2021-11-26

Information Theoretic Error Bounds on NISQ Learning Systems

B.Tech Project I

Sankalp Gambhir

Email sgambhir@iitb.ac.in

Abstract Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Contents

1 Introduction

There has been long standing interest in constructing systems capable of learning from experience since even before computers in their modern form have existed. In the last few decades, with computing power skyrocketing exponentially coupled with leaping advances in theory of learning systems and statistical inference, these problems became tractable and eventually came into use ubiquitously. With applications ranging from facial detection systems for surveillence to identifying cosmic objects for cosmology, they have found widespread adoption in industry and academia. These systems circumvent the need to produce a precise mathematical model for the problem at hand, exploiting general techniques to instead infer a model from available data. With their advent, however, has come an ever rising need for computing power to facilitate their operation. This has found data centers of unprecendented scales consuming enormous amounts of power to provide the instant predictions we've come to rely on.

With snowballing energy and space requirements of classical computers in the form of GPU clusters and Application Specific Integrated-Circuits (ASICs), there has been a spark of interest in offloading this computation onto quantum computers, which, till recently, have largely remained a rare species spotted only in labs surrounded by helium-cooled superconductors and white-coated predators. Current scales of available quantum computers, however, still lack the power required to fully tackle these challenges while maintaining reliable error-levels or adding their own error checking and correction. This has motivated using quantum computers to run bottnecked computational subroutines with classical control systems. These systems generally lack error correction, and thus earn themselves the title of 'noisy'. These form the basis of computation considered in this thesis, Noisy Intermediate-Scale Quantum (NISQ) computers.

Connecting a quantum computer to a classical puppeteer is not expected to come without its own issues either. It constrains the architecture and is itself bottlenecked on both ends, first by the parameter transfer and configuration from the classical to the quantum, then finally by the detectors on the quantum side to the classical. In this thesis, we focus on the former, discussing the limits of computation and computational precision achievable with this hybrid architecture.

1.1 Structure

In section 2, definitions and relevant results in classical computing, physics, and quantum information are presented. [Note: Extend this.]

1.2 Outline of New Results

[Note: Add summary of results at the end.]

2 Preliminaries

2.1 Classical Computing

2.1.1 Learning Problem

Learning [1] can be broadly defined as attempting to learn the input-output pattern given sample data. For this thesis, we consider three major categories of learning problems:

- Binary Classification — input points in a chosen domain, and a binary output label for each point.
- Multi-Label Classification — input points in a chosen domain, and one of n labels as output for each point.
- Regression — input points in a chosen domain with real-valued output.

2.1.2 Classification Problem

We take as input elements $\{x_i\}$, generally called *feature vectors*, in a chosen domain X called the *feature space* and output an element from a finite set $L = l_i$ of labels. [Note: Add a nice picture]

The problem is called binary classification if $|L| = 2$.

Formally, we attempt to learn a function $f : X \to L$ given a set of inputs in the domain, and possibly paired output labels.

The problem proceeds in two manners given the form of inputs: if provided input-output pairs, the problem is called a *supervised learning problem*, while attempting to learn a set of labels given just (clustered) inputs is called *unsupervised* learning. We focus on supervised classification here.

The set of input-output pairs provided is called the *training data*.

Given the difficulty of working with discretized domains, the input domain is generally converted to be a subset of a Euclidean space, using a suitable *embedding function*.

2.1.3 Embedding

An embedding of X in Y is a function $f : X \to Y$ that is injective and structure-preserving. The exact restrictions on the map to be structure-preserving depend on the structures of the domain and the codomain [2]. It is denoted here as $f : X \hookrightarrow Y$.

