A Quantum view of Machine Learning for Big Data Classification

Can quantum computing provide an edge
in machine learning algorithms?

By

Student Number: UXXXXXXX

A Dissertation submitted in partial fulfilment of the requirements
for the degree of
MSc in Big Data and Digital Futures

Supervised by XXXXXXX

University of Warwick
Centre for Interdisciplinary Methodologies
August 2020

ABSTRACT

New and emerging challenges give rise to new ways of thinking. The onslaught of Big-Data with all its bells and whistles is one such challenge that has garnered intense research focus in finding solutions beyond the conventional and classical means. To that end, quantum computation has fuelled optimism with the promise of unconventional speedups in processing and therefore it becomes not only natural but rather imperative to seek connections between it, and the computationally demanding fields of data mining and machine learning. Our understanding of the physics of quantum world has allowed us to unleash the power of nature in doing the computational heavy lifting, and consequently presented itself as a significant tool in field of machine learning, which as we will see in this work, is based on clear scientific evidence and shared mathematical foundations.

Machine learning algorithms in data mining have in recent decades come a long way from being abstract models for classification, regression and pattern recognition to actual manifestations of commercial and critical essence. However, the three cornerstones of big data – volume, velocity and variety – pose serious concerns for the future by outpacing the capacity and complexity of data analysis tools currently at our disposal. This dissertation aims to explore prospects in the interdisciplinary field of 'Quantum Machine Learning' that has been heralded as a new way to harness the power of Quantum Computers in order to alleviate those concerns by finding solutions to some of the pressing problems in traditional machine leaning which are presumably intractable via classical techniques.

This work is mostly an abstract and theoretical study of quantum approaches to machine learning with a tangible attempt, in the end, towards an implementation of a supervised classification algorithm on a real quantum device. Overall in a broader perspective it will dissertate on the question : '*Can quantum computing provide an edge over classical machine learning techniques?*' and examine how quantum architectures can be exploited, in either standalone or hybrid quantum-classical setups to speedup existing or potentially invent new learning algorithms with quantum advantages that could be of theoretical and practical interest to academia and businesses.

List of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my dissertation supervisor XXXXXXXX for his invaluable advice and guidance throughout this work and also for his confidence and support for such an unconventional research topic for this dissertation.

I would also like to extend gratitude to XXXXXXX, Department of Computer Science, at the university for his brilliant Data Mining module, which inspired me to research the field in greater depth and in the course of which, this topic was formed while exploring alternate computing paradigms in machine learning.

Last but not the least my personal tutor, XXXXXXXX, who always ensured her availability to answer any queries and provide support and encouragement throughout the programme.

Chapter 1

Classical Machine Learning

1.1 Background

Machine learning (ML) contained within a much broader domain of Artificial Intelligence (AI), concerns itself with intelligent systems that learn (Russell, 1996). By way of generalizing through computation from examples – i.e. data organized as datasets, these systems pose an attractive and cost-effective – and often the only – generalized alternative to deriving knowledge from big data. Since modern computers are universal digital computing machines which can theoretically solve any *computable* problem, they are potentially well suited to become such intelligent systems themselves. This realization led to the eruption of an algorithmic modelling culture (Breiman, 2001) to define learning models, both specific and generic, for a variety of ambitious problems ranging from handwriting recognition to playing (and even beating humans at) the ever-complex games of Chess and Go (Silver et al, 2016)

A system can learn either in a supervised or un-supervised setup depending on the available data, the problem statement and the end goal. In a supervised setup, the learning process by such a system can be understood in terms of its three main components (Figure 1) i.e. – **R**epresentation, **E**valuation and **O**ptimization (Domingos, 2012).

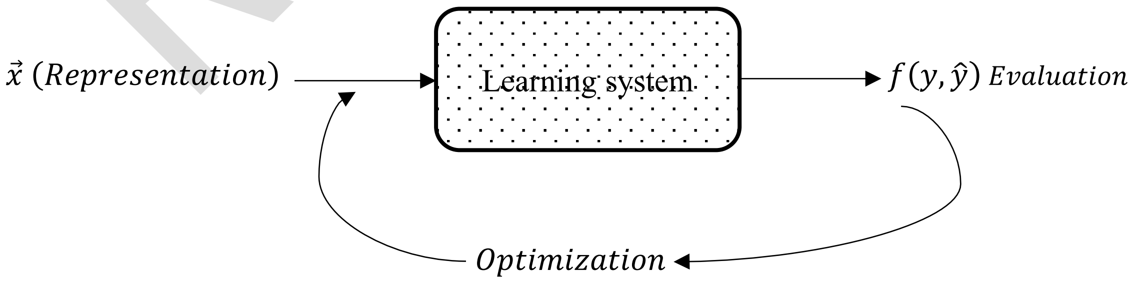


Figure 1 Components of a supervised learning process.

