Eigen Component Analysis: A Quantum Theory Incorporated Machine Learning Technique to Find Linearly Maximum Separable Components

Chen Miao[1,2] and Shaohua Ma[1,2,3,4]

[1]*Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, Shenzhen, 518055, China*
[2]*Tsinghua University Shenzhen International Graduate School, Shenzhen, 518055, China*
[3]*Corresponding email: ma.shaohua@sz.tsinghua.edu.cn*
[4]*Lead Contact*

March 24, 2020

Abstract

For a linear system, the response to a stimulus is often superposed by its responses to other decomposed stimuli. In quantum mechanics, a state is the superposition of multiple eigenstates. Here, we propose eigen component analysis (ECA), an interpretable linear learning model that incorporates the principle of quantum mechanics into design of algorithms capable of feature extraction, classification, dictionary and deep learning, and adversarial generation, etc. The simulation of ECA, possessing a measurable *class-label* $\mathcal{H}$, on a classical computer outperforms the existing classical linear models. An enhanced eigen component analysis network (ECAN), a network of concatenated ECA models, gains the potential to be not only integrated with nonlinear models, but also an interface for deep neural networks to implement on a quantum computer, by analogizing a data set as recordings of quantum states. Therefore, ECA and its derivatives contribute to expanding the feasibility of linear learning models, by adopting the strategy of quantum machine learning to replacing heavy nonlinear models with succinct linear operations in tackling complexity.

Keywords Quantum Mechanics · Machine Learning · Degeneracy · Component analysis · Linear separability

1 Introduction

Machine learning is widely used in areas ranging from chemistry [1, 2, 3], biology [4, 5, 6], materials [7] to medicine [8] . It has also been used in quantum mechanics [9, 10, 11, 12] and quantum chemistry [13, 14]. Quantum mechanics has also inspired many machine learning algorithms [15, 16, 17, 18], which, in turn, facilitate physics growth per se [19, 20]. The entanglement between machine learning and quantum mechanics has started to produce increasing cross-disciplinary breakthrough in physics, chemistry, artificial intelligence and even social sciences [21], and emerged Quantum machine learning (QML), an interdisciplinary field that employs quantum mechanics principles into machine learning. In fact, quantum mechanics share high similarity with machine learning in both of their underlying principles and prediction manners [12, 20].

In machine learning, the features of a data set are usually redundant [22]. Feature extraction refers to enriching the features of interest and suppressing or discard features out of interest. A number of classical dimension reduction methods has been proposed to learn the similarity or difference among the features of a data set or across multiple data sets. Principal component analysis (PCA) seeks an orthogonal transformation to maximize the variance and separate the data, but it is incapable of exploiting class labels or performing inter-class differentiation. Linear discriminative analysis (LDA) takes the advantage of class label to differentiate inter-class data, but it is conditioned on Gaussian distribution. T-distributed stochastic neighbor embedding (t-SNE) is a nonlinear feature extraction model that finds a low dimensional representation of a high dimensional data, but could suffer false clustering if with low perplexity. independent component analysis (ICA) decouples a mixed signal into multiple source signals, but it's limited to

non-Gaussian data distribution. Classification can be conducted in an easier manner with features extracted. The goal of classification is to assign a class label on a given sample. If several classes are linearly separable, the classifier is termed linear classifier. Usually, linear separation refers to one or several hyperplanes existed to separate the data. Logistic regression (LoR) finds a hyperplane and converts the distance between a new input and the hyperplane into probability of the data belonging to a class. Taking this one step further, support vector machine (SVM) maximizes the two margins on each side of the hyperplane. LDA finds several hyperplanes at once, with each one being similar to the ones found by LoR. Empirically, it's generally assumed that a linear model is less robust and powerful than a nonlinear model. However, this is not the case for eigen component analysis (ECA). By utilizing the linearity of a quantum system that superimpose eigenfeatures (i.e. eigenstates), it functions as a linear model that could couple with most of the machine learning subjects, including but not limited to classification, generative model, feature extraction, dictionary learning, deep neural networks (DNN), convolutional neural networks (CNN) or adversarial generation. ECA provides a solution toward image generation by learning from a few coefficient generators, such as normal and uniform distribution. The image generation from known coefficients are analogous to prepare a 'cuisine' by following an established 'recipe'.

However, it is demanded to establish a method that learns the similarities, i.e. the common features possessed by the data in a set across classes, and differences, i.e. the features possessed by the data belonging to a specific class, simultaneously. The similarities are used to denoise or remove the background, such as telling wheat from chaff. The differences are used to 'tell the wheat from rye'. PCA and LDA pursuit orthogonal bases whose subsets could be used as undercomplete dictionaries. However, it's not required for the dictionary to be orthogonal in generic dictionary learning (DictL). From feature extraction to linear classification and dictionary learning, a model is required to have good performance as well as high interpretability. The principle of superposition in a quantum linear system empowers it being of higher significance than classical linear and nonlinear systems in interpretation.

The responses of a linear system to two or more stimuli are superposed. It allows inspection on the response of individual stimulus to obtain the overall response by superposition. The decomposing strategy could also be extended to approximate nonlinear systems, by dividing one composite signal into multiple basis signals in their analysis. A well chosen basis is important in decomposing such a signal. For example, the Fourier transform can decompose a signal into infinite orthogonal basis signals composed by sine and cosine functions. The overall response is the superposed consequence of responses to individual decomposed signals. In analysis of a basis signal, such as the sinusoid function,

$$f(x) = A\sin(\omega x + \phi), x \in (-2\pi, 2\pi] \tag{1}$$

it could be distinguished by its amplitude A, frequency ω, and phase shift ϕ. From a local observer's point of view at $x_0 \in (-2\pi, 2\pi)$, only the amplitude and phase difference could be sensed. Hence, we define two types of separable components, the amplitude component and the phase component. We use 'amplitude' and 'phase' here to avoid confusion with the concept of space, time and frequency domains in Fourier transform. The amplitude differences is related to amplitude, magnitude, and coordinate differences. The phase difference is related to phase, frequency, direction and eigenvalue. There exists many classical algorithms to distinguish the coordinate difference or amplitude difference. K-nearest neighbor (KNN), k-means clustering (KMC) and mean shift classify or cluster data by a metric of 'distance' from a centroid [23, 24, 25]. PCA maximizes the coordinate gap of the projection on one dimension of the vector space in a data set, whereas LDA maximizes the coordinate gap among classes. T-SNE is a nonlinear transformation method that presents the coordinate difference in a lower dimensional feature space. LoR finds the hyperplane to tell the coordinate difference and SVM maximizes the margins of a hyperplane upon their distance between coordinates.

Phase differences substantially exist among various data sets. Each signal can be decomposed into several phases with varied probabilities. Meanwhile, a phase may be prevalent in some classes but rare in the rest, which suggests differential probability one eigenfeature belonging to each class. Here, we propose ECA that is a quantum theory-based algorithm and focuses on the phase differences. ECA identifies the phase differences for a l-class data set. Benefiting from ECA, the tasks such as feature extraction, classification, clustering, and dictionary learning can all be performed in linear models. In classical machine learning, the class label follows a Bernoulli or categorical distribution. ECA challenges it with a more rational assumption, i.e. l class labels following l independent Bernoulli distributions. First, a m-dimensional data is prepared as a state on $\log(m)$ qubits. Second, measurements on l measurables with l commutative operators are taken on this state in an arbitrary order and the measured results are recorded. Last, optimization is performed on a classical computer based upon the parameters of the operators, probabilities in measurements and ground truth of each prepared state.
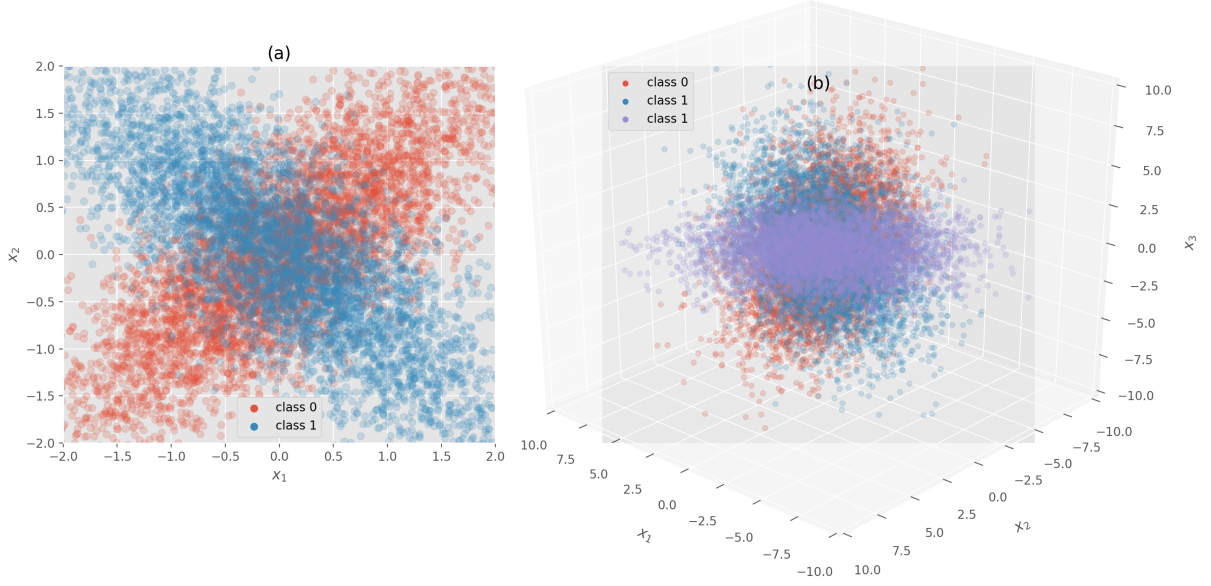
Figure 1: Two artificial intersected data sets. (a) 2D data set. Data in red belong to class 0 and data in blue belong to class 1; (b) 3D data set. Data in red belong to class 0, and data in blue and purple belong to class 1.

2 Results

2.1 Background and eigen component analysis (ECA) mechanism

First, we clarify the notations used in this paper. A quantum algorithm is intrinsically simple and intuitive, but also abstractive. Its implementation can be simulated on a classical computer. To help understand the concept and verify the simulation, we describe the simulation algorithm in a classical machine learning language, but meanwhile, follow the conventional notations as in quantum mechanics to be consistent with the quantum algorithm, unless specified otherwise. For example, to avoid the confusion caused by using 'observe' and its derivatives, we adopt their meanings as in quantum mechanics in this article. Likewise, the samples in the data set $\{x^{(n)}\}$ are termed *sample*, *input*, *state*, *recording* or just *vector*. Furthermore, we use notations of convention in real coordinate spaces when no complex numbers are involved.

It should be paid attention that in quantum algorithms, the eigenvalues of measurable for a qubit or a composite system, are defined as -1 and $+1$, representing 'false' and 'true' that if an input or eigenfeature belongs to a class. In classical simulation, the counterparts of these eigenvalues, indicating the class label of an eigenfeature or input vector, are defined as 0 and 1. Moreover, the term 'class label' could mean both the original class label given by the data set and the derived class label that if an input belongs to one class. For example, the class label '3' of a sample in a 10-class data set could derive ten class labels, which are '+1' for being a sample from class '3' and '-1' for the corresponding input belonging to classes other than '3'.

All the sets used in this paper are 0-indexed. The values i, j, k index the sample data set (with size n), input vector (with size m), and class label (with size l), respectively. $\{\mathring{x}^{(i)}\}$ refers to a data set with n samples, where $i = 0, 1, \cdots, n-1$, together with a finite set $\mathcal{C}$ of class labels $\{k, k = 0, 1, ..., l-1\}$. The corresponding target values compose the set $\{y^{(i)}\}$. Notation $\mathring{(\cdot)}$ refers to an initially non-normalized vector. The data set is then normalized to $\{(x^{(i)}, y^{(i)}), i = 0, 1, \cdots, n-1\}$. All the vector in vanilla eigen component analysis (VECA) are normalized and the magnitude information are discarded if without notation $\mathring{(\cdot)}$ or unspecified. The normalized data set is the recordings of states and their measured values. We also denote one indicator function $1\{y = y^{(i)}\}$ and one-hot vector function
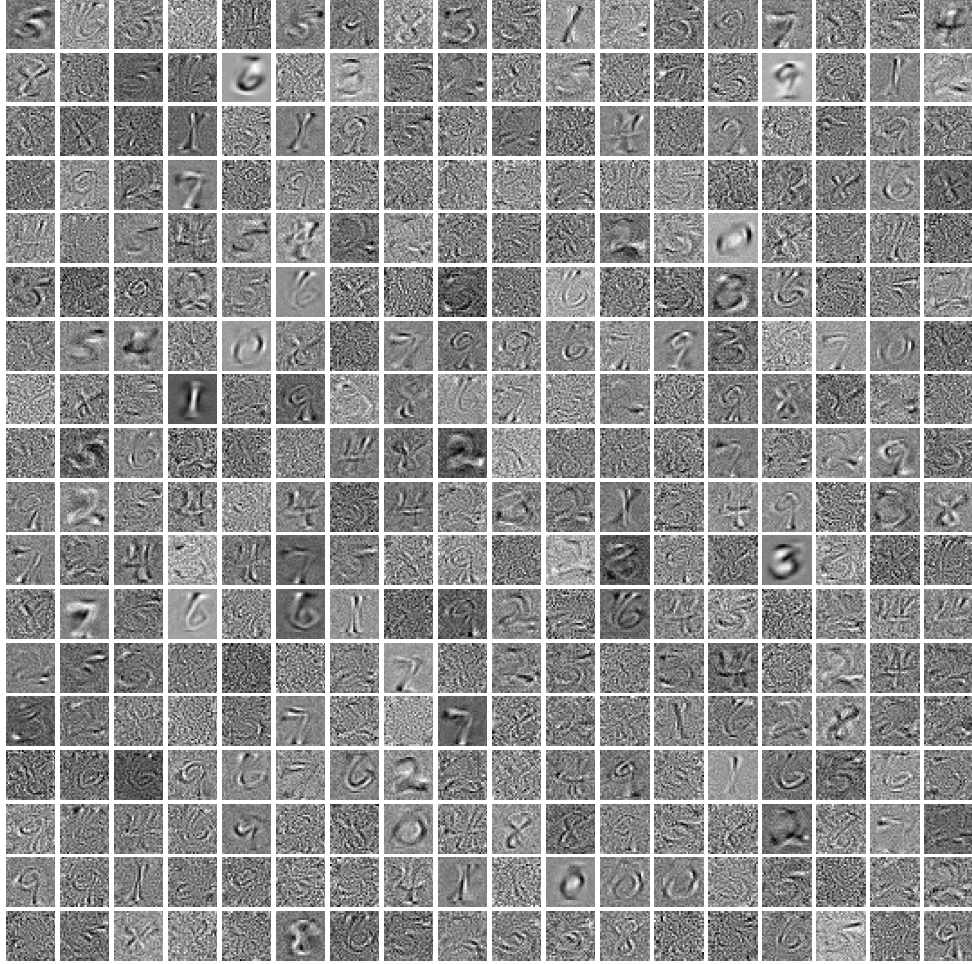
Figure 2: Some 324 pure eigenfeatures (PEs) randomly choosed from all 328 PEs learnt on MNIST data set by approximated eigen component analysis (AECA). The total number of eigenfeatures is 784.

Figure 3: These images are coarsely generated with weighted sum of pure eigenfeatures (PEs) of MNIST data set. For each class, the weight of a PE is the mean projection of all training samples on that PE.

$y^{(i)} = \mathbb{1}\{y = y^{(i)}\}$, where $i = 0, 1, \cdots, n - 1$.$y^{(i)}$ is a $l \times 1$ vector with its $y^{(i)}$-th element being 1 and otherwise 0. In addition to that, we denote a one-hot matrix of stacked Bernoulli one-hot vector as

$$\mathbb{y}^{(i)} = \begin{bmatrix} y^{(i)} & \widetilde{y^{(i)}} \end{bmatrix} = \begin{bmatrix} \mathbb{1}^T\{y_0 = y_0^{(i)}\} \\ \mathbb{1}^T\{y_1 = y_1^{(i)}\} \\ \vdots \\ \mathbb{1}^T\{y_{l-1} = y_{l-1}^{(i)}\} \end{bmatrix}_{l \times 2} \tag{2}$$

where the operator $\widetilde{y}$ takes the one's complement of each element in y. In the classical simulation, for discrete ECA, i.e. the observed values are discrete, the ket-vectors (or kets) $|x\rangle$ is the same as x. The probability $p(y| |x\rangle)$ without a superscript specified on y means a vector representation of its probability mass function (p.m.f.). The bold font $\mathbf{p}$ indicates a vector of stacked probabilities of independent random variables. The outline font $\mathbb{p}$ indicates a matrix of stacked p.m.f. of independent Bernoulli random variables. The element y_k of y is a Bernoulli random variable if without a superscript. Thus, with a superscript, $y^{(i)}$ is one-hot vector, yet without superscript, it is a vector of stacked independent random variables. For some situation, usually in general discussion, the superscript of a numerical value like $y^{(i)}$, $y^{(i)}$, or $x^{(i)}$ is omitted for simplicity, when it could be inferred from the context . The subscript and superscript are omitted when there is no risk of ambiguity in the rest of the paper.