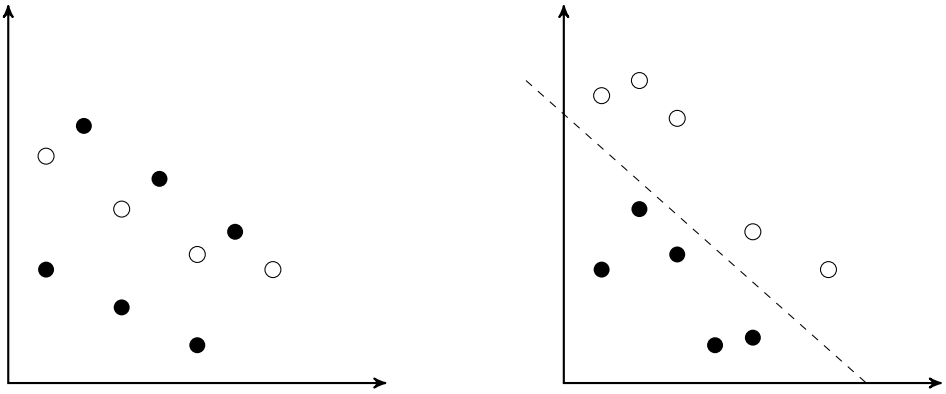
For example, a topological embedding, i.e., the embedding of a topological space, will be restricted to preserve its associated structure of open sets. A field embedding, similarly, will be restricted to preserve the field operations + and $\times$.

For a given arbitrary feature space X, it is generally embedded into $\mathbb{R}^n$ for some n.

2.1.4 Linear Classification

Classification generally proceeds by producing linear functions as candidate (supplemented with a discretization function) labelling functions and fitting them to the training data. For simplicity, we first restrict the discussion to binary classifiers.

Given a set of points which, due to an embedding, may be assumed to be in $X \subseteq \mathbb{R}^n$, attempting to classify them may still be an arduous task if the spatial regions corresponding to the labels are intertwined. Thus, to make the problem tractable, we restrict the data to be *strongly* separated, i.e., assume that there always exists a set hyperplanes (of size $|L| - 1$) that isolates the points with each label within the domains created by intersection. In the binary case, this is just one label on either side of the hyperplane.



(a) *Unseparated data* **(b)** *Strongly separated data*

Figure 1: *The magic of the (strong) separation axiom.*

2.1.5 Support Vector Machine

A support vector machine is a classifier model which constructs a hyperplane or a set of hyperplanes in the feature space optimising classifier separation depending on the objective [3].

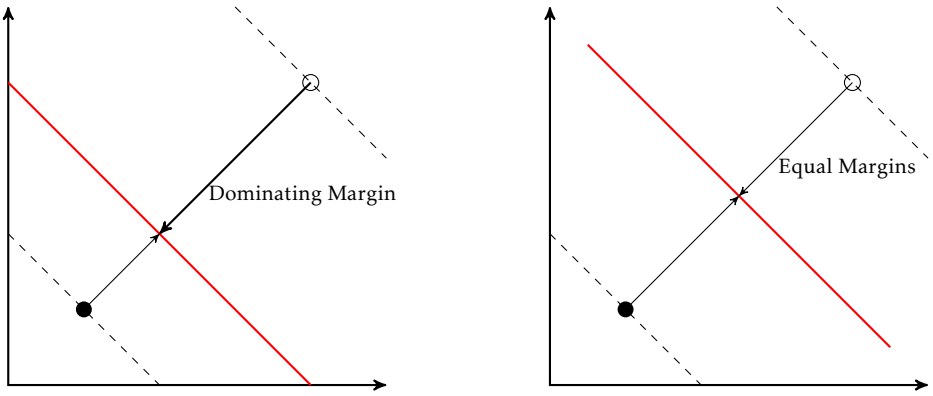
We will synonymously use the terms 'Support Vector Machine' and that of its common model 'Maximal Margin Classifier', which is more appropriately what we use here.

As the name suggests, a maximal margin classifier SVM tries not only to construct a set of hyperplanes, but to find the set such that their margin from the data is maximised. This builds upon the intuitive idea of a good separator being further away from the given data points. See Figure 2.