The representation aspect deals with the formal language that encodes the hypothesis space of the underlying model that the system can understand. A classification problem can be represented for example via a simple logistic regressor, a Support Vector Machine (SVM) classifier or a Multi-layer perceptron (MLP) of a neural network. It also addresses the question of how to numerically represent the input dataset i.e. in forms of vectors, matrices or tensors.

Evaluation (also called *objective* or *loss* function) is a means to compare the naïve output ($\hat{y}$) of the system against the true value (y) available from examples. Mean Squared error, Accuracy and Error rates are some formulations that capture this performance metric to estimate the quality of model. The Optimization part concerns itself with a way to find, from the space of all represented models, the one which evaluates to the optimal performance. For e.g. in an unconstrained continuous space, this is usually implemented as gradient descent, whereas a system with constraints in practice employs tools such as linear programming to achieve this optimality.

In the following sections we can see these components more concretely in the light of mathematical formalism, first to describe a classical support vector machine (SVM), and then extend it to describe its equivalent quantum version in a supervised setup for multi-class classification of dataset.

1.2 Formalization

In the context of classical machine learning (valid with slight modifications to the domain of Quantum Machine Learning) the pipeline starts with data points or samples or examples. A single data point $\boldsymbol{x}$ in its most generic form is denoted as a *feature vector* in an *N*-dimensional real vector space as

$$\boldsymbol{x} = \{x_1, x_2, \dots, x_N\}\,;\; x_i \in \mathbb{R} \qquad (1.0)$$
$$i.e\; \boldsymbol{x} \in \mathbb{R}^N$$

where, x_i are selected features representing information contained within it. Similarly, the class label y_j for data points are just real numbers for each of the *M* relevant classes, i.e.

$$\boldsymbol{y} = \{y_1, y_2, \dots, y_M\} \qquad (1.1)$$

Taken together, the set of $\{\boldsymbol{x}, \boldsymbol{y}\}$ form the training dataset T used by classifier to learn the optimal state representation.

A representation for e.g. of a simplistic linear classifier is then, a map from input space of feature vectors to the output space of labels. i.e.

$$y_j = f(\boldsymbol{x}\,;\, \boldsymbol{w}) = w_1 x_1 + \; w_2 x_2 + \cdots w_N x_N + b \tag{1.2}$$

or equivalently in dot product notation,

$$y_j = f(\boldsymbol{x};\, \boldsymbol{w}) = w^T \boldsymbol{x} + b \tag{1.3}$$

where, $\boldsymbol{w}$ is the weight vector (with w^Tas its transpose), b is some bias or offset.

With suitable pre-processing of training data, a binary classifier such as above having two labels $y_j \;\in \{+1, -1\}$ can be used to predict the class of any example from the set of unseen data by looking at the output of discriminant function $f(\boldsymbol{x};\, \boldsymbol{w}) > 0$ for the class '+1' or $f(\boldsymbol{x};\, \boldsymbol{w}) < 0$ for '-1'. The empirical risk or error for this classifier is defined as defined as the difference between the predicted and actual classes of the training data points. By multiplying the predicted and actual classes together i.e. $y_j . f(\boldsymbol{x};\, \boldsymbol{w})$ we can see that the product is always positive where prediction matches actual class and negative otherwise. This is utilized in the formulation of the empirical error function *(L)* as:

$$L(X, Y; w) = \tfrac{C}{N} \textstyle\sum_{i=1}^{N} max\{0{,}1 - \; y_j . f(\boldsymbol{x};\, \boldsymbol{w})\} \tag{1.4}$$

Where C is a hyperparameter that controls the contribution of empirical error in the optimization process. As can be seen in Fig. 2a, a whole family of solutions $f(\boldsymbol{x};\, \boldsymbol{w}) = 0$ represented by various straight lines exist which can act as classifier. Finding an optimal discrimination boundary that has maximum margins then becomes a constrained optimization problem involving a quadratic function of weight parameters. The function $f(\boldsymbol{x};\, \boldsymbol{w})$ with the minimum value of the weight vector $(\boldsymbol{w})$ gives the optimal boundary with least structural risk $R(w)$ and regularizes any large fluctuations in the predicted class of the data due to small variations in the input data. The structural risk can be written as

$$R(w) = \min \tfrac{1}{2} w^T \boldsymbol{w} \tag{1.5}$$

The optimal classifier is obtained via the minimization of both the structural risk (1.5) and the empirical risk (1.4) combined together and is known as the dual optimization problem (Grinblat et al, 2010). Classical SVMs proposed as supervised algorithm for classification and regression (Boser et al, 1992) resolve this problem by utilizing the inductive principle of Structural Risk Minimization (SRM) which in turn is based on the optimal margin algorithm originally developed by Vapnik (Vapnik & Chervonenkis, 1974).

SVMs became popular as they learned not just from the training data set alone (via empirical error minimization) but also from the structural error. By gaining control over the structure of classifier and putting an upper bound on the generalization error it minimizes the structural risk leading to greater stability and robust classifier. The resultant learning model (Fig. 2b) with the optimal margin thus has greater tolerance for noise when trained over finite data set and also avoids over-fitting. An optimal decision boundary $f(\boldsymbol{x};\, \boldsymbol{w}) = 0$ generated by SVMs is also structurally sound due to the orthogonal "support vectors", which in addition to giving SVMs their name, also give rise to model 'regularization' i.e. the SVM classifier does not produce huge changes in response to small changes in the input data set.