In quantum mechanics, a vector representation of an object or state is 'measurable' as long as we know the measurable. We could also predict these measurements, once we know the mathematical expression of the measurable and its state. Likewise, we could abstract a real world state or object, such as a spin or image, as a vector, and construct its measurable, no matter it's momentum or *class-label*.

As we addressed, samples in a data set have two types of variance. One is amplitude-based and the other one is phase-based. Unlike the off-the-shelf algorithms like SVM, LDA or KNN that have proven extraordinary performance on telling the difference on amplitudes, our ECA focuses on identifying the phase-based differences of a data set.

For a vector, in general, a linear classifier or even some kernel-based classifiers treat each element of the vector as one feature. However, these 'features' may not fully represent the real property of the data. For a vector possessing more complicated structures, all the elements in the vector become necessary to constitute a reliable feature. In other words, it's a choice to express the data on a well-defined basis (see Figure 1 (a)) or a standard basis (see Figure 1 (b)). Therefore, for a vector x

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{bmatrix}, \tag{3}$$

each x_i $(i = 0, 1, \cdots, m - 1)$ is viewed as a feature of this vector upon a standard basis. For some complicated structures (e.g. the edges of an object in an image) , one single element of x i.e. a vector in a standard basis, cannot tell
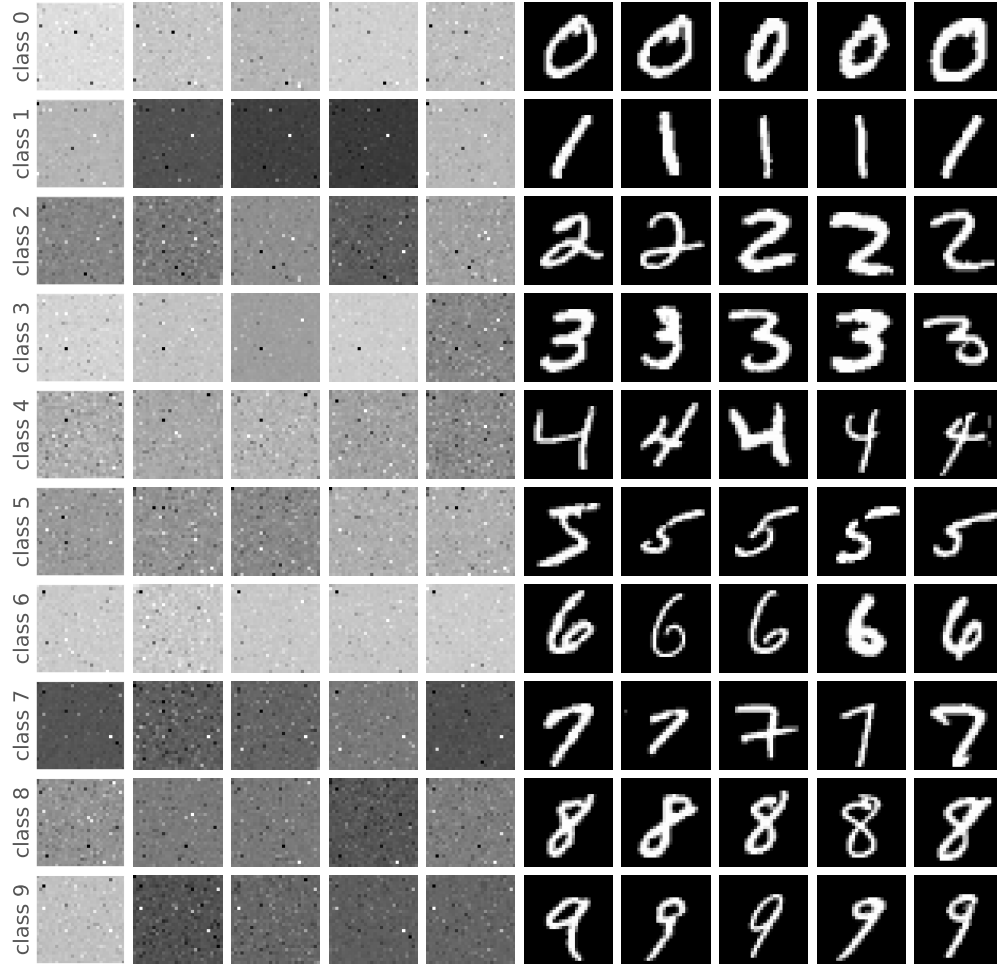
Figure 4: Randomly chosen images from MNIST data set upon basis learnt by vanilla eigen component analysis (AECA) and the standard basis. The images on the right half are input images to train AECA. The images on the left are corresponding result of a basis transformation with learnt basis, i.e. eigenfeatures, from right half. The darkest and brightest pixels on the left half are dominant eigenfeatures, which remain constant regardless of digit features on the right.

the whole story. If we finds an orthogonal basis (i.e. eigenfeatures, see Figure 2) $\mathcal{B}$, all the vectors could be a unique linear combination of vectors in this basis (see Figure 3). If $\mathcal{B} = \{|\lambda_0\rangle, |\lambda_1\rangle, \cdots, |\lambda_{m-1}\rangle\}$, then

$$\mathsf{x} = \psi_0 |\lambda_0\rangle + \psi_1 |\lambda_1\rangle + \cdots + \psi_{m-1} |\lambda_{m-1}\rangle \tag{4}$$

in which ψ_j is nomalized coefficients (see Figure 4), with

$$\sum_j \langle \psi_j | \psi_j \rangle = 1. \tag{5}$$

In quantum mechanics, the state x would collapse on eigenstate $|\lambda_j\rangle$ $(j = 0, 1, \cdots, m-1)$ with the probability

$$p_j = \langle \psi_j | \psi_j \rangle. \tag{6}$$

Not only the input vector but also its class label have quantum interpretation. Classical machine learning algorithms usually assume the class labels of input or basis vectors following Bernoulli or categorical distribution, which is true for the eigenvectors in Figure 5 (a)-(f). For a 2-class data set (Figure 1 (a)), the probabilities of the predictions could be described by a probability matrix

$$\rho = \begin{bmatrix} \rho_0(1 - \rho_1) & \rho_0 \rho_1 \\ (1 - \rho_0)(1 - \rho_1) & \rho_1(1 - \rho_0) \end{bmatrix} \tag{7}$$

in which ρ_0 and ρ_1 are the probability of an input vector belonging to class '0' and class '1', respectively. Under the classical assumption, the trace of ρ equals 1, i.e.

$$Tr(\rho) = 1. \tag{8}$$

If a new input vector unambiguously belongs to one class, then either p_0 or p_1 equals 1, reaching

$$\rho^2 = \rho. \tag{9}$$

However, this could be wrong. For the data points in the center of Figure 1 (a), it could belong to either class. Meanwhile, many data points in the two-dimensional (2D) space do not belong to any of these two classes. A data set usually occupies a compact space or spans a subspace in the full vector space. For a l-class data set, the total number of decisions in the full space could reach 2^l, but the mutual exclusive decision is only a small subset of size l. If we assume that each input vector as well as its basis vectors have considerable possibilities belonging to one or more classes (Figure 5 (g)-(l)), we could choose to inspect each class independently. If we prepare a new input vector as a quantum state, it's assumed an apparatus could be constructed to measure the class label indicating whether an input vector belongs to a to-be-decided class. For the l classes, l sized apparatuses could be built to take measurements on l identical copies of the state. Second, an operator on a m-dimensional state vector has m eigenvalues, with each eigenvalue has two possible outcomes, -1 ('false') and $+1$ ('true'), to represent if it belongs to a specific class. The candidates of all these eigenvalues of each operator should be arranged to degenerate to -1 and $+1$. Taking all classes into account, we should assume all these corresponding operators of these measurables share a complete basis of simultaneous eigenvectors. Hence, for a data set with l classes and m-dimensional features, the task of degenerating the data set into two distinctive states is converted into a measurement on l independent systems with $l\log(m)$ qubits in total. In these measurements, for each measurable, the measurement on the whole system could be product of observed values taken on each qubit. For more concurrency, $l\log(m)$ qubits could be used. As these operators commute, all these operators share a complete basis of simultaneous eigenvectors but has its own eigenvalues.

Therefore, in classical simulation, instead of learning which class a vector belongs to, we learn which classes each eigenfeature of a vector belongs to. Furthermore, the decision-making should be conducted independently on each class. Next, we have a mapping table between eigenfeatures and class labels. This leads us to learning an eigenfeature matrix (EFM) representing the unitary operator with a complete basis of simultaneous eigenvectors and eigenfeature-class mapping matrix (ECMM). ECMM bridges the superposition of the probabilities of the class label of each eigenfeature and the class label of the input vector. All we need is to sum up the probabilities of each eigenfeature assigned for all classes independently. Afterward, we could obtain the combined probabilities one vector belonging to each class. For l classes, the probabilities of an input vector belonging to each class is $p_0, p_1, \cdots, p_{l-1}$. The mutual exclusive probability for class 'c' is calculated by

$$p(y^{(i)} = c|\mathsf{x}; \theta) = \prod_{k \neq c} (1 - p_k) p_c.$$
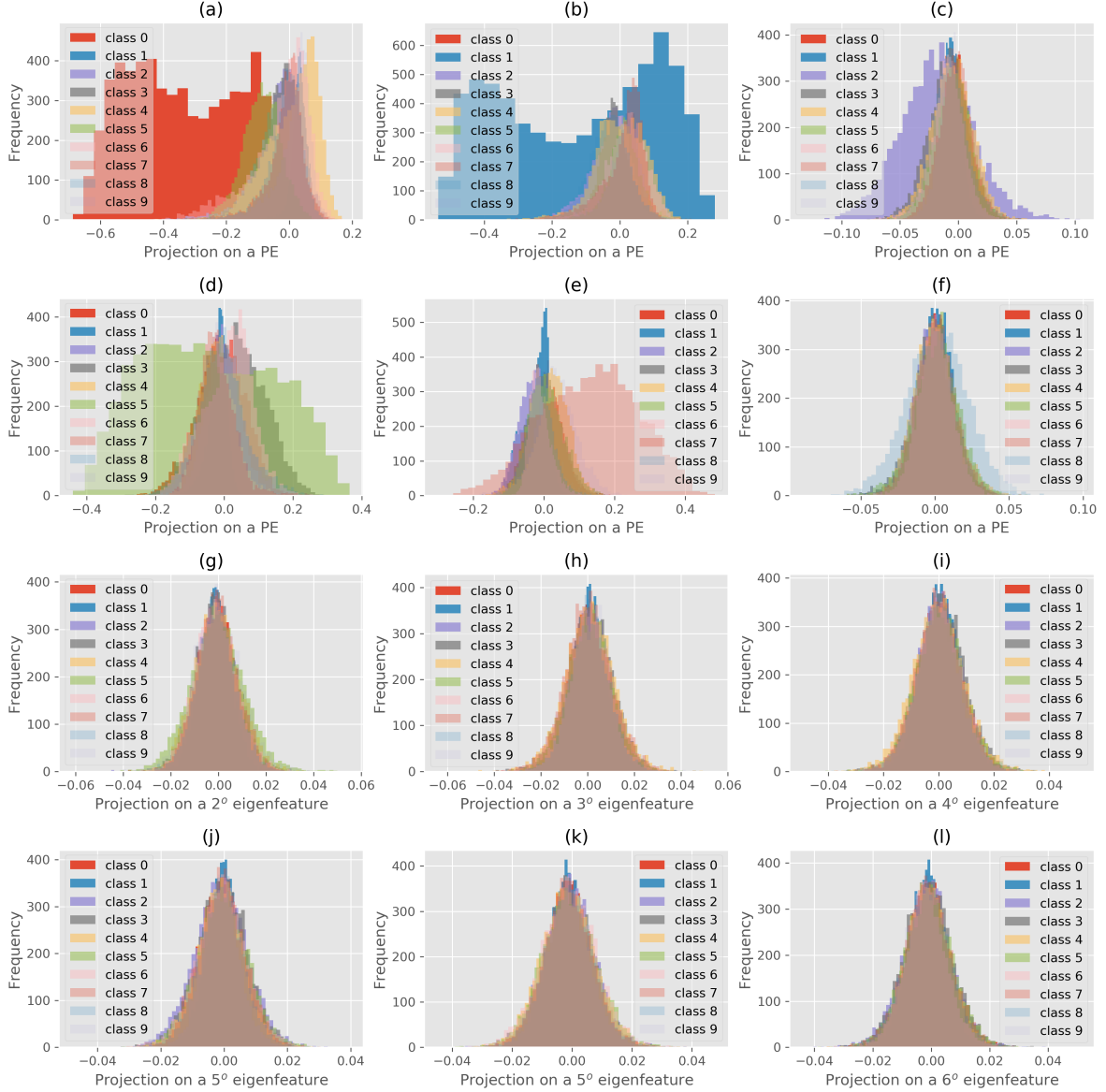
7

Figure 5: The distribution of projections of all training samples on 12 eigenfeatures selected from eigenfeatures learnt on MNIST data set using approximated eigen component analysis (AECA). The superscript k^o denotes k degree of overlapping. (a) - (f) Frequency distribution of normalized projections on pure eigenfeatures (PEs) corresponding to the class label '0', '1', '2', '5', '7', and '8' sequentially. That PEs used in (a) - (c) are corresponding to the least degenerated and these in (d) - (f) are corresponding to the most degenerated. (g) - (i) Frequency distributions of randomly choosed 2°, 3° and 4° eigenfeatures in turn. (j) - (l) Frequency distributions of the most overlapped three eigenfeatures with 5° for eigenfeature in (g) and (h) and 6° for (i).

The unitarity of EFM could guarantee that the difference is kept in a change of basis transformation. For EFM, the variance of projection on eigenfeature is maximized (see Figure 4). In the left half of Figure 4, the bright and dark pixels indicate significant signals, whereas the gray pixels are trivial, enabling for dimension reduction. Also, the stable positioning of these bright and dark pixels among different inputs suggests it being appropriate as a classifier.

For prediction, we prove that an independent decision can be made without calculating the mutual exclusive probabilities for a data set with two classes. The proof can be easily extended to data sets with multiple classes using mathematical induction. For a given input, the two mutual exclusive probabilities of each class label are

$$p(\begin{bmatrix}0\\1\end{bmatrix}|\mathbf{x}^{(i)};\theta) = p_0(1-p_1)$$

and

$$p(\begin{bmatrix}1\\0\end{bmatrix}|\mathbf{x}^{(i)};\theta) = p_1(1-p_0)$$

Then the proof is given as

Proof. Without loss of generality, suppose $p_0 > p_1$, such that

$$p(\begin{bmatrix}0\\1\end{bmatrix}|\mathbf{x}^{(i)};\theta) - p(\begin{bmatrix}1\\0\end{bmatrix}|\mathbf{x}^{(i)};\theta)$$
$$=p_0(1-p_1) - p_1(1-p_0)$$
$$=p_0 - p_1 > 0$$
$$\Rightarrow p(\begin{bmatrix}0\\1\end{bmatrix}|\mathbf{x}^{(i)};\theta) > p(\begin{bmatrix}1\\0\end{bmatrix}|\mathbf{x}^{(i)};\theta)$$

□

Further, we can build a τ-fold eigen component analysis network (ECAN) by concatenating τ ECA models. ECAN gives ECA the capability to integrate nonlinear models such as DNNs. A dimension operator assuming the nonlinearity, which can be specially designed or a classical DNN, is installed between consequent ECA models.

2.2 Related work

The related work include several classical algorithms like ICA, DictL, and also QML algorithms.

2.2.1 Independent component analysis (ICA)

ICA shares the similar goal as ECA. Both algorithms try to find independent components which could generate the data from some independent sources. ICA decouples a mixed signal by multiple recorders depending on the varied combination of source signals. The recorders can be regarded as another kind of label because they record the intensity of the source variably. Thus, ICA could be replaced by ECA as ECA is dependent on the most intensive source. The major advantage of ECA over ICA is that there is no necessary for prior distribution assumption.

2.2.2 Dictionary learning (DictL)

DictL is similar to ECA because they both want to find a sparse representation of the data set. A supervised dictionary learning method presented in [26] is like a second cousin of ECA in appearance. A discriminative task is added to the objective while the reconstruction term is reserved. Likewise, in VECA, our objective is also to identify the independent eigenfeatures, based on which the data classification is conducted. In comparison, ECA is easier to train because it comprises less hyperparameters. Moreover, ECA is less likely for loss of information because it takes all classes and a complete profile of basis into account and preserves the difference.