Formally, we characterize a hyperplane in $\mathbb{R}^n$ as a pair $(\boldsymbol{w}, b)$, with $\boldsymbol{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, such that for all points $\boldsymbol{x}$ on the hyperplane

$$\langle \boldsymbol{w} \cdot \boldsymbol{x} \rangle + b = 0 \,.$$

Geometrically, $\boldsymbol{w}$ is the vector normal to the hyperplane, and b is the bias or offset from origin.



(a) *Unoptimized margins* **(b)** *Maximal margin classifier*

Figure 2: *Illustration of different margins for hyperplanes.*

Note that by moving to $\mathbb{R}^{n+1}$, we can convert the hyperplane to one without bias (passing through the origin)

$$\langle \boldsymbol{w} \cdot \boldsymbol{x} \rangle + b = 0 \,,$$
$$\langle (\boldsymbol{w} \oplus \begin{pmatrix} b \end{pmatrix}) \cdot (\boldsymbol{x} \oplus \begin{pmatrix} 1 \end{pmatrix}) \rangle + 0 = 0 \,.$$

which is the hyperplane $(\boldsymbol{w} \oplus \begin{pmatrix} b \end{pmatrix}, 0)$ in $\mathbb{R}^{n+1}$. So, without loss of generality, we work with hyperplanes without bias.

Now, given the training dataset $(\boldsymbol{x_i}, y_i)$, with $y_i = \pm 1$, we can write constraints on $\boldsymbol{w}$ as

$$\forall i \; y_i \cdot \langle \boldsymbol{w} \cdot \boldsymbol{x_i} \rangle > 0 \,, \tag{1}$$

that is, x_i is on the same side of the hyperplane as indicated by y_i as the sign of the inner product corresponds to the same.

By scaling $\boldsymbol{w}$ (without changing the hyperplane), we can construct the constraint system

$$\forall i \; y_i \cdot \langle \boldsymbol{w} \cdot \boldsymbol{x_i} \rangle \geq 1 \,. \tag{2}$$

Since we are scaling $\boldsymbol{w}$, we choose an appropriate optimization target, its norm.

Since this is a constrained optimization, we write its Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) = \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + \sum_i \alpha_i \left[y_i \cdot \langle \boldsymbol{w} \cdot \boldsymbol{x_i} \rangle \right] , \tag{3}$$

where $\{\alpha_i\}$ are the Lagrangian multipliers. For the optimal solution, the Lagrangian is stationary, i.e.,

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha})}{\partial \boldsymbol{w}} &= 0 , \\ \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x_i} &= 0 . \end{aligned} \tag{4}$$

Substituting this expression for $\boldsymbol{w}$ in the Lagrangian itself, we get [Note: work out the substitution]

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) = \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + \sum_i \alpha_i \left[y_i \cdot \langle \boldsymbol{w} \cdot \boldsymbol{x_i} \rangle \right] , \tag{5}$$

By maximising this Lagrangian with respect to the parameters $\{\alpha_i\}$, we obtain an optimal $\boldsymbol{w}$ which is the maximum margin classifier.

With $\boldsymbol{w}$ fixed at its optimal value, we get a simple computational method to classify all new incoming points $\boldsymbol{x} \in \mathbb{R}^n$, given by

$$\mathrm{sgn}(\langle \boldsymbol{w} \cdot \boldsymbol{x} \rangle) \tag{6}$$

returning a label ± 1 (or anomalously zero, if you happen to pick a point on the hyperplane, which can be remedied by making one side's boundary soft).

As the major 'quantum' modification, we will discuss how the bottlenecked linear algebra computation is offloaded to a quantum circuit in section 3.

2.1.6 Optimisation Techniques

The discussion of optimization techniques in classical computing is a long and arduous one. We refer the reader to a common text on the matter for a detailed discussion [Note: add some citations], while reviewing the general ideas briefly here.

In a NISQ system, there is generally little to no attempt at error correction, and the general goal is to capitalize on what is possible with the short available coherence times, without devoting a majority of the system's resources to error checking and correction. As such, these systems are unable to support high-depth circuits with computationally involved analytical gradient based approaches. An effective optimizer in control of such a temporally-bound circuit should try to utilize techniques minimizing the number of measurements or function evalutations, as the relevant modules generally form the bottleneck of the computation [see 4, chapter II.D].