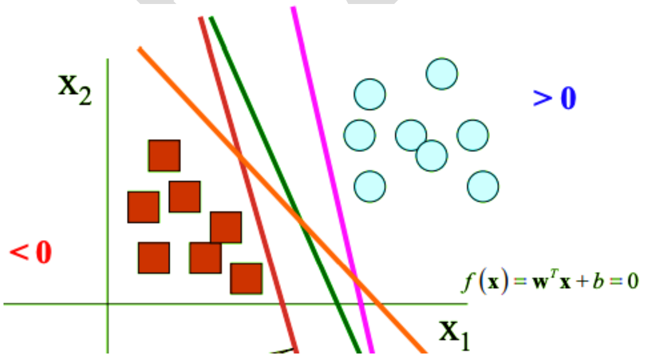


Figure 2a Multiple decision boundaries as classifiers.

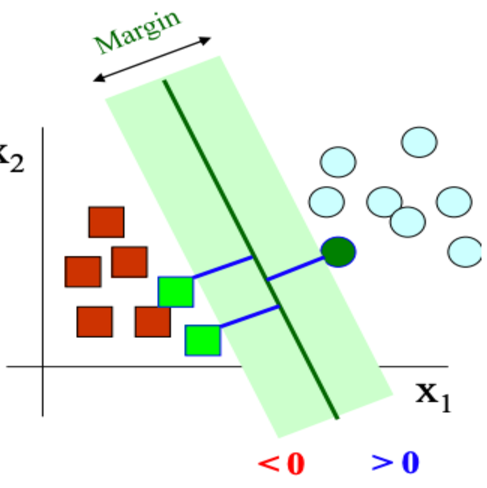


Figure 2b Optimal margin boundary with orthogonal support vectors (Source: CS909 Data Mining, University of Warwick)

SVMs are a special class of more generalized models called Maximum Margin Classifiers (MMC) which aim to maximize the margin between the class boundary and training patterns as an alternative to other training methods such as the mean squared error (MSE) in order to optimize the loss function. The algorithm and its variations have been compared to traditional artificial neural networks (ANN) in the context of practical industrial applications and have shown higher accuracy and robustness especially for complex non-linear data sets (Balabin & Lomakina, 2011) .

However, this problem of classification becomes increasingly non-trivial as the number of dimensions *N* becomes immensely huge, and even more so when the dataset is not linearly separable i.e. insolvable via a linear classifier shown in eq. 1.2. With increasing dimensionality and non-linearity of the input feature space, the task of transforming the problem into linearly separable one becomes not only hard but even intractable. This is where kernel functions or '*kernel trick*', as they are referred to sometimes, provide an interesting solution. By applying a non-linear function $\emptyset()$, the original dataset is mapped into a higher dimensional feature space where the problem of classification is transformed into a linearly separable one (Fig. 3), and only requires calculating how close the data points are to each other in the transformed space (Schölkopf, 2000).

Kernels are specialized functions which result in a transformed feature space that is equipped with the property of inner products – a characteristic of *Hilbert Spaces* – which allows measurement of this closeness through the inner product of two feature vectors x and y belonging to transformed feature space. Denoted as $\langle . \ , \ . \rangle$ the inner product is essentially a "measure of similarity" between the features and defines the kernel matrix by computing the pair-wise inner product of data points in the transformed space.

$$k(x, y) = \langle \emptyset(x), \emptyset(y) \rangle \tag{1.6}$$

Where, $\emptyset(x)$ and $\emptyset(y)$ are mappings of original data points in the transformed space.

Computing all possible pair-wise inner products of a training set containing *N* data points gives rise to what is called a 'kernel matrix' of size *N* x *N* and allows solving the dual problem easily. However, in practical implementations involving real world data analysis problems, often the function $\emptyset$ is unknown and the cost of computing all possible inner products becomes prohibitive. It has been mathematically proven that the class of positive definite matrices can be substituted for the dot product in e.q. 1.6 and that such kernel matrices $k(x, y)$ can be expressed via expansion of some functions using representer theorem (Hofmann et al, 2008). The approximation function can be linear, polynomial, radial basis functions (rbf), sigmoid depending on the desired performance. For e.g. a SVM based on Gaussian kernel can be estimated using a finite-dimensional polynomial representation obtained through Taylor expansion of the exponential function (Cotter et al, 2011).

As we will see later, this is where QML can provide us with an advantage both in terms of dealing with the '*curse of dimensionality*' (Bellman, 1961) and the intractability of finding such a kernel in a large hypothesis space of all possible kernels by calculating the inner products of the data pairs through quantum parallelism and forms the basis of speedups in the quantum version of SVM..