2.2.3 Quantum machine learning (QML)

QML is a wide range of machine learning algorithms including quantum computation-based machine learning or inspired and facilitated by quantum mechanics. The method presented in [17] assumes a prior for the input feature states and obtains a number of template classes. However, the identities of these template states are ignored. In their method, the need to find some template states which are all linear combination of some pure quantum states. In contrast, in our ECA, all the inputs are linear combinations of pure quantum states, the identities of which are utilized for further pursuit.

2.3 Preliminary performance test of eigen component analysis (ECA)

Before moving forward to the algorithm, we prepared an example that illustrated two ideal cases with two artificial data sets (see Figure 1). To birth some epiphanies, this informal discussion is based upon *guess* and intuition.

Two data sets showed in Figure 1. The data of each class intersected with each other. For the 2D and three-dimensional (3D) data set , we could *guess* an EFM (which is an unitary operator) $\breve{\hat{P}} \in \mathbb{C}^{m \times m}$ of a linear seprator $\breve{\hat{H}}$ with eigenfeature as column vector and ECMM $\breve{\hat{\mathfrak{L}}} \in \mathbb{R}^{m \times l}$ (of which the elements $\breve{\hat{\mathfrak{L}}}_{jk} \in \{0, 1\}$) which are

$$\breve{\hat{P}}_{2D} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \quad \breve{\hat{\mathfrak{L}}}_{2D} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

and

$$\breve{\hat{P}}_{3D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \breve{\hat{\mathfrak{L}}}_{3D} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \text{ or } \breve{\hat{\mathfrak{L}}}_{3D}^{sparse} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

respectively, in which the symbol $\breve{(\cdot)}$ indicates a numerical estimation and $\breve{\hat{\mathfrak{L}}}_{3D}^{sparse}$ is an equivalent sparse representation of $\breve{\hat{\mathfrak{L}}}_{3D}$. For the 3D data set, each columns of EFM represent a pivot axis or principal component of the data set. The element $\breve{\hat{\mathfrak{L}}}_{3D}^{00} = 0$ means the 0th eigenfeature of this data set doesn't belong to class '0' and $\breve{\hat{\mathfrak{L}}}_{3D}^{10} = 1$ indicates the 1st eigenfeature belongs to class '0'. The two 1s in the third row of $\breve{\hat{\mathfrak{L}}}_{3D}$ represents that the 2nd eigenfeature could be noise or background shared among the two classes. For a new input, we only need to sum up the probabilities that the input projects on the 1st and 2nd eigenfeature to decide the probability whether this input belongs to class '0'. One should notice the decision of class label of a input is not mutual exclusive because the summation of each class could be equal and the sum of the two probabilities could surpass 1 as they are independent decision.

We give a more concrete development for this informal discussion. For vector $\mathring{x}$ in aforementioned 3D data set,

$$(\hat{\mathring{x}}^T \breve{\hat{P}})(\hat{\mathring{x}}^T \breve{\hat{P}})^T = 1 \tag{10}$$

in which $\hat{(\cdot)}$ is the nomalization operator. Hence we denote the elements of

$$\psi = \breve{\hat{P}}^T \hat{\mathring{x}} \tag{11}$$

as ψ_j and we have

$$\sum_{j=0}^{m-1} \psi_j^2 = 1. \tag{12}$$

For the aforementioned 3D data set, instead of treating the class label of a vector as a single categorical distribution, the p.m.f. of each class label c given x is assumed to follow a independent Bernoulli distribution, such that the probability of one vector belongs to class '0' is

$$p_0(y_0 = 1 | x^{(i)}; \theta) = [(x^{(i)T} \breve{\hat{P}}_{3D}) \odot (x^{(i)T} \breve{\hat{P}}_{3D})](\breve{\hat{\mathfrak{L}}}_{3D})_{\bullet,0} \tag{13}$$

in which θ is the all unknown parameters, $\odot$ is element-wise Hadamard product operator, $\bullet$ is a placeholder for taking the entire column (as first index) or row (as second index) and $(\breve{\hat{\mathfrak{L}}}_{3D})_{\bullet,0}$ means the 0-th column of $\breve{\hat{\mathfrak{L}}}_{3D}$. Thus, the combined probabilities of these two Bernoulli random variables with observed event as $\mathbb{1}_{2 \times 1}$ could be defined as

$$\mathbf{p}(y = \mathbb{1}_{2 \times 1} | x^{(i)}; \theta) = \begin{bmatrix} p_0 \\ p_1 \end{bmatrix} = \{[(x^{(i)T} \breve{\hat{P}}_{3D}) \odot (x^{(i)T} \breve{\hat{P}}_{3D})](\breve{\hat{\mathfrak{L}}}_{3D})\}^T \tag{14}$$

and the complement probabilities could be

$$\mathbf{p}(\mathsf{y} = \mathbb{0}_{2\times 1}|\mathsf{x}^{(i)};\theta) = \mathbb{1}_{2\times 1} - \mathbf{p}(\mathsf{y}_0^{(i)}|\mathsf{x}^{(i)};\theta) = \begin{bmatrix} 1 - p_0 \\ 1 - p_1 \end{bmatrix} = \{[(\mathsf{x}^{(i)T}\breve{P}_{3D}) \odot (\mathsf{x}^{(i)T}\breve{P}_{3D})](\mathbb{1}_{3\times 2} - \breve{\mathfrak{L}}_{3D})\}^T \quad (15)$$

in which the outline font $\mathbb{0}, \mathbb{1}$ are vector or matrix with corresponding digits.

For simplicity, we could write these two p.m.f. of Bernoulli random variable y_0 and y_1 together. With the feature matrix and the mapping matrix, these stacked or combined p.m.f. of combined Bernoulli random variable $\mathsf{y}_k(k = 0, 1)$ given x (i.e. $\mathsf{y}_k|\mathsf{x}^{(i)}, i = 0, 1, \cdots, n-1$) could be written in

$$\mathbb{p}(\mathsf{y}|\mathsf{x}^{(i)};\theta) = [\mathbf{p} \quad \mathbb{1}_{2\times 1} - \mathbf{p}] \tag{16}$$

where the *rows* are the vectors of corresponding p.m.f..

To obtain the mutual exclusive decision on the class label, the unambiguous probability of x belonging to class '0' could be calculated as

$$p_0(\mathsf{x} \text{ belongs to class '0' and not '1'}|\mathsf{x}^{(i)};\theta) = p(\begin{bmatrix} 0 \\ 1 \end{bmatrix}|\mathsf{x}^{(i)};\theta) = \mathbb{p}_{00}\mathbb{p}_{11} = \mathbf{p}_0(1 - \mathbf{p}_1) = p_0(1 - p_1) \tag{17}$$

2.4 Experiment results

We compared our model with LoR, LDA, quadratic discriminative analysis (QDA), SVM, kernel support vector machine (KSVM) with an radial basis function (RBF) kernel.

2.4.1 Counting the parameters

For a data set with m features and l classes, the number of total parameters could be calculated as:

LoR $m + 1$

LDA $lm + (l - 1)$

QDA $lm + (l - 1) + l\frac{m(m+1)}{2}$

SVM $m + 1$

ECA $lm + m(m + 1)$.

We don't count the parameters of KSVM in these experiments.

2.4.2 Two artificial data sets (2D and 3D)

2D One class of the 2D data set (Figure 1) is random normally generated with mean of $(0 \quad 0)^T$ and covariance matrix $\begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$; and the another class is gengerated with mean $(0 \quad 0)^T$ and covariance matrix $\begin{pmatrix} 1 & -0.8 \\ -0.8 & 1 \end{pmatrix}$.

The P and $\dot{L}$ obtained by ECA is

$$P_{2D} = \begin{bmatrix} -0.7199738 & -0.7047859 \\ -0.6945969 & 0.70967793 \end{bmatrix} \text{ and } \dot{L}_{2D} = \begin{bmatrix} 1.0000000e + 00 & 1.0823610e - 20 \\ 1.6251015e - 20 & 1.0000000e + 00 \end{bmatrix}$$

respectively, with which the model could obtain an accuracy of 81.39% on the validation data which is on par with QDA and outperforms the rest (Table 1).

3D One class of 3D data set (Figure 1 (b)) is random normally generated with mean of $(0 \quad 0 \quad 0)^T$ and covariance matrix $\begin{pmatrix} 0.10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}$; and another one is random normally generated with mean $(0 \quad 0 \quad 0)^T$ and covariance matrix $\begin{pmatrix} 10 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 10 \end{pmatrix}$ and mean $(0 \quad 0 \quad 0)^T$ and covariance matrix $\begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}$.

Table 1: Compare with other classifiers of 2D data set

Name	Metrics		Parameters
	Accuracy	Confustion Matrix	
LoR	0.5242	$\begin{pmatrix} 0.5817 & 0.5313 \\ 0.4183 & 0.4687 \end{pmatrix}$	3
LDA	0.5239	$\begin{pmatrix} 0.5811 & 0.5313 \\ 0.4189 & 0.4687 \end{pmatrix}$	5
QDA	0.8124	$\begin{pmatrix} 0.8229 & 0.1977 \\ 0.1771 & 0.8023 \end{pmatrix}$	11
SVM	0.6048	$\begin{pmatrix} 0.8168 & 0.5998 \\ 0.1832 & 0.4002 \end{pmatrix}$	3
KSVM	0.8063	$\begin{pmatrix} 0.802 & 0.1894 \\ 0.198 & 0.8106 \end{pmatrix}$	
ECA	0.8139	$\begin{pmatrix} 0.8192 & 0.1912 \\ 0.1808 & 0.8088 \end{pmatrix}$	8

The P and $\dot{L}$ obtained by ECA is

$$P_{3D} = \begin{bmatrix} 0.01377437 & -0.00443745 & 1.0064974 \\ 0.02682977 & -0.9951887 & -0.00494725 \\ 1.0032624 & 0.02540882 & -0.01346647 \end{bmatrix}$$

$$\dot{L}_{3D} = \begin{bmatrix} 6.8667921e-22 & 7.7224560e-22 \\ 1.0000000e+00 & 3.4256167e-22 \\ 1.8692240e-21 & 1.0000000e+00 \end{bmatrix}$$

with which the model could obtain an accuracy of 94.24% on the validation data. It outperforms any other linear models included in the table. Meantime. We would obtain the equivalent form of $\dot{L}$ if we used approximated eigen component analysis (AECA) with Equation (55). That is

$$\mathfrak{L}_{3D}^{dense} = \lfloor \dot{L}_{3D}^{dense} \rceil = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

to which the result is rounded. And we won't mention if we round the result in the rest of the paper.

2.4.3 MNIST data set (using approximated eigen component analysis (AECA), vanilla eigen component analysis (VECA) and eigen component analysis network (ECAN))

This experiment exhibited the capability of dimension reduction of ECA. Meantime, it's a good illustration of the extensionality of ECA.

- AECA

  With no more than 12 epochs of training, we could obtain an accuracy of 91.82% on the MNIST data set, which outperforms LDA (87.30%). QDA collapsed on this data set. The corresponding confusion matrices of ECA, LDA, QDA are listed together with their accuracy (Figure 6).

  Part of the learnt eigenfeatures are displayed in Figure 2. These overlapped eigenfeatures (mapped to two and more classes) could be separated by amplitude based separator or raising dimension. The crowdedness of eigenfeatures (Figure 8) showed that the digit '1' needs the least eigenfeatures to express itself and the digit '8' needs the most eigenfeatures to express. From the overlapping histogram of classes on eigenfeatures (Figure 9), we could found that more than 300 eigenfeatures are mapped to a single class.

Table 2: Compare with other classifiers of 3D data set

| Name | Metrics | | Total Parameters |
	Accuracy	Confustion Matrix	
LoR	0.6671	$\begin{pmatrix} 0.1974 & 0.1 \\ 0.8026 & 0.9 \end{pmatrix}$	4
LDA	0.6667	$\begin{pmatrix} 0.2005 & 0.1021 \\ 0.7995 & 0.8979 \end{pmatrix}$	7
QDA	0.9368	$\begin{pmatrix} 0.9683 & 0.0789 \\ 0.0317 & 0.9211 \end{pmatrix}$	19
SVM	0.6684	$\begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix}$	4
KSVM	0.4682	$\begin{pmatrix} 0.9921 & 0.7915 \\ 0.0079 & 0.2085 \end{pmatrix}$	
ECA	0.9424	$\begin{pmatrix} 0.9439 & 0.0583 \\ 0.0561 & 0.9417 \end{pmatrix}$	15

Part of our obtained P and $\dot{L}$ is

$$P_{MNIST} = \begin{bmatrix} -0.00413941 & 0.00901582 & 0.04932139 & \ldots & 0.02962722 & 0.03481099 & -0.00057647 \\ 0.00875694 & 0.00608521 & 0.01776391 & \ldots & 0.06011203 & -0.00239336 & 0.01214911 \\ 0.02980248 & -0.05551565 & -0.05156798 & \ldots & 0.06693095 & 0.04165073 & 0.03867088 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -0.0261358 & 0.00822736 & -0.07249103 & \ldots & -0.00446923 & 0.09919091 & 0.01148881 \\ -0.00769167 & 0.02939839 & 0.04298031 & \ldots & -0.01120429 & -0.00835039 & 0.00816467 \\ -0.02147239 & -0.03735997 & -0.03697227 & \ldots & -0.00956103 & 0.02511721 & -0.00121021 \end{bmatrix}$$

and

$$\dot{L}_{MNIST} = \begin{bmatrix} 4.28501236e-16 & 3.17267414e-16 & \ldots & 4.07497572e-16 & 3.69587051e-16 \\ 2.94324218e-11 & 1.02724047e-12 & \ldots & 1.69453180e-08 & 1.83475474e-11 \\ 2.42868087e-11 & 6.55227218e-12 & \ldots & 9.99977827e-01 & 8.40801428e-10 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 8.78414095e-13 & 4.65978872e-13 & \ldots & 5.07113839e-12 & 9.25257024e-11 \\ 6.30129141e-13 & 3.91200852e-13 & \ldots & 6.63759282e-12 & 1.00000000e+00 \\ 7.75741233e-15 & 8.41619221e-15 & \ldots & 1.31394444e-14 & 1.69584344e-13 \end{bmatrix}$$

such that

$$\mathfrak{L}_{MNIST} = \lfloor \dot{L}_{MNIST} \rceil = \begin{bmatrix} 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 1 \\ 0 & 0 & \ldots & 0 & 0 \end{bmatrix}.$$

- VECA

  The $\mathfrak{L}$ learnt by VECA is extremely sparse, thus our eigenfeature is rather abstract. Using VECA, we achieved an validation accuracy of 90.48%. The reason that this accuracy is lower than that obtained by AECA is that VECA is less tolerant to weak mapping between eigenfeatures and their class labels. VECA intends to learn each element of an ECMM as unambiguous as possible. Nevertheless, the most intriguing part is that we learnt only 110 pure eigenfeatures (PEs) (???????) attribute to the unambiguity. In ?? (i), we found the class '7' and the class '9' are both distant from distributions of other classes. However, this PE is unambiguously assigned to the class '9'. This phenomenon indicates that VECA might be less tolerant to weak mappings between eigenfeature and their class labels, which is consistent to our objective in development of this two models.
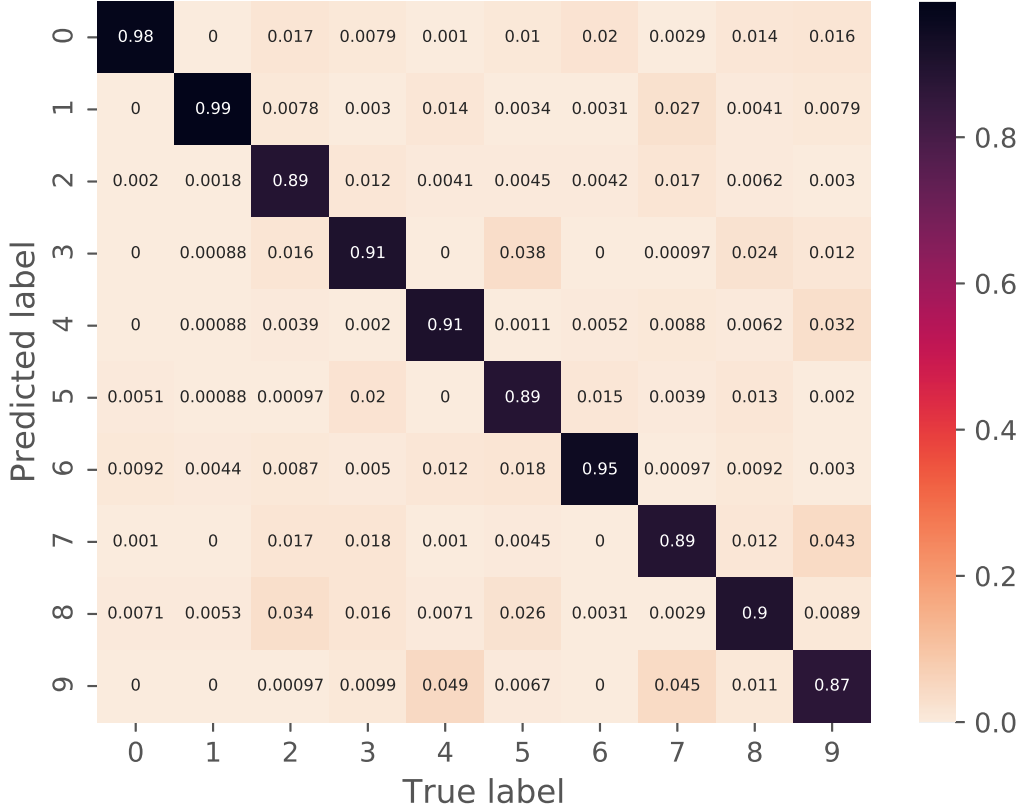
13

Figure 6: Confusion matrix of approximated eigen component analysis (AECA) model on MNIST data set (Accuracy of ECA, LDA, QDA are 0.918, 0.873, 0.144 respectively.)