2.2 Quantum Regime

[Note: add a general introduction]

2.2.1 Hilbert Space

Definition 1. *A Hilbert space is a vector space $\mathcal{H}$ equipped with an inner product $\langle f, g\rangle \; \forall f, g \in \mathcal{H}$ such that the norm defined by*

$$\|f\| = \sqrt{\langle f, f\rangle}$$

turns $\mathcal{H}$ into a complete metric space [5].

Physical quantities — such as energy, momentum, and position — are represented as operators over a Hilbert space $\mathcal{H}$ to which the wavefunctions belong [6].

Herein, the inner product is assumed to be linear in the second factor, i.e.,

$$\langle f, \lambda g\rangle = \lambda\langle f, g\rangle; \quad \langle \lambda f, g\rangle = \bar{\lambda}\langle f, g\rangle$$

$\forall f, g \in \mathcal{H}$ and λ in the base field $\mathbb{F}$. For our purposes, we will assume $\mathcal{H}$ to always have as its base field the field of complex numbers, $\mathbb{C}$.

We will assume relevant quantities to be linear operators $\cdot : \mathcal{H} \to \mathcal{H}$ with adjoints where necessary, [see 6, Appendix A] for details.

2.2.2 Schrödinger Equation

Given a *Hamiltonian* operator $\hat{H}$ for a system represented by a wavefunction ψ, its time evolution is given by the Schrödinger equation,

$$\frac{\mathrm{d}\psi}{\mathrm{d}t} = \frac{1}{\imath\hbar}\hat{H}\psi \,. \tag{7}$$

2.2.3 Quantum Computation

2.2.4 State Construction and Embedding

2.2.5 Measurement

2.2.6 Information Matrices and Distance Measures

Fubini-Study and Fisher Information, parametrized quantum states [7].

3 Variational Quantum Algorithms

With the goal of optimizing learning computations using quantum computers in mind, we need an abstract idea of how to implement this connection. A *Variational Quantum Algorithm* (VQA) is any such system based on a proposed architecture for a classically controlled quantum computer [4]. Figure 3 presents the proposed architecture. The following subsection presents an expanded view of the computation.

3.1 Building Blocks

A VQA computation has 4 major components, as shown in Figure 3:

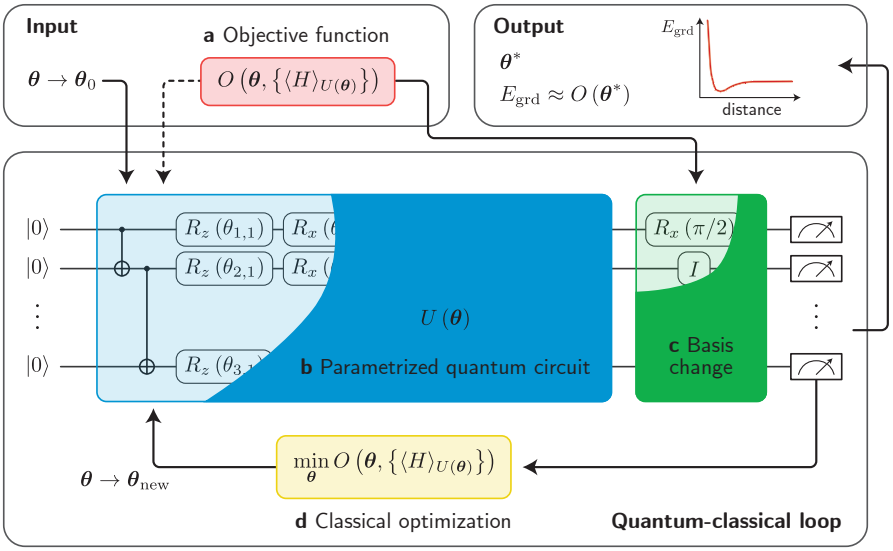


Figure 3: *Diagrammatic representation of a Variational Quantum Algorithm (VQA) [taken from 4, Figure 2].*

- objective function — the encoding of the problem at hand as an optimization,
- parametrized quantum circuit (PQC) — circuit encoding a unitary operator parametrized by classically controlled parameters $\boldsymbol{\theta}$,
- measurement scheme — the system performing basis changes and transferring outputs to the control system, and
- classical optimizer — a classical objective minimizer which controls the PQC.

