Uetani M, Tateyama T, Kohara S, Han X H, Chen Y W, Kanasaki S, Furukawa A and Wei X 2015 Computer-aided diagnosis of liver cirrhosis based on multiple statistical shape models *Int. Conf. on Computer Information Systems and Industrial Applications*
[12] Dai Z, Yan C, Wang Z, Wang J, Xia M, Li K and He Y 2012 Discriminative analysis of early Alzheimer's disease using multi-modal imaging and multi-level characterization with multi-classifier (M3) *NeuroImage* **59** 2187
[13] Naik G, Selvan S and Nguyen H 2015 Single-Channel EMG classification with ensemble-empirical-mode-decomposition-based ICA for diagnosing neuromuscular disorders *IEEE Trans. Neural Syst. Rehabil. Eng.* (doi:10.1109/TNSRE.2015.2454503)
[14] Krusienski D J, Sellers E W, Cabestaing F, Bayoudh S, McFarland D J, Vaughan T M and Wolpaw J R 2006 A comparison of classification techniques for the P300 Speller *Neural Eng.* **3** 299
[15] Suárez-Cuenca J J, Guo W and Li Q 2015 Integration of multiple classifiers for computerized detection of lung nodules in CT *Biomed. Eng. Appl. Basis Commun.* **27** 1550040
[16] Esener I I, Ergin S and Yuksel T 2015 A new ensemble of features for breast cancer diagnosis *38th Int. Convention on IEEE Information and Communication Technology, Electronics and Microelectronics (MIPRO)*
[17] Alkan A and Günay M 2012 Identification of EMG signals using discriminant analysis and SVM classifier *Expert Syst. Appl.* **39** 44
[18] Dixon S J and Brereton R G 2009 Comparison of performance of five common classifiers represented as boundary methods: euclidean distance to centroids, linear discriminant analysis, quadratic discriminant analysis, learning vector quantization and support vector machines, as dependent on data structure *Chemometr. Intell. Lab. Syst.* **95** 1
[19] Hastie T, Tibshirani R and Friedman J 2009 *The Elements of Statistical Learning* (Berlin: Springer)
[20] Ahuja A and Kapoor S 1999 arXiv:quant-ph/9911082v1
[21] Mika S, Rätsch G, Weston J, Schölkopf B and Müller K-R 1999 Fisher discriminant analysis with kernels (unpublished)
[22] Cleve R, Ekert A, Macchiavello C and Mosca M 1999 Quantum algorithms revisited (arXiv:quant-ph/9708016)
[23] Li T, Zhu S and Ogihara M 2006 Using discriminant analysis or multi-class classification: an experimental investigation *Knowl. Info. Syst.* **10** 4
[24] Mermin D 2007 *Quantum Computer Science* (Cambridge: Cambridge University Press)
[25] Shor P 1997 polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer *SIAM J. Comput.* **26** 1484
[26] Nakahara M and Ohmi T 2008 *Quantum Computing: From Linear Algebra to Physical Realizations* (Boca Raton, FL: CRC) ch 8
[27] Boyd S and Vandenberghe L 2004 *Convex Optimization* (Cambridge: Cambridge University Press) ch 5
[28] Grover L and Rudolpha T Creating superpositions that correspond to efficiently integrable probability distributions arXiv: quant-ph/0208112
[29] Kaye P and Mosca M 2001 *Proc. Int. Conf. on Quantum Information* (New York: Rochester) (arXiv:quant-ph/0407102)
[30] Soklakov A N and Schack R 2006 *Phys. Rev.* A **73** 012307
[31] Williams V V 2012 Multiplying matrices faster than Coppersmith-Winograd *Proc. 44th ACM Symp. on Theory of Computing*
[32] Nielsen M A and Chuang I L 2011 *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press)