- ECAN

  We implemented several 2-fold ECANs in this experiment. All these experiments are trained in 12 epochs. An indentity dimension operator is implemented if a rasing dimension operator (RaDO) or reducing dimension operator (ReDO) is not mentioned.

  Since the major task in this demonstatrion is extensionability, the margin of accuracy for parameter tuning is possible for suited dimension operator. As limited by finding the orthogonal dictionary and linear operation, the performance in prediction accuracy is marginally underperformed than standard DNNs. Also, in classical simulation, the training time is at least doubled than standard DNNs because the extra linear operation which won't be a problem on a quantum computer.

  The validation accuracy of the second fold achieved 94.58% when we implemented a specially designed ReDO (see Equation (59)) in that reduced the dimension to 128 in the first fold of ECAN.

  Moreover, a non-quadratic RaDO (see Equation (60)) and ReDO (see Equation (61)) has been implemented in neural network with rectified linear unit (ReLU) activation function which is on par with these quadratic operators. The accuracies of each folds get to 0.9686 and 96.83%, 91.43% and 94.66%, 96.6% and 97.6% for the three subexperiments with RaDO, ReDO, and both operators.

  Instead of RaDO or ReDO, we implemented fully connected neural networks as a dimension operator. The accuracies of each folds get to 96.33% and 96.37%, 91.29% and 94.52%, 96.6% and 96.87% with a fully connected neural network with 128 units implemented at the position of RaDO, ReDO and both place in the first fold.
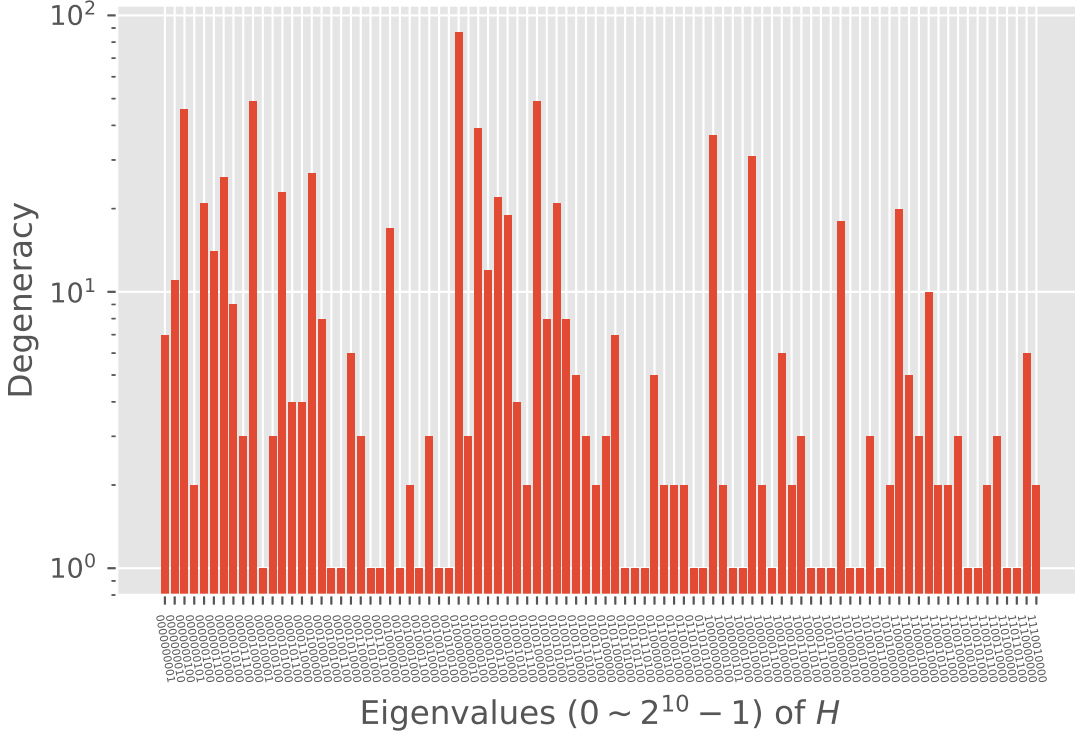
Figure 7: Degeneracy of all 98 distinctive eigenvalues of H on the MNIST data set using approximated eigen component analysis (AECA). The degeneracy of eigenvalue $(0000000000)_2$ is 0 which means there aren't eigenfeatures assume the similarities of the data set. The eigenvalue with the largest degeneracy (171) is $(0100000000)_2$, all corresponding to pure eigenfeatures (PEs) mapped to class label '8' ($\equiv \log_2(0100000000)$). The largest eigenvalue is $(1110010000)_2$ ($\equiv 912$) with degeneracy 2.

2.4.4 Two breast cancer data sets

This experiment used two data sets which could be used to illustrate the high interpretability of ECA. We analyzed two eigenfeature of the first data set to explain the meaning of what we've obtained.

In this two experiments, we used two data sets downloaded from UCI machine learning repository, which was originally obtained from the University of Wisconsin Hospitals, Madison by Dr. William H. Wolberg) One data set was published in 1992 (abbreviated as Wis1992) and the other one was in 1995 (Wis1995). ECA achieved the validation accracies of 90.04% and 94.14%, respectively, with all other a.fermentioned failed on these two data sets.

Wis1992   With VECA, we achieve an accuracy 90.04% (Table 3). The eigenvalue of H and its corresponding degeneracy is listed below.

Eigenvalue	Binary eigenvalue	Class label of PE	Degeneracy
0	00		5
1	01	$0 = \log_2(01)$	2
2	10	$1 = \log_2(10)$	2
3	11		0

In Wis1992, the 9 original features are 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape', 'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin', 'Normal Nucleoli' and 'Mitoses'. For this data set, the ECMM $\dot{L}$ we obtained is
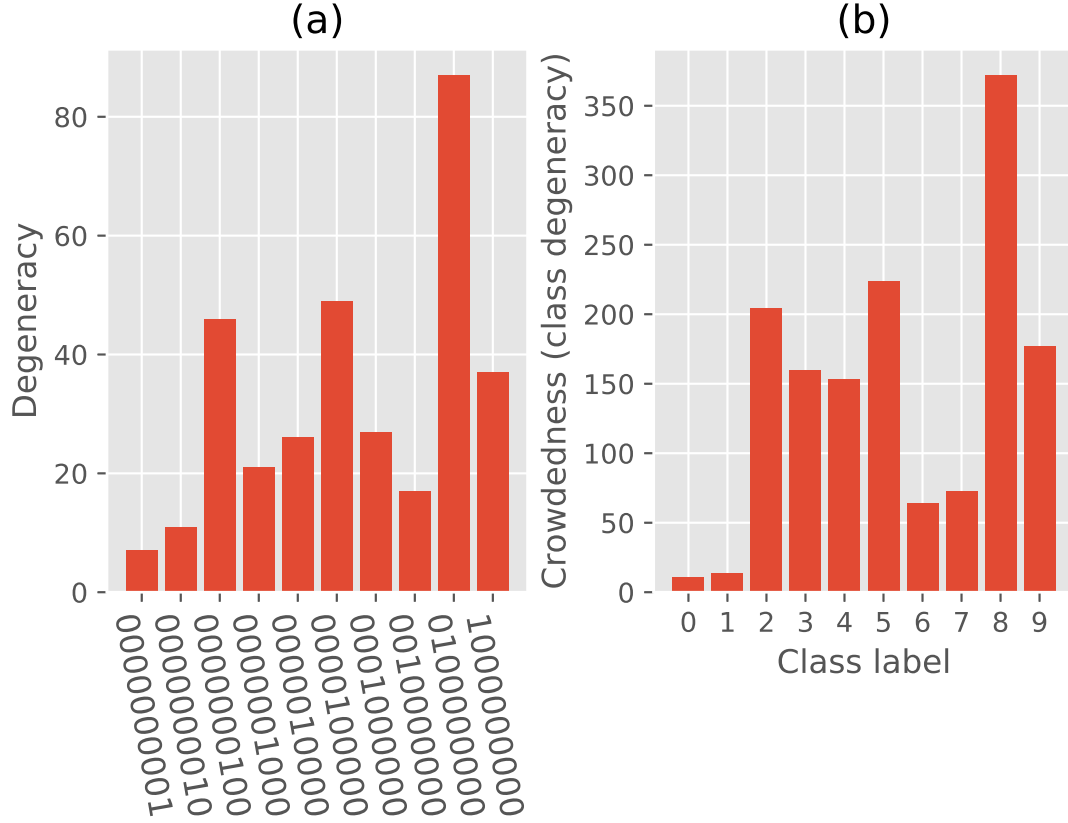
15

(a)

(b)

Eigenvalues of pure eigenfeatures (PEs) of H

Figure 8: (a) Degeneracy of pure eigenfeatures (PEs) of MNIST data set with approximated eigen component analysis (AECA); (b) Crowdedness of classes on MNIST data set with AECA.

Table 3: Comparison with other classifiers on the Wis1992 data set

Name	Metrics		Total Parameters
	Accuracy	Confusion Matrix	
LoR	0.3420	$\begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix}$	10
LDA	0.3420	$\begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix}$	19
QDA	0.3420	$\begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix}$	109
SVM	0.3420	$\begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix}$	10
KSVM	0.3420	$\begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix}$	
ECA	0.9004	$\begin{pmatrix} 0.9539 & 0.2025 \\ 0.0461 & 0.7975 \end{pmatrix}$	99

$$\dot{L}_{Wis1992} = \begin{bmatrix} 1.0 & 1.2813353e-21 \\ 1.544713e-21 & 1.0 \\ 1.0064267e-20 & 9.3284e-21 \\ 1.0 & 1.7838357e-15 \\ 4.8638498e-20 & 4.5091693e-19 \\ 1.3446889e-13 & 4.2358635e-17 \\ 2.2195558e-13 & 1.1143089e-15 \\ 2.4169718e-14 & 1.0 \\ 0.0019637463 & 1.0590642e-09 \end{bmatrix}$$
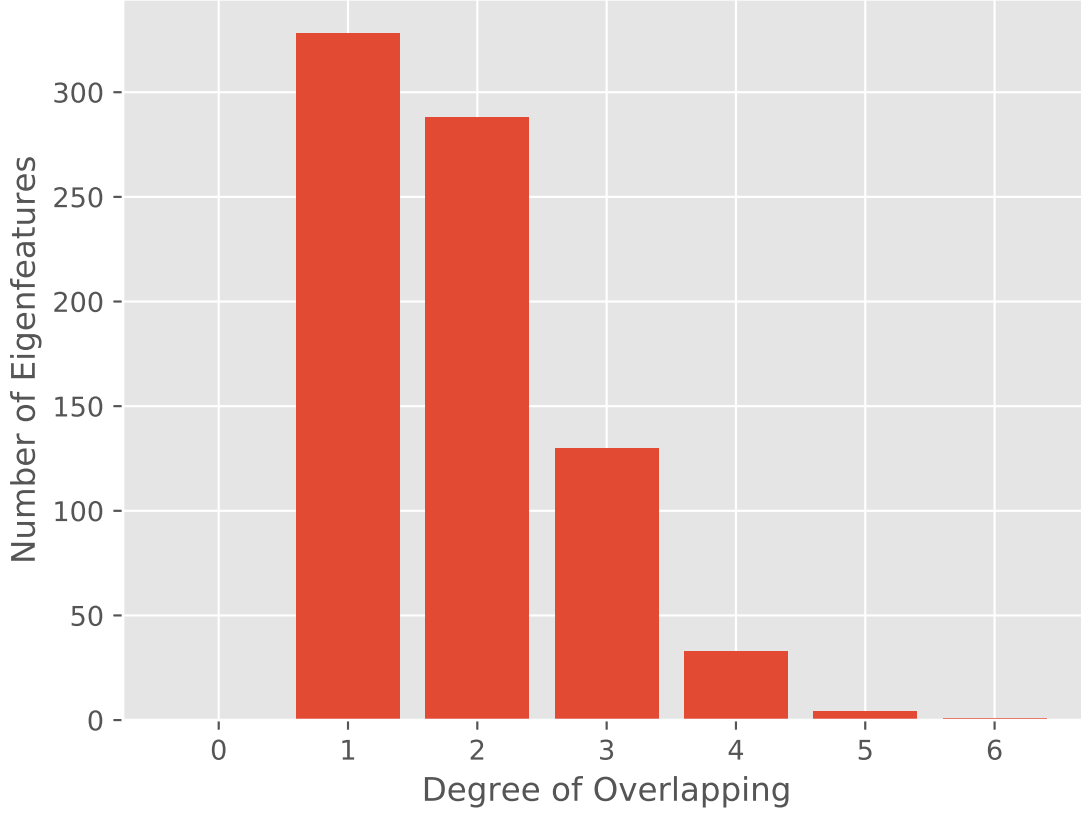
Figure 9: Overlapping of classes on eigenfeatures of MNIST data set with approximated eigen component analysis (AECA).

First, we choose the 0th eigenfeature from EFM P as obtained. With the ECMM, we know this eigenfeature is a PE mapping to the class '0' (i.e. 'benign' tumor). This eigenfeature and its squared value are

$$P_0 = \begin{bmatrix} 0.15118523 \\ 0.2486596 \\ -0.5196432 \\ -0.020636577 \\ 0.051793348 \\ -0.3954467 \\ 0.6642099 \\ 0.03258596 \\ -0.18750139 \end{bmatrix} \text{ and } (P_0)^2 = \begin{bmatrix} 0.022856973 \\ 0.061831594 \\ 0.27002904 \\ 0.00042586832 \\ 0.0026825508 \\ 0.15637809 \\ 0.4411748 \\ 0.0010618448 \\ 0.03515677 \end{bmatrix}$$

What's the meaning of P_0 and $(P_0)^2$? In analysis of a new input vector, we could use EFM. To analyze the eigenfeatures in EFM, we should use a special EFM, the identity matrix. First of all, P_0 is a paradigm or a textbook solution of 'benign' tumor indicator. For P_0, the value of its elements represent the relative intensity on each original feature. In more detail, high 'Bland Chromatin', and relatively high 'Clump Thickness' and 'Uniformity of Cell Size' with low 'Uniformity of Cell Shape','Bare Nuclei', and 'Mitoses' tend to be symptoms of 'benign' tumor. This 'benign' PE indicates how we take into each original feature into account when we make the decision of the tumor being 'benign'. Considering a special EFM $I_{9\times9}$, the 6th value in $(P_0)^2$ is 0.441748, which means one should take 44.17% into account of 'Bland Chromatin' together with 'Uniformity of Cell Shape' (27.00%) and 'Bare Nuclei' (15.64%).

17

Next, we take the 1st eigenfeature which is a 'malignant' PE into inspection. This eigenfeature and its squared value are

$$P_1 = \begin{bmatrix} 0.009413117 \\ -0.40779403 \\ 0.09775494 \\ 0.21722561 \\ 0.26575032 \\ 0.16900736 \\ 0.103898466 \\ -0.048212416 \\ -0.81856734 \end{bmatrix} \text{ and } (P_1)^2 = \begin{bmatrix} 8.8606765e-05 \\ 0.16629598 \\ 0.009556028 \\ 0.047186967 \\ 0.070623234 \\ 0.028563488 \\ 0.010794891 \\ 0.002324437 \\ 0.67005247 \end{bmatrix}$$

No doubt the 'Mitoses' is the factor that we should consider the most (67.01%) to decide a tumor being 'malignant', together with 'Clump Thickness'(16.63%). If a patient with less 'Mitoses' and 'Uniformity of Cell Size' and relatively high 'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', and 'Bland Chromatin', a 'malignant' diagnosis might be on the way.

Wis1995 With VECA, we achieved an accuracy 94.14% (Table 4). The eigenvalue of H and its corresponding degeneracy is listed below.