These components form a modular computation model where each of the components can be swapped and improved individually to relieve bottnecks and adapt to the problem at hand, to control the expressiveness of the system or avoid treacherous optimization landscapes [8].

3.1.1 Objective Function

The *objective* or *loss function* [8] forms the target of the optimization problem at hand. This can be any function that can be encoded in an operational form, i.e.,

written as or decomposed into quantum operators. In most cases, this can be expected to be something akin to the Hamiltonian of a system [4], thus making the minimal, ground state, energy the optimization target. This may also be called the *parametrized cost* of the computation. Subject to the optimization constraints, the target of the system is then to find the optimal parameter input

$$\theta_* = \arg\min_\theta \mathcal{L}(\theta, p_0(\theta)) \,,$$

where $p_0(\theta)$ represents the parametrised probability to measure the output in the state $|0\rangle$.

3.1.2 Parametrised Quantum Circuits (PQCs)

The module central to the computation is the parametrised quantum circuit given by $U(\theta)$. It is the component of the circuit which performs the actual 'computation' and outputs the state that best meets the objective. It does so by acting on the input state a series of unitary transformations parametrised by controllable inputs. We assume the circuit to have an L-layered structure as

$$U(\theta) = \prod_{l=1}^{L} U_l(\theta_l), \quad U_l(\theta) = \prod_{k=1}^{K} e^{-\imath\theta_{lk}H_k} \,, \tag{8}$$

where the index l indicates the layer, and the index k spans the traceless Hermitian operators $\{H_k\}$ that generate the space of unitaries for the chosen ansatz. Here, θ decomposes as a set of vectors of parameters θ_l for each of the indexed layers, which in turn map to individual parametrised unitary actions indexed by k. Finally, $M = K \cdot L$ is the number of trainable parameters of the system [see 8, section II.A].

This general description of a PQC subsumes most ansatzes studied in literature [9]. These include the hardware-efficient ansatz [10], quantum alternating operator ansatz (QAOA) [11], Hamiltonian variational ansatz [12], quantum optimal control ansatz [13], among others [14, 15, 16]. These correspond to specific configurations of layer sizes and choices of the generators.

The choice of generators is intimately tied to the reachable states of the system, and the landscape needed to be traversed to get there. This is discussed in further detail in subsection 3.2.

Assuming for now that the space spanned by the generators contains our target unitary, we proceed with the discussion of the computation. After the application of the PQC, the initial state $|\Psi_0\rangle$ is transformed as

$$|\Psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\Psi_0\rangle \ . \tag{9}$$

Typically, the input state is chosen to be a zero-valued product state in the computational basis representation, i.e., $|\Psi_0\rangle = |00\ldots00\rangle = |0\rangle^{\otimes n}$. Other choices of the initial state may be made based on the problem requirements, possibly even to depend on some variational parameters itself as $|\Psi_0\rangle = P(\phi)|0\rangle^{\otimes n}$, with $P(\phi)$ a parametrised unitary, and ϕ the set of variational parameters. We discuss these as subjects of study in the future in section 6.

3.1.3 Measurement Scheme

[Note: Talk to Karthik/Sai about it]

3.1.4 Parameter Optimisation and Classical Control

After obtaining the loss function from the measurement and post-processing, a classical control system treats it as output from a black box, treating it the same as a classical optimizer, and readjusts the parameters according to an update rule of choice corresponding to a chosen optimization algorithm (gradient descent, finite element, etc.).

