2021-11-20

# 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 $f : X \hookrightarrow Y$.

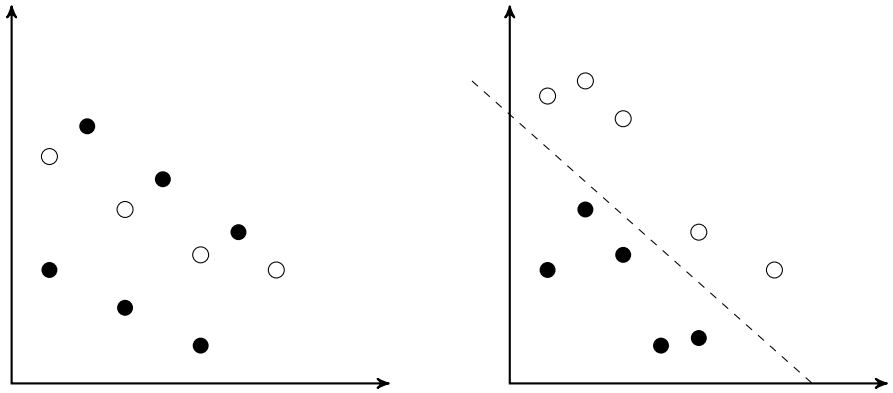
For example, a topological embedding, i.e., the embedding of a topological space, will be restricted to preserve its associated structure of open sets. A field embedding, similarly, will be restricted to preserve the field operations $+$ and $\times$.

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., any labelling function $f : X \to L$ that agrees with the training data [Note: try to write separation properly].



**(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].

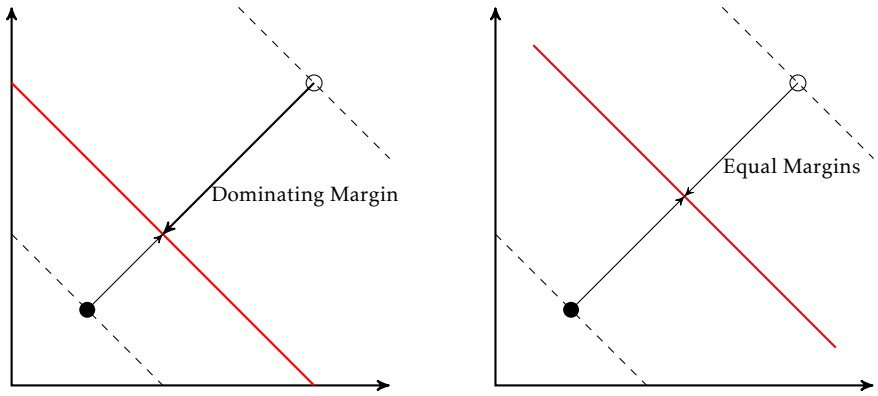
We will synonymously use the terms 'Support Vector Machine' and that of its common model 'Maximal Margin Classifier', which is more appropriately what we use here.

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)

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

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

### 2.1.6 Optimisation Techniques

Gradient descent or st. [Note: expand]

### 2.1.7 Generalisation Error

Hello [Note: read about gen error and write here]

## 2.2 Quantum Regime

[Note: add a general introduction]

### 2.2.1 Schrödinger Equation

### 2.2.2 Hilbert Space

### 2.2.3 Quantum Computation

### 2.2.4 State Construction and Embedding

### 2.2.5 Measurement

# 3 Variational Quantum Algorithms

Hello. [4]. [Note: Write this section ig]. [Note: borrow citations from K Bharti].

## 3.1 Building Blocks

### 3.1.1 Objective Function

### 3.1.2 Parametrised Quantum Circuits

Expressiveness [5].

### 3.1.3 Measurement

### 3.1.4 Parameter Optimisation

# 4 Quantum Support Vector Machine

# 5 Information Theoretic Limits

## 5.1 Bounds on PQC Optimisation

## 5.2 Bounds Inherited by QSVMs

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