Eigenvalue	Binary eigenvalue	Class label of PE	Degeneracy
0	00		4
1	01	$0 = \log_2(01)$	17
2	10	$1 = \log_2(10)$	9
3	11		0

Table 4: Compare with other classifiers on Wis1995 data set

Name	Metrics		Total Parameters
	Accuracy	Confustion Matrix	
LoR	0.4043	$\begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix}$	31
LDA	0.4043	$\begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix}$	61
QDA	0.3032	$\begin{pmatrix} 0.0 & 0.25 \\ 1.0 & 0.75 \end{pmatrix}$	991
SVM	0.4043	$\begin{pmatrix} 0.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix}$	31
KSVM	0.5957	$\begin{pmatrix} 1.0 & 1.0 \\ 0.0 & 0.0 \end{pmatrix}$	
ECA	0.9414	$\begin{pmatrix} 0.9911 & 0.1316 \\ 0.0089 & 0.8684 \end{pmatrix}$	960

3 Discussions

We proposed a new quantum machine learning algorithm. This algorithm could be simulated on a classical computer. We used VECA for data classification which outperforms LoR, LDA, QDA, SVM and KSVM (with RBF kernel). One drawback of VECA is that it ignores the amplitude difference but focuses on the phase difference. The magnitude information was discarded in its classification performance, though, in practice, we found this magnitude information did not show substantial influence on the model. One solution to recover the lost information is to wrap the magnitude into our original vector. Other solutions include raising dimension before normalization or adopting a parallel fragment neural network (FNN) on magnitude (see more about this in **??**). With extension, our algorithm could work with amplitude difference along each eigenfeature. Thus a combination method based upon these two components is expected to build a more robust linear classifier. ECAN can further improve the performance by integrating with nonlinear

models such as deep networks. This method could be used in text classification and sentiment analysis as text usually has intricate linearity.

The advantages of ECA can be found in several aspects. First, as a classifier, any hyperplane separating data can be separated by ECA with no more than one auxilliary dimension, i.e. an extra dimension with unit or constant length (details omitted due to space limitation). In addition, for classification, ECA could process more than two classes simultaneously. Unlike PCA or LDA, ECA does not need to specify the number of dimensions for a lower dimensional feature space. The concrete number could be calculated from ECMM neither more nor less. Not only can ECA work as a good classifier, but it can also obtain a good dictionary.

Moreover, as this method is inspired by quantum mechanics, we introduce the concept of degeneracy in quantum mechanics as redundancy in machine learning problem. With degeneracy, we could also learn an undercomplete or overcomplete dictionary. With ECA, a complete dictionary could also be nontrivial. Indeed, the final dictionary (composed by PEs) will be a subset of all eigenfeatures based upon the obtained ECMM. Besides, this redundancy introduced could be not only used in our method, but also anywhere independently tackling a machine learning problem to avoid overfitting on linearity. In conclusion, ECA is an algorithm which deeply exploits the divide-and-conquer strategy.

4 Methods

We first develop a classical approximation of the algorithm which could be implemented on a classcial computer. Afterward, a quantum algorithm could be presented to implement on a quantum computer. We begin our development from the quantum intuition and fade out to the classical simulation followed by the full quantum algorithm.

For an observable, we could develop a 'machine' or apparatus to measure it. In quantum mechanics, such an observable or measurable are represented by linear operators H. These eigenvalues of the operator are the possible to-be-observed results of a measurement. Corresponding eigenvectors of these eigenvalues represents unambiguously distinguishable states. When we measure on an observable H at state A, the probability to observer value λ_j is given by

$$p(\lambda_j) = \langle A|\lambda_j \rangle \langle \lambda_j|A \rangle . \tag{18}$$

Now we define a new observable or measurable the *class-label* $\mathcal{H}$ and its corresponding linear operator H. In addition to all the principles or assumptions in quantum mechanics, there are two assumptions needed to be hold in the development of VECA:

- Assumption 1: Any system with a measurable *class-label* is a quantum system:

- Assumption 2:
    - On a quantum computer, for each class, l commutative measurable could be built for each class label;
    - On a classical computer, each class label of each eigenfeature follows independent Bernoulli distribution.

If we view our vector representation $|\mathsf{x}\rangle$ of a to-be-classified object $|\mathsf{o}\rangle$ as a quantum state, the measurement could be expressed as

$$\mathcal{H} |\mathsf{o}\rangle \tag{19}$$

or

$$H |\mathsf{x}\rangle = H \sum_{\lambda} \psi_\lambda |\lambda\rangle = \sum_{\lambda} \psi_\lambda \lambda |\lambda\rangle . \tag{20}$$

in which λ is the eigenvalue and $|\lambda\rangle$ is the eigenvector of H. We conduct a series of measuremnts on states x to obtain its correponding observed value y. Then we acquire a data set of measurement results on states $\{|x^{(i)}\rangle\}$ with observed values (class label) $\{y^{(i)}\}$ (where $i = 0, 1, \cdots, n-1$). One ideal situation for totally unambiguous states, the measurement on *class-label* could observe the class label (which is an integer). Hence the class label is the corresponding eigenvalue of H. That is

$$H |\mathsf{x}^{(i)}\rangle = y^{(i)} |\mathsf{x}^{(i)}\rangle \quad (i = 0, 1, \cdots, n-1). \tag{21}$$

For a data set with a full rank design matrix X whose states are unambiguous, we could obtain an analytic solution for Equation (21), such that

$$
\begin{aligned}
HX^T &= Y^T \star X^T \\
\Rightarrow HX^TX &= (Y^T \star X^T)X \\
\Rightarrow H &= (Y^T \star X^T)X(X^TX)^{-1}
\end{aligned}
\tag{22}
$$

in which Y is the vector of all the class labels and $\star$ is element-wise scalaring along columns. Given a new state $|x\rangle$, the prediction could be obtained by the rounded result of the expectation value of H given state $|x\rangle$. This analytic solution is implemented with performance over several classical algorithms mentioned in experiments section. However, one eigenstate (i.e. eigenfeatures) could be overlapped by several classes and states in one class could project on several eigenfeatures. What we could predict, in the measurement of $\mathcal{H}$ on a state $|x\rangle$, would be the expectation of the observable. With Equation (20), we have

$$
\langle H \rangle = \langle x|H|x\rangle = \sum_{\lambda} \lambda p_{\lambda}.
\tag{23}
$$

These eigenvalues crushed together, which cannot give us the information about the p.m.f. of how the input state would collapse on eigenstates. Hence it couldn't give us the information which the classes it belongs to. Thus for a l-class data set, we need to define l commutative operators

$$
H_0, H_1, \cdots, H_{l-1}.
\tag{24}
$$

To identify the class label, we need to know the p.m.f. p_{λ} firstly. For matrix form of H (we'll assume H is a matrix here and non-matrix H would be discussed in appendix), any Hermitian matrix could be diagonalized, such that

$$
H_k = P\Lambda_k P^{\dagger}, k = 0, 1, \cdots, l-1
\tag{25}
$$

in which Λ is composed by eigenvalues of H_k on its diagonal and unitary operator P is composed by simultaneous eigenvectors of all l operators H as its column vector. Thus, for all these l operators, we want to find a complete basis of simultaneous eigenvectors of all these l operators and the eigenvalues of each measurable.

Then, with (Equation (18)), the p.m.f. of collapsing on each eigenstate $|\lambda\rangle$ given a state $|x\rangle$ is

$$
p(|\lambda\rangle \,|\, |x\rangle) = (\langle x|\, P)^* \odot (\langle x|\, P)
\tag{26}
$$

To identify the unambiguous relationship between eigenfeature and class label, we assume the Bernoulli random variable y_k if one eigenfeature or vector is belonging to a class k follows independent Bernoulli distributions (development based upon a categorical distribution assumption would be attached in appendix). Thus, the p.m.f. of the classes to which an eigenfeature is belonging could be described as

$$
p_{k,\lambda}(y_k = 1|\,|\lambda\rangle).
\tag{27}
$$

Hence , by the principle of superposition, the p.m.f. of y_k given $|x\rangle$ which is the decision if one state belongs to one class k would be

$$
p_k(y_k = 1|\,|x\rangle) = \sum_{\lambda} p_{k,\lambda}(y_k|\,|\lambda\rangle)p(\lambda|\,|x\rangle)
\tag{28}
$$

For all classes, the matrix composed by stacked or combined p.m.f. of y with combined Bernoulli random variables y_k given $|x\rangle$ could be denoted as

$$
\mathbb{p}(y|\,|x\rangle) = \begin{pmatrix}
\sum_{\lambda} p_{0,\lambda}^T(y_0 = 1|\,|\lambda\rangle)p(\lambda|\,|x\rangle) & \sum_{\lambda} p_{0,\lambda}^T(y_0 = 0|\,|\lambda\rangle)p(\lambda|\,|x\rangle) \\
\sum_{\lambda} p_{1,\lambda}^T(y_1 = 1|\,|\lambda\rangle)p(\lambda|\,|x\rangle) & \sum_{\lambda} p_{1,\lambda}^T(y_1 = 0|\,|\lambda\rangle)p(\lambda|\,|x\rangle) \\
\vdots & \vdots \\
\sum_{\lambda} p_{l-1,\lambda}^T(y_{l-1} = 1|\,|\lambda\rangle)p(\lambda|\,|x\rangle) & \sum_{\lambda} p_{l-1,\lambda}^T(y_{l-1} = 0|\,|\lambda\rangle)p(\lambda|\,|x\rangle)
\end{pmatrix}_{l\times 2} = \sum_{\lambda} \mathbb{p}_{\lambda}(y|\,|\lambda\rangle)p(\lambda|\,|x\rangle)
\tag{29}
$$

in which the bold font $\mathbb{p}(\cdot)$ indicates a matrix and the subscript indicates the size of the vector or matrix.

Furthermore, the mapping between eigenfeature and class label follow the rule of winner-take-all, i.e the probability would be rounded to 0 or 1, such that

$$p_k(\mathsf{y}_k = 1 | |\mathsf{x}\rangle) = \sum_\lambda \lfloor p_{k,\lambda}(\mathsf{y}_k | |\lambda\rangle) \rceil p(\lambda | |\mathsf{x}\rangle) \tag{30}$$

and

$$\mathbb{p}(\mathsf{y} | |\mathsf{x}\rangle) = \sum_\lambda \lfloor \mathbb{p}_\lambda(\mathsf{y} | |\lambda\rangle) \rceil p(\lambda | |\mathsf{x}\rangle). \tag{31}$$

in which $\lfloor \cdot \rceil$ is round operation.

Then we put the rounded distribution of y_k given $|\lambda\rangle$ together to form a $m \times l$ matrix $\mathfrak{L}$. By substituting Equation (26) into eq. (29), the combined probabilities of y given $|\mathsf{x}\rangle$ could be written in

$$\mathbb{p}(\mathsf{y} | |\mathsf{x}\rangle; \theta) = \left\{ \left\{ [(\langle\mathsf{x}| P)^* \odot (\langle\mathsf{x}| P)] \mathfrak{L} \right\}^T \quad \left\{ [(\langle\mathsf{x}| P)^* \odot (\langle\mathsf{x}| P)](\mathbb{1}_{m \times l} - \mathfrak{L}) \right\}^T \right\}_{l \times 2} \tag{32}$$

in which θ denotes all the unknown parameters and $\mathbb{1}_{m \times l}$ is a $m \times l$ all-ones matrix. When one eigenfeature only belongs to one class, the corresponding row of $\mathfrak{L}$ is a bitwise representation of the binary digit of the class label. With these eigenfeature overlapped by several calsses, we define its eigenvalue the binarized number of the corresponding reversed row of $\mathfrak{L}$. Hence, we define

$$\Lambda \equiv Diag(\mathfrak{L}_{\overleftarrow{2}}). \tag{33}$$

in which the subscript $\overleftarrow{2}$ means reversely binarizing the row vector of $\mathfrak{L}$ and $Diag(\cdot)$ denotes the operation diagnalizing a vector into a matrix with elements on diagonal and otherwise 0. With Λ or $\mathfrak{L}$, in classical simulation, the l commutative operators $(H_0, H_1, \cdots, H_{l-1})$ could be combined into a single operator H such that

$$H = P \Lambda P^\dagger \tag{34}$$

In the example of the 2D and 3D data set (see Figure 1 and section 2.3), the corresponding separator has eigenvalues $\left\{ (01)_2, (10)_2 \right\}$ and $\left\{ (10)_2, (01)_2, (11)_2 \right\}$ respectively, in which $(\cdot)_2$ indicates that the number or the number in a set is binary. Correspondingly, we also convert the class labels as reversely bitwise view of its one-hot vector expression. The conversion is depicted as

$$\mathcal{C} = \{0, 1, \cdots, l-1\} \Rightarrow \begin{Bmatrix} \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^T, \\ \begin{bmatrix} 0 & 1 & \dots & 0 \end{bmatrix}^T, \\ \vdots, \\ \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}^T \end{Bmatrix} \Rightarrow \tag{35}$$

$$\mathcal{C}_2 = \left\{ (000\cdots001)_2, (000\cdots010)_2, \cdots, (100\cdots000)_2 \right\}.$$

with which we could easily to determine which classes an eigenfeature belongs to. These eigenfeatures that only belongs to one class are called PE. In our terminology, PE are 1° eigenfeature in which the superscript k^o denotes k degree of overlapping. For a PE $\mathfrak{f}$ corresponding to eigenvalue λ, the class label c of it could be calculated trivially such that

$$c_\mathfrak{f} = \log_2(\lambda). \tag{36}$$

Therefore, to predict the measurements on measurable *class-label* $\mathcal{H}$, instead of learning H directly (to build neural networks simulating a function H to represent $\mathcal{H}$), we could learn P and Λ (i.e. $\mathfrak{L}$) to construct our H in a classical simulation. In the quantum algorithm, a subtle difference is that H could be learned directly with a quantum computer.

Given a data set $\{(\mathsf{x}^{(i)}, \mathsf{y}^{(i)}), i = 0, 1, \cdots, n - 1\}$, with Equation (30), we could denote the combined probabilities of observing $\mathbb{1}_{l \times 1}$ given $\mathsf{x}^{(i)}$ as

$$\mathbf{p}(\mathsf{y} = \mathbb{1}_{l \times 1} | \mathsf{x}^{(i)}; \theta) = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{l-1} \end{bmatrix} \qquad (37)$$

and then with Equation (29) we have

$$\mathbb{p} = \begin{bmatrix} \mathbf{p} & \mathbb{1}_{l \times 1} - \mathbf{p} \end{bmatrix}. \qquad (38)$$

As we assume the decision on each class label of each eigenfeature are independent, the probability of a measurement of $\mathsf{y}^{(i)}$ given $\mathsf{x}^{(i)}$ is

$$p(\mathsf{y}^{(i)} | \mathsf{x}^{(i)}; \theta) = p(\mathsf{y}_0^{(i)}, \mathsf{y}_1^{(i)}, \cdots, \mathsf{y}_{l-1}^{(i)} | \mathsf{x}^{(i)}; \theta) = \prod_{k=0}^{l-1} p_k(\mathsf{y}_k^{(i)} | \mathsf{x}^{(i)}; \theta_k). \qquad (39)$$

Then the log-likelihood function is

$$\mathcal{L}(\theta; \mathsf{x}, \mathsf{y}) = \log(\prod_{i=0}^{n-1} p(\mathsf{y}^{(i)} | |\mathsf{x}^{(i)}\rangle; \theta)). \qquad (40)$$

To learn P and $\mathfrak{L}$, our objective could be

$$\arg\max_{P, \mathfrak{L}} \log(\prod_{i=0}^{n-1} p(\mathsf{y}^{(i)} | |\mathsf{x}^{(i)}\rangle; \theta)) \qquad (41)$$
$$\text{subject to } H^{\dagger} = H$$

in which the constraint on H is a shorthand for the constraints on P and $\mathfrak{L}$. Actually, the optimization of this objective is NP-hard.

By substituting Equation (39) into Equation (41), then with expansion and regrouping the objective could be simplified as

$$\begin{aligned}
\mathcal{L}(\theta; \mathsf{x}, \mathsf{y}) &= \log(\prod_{i=0}^{n-1} p(\mathsf{y}^{(i)} | |\mathsf{x}^{(i)}\rangle; \theta)) \\
&= \log\left(\prod_{i=0}^{n-1} \prod_{k=0}^{l-1} p_k(\mathsf{y}_k^{(i)} | |\mathsf{x}^{(i)}\rangle; \theta_k)\right) \\
&= \log\left(\prod_{i=0}^{n-1} \prod_{k=0}^{l-1} \prod_{\kappa=0}^{1} p_k(\kappa | |\mathsf{x}^{(i)}\rangle; \theta_k)^{1\{\kappa = \mathsf{y}_k^{(i)}\}}\right) \\
&= \sum_{i=0}^{n-1} \sum_{k=0}^{l-1} \sum_{\kappa=0}^{1} 1\{\kappa = \mathsf{y}_k^{(i)}\} \log p_k(\kappa | |\mathsf{x}^{(i)}\rangle; \theta_k) \\
&= \sum_{i=0}^{n-1} \sum_{k=0}^{l-1} \mathbb{1}^{T}\{\mathsf{y}_k^{(i)} = 1\} \log p_k(\mathsf{y}_k | |\mathsf{x}^{(i)}\rangle; \theta_k) \\
&= \sum_{i=0}^{n-1} \mathsf{y}^{(i)T} \log \mathbf{p}^{(i)} + (\mathbb{1}_{1 \times l} - \mathsf{y}^{(i)T}) \log(\mathbb{1}_{l \times 1} - \mathbf{p}^{(i)}) \\
&= \sum_{i=0}^{n-1} \text{Tr}(\mathsf{y}^{(i)T} \log \mathbb{p}^{(i)})
\end{aligned} \qquad (42)$$

22

in which the $\widetilde{\mathsf{y}}$ is the one's complement of each element of y, $\mathrm{Tr}(\cdot)$ is the trace of corresponding matrix and $\mathbf{p}^{(i)}$ and $\mathbb{p}^{(i)}$ are $\mathbf{p}(\mathsf{y}^{(i)} = \mathbb{1}_{l \times 1} | \, |\mathsf{x}^{(i)}\rangle ; \theta)$ and $\mathbb{p}(\mathsf{y} | \, |\mathsf{x}^{(i)}\rangle ; \theta)$ in short respectively.