3.2 Quantum Landscape Theory

In subsubsection 3.1.2, we suggested the problem of the target unitary not existing in the space reachable in our circuit configuration. In this section, we elaborate on this issue, and discuss the related problems of studying the loss landscape, how it emerges, and how it affects the optimization process. We begin with a review of Quantum Landscape Theory [see 8, chapter II.B].

To study the landscapes, one must first be aware of the spaces each of the objects relevant to the computation belong to. This is diagramatically illustrated in Figure 4. First, the input parameter set $\boldsymbol{\theta}$ is seen as a vector in $\mathbb{R}^M$. The PQC

then represents an embedding of $\mathbb{R}^M$ into the unitary space of appropriate size $d \in \mathbb{N}$, $\mathcal{U}(d)$. Its action on the input state is the map $U(\boldsymbol{\theta}) : \mathcal{H} \to \mathcal{H}$. Finally, the measurement scheme maps the output state to a real-valued loss, which is used by the optimiser to recompute the parameters. Succinctly, the action of the model arises from the following tranformations

$$\mathbb{R}^M \to \mathcal{U}(d) \to \mathcal{H} \to \mathbb{R}\,. \tag{10}$$

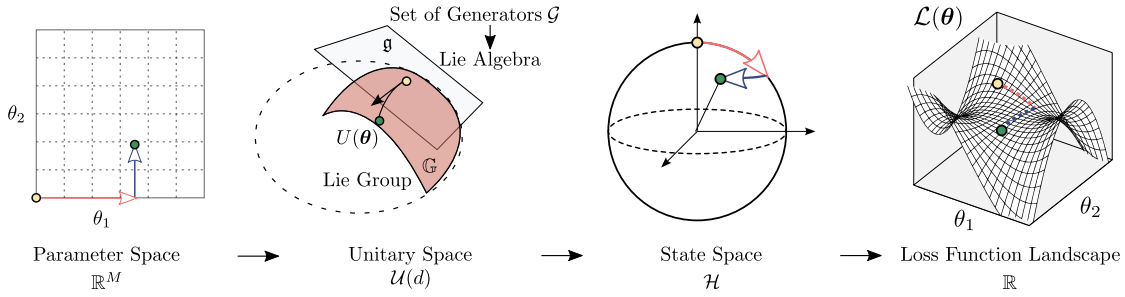


Figure 4: *Relevant mathematical spaces for VQA [taken from 8, Figure 2].*

3.2.1 Parameter Space to Unitary Group ($\mathbb{R}^M \to \mathcal{U}(d)$)

The first map, i.e., the map from the space of parameters to the unitary group, will be the focal point of the rest of this section. The unitaries generated by this map, and thus the chosen ansatz, are characterized by an object called the Dynamical Lie Algebra (DLA) of the system [see 17, chapter 3].

Definition 2 (Set of Generators)**.** *Consider a PQC of the form Equation 8. The set of generators $\mathcal{G} = \{H_k\}_{k=0}^K$ is defined as the set (of size K) of the Hermitian operators that generate the unitaries in a single layer of $U(\boldsymbol{\theta})$.*

Definition 3 (Dynamical Lie Algebra (DLA))**.** *Given a set of generators $\mathcal{G}$, its DLA $\mathfrak{g}$ is defined as the span of its Lie closure, or the space generated by $\mathcal{G}$ after closure with repeated nested commutation. Mathematically,*

$$\mathfrak{g} = \text{span}\,\langle \imath H_1, \imath H_2, \ldots, \imath H_k \rangle_{Lie}\,,$$

where $\langle S \rangle_{Lie}$ denotes the Lie or the nested-commutator closure of S.

The set of reachable unitaries is then a subset of the Lie group $\mathbb{G}$ generated by $\mathfrak{g}$,

$$\{U(\boldsymbol{\theta})\}_{\boldsymbol{\theta}} \subseteq \mathbb{G} \subseteq \mathcal{SU}(d) \,. \tag{11}$$

$\mathbb{G}$ can also be generated completely from the underlying Lie algebra as $e^{\mathfrak{g}}$.