Hence our objective becomes

$$\underset{P,\mathfrak{L}}{\arg\max} \sum_{i=0}^{n-1} \mathrm{Tr}(\mathsf{y}^{(i)T} \log \mathbb{p}^{(i)}) \tag{43}$$

$$\text{subject to } H^{\dagger} = H.$$

With Equation (18), we have

$$\underset{P,\mathfrak{L}}{\arg\min} - \sum_{i=0}^{n-1} \left\{ \mathsf{y}^{(i)T} \log\{[(\langle \mathsf{x}^{(i)}| \, P)^* \odot (\langle \mathsf{x}^{(i)\mathsf{T}}| \, P)]\mathfrak{L}\}^T \right.$$
$$\left. + (\mathbb{1}_{1 \times l} - \mathsf{y}^{(i)T}) \log\{[(\langle \mathsf{x}^{(i)}| \, P)^* \odot (\langle \mathsf{x}^{(i)}| \, P)](\mathbb{1}_{m \times l} - \mathfrak{L})\}^T \right\} \tag{44}$$
$$\text{subject to } H^{\dagger} = H.$$

The round operation is not differentiable. Thus, we replace $\mathfrak{L}$ with $\dot{L}$ which is sigmoid function of parameters L (Figure 10 (a)):

$$\dot{L}_{jk} = \frac{1}{1 + e^{-\chi L_{jk}}} \tag{45}$$

in which χ is the imambiguity factor to make the probability more concentrate on 0 or 1 and $\dot{(\,)}$ is the sigmoid function operator. The higher χ then the probability would be more concentrated on 0 or 1. Then the round operation on $\dot{L}$ would be less risky. As we want $\mathfrak{L}$ be a binary matrix, to make this constraint more neat, we add an auxiliary sinusoid function on L (Figure 10 (d)) such that

$$\dot{L}_{jk} = \frac{1}{1 + e^{-\chi \sin(\omega L_{jk})}}. \tag{46}$$

The sigmoid function could output 0 or 1 when the input approches $-\infty$ or $+\infty$ respectively. Nevertheless, the elements of $\dot{L}$ could never approch 0 or 1 theoretically. In practice, with a relatively large χ, $\dot{L}$ works as a good approximation of sigmoid function.

Thus, we replace the constraint on H with P and L. Our objective becomes

$$\underset{P,L}{\arg\min} - \sum_{i=1}^{n} \left\{ \mathsf{y}^{(i)T} \log\{[(\langle \mathsf{x}^{(i)}| \, P)^* \odot (\langle \mathsf{x}^{(i)\mathsf{T}}| \, P)]\dot{L}\}^T \right.$$
$$\left. + (\mathbb{1}_{1 \times l} - \mathsf{y}^{(i)T}) \log\{[(\langle \mathsf{x}^{(i)}| \, P)^* \odot (\langle \mathsf{x}^{(i)}| \, P)](\mathbb{1}_{m \times l} - \dot{L})\}^T \right\} \tag{47}$$
$$\text{subject to } (P^{\dagger}P)^* \odot (P^{\dagger}P) = I$$
$$\sin^2(L) = \mathbb{1}_{m \times l} \ (\text{i.e. } \cos(2L) = -\mathbb{1}_{m \times l}).$$

Then we obtain our objective function using Frobenius norm which is

$$\underset{P,L}{\arg\min} - \sum_{i=1}^{n} \left\{ \mathsf{y}^{(i)T} \log\{[(\langle \mathsf{x}^{(i)}| \, P)^* \odot (\langle \mathsf{x}^{(i)\mathsf{T}}| \, P)]\dot{L}\}^T \right.$$
$$+ (\mathbb{1}_{1 \times l} - \mathsf{y}^{(i)T}) \log\{[(\langle \mathsf{x}^{(i)}| \, P)^* \odot (\langle \mathsf{x}^{(i)}| \, P)](\mathbb{1}_{m \times l} - \dot{L})\}^T \right\}$$
$$+ \xi \|I - (P^{\dagger}P)^* \odot (P^{\dagger}P)\|_F^2 \tag{48}$$
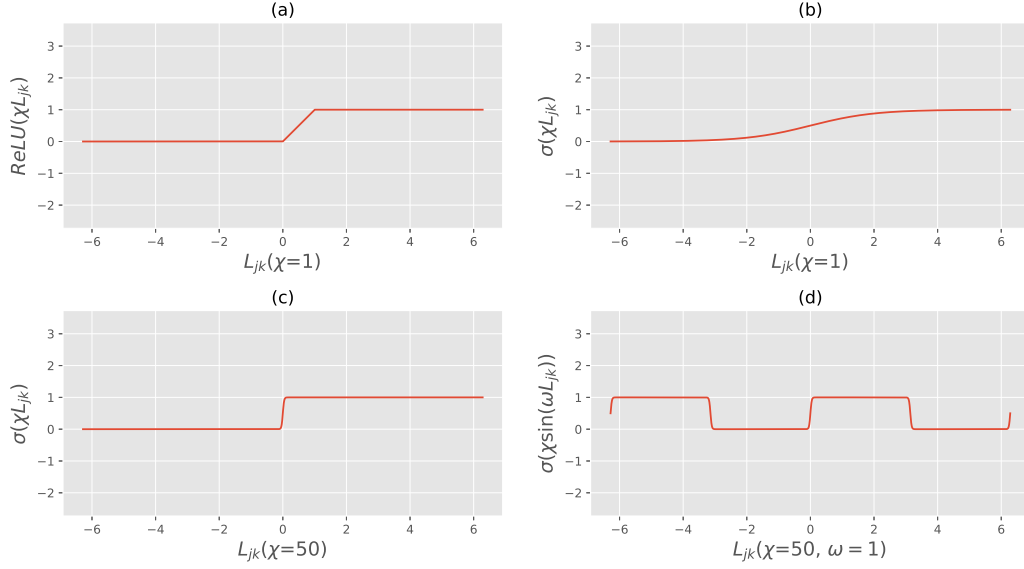$$+ \gamma \|\mathbb{1}_{m \times l} + \cos(2L)\|_F^2.$$

23

Figure 10: Several hypothesis of eigenfeature-class mapping matrix (ECMM). (a)ReLU function; (b)Sigmoid function; (c)Sigmoid function which is more concentrated on 0 or 1; (d)Sigmoid function on sinusoid which periodicly concentrates on 0 or 1.

Since most samples we meet are in real coordinate space, for simplicity and without loss of generality, we assume all the vectors would always project only on real space (i.e. the imaginary part always equals 0). The real version of our objective could be

$$
\begin{aligned}
\arg\min_{P,L} - \sum_{i=1}^{n} & \left\{ \mathsf{y}^{(i)T} \log\{[(\mathsf{x}^{(i)T}P) \odot (\mathsf{x}^{(i)\mathsf{T}}P)]\dot{L}\}^T \right. \\
& + (\mathbb{1}_{1\times l} - \mathsf{y}^{(i)T}) \log\{[(\mathsf{x}^{(i)T}P) \odot (\mathsf{x}^{(i)T}P)](\mathbb{1}_{m\times l} - \dot{L})\}^T \} \\
& + \xi \|I - P^T P\|_F^2 \\
& + \gamma \|\mathbb{1}_{m\times l} + \cos(2L)\|_F^2.
\end{aligned}
\tag{49}
$$

4.1 Approximation of eigen component analysis (ECA)

Furthermore, for a relative large data set, the combined probabilities of a combined random vector y of Bernoulli random variables y_k could be an approximation of the p.m.f. of y, such that these combined probabilities

$$
\mathbf{p}(\mathsf{y}^{(i)}|\mathsf{x}^{(i)};\theta)
\tag{50}
$$

could be used to estimate p.m.f. of y given x

$$
p(y|\mathsf{x}^{(i)};\theta).
\tag{51}
$$

24

The log-likelihood function becomes

$$
\begin{aligned}
\mathcal{L}(\theta; y, \mathsf{x}) &= \log \prod_{i=0}^{n-1} \prod_{i=k}^{l-1} p(k|\mathsf{x}^{(i)}; \theta)^{1\{k=y^{(i)}\}} \\
&= \sum_{i=0}^{n-1} \sum_{i=k}^{l-1} 1\{k = y^{(i)}\} \log p(k|\mathsf{x}^{(i)}; \theta) \\
&= \sum_{i=0}^{n-1} \mathbb{1}\{y = y^{(i)}\} \log p(y|\mathsf{x}^{(i)}; \theta). \\
&= \sum_{i=0}^{n-1} \mathsf{y}^{(i)} \log p(y|\mathsf{x}^{(i)}; \theta).
\end{aligned}
\tag{52}
$$

Then the objective of AECA could be written in

$$
\begin{aligned}
\arg\min_{P,L} - &\sum_{i=1}^{n} \mathsf{y}^{(i)T} \log\{[(\mathsf{x}^{(i)T}P) \odot (\mathsf{x}^{(i)\mathsf{T}}P)]\dot{L}\}^T \\
&+ \xi \|I - P^T P\|_F^2 \\
&+ \gamma \|\mathbb{1}_{m \times l} + \cos(2L)\|_F^2.
\end{aligned}
\tag{53}
$$

In default, the combined probabilities of y given x is

$$
\mathbf{p}(\mathsf{y}^{(i)} | \, |\mathsf{x}^{(i)}\rangle \,; \theta) = [(\langle \mathsf{x}^{(i)} | \, P)^* \odot (\langle \mathsf{x}^{(i)} | \, P)]\dot{L}.
\tag{54}
$$

As for a specific eigenvalue might be overlapped by more than one class, the probability on that eigenvector should be weighted. The modified probabilities or approximated p.m.f. is

$$
\mathbf{p}(\mathsf{y}^{(i)} | \, |\mathsf{x}^{(i)}\rangle \,; \theta) = [(\langle \mathsf{x}^{(i)} | \, P)^* \odot (\langle \mathsf{x}^{(i)} | \, P)]\dot{L}/o(\dot{L})
\tag{55}
$$

in which $o(\dot{L})$ is the degree of overlapping of each eigenfeature. With the modification, then the sum of probabilities in $\mathbf{p}$ could sum up to 1.

4.2 Raising and reducing dimension and eigen component analysis networks (ECANs)

In dealing with image or other interpolation tolerant data, the simplest method for rasing dimension might be resizing. For more advanced raising dimension stratergy, we introduce a RaDO and a ReDO. In the development of this section, the vector x are all originally non-normalized vector. A layer of nerual network with m-dimensional input and m' units of neurons is defined as

$$
W^T_{m' \times (m+1)} \begin{bmatrix} 1 \\ \mathsf{x} \end{bmatrix}_{(m+1) \times 1}
\tag{56}
$$

in which the extra 1 is viewed as an auxilliary dimension (we could call it the God-dimension because it provides the perspective over all other dimensions) with unit lenghth in ECA. Instead of neurons, all the units are viewed as dimensions in ECA. To deal with the dimension, in ECA, raising dimension is a process of unfold these wrapped dimension. Hence, this process is totally reversible as we project these unfolded dimension onto original dominated dimension. This is the same as to dimension reduction, with simple restriction, we combine several dimension whose corresponding eigenfeatures are degenerated. However, the restoration of reduced dimension could not guarantee intactness as the original one.

We could define our RaDO A that keep the original vector intact and unfold these wrapped dimensions, such that

$$
A\mathsf{x} = \begin{bmatrix} \mathsf{x} \\ f_1(\mathsf{x}) \\ f_2(\mathsf{x}) \\ f_3(\mathsf{x}) \\ \vdots \\ f_{m-\mathfrak{p}}(\mathsf{x}) \end{bmatrix}.
\tag{57}
$$

We could also learn these parameters for rasing dimension. For a RaDO A, it could be defined as

$$\mathsf{x}_{m\times 1} = A_{m\times \mathfrak{p}}\mathsf{x}_{\mathfrak{p}\times 1} = \begin{bmatrix} \mathsf{x}_{\mathfrak{p}\times 1} \\ A'_{(m-\mathfrak{p})\times(2\mathfrak{p}+1)} \begin{pmatrix} 1 \\ \mathsf{x}_{\mathfrak{p}\times 1} \\ \mathsf{x}^2_{\mathfrak{p}\times 1} \end{pmatrix} \end{bmatrix}, \quad m > \mathfrak{p}. \tag{58}$$

For dimension reduction operator D, that could be defined as

$$\mathsf{x}_{m\times 1} = D_{m\times \mathfrak{p}}\mathsf{x}_{\mathfrak{p}\times 1} = \sqrt{D'_{m\times(\mathfrak{p}+1)} \begin{bmatrix} 1 \\ \mathsf{x}^2_{\mathfrak{p}\times 1} \end{bmatrix}}, \quad m < \mathfrak{p} \text{ and } D' > 0. \tag{59}$$

Additionally, a non-quadratic RaDO could be defined as

$$\mathsf{x}_{m\times 1} = A_{m\times \mathfrak{p}}\mathsf{x}_{\mathfrak{p}\times 1} = \begin{bmatrix} \mathsf{x}_{\mathfrak{p}\times 1} \\ \mathrm{ReLU}\left(A'_{(m-\mathfrak{p})\times(2\mathfrak{p}+1)} \begin{pmatrix} 1 \\ \mathsf{x}_{\mathfrak{p}\times 1} \end{pmatrix} \right) \end{bmatrix}, \quad m > \mathfrak{p} \tag{60}$$

and a non-quadratic ReDO as well as a fully connected neural network could be

$$\mathsf{x}_{m\times 1} = D_{m\times \mathfrak{p}}\mathsf{x}_{\mathfrak{p}\times 1} = \mathrm{ReLU}\left(D'_{m\times(\mathfrak{p}+1)} \begin{bmatrix} 1 \\ \mathsf{x}_{\mathfrak{p}\times 1} \end{bmatrix} \right), \quad m < \mathfrak{p} \tag{61}$$

in which $\mathrm{ReLU}(\cdot)$ is the activation function of ReLU.

Moreover, we define two special dimension operator A_I and D_I with which

$$A_I\mathsf{x} = \mathsf{x} \text{ and } D_I\mathsf{x} = \mathsf{x}. \tag{62}$$

The reduced dimension could be dynmaicly adapted to the overlapping and crowdedness of ECMM. By raising dimension, these amplitude variances could be transformed into phase variances.

4.2.1 Eigen component analysis networks (ECANs)

In the development in this section, we also use the combined probabilities

$$\mathbf{p}(\mathsf{y}^{(i)}|\mathsf{x}^{(i)};\theta) \tag{63}$$

as an approximation of p.m.f. of y given x

$$p(y|\mathsf{x}^{(i)};\theta). \tag{64}$$

For each P, the corresponding state should be normailzed. That a τ-fold ($\tau = 0, 1, \cdots$) ECAN (see Figure 11) is defined as

$$f_\tau(|\mathsf{x}\rangle) = \prod_{t=1}^{\tau} D_t P_t^\dagger \widehat{A_t |\mathsf{x}\rangle} \tag{65}$$

$$\text{and } g_t(|\mathsf{x}\rangle) = P_t^\dagger \widehat{A_t |f_{t-1}(|\mathsf{x}\rangle)\rangle}, \text{ where } f_0(|\mathsf{x}\rangle) = |\mathsf{x}\rangle$$

in which $\widehat{(\cdot)}$ is the normalization operator. One subtle point that is worth noticing, in implementation this algorithm, the ReDO or RaDO should use the identity operator alternatively because two consequent dimension operators are trival.

Specifically, a 0-fold ECAN is

$$p(y|\,|\mathsf{x}\rangle\,;\theta) = [(\widehat{\langle\mathsf{x}|P_I})^* \odot (\widehat{\langle\mathsf{x}|P_I})]^T \mathfrak{L}_\tau \tag{66}$$
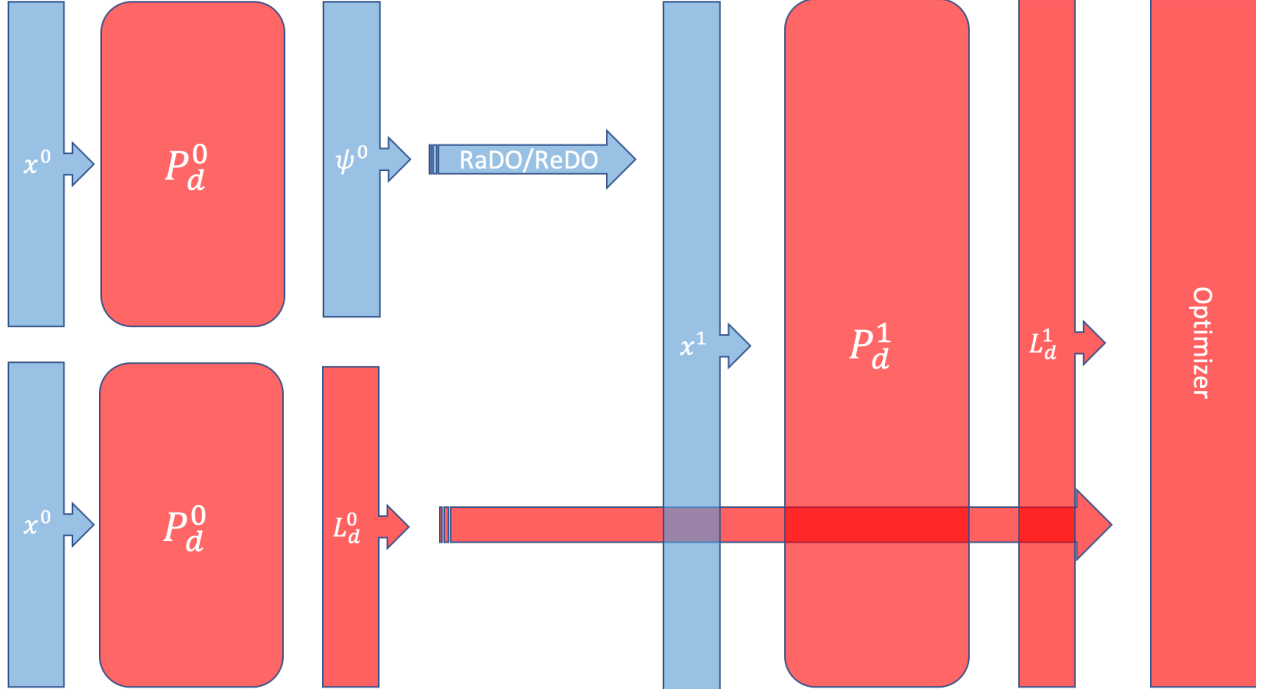
Figure 11: A 2-fold eigen component analysis network (ECAN) with a rasing dimension operator (RaDO) or reducing dimension operator (ReDO) in between.

in which P_I is an EFM of the identity matrix .

Thus, the p.m.f. of y given x is

$$p(y|\left|\mathsf{x}\right\rangle;\theta) = [g_\tau^*(\left|\mathsf{x}\right\rangle) \odot g_\tau(\left|\mathsf{x}\right\rangle)]^T \mathfrak{L}_\tau. \tag{67}$$

Then the objective becomes

$$\arg\min_{P,\mathfrak{L}} -\sum_{t=1}^{\tau} \pi_t \sum_{i=1}^{n} \mathsf{y}^{(i)\mathsf{T}} \log\{[g_t^*(\left|\mathsf{x}\right\rangle) \odot g_t(\left|\mathsf{x}\right\rangle)]^T \mathfrak{L}_t\}^T \tag{68}$$

$$\text{subject to } H_t^{\dagger} = H_t \text{ for } t = 1, 2, \cdots, \tau-1.$$

4.3 Generative model of eigen component analysis (ECA)

Not only ECA could be used to develop a model with generative learning interpretation, but also we could also build generative eigen component analysis network (GECAN).Empirically, We found the projections on the eigenfeature are drawn from multimodal normal distribution, such that

$$\left\langle \mathsf{x}|\lambda \right\rangle |y \sim \mathcal{M}(\mu_q, \sigma_q), q = 0, 1, \cdots \tag{69}$$

in which $\mathcal{M}$ denotes the multimodal distribution of several complex normal distributions with mean μ_q and variance σ_q for each. Indeed, for simplicity ,we could still assume independent complex normal distribution of projection on each eigenfeature for each class, such that

$$\left\langle \mathsf{x}|\lambda_i \right\rangle |y \sim \mathcal{N}_{\mathbb{C}}(\mu_{\lambda_i}, \sigma_{\lambda_i}) \tag{70}$$

or a real normal distribution

$$\left\langle \mathsf{x}|\lambda_i \right\rangle |y \sim \mathcal{N}_{\mathbb{R}}(\mu_{\lambda_i}, \sigma_{\lambda_i}). \tag{71}$$

We assume class label y follows a categorical distribution, such that

$$y \sim \text{Categorical}(\phi_k), \text{ for } k = 0, 1, \cdots, l-1. \tag{72}$$

For a classical generative model, the objective is

$$\arg\max_{y} p(\langle x|\,|y)p(y). \tag{73}$$

For the EFM P and ECMM $\mathfrak{L}$ to be determined, we denote the column vector of P and $\mathfrak{L}$ as P_j $(j = 0, 1, \cdots, m-1)$ and $\mathfrak{L}_k$ $(k = 0, 1, \cdots, l-1)$ respectively.

Given y, since we assume eigenfeatures are independent, the probabilty of x gcould be en y is a joint distribution which could be

$$
\begin{aligned}
p(\langle x|\,|y) &= p(\langle x|P_0\rangle, \langle x|P_1\rangle, \cdots, \langle x|P_j\rangle, \cdots, \langle x|P_{m-1}\rangle\,|y), \text{ for } j \in \{j|\mathfrak{L}_{jy} = 1, j = 0, 1, \cdots, m-1\} \\
&= \prod_{j=0}^{m-1} p(\langle x|P_j\rangle\,|y)^{\mathfrak{L}_{jk}}
\end{aligned}
\tag{74}
$$

Given the data set $\{(x^{(i)}, y^{(i)})\}$, by substituting Equation (74) into Equation (73), the log-likelihood function is given by

$$
\mathcal{L}(\mu, \sigma, P, L, \phi) = \log \prod_{i=0}^{n-1} \prod_{j=0}^{m-1} p(\langle x|P_j\rangle\,|y^{(i)})^{\mathfrak{L}_{jy^{(i)}}} \mathfrak{L}_{y^{(i)}} \phi_{y^{(i)}} \tag{75a}
$$

$$
= \sum_{i=0}^{n-1} \sum_{i=0}^{m-1} \mathfrak{L}_{jy^{(i)}} \log p(\langle x|P_j\rangle\,|y^{(i)}) + \sum_{i=0}^{n-1} \log \phi_{y^{(i)}} \tag{75b}
$$

$$
= \sum_{i=0}^{n-1} \sum_{i=0}^{m-1} \mathfrak{L}_{jy^{(i)}} \log \prod_{k=0}^{l-1} p(\langle x|P_j\rangle\,|k)^{1\{k=y^{(i)}\}} + \sum_{i=0}^{n-1} \log \prod_{k=0}^{l-1} \phi_k^{1\{k=y^{(i)}\}} \tag{75c}
$$

$$
= \sum_{i=0}^{n-1} \sum_{i=0}^{m-1} \mathfrak{L}_{jy^{(i)}} \sum_{k=0}^{l-1} 1\{k = y^{(i)}\} \log p(\langle x|P_j\rangle\,|k) + \sum_{i=0}^{n-1} \sum_{k=0}^{l-1} \log \phi_k^{1\{k=y^{(i)}\}} \tag{75d}
$$

$$
= \sum_{i=0}^{n-1} \sum_{i=0}^{m-1} \mathfrak{L}_{jy^{(i)}} \mathbb{1}^T \{y = y^{(i)}\} \log p(\langle x|P_j\rangle\,|y) + \sum_{i=0}^{n-1} \mathbb{1}^T \{y = y^{(i)}\} \log \phi \tag{75e}
$$

$$
= \sum_{i=0}^{n-1} \sum_{i=0}^{m-1} [\mathfrak{L}_j \odot \mathbb{1}^T \{y = y^{(i)}\}] \log p(\langle x|P_j\rangle\,|y) + \sum_{i=0}^{n-1} \mathbb{1}^T \{y = y^{(i)}\} \log \phi \tag{75f}
$$

$$
= \sum_{i=0}^{n-1} \mathbb{1}^T \{y = y^{(i)}\} \mathfrak{L}^T_{l \times m} \log p^T (\langle x|\,P|y)_{m \times l} \mathbb{1}\{y = y^{(i)}\} + \sum_{i=0}^{n-1} \mathbb{1}^T \{y = y^{(i)}\} \log \phi \tag{75g}
$$

$$
= \sum_{i=0}^{n-1} y^{(i)T} \mathfrak{L}^T_{l \times m} \log p^T (\langle x|\,P|y)_{m \times l} y^{(i)} + \sum_{i=0}^{n-1} y^{(i)T} \log \phi \tag{75h}
$$

in which the matrix operation in Equation (75h) increased the computation l^2 times futilely.

Then our objective could be

$$
\begin{aligned}
\arg\max_{\mu, \sigma, P, L, \phi} \quad & \mathcal{L}(\mu, \sigma, P, L, \phi) \\
\text{subject to } & H^\dagger = H.
\end{aligned}
\tag{76}
$$

In practice, these projections usually happen in real coordinate space. Thus, we substitute the real normal distribution

$$
p(x^T P_j|y = k; \mu_{jk}, \sigma_{jk}) = \frac{1}{\sigma_{jk}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x^T P_j - \mu_{jk}}{\sigma_{jk}}\right)^2} \tag{77}
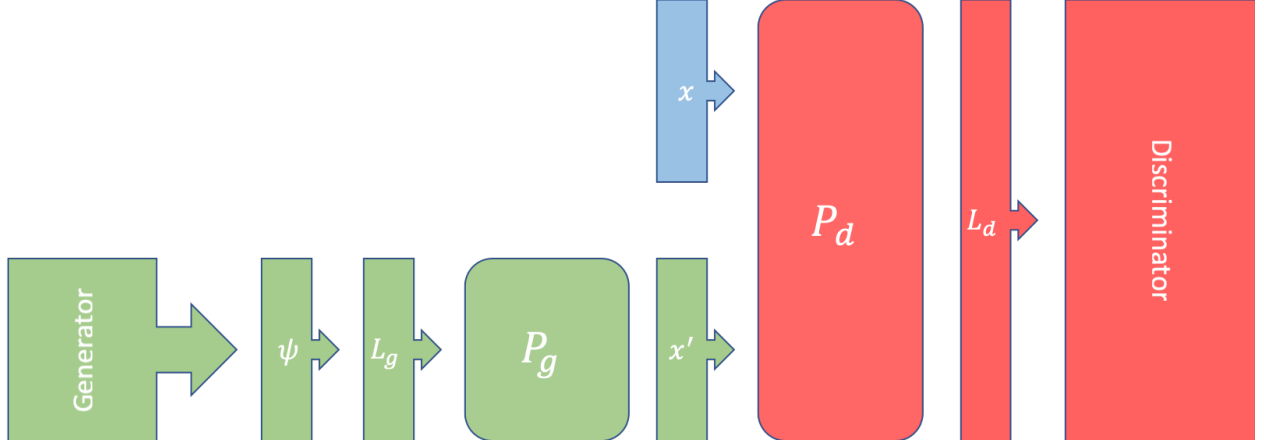$$

Figure 12: The architecture of an eigen component analysis-based generative adversarial network (ECAbGAN). Blocks in green belongs to generator. These blue blocks is related to ground truth and its evolution. The red blocks consists of the discriminator.

into Equation (75h). Thus, the log-likelihood function becomes

$$\mathcal{L}_{\mathbb{R}}(\mu,\sigma,P,L,\phi) = \sum_{i=0}^{n-1} \mathsf{y}^{(i)T} \mathfrak{L}^T{}_{l\times m} \log \left\{ \frac{1}{(\sigma \mathsf{y}^{(i)})^T \sqrt{2\pi}} \odot e^{-\frac{1}{2}\left((x^T P \ominus (\mu \mathsf{y}^{(i)})^T \odot \frac{1}{(\sigma \mathsf{y}^{(i)})^T} \right)^2} \right\}^T + \sum_{i=0}^{n-1} \mathsf{y}^{(i)T} \log \phi$$

(78)

in which μ and σ are $m \times l$ matrix.

Then our objective could be

$$\begin{aligned} \underset{\mu,\sigma,P,L,\phi}{\arg\min} \quad & -\mathcal{L}_{\mathbb{R}}(\mu,\sigma,P,L,\phi) \\ & + \xi \| I - P^T P \|_F^2 \\ & + \gamma \| \mathbb{1}_{m\times l} + \cos(2L) \|_F^2. \end{aligned}$$

(79)

By taking the gradient on ϕ and set that gradient to zero, we could obtain the solution of ϕ.

4.4 Eigen component analysis (ECA) and generative adversarial networks (GANs)

Learning cuisine in the field is way harder than in the kitchen of a top chef. In the field, one need to care about those ingridents as well as recipe. In the kitchen of a top chef, one is less risky to preparable an extremly terrible meal. As all the ingridents are well prepared, one can still 'mix' a not-that-bad meal even if they know nothing about cook. Hence, to learn cuisine, the only thing that one need to learn narrows to the recipe. Therefore, a cuisine learning problem could be devided into a ingredient learning and a recipe learning problem.

We could build up an eigen component analysis-based generative adversarial network (ECAbGAN) or eigen component analysis network-based generative adversarial network (ECANbGAN) which learns the ingredients firstly then the recipe (Figures 12 and 13).

4.5 Unsupervised eigen component analysis (UECA)

For unlabeled data set, we still want to find some userful structure. For a data set $\{\mathsf{x}^{(i)}, i = 0, 1, \cdots, n-1\}$, we want to fit a model of $p(x,z)$ to the data, in which z is the latent variable. Thus the log-likelihood function is

$$\begin{aligned} \mathcal{L}(\theta) &= \sum_{i=0}^{n-1} \log p(\mathsf{x}) \\ &= \sum_{i=0}^{n-1} \log \sum_z p(\mathsf{x}, \mathsf{z}; \theta) \end{aligned}$$
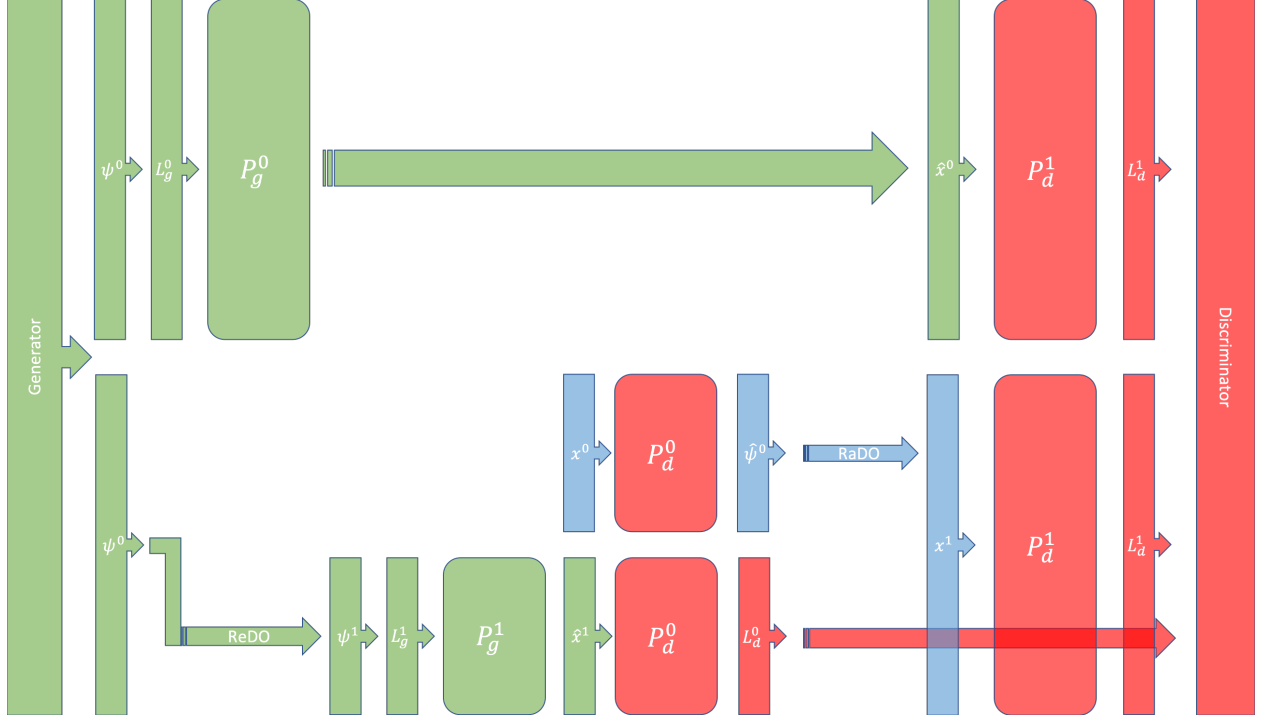
(80)

29

Figure 13: The architecture of a 2×2 eigen component analysis network-based generative adversarial network (ECANbGAN) was illustrated. In the first fold of eigen component analysis network (ECAN) in generator and discriminator, the implicit rasing dimension and dimension reduction operator are all identity operator.