It would seem at first glance that a configuration of generators should be chosen so as to be as expressive as possible, which is to have $\mathbb{G}$ be as close to $\mathcal{SU}(d)$ as possible, however, this often leads to trainability issues such as barren plateaus due to randomly chosen initial parameters [9, 18, 19]. As such, the ansatz is generally either chosen to make the problem convenient, i.e. problem-inspired ansatz [13], or to make the implementation convenient, i.e. harware-efficient ansatz [20].

3.2.2 Unitary Group to State Space ($\mathcal{U}(d) \to \mathcal{H}$)

Recall that for the map from the unitary group to the state space, the unitary output from the first map acts on states in the input set. Specifically, choosing the input set to be a training set $\mathcal{S} = |\psi_\mu\rangle$. Then, the second map (now parametrized by μ) is defined as

$$U(\boldsymbol{\theta}) \mapsto U(\boldsymbol{\theta})|\psi_\mu\rangle \,. \tag{12}$$

This set of reachable states is often called the *orbit*. In many cases, when the states in $\mathcal{S}$ have certain symmetries, the DLA in turn decomposes as the direct sum of the subspaces invariant under the symmetries

$$\mathfrak{g} = \bigoplus_{\nu} \mathfrak{g}_\nu \,. \tag{13}$$

There is no restriction on whether the states in the training set share or respect any symmetries of the PQC itself. In this way, the DLA serves as a focal point to determine the expressiveness in terms of unitaries as well as the set of reachable states in the Hilbert space.

Next, we wish to see how the output state changes with varying parameters $\boldsymbol{\theta}$. So, consider an infinitesmal perturbation to the parameters $\boldsymbol{\delta} \in \mathbb{R}^M$ and we can then quantify the distance between the initial and perturbed state. Define

$|\psi_\mu(\boldsymbol{t})\rangle = U(\boldsymbol{t})|\psi_\mu\rangle \ \forall t \in \mathbb{R}^m$. Writing the distance function (second order) as discussed in subsubsection 2.2.6

$$d(|\psi_\mu(\boldsymbol{\theta})\rangle, |\psi_\mu(\boldsymbol{\theta}+\boldsymbol{\delta})\rangle) = \frac{1}{2}\boldsymbol{\delta}^\top F_\mu(\boldsymbol{\theta})\boldsymbol{\delta}\,, \tag{14}$$

where $F_\mu(\boldsymbol{\theta})$ is the Quantum Fisher Information Matrix (QFIM) for $|\psi_\mu(\boldsymbol{\theta})\rangle$. Since the QFIM is related to the curvature of the state space, it plays a crucial role in quantum-aware optimizers such as the quantum natural gradient descent [21, 22, 23, 24]. Further, the rank of the QFIM quantifies the number of independent parameters in the state space that changing the parameters can allow us to explore.

3.2.3 State Space to Loss Landscape ($\mathcal{H} \to \mathbb{R}$)

Finally, the loss landcape generated by the composition map is characterized by the classically computed landscape of the real-valued loss function, i.e., using the $M \times M$ Hessian matrix

$$\left[\nabla^2 \mathcal{L}(\boldsymbol{\theta})\right]_{ij} = \partial_i \partial_j \mathcal{L}(\boldsymbol{\theta})\,. \tag{15}$$

Computing the gradient and Hessian matrix allows us to form a quadratic model of the loss function, with the Hessian's eigenvectors characterizing curvature directions at each point. The rank of the Hessian once again relates to the number of independent directions explorable by change in parameters, emphasizing how the QFIM functions as a curvature measure in the state space.

4 Information Theoretic Limits

4.1 Bounds on PQC Optimisation

4.2 Bounds Inherited by QSVMs

5 Applications - Quantum Support Vector Machine

6 Conclusion and Future Work

We will do st probably.