To optimize this function, we could use Expectation-maximization (EM) algorithm. We assume z follows categorical distribution such that

$$z \sim \text{Categorical}(\phi_k), \text{ for } k = 0, 1, \cdots, \tilde{l} - 1, \tag{81}$$

in which $\hat{l}$ is a hyperparamter of to-be-classified classes in the data set.

Then the algorithm could be described as:

Repeat untill convergence {

(E-step) For each i, set

$$
\begin{aligned}
Q_i(z^{(i)}) :=& p(z^{(i)}|\mathsf{x}^{(i)}; P, L) \\
=& [\langle \mathsf{x}^{(i)}| \, P \odot \langle \mathsf{x}^{(i)}| \, P]^T \mathfrak{L}_{z^{(i)}}
\end{aligned}
\tag{82}
$$

(M-step)

$$
\begin{aligned}
Q_i(z^{(i)}) :=& \underset{\mu,\sigma,P,L,\phi}{\arg\max} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(\mathsf{x}^{(i)}, z^{(i)}; \mu, \sigma, P, L, \phi)}{Q_i(z^{(i)})} \\
& \text{subject to } H^\dagger = H \\
=& \underset{\mu,\sigma,P,L,\phi}{\arg\max} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(\mathsf{x}^{(i)}|z^{(i)}; \mu, \sigma, P, L) p(z^{(i)}; \phi)}{Q_i(z^{(i)})} \\
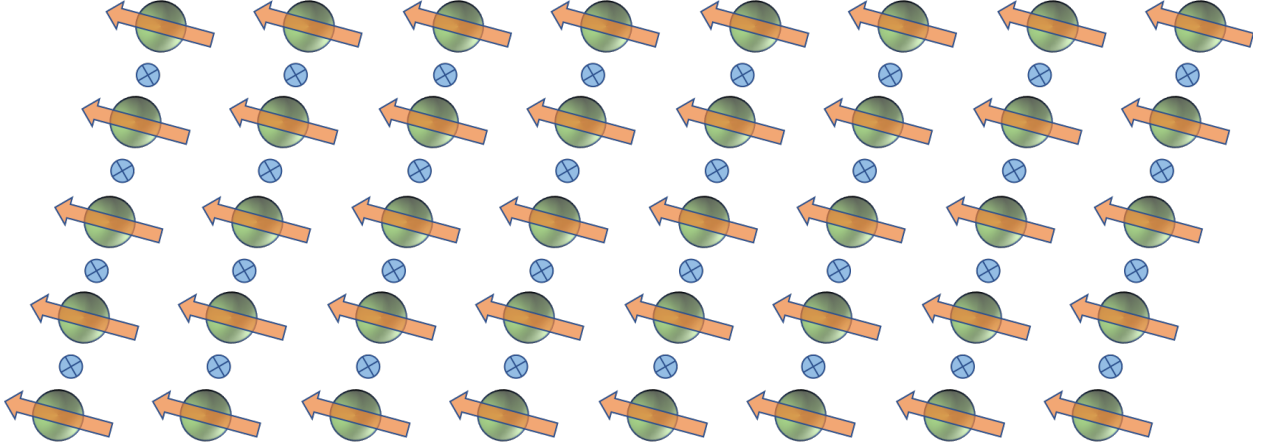& \text{subject to } H^\dagger = H
\end{aligned}
\tag{83}
$$

}

Figure 14: A prototype of a quantum computer for eigen component analysis (ECA) with 5×8 qubits. With this computer, a data set with 32-dimenional state vector and 8 classes could be parallelly processed. On the other hand, this computer could be used to process data with at most 2^{40}-dimensional state vector. This ajustment depends on the task and the blance between the cost for preparing a state and adding qubits.

4.6 Eigen component analysis (ECA) on a quantum computer

The quantum version of this algorithm is litte bit different from the classical version in the state representation and measurements. However, all the underneath motivation is the same. We need to build a 'machine' or appratus to finish the measurement. For a m-dimensional data, all of its states could be represented by $\lceil \log(m) \rceil$ qubits (for simplicity, we always assume $\log(m)$ is an integer). For a single qubit, the operator of its corresponding measurable H_b could be written in spectral decomposition with unitary matrix U_b and diagonal matrix Λ_b such that

$$H_b = U_b \Lambda_b U_b^{\dagger}. \tag{84}$$

To decide the measurable, we need to determine its corresponding unitary operator as well as the eigenvalues in Λ_b. For each qubit, the eigenvalues could be -1 or $+1$. The unitary operator of a qubit could be written in

$$U_b = e^{i\varphi/2} \begin{bmatrix} e^{i\varphi_0} \cos\theta & e^{i\varphi_1} \sin\theta \\ -e^{i\varphi_1} \sin\theta & e^{i\varphi_0} \cos\theta \end{bmatrix}. \tag{85}$$

Together with the parameters for two eigenvalues, for a single qubit, there are 6 parameters needed to be determined for a H_b in tatal (complex number counted as a single parameter).

For the composite system $\mathcal{H}$, its corresponding measurable H could be the tensor product

$$H = H_{b_0} \otimes H_{b_1} \otimes \cdots \otimes H_{b_{\log(m)-1}}. \tag{86}$$

On the other hand, we need l operators

$$H_0, H_1, \cdots, H_{l-1}$$

to measure on l classes. Actually, as we want to find a complete set of simultaneous vectors, these l operators commute because for $c, d = 0, 1, \cdots, l - 1$

$$[H_c, H_d] = H_c H_d - H_d H_c = U \Lambda_c U^{\dagger} U \Lambda_d U^{\dagger} - U \Lambda_d U^{\dagger} U \Lambda_c U^{\dagger} = 0$$

for which we could measure these l measurables in arbitary order.

The measurement of H could be the same as meansuring each subsystem H_b (i.e qubit). As each measurement destroys a measuring state, if not for no coloning theorem, we could take Q mesurements on each qubit and the observed value of the whole system could be caculated as product of the measurments of each qubit. Therefore, for '1' observed r times, the probability $\frac{r}{Q}$ could be used for optimizing the design of our measurable.

31

For a data set with l classes, measurement should be taked on each class with the same unitary operator but different eigenvalues. If we measure l classes parallely, all the qubits needed is $l \log(m)$. The concurrency could be further improved with another Q times qubits. The total number of parameters is $(4 + 2l) \log(m)$. One thing shoud be noticed, as theese l operators commute, that a prepared state could be reused for l times (if the number of qubits is limited and preparation is relatively expensive). A prototype of the quantum computer is depicted in Figure 14. Such a quantum computer could be used for processing a data set with 2^5 eigenstates and 8 classes parallelly.

Once the measurements finished, a classical computer is needed for optimizing these operators. Together with the data set, the corresponding probability $\frac{r}{Q}$ could be used for optimizing these parameters on a classical computer using any off-the-shelf optimizing algorithm.

To sum up, the algorithm could be described as

Repeat untill convergence {

(1) Initialize or update parameters of measurable $\mathcal{H}_b$ of each qubit for $l \log(m)$ qubits of all operators H for l measurables $\mathcal{H}$ on a classical computer. These parameters then could be used for buidling the 'machine' or apparatus for measurement on a quantum computer.

(2) Prepare the input vector as l identical copies of states on $l \log(m)$ qubits on a quantum computer.

(3) Take measurments on all qubits for Q times. For each measurable $\mathcal{H}$, the observed value is the product of all the $\log(m)$ qubits of that measurable. In the case of observing '1' for r times, then the l probabilities $\frac{r_k}{Q_k} (k = 0, 1, \cdots, l-1)$ are recorded.

(4) Together with the ground truth of the data set, these l probabilities $\frac{r_k}{Q_k} (k = 0, 1, \cdots, l-1)$ could be used for optimization of the parameters of operators for measurable $\mathcal{H}_b$ of each qubit on a classical computer with a classical optimization.

}

5   Acknowledgements

References

[1] Connor W Coley, Dale A Thomas, Justin AM Lummiss, Jonathan N Jaworski, Christopher P Breen, Victor Schultz, Travis Hart, Joshua S Fishman, Luke Rogers, and Hanyu Gao. A robotic platform for flow synthesis of organic compounds informed by ai planning. *Science*, 365(6453):eaax1566, 2019.

[2] Sebastian Steiner, Jakob Wolf, Stefan Glatzel, Anna Andreou, Jarosław M Granda, Graham Keenan, Trevor Hinkley, Gerardo Aragon-Camarasa, Philip J Kitson, and Davide Angelone. Organic synthesis in a modular robotic system driven by a chemical programming language. *Science*, 363(6423):eaav2211, 2019.

[3] Ashley R. G. Libby, Demarcus Briers, Iman Haghighi, David A. Joy, Bruce R. Conklin, Calin Belta, and Todd C. McDevitt. Automated design of pluripotent stem cell self-organization. *cels.*, 9(5):483–495.e10, 2019.

[4] Hongda Wang, Yair Rivenson, Yiyin Jin, Zhensong Wei, Ronald Gao, Harun Günaydın, Laurent A. Bentolila, Comert Kural, and Aydogan Ozcan. Deep learning enables cross-modality super-resolution in fluorescence microscopy. *Nat. Methods*, 16(1):103–110, 2019.

[5] Yichen Wu, Yair Rivenson, Hongda Wang, Yilin Luo, Eyal Ben-David, Laurent A. Bentolila, Christian Pritz, and Aydogan Ozcan. Three-dimensional virtual refocusing of fluorescence microscopy images using deep learning. *Nature Methods*, 16(12):1323–1331, 2019.

[6] Yair Rivenson, Hongda Wang, Zhensong Wei, Kevin de Haan, Yibo Zhang, Yichen Wu, Harun Günaydın, Jonathan E. Zuckerman, Thomas Chong, Anthony E. Sisk, Lindsey M. Westbrook, W. Dean Wallace, and Aydogan Ozcan. Virtual histological staining of unlabelled tissue-autofluorescence images via deep learning. *Nat. Biomed. Eng.*, 3(6):466–477, 2019.

[7] Michael Fernandez, Jose I Abreu, Hongqing Shi, and Amanda S Barnard. Machine learning prediction of the energy gap of graphene nanoflakes using topological autocorrelation vectors. *ACS combinatorial science*, 18(11):661–664, 2016.

[8] Alvin I. Chen, Max L. Balter, Timothy J. Maguire, and Martin L. Yarmush. Deep learning robotic guidance for autonomous vascular access. *Nature Machine Intelligence*, 2(2):104–115, 2020.

[9] Matthias Rupp. Machine learning for quantum mechanics in a nutshell. *International Journal of Quantum Chemistry*, 115(16):1058–1073, 2015.

[10] Peter Broecker, Juan Carrasquilla, Roger G Melko, and Simon Trebst. Machine learning quantum phases of matter beyond the fermion sign problem. *Scientific reports*, 7(1):8823, 2017.

[11] Ghanshyam Pilania, Chenchen Wang, Xun Jiang, Sanguthevar Rajasekaran, and Ramamurthy Ramprasad. Accelerating materials property predictions using machine learning. *Scientific reports*, 3:2810, 2013.

[12] Peter Sadowski, David Fooshee, Niranjan Subrahmanya, and Pierre Baldi. Synergies between quantum mechanics and machine learning in reaction prediction. *Journal of chemical information and modeling*, 56(11):2125–2128, 2016.

[13] O Anatole Von Lilienfeld. Quantum machine learning in chemical compound space. *Angewandte Chemie International Edition*, 57(16):4164–4169, 2018.

[14] KT Schütt, Michael Gastegger, Alexandre Tkatchenko, K-R Müller, and Reinhard J Maurer. Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions. *Nature communications*, 10(1):1–10, 2019.

[15] Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised generative modeling using matrix product states. *Physical Review X*, 8(3):031012, 2018.

[16] Mohammad H Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, and Roger Melko. Quantum boltzmann machine. *Physical Review X*, 8(2):021050, 2018.

[17] Masahide Sasaki and Alberto Carlini. Quantum learning and universal quantum matching machine. *Physical Review A*, 66(2):022303, 2002.

[18] Chun-Lin Chen and Dao-Yi Dong. Superposition-inspired reinforcement learning and quantum reinforcement learning. *IntechOpen*, 2008.

[19] Roger G Melko, Giuseppe Carleo, Juan Carrasquilla, and J Ignacio Cirac. Restricted boltzmann machines in quantum physics. *Nature Physics*, 15(9):887–892, 2019.

[20] Sankar Das Sarma, Dong-Ling Deng, and Lu-Ming Duan. Machine learning meets quantum physics. *arXiv preprint arXiv:1903.03516*, 2019.

[21] Ji-An Li, Daoyi Dong, Zhengde Wei, Ying Liu, Yu Pan, Franco Nori, and Xiaochu Zhang. Quantum reinforcement learning during human decision-making. *Nat. Hum. Behav.*, page 1–14, 2020.

[22] Isabelle Guyon and André Elisseeff. *An introduction to feature extraction*, pages 1–25. Springer, 2006.

[23] XiaoHong Han, Long Quan, XiaoYan Xiong, and Bing Wu. Facing the classification of binary problems with a hybrid system based on quantum-inspired binary gravitational search algorithm and k-nn method. *Eng. Appl. Artif. Intell.*, 26(10):2424–2430, 2013.

[24] Bahareh Nikpour, Mahin Shabani, and Hossein Nezamabadi-pour. Proposing new method to improve gravitational fixed nearest neighbor algorithm for imbalanced data classification. *2017 2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, page 6–11, 2017.

[25] Mahin Shabani-kordshooli, Bahareh Nikpour, and Hossein Nezamabadi-pour. An improvement to gravitational fixed radius nearest neighbor for imbalanced problem. *2017 Artificial Intelligence and Signal Processing Conference (AISP)*, page 262–267, 2017.

[26] Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R. Bach. Supervised dictionary learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1033–1040. Curran Associates, Inc., 2009.

[27] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Machine learning in a quantum world. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 431–442. Springer.

[28] Hafeez Ullah Amin, Aamir Saeed Malik, Rana Fayyaz Ahmad, Nasreen Badruddin, Nidal Kamel, Muhammad Hussain, and Weng-Tink Chooi. Feature extraction and classification for eeg signals using wavelet transform and machine learning techniques. *Australas. Phys. Eng. Sci. Med.*, 38(1):139–149, 2015.

[29] Kristin P Bennett and Olvi L Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, 1(1):23–34, 1992.

[30] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549:195–202, 2017.

[31] Youngjune Gwon, Miriam Cha, and H. T. Kung. Deep sparse-coded network (dsn). *2016 23rd International Conference on Pattern Recognition (ICPR)*, page 2610–2615, 2016.

[32] Sae Hwang, JungHwan Oh, Wallapak Tavanapong, Johnny Wong, and Piet C De Groen. Stool detection in colonoscopy videos. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3004–3007. IEEE.

[33] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[34] Julien Mairal, Francis Bach, and Jean Ponce. Task-driven dictionary learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):791–804, 2012.

[35] 2009.

[36] J Mairal, F Bach, J Ponce, G Sapiro, and A Zisserman. Learning discriminative dictionaries for local image analysis. In *26th IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[37] Angshul Majumdar and Rabab Ward. Robust greedy deep dictionary learning for ecg arrhythmia classification. *2017 International Joint Conference on Neural Networks (IJCNN)*, page 4400–4407, 2017.

[38] Olvi L Mangasarian, Rudy Setiono, and William H Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis. 1990.

[39] Olvi L Mangasarian and William H Wolberg. Cancer diagnosis via linear programming. Report, University of Wisconsin-Madison Department of Computer Sciences, 1990.

[40] Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. 2013.

[41] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

[42] Manel Martínez Ramón, Thomas Atwood, Silvio Barbin, and Christos G. Christodoulou. Signal classification with an svm-fft approach for feature extraction in cognitive radio. *2009 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*, page 286–289, 2009.

[43] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[44] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.*, 30(3):83–98, 2013.

[45] Vanika Singhal, Shikha Singh, and Angshul Majumdar. How to train your neural network with dictionary learning. *2017 Data Compression Conference (DCC)*, page 460, 2017.

[46] Snigdha Tariyal, Hemant Aggarwal, and Angshul Majumdar. Greedy deep dictionary learning for hyperspectral image classification. *2016 8th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, page 1–4, 2016.

[47] Ivana Tošić and Pascal Frossard. Dictionary learning. *IEEE Signal Process. Mag.*, 28(2):27–38, 2011.

[48] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J. Pal. Deep complex networks. 2017.

[49] William H Wolberg and Olvi L Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the national academy of sciences*, 87(23):9193–9196, 1990.

[50] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[51] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.