References

[1] Nello Cristianini, John Shawe-Taylor et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[2] Hanamantagouda P Sankappanavar and Stanley Burris. 'A course in universal algebra'. In: *Graduate Texts Math* 78 (1981).

[3] Corinna Cortes and Vladimir Vapnik. 'Support-vector networks'. In: *Machine learning* 20.3 (1995), pp. 273–297.

[4] Kishor Bharti et al. 'Noisy intermediate-scale quantum (NISQ) algorithms'. In: *arXiv preprint arXiv:2101.08448* (2021).

[5] Giovanni Sansone. *Orthogonal functions*. Vol. 9. Courier Corporation, 1959.

[6] Brian C Hall. *Quantum theory for mathematicians*. Vol. 267. Springer, 2013.

[7] Johannes Jakob Meyer. 'Fisher Information in Noisy Intermediate-Scale Quantum Applications'. In: *Quantum* 5 (Sept. 2021), p. 539. ISSN: 2521-327X. DOI: 10.22331/q-2021-09-09-539. URL: http://dx.doi.org/10.22331/q-2021-09-09-539.

[8] Martin Larocca et al. 'Theory of overparametrization in quantum neural networks'. In: *arXiv preprint arXiv:2109.11676* (2021).

[9] Martin Larocca et al. *Diagnosing barren plateaus with tools from quantum optimal control*. 2021. arXiv: 2105.14377 [quant-ph].

[10] Abhinav Kandala et al. 'Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets'. In: *Nature* 549.7671 (2017), pp. 242–246.

[11] Edward Farhi, Jeffrey Goldstone and Sam Gutmann. 'A quantum approximate optimization algorithm'. In: *arXiv preprint arXiv:1411.4028* (2014).

[12] Dave Wecker, Matthew B Hastings and Matthias Troyer. 'Progress towards practical quantum variational algorithms'. In: *Physical Review A* 92.4 (2015), p. 042303.

[13] Alexandre Choquette et al. 'Quantum-optimal-control-inspired ansatz for variational quantum algorithms'. In: *Physical Review Research* 3.2 (2021), p. 023092.
[14] Stuart Hadfield et al. 'From the quantum approximate optimization algorithm to a quantum alternating operator ansatz'. In: *Algorithms* 12.2 (2019), p. 34.
[15] Linghua Zhu et al. 'An adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer'. In: *arXiv preprint arXiv:2005.10258* (2020).
[16] Juneseo Lee et al. 'Progress toward favorable landscapes in quantum combinatorial optimization'. In: *Physical Review A* 104.3 (2021), p. 032401.
[17] Domenico d'Alessandro. *Introduction to quantum control and dynamics*. Chapman and Hall/CRC, 2021.
[18] Zoë Holmes et al. 'Connecting ansatz expressibility to gradient magnitudes and barren plateaus'. In: *arXiv preprint arXiv:2101.02138* (2021).
[19] Jarrod R McClean et al. 'Barren plateaus in quantum neural network training landscapes'. In: *Nature communications* 9.1 (2018), pp. 1–6.
[20] Marcello Benedetti, Mattia Fiorentini and Michael Lubasch. 'Hardware-efficient variational quantum algorithms for time evolution'. In: *Physical Review Research* 3.3 (July 2021). ISSN: 2643-1564. DOI: 10.1103/physrevresearch.3.033083. URL: http://dx.doi.org/10.1103/PhysRevResearch.3.033083.
[21] James Stokes et al. 'Quantum natural gradient'. In: *Quantum* 4 (2020), p. 269.
[22] Bálint Koczor and Simon C Benjamin. 'Quantum natural gradient generalised to non-unitary circuits'. In: *arXiv preprint arXiv:1912.08660* (2019).
[23] Julien Gacon et al. 'Simultaneous perturbation stochastic approximation of the quantum fisher information'. In: *arXiv preprint arXiv:2103.09232* (2021).
[24] Tobias Haug and MS Kim. 'Natural parameterized quantum circuit'. In: *arXiv preprint arXiv:2107.14063* (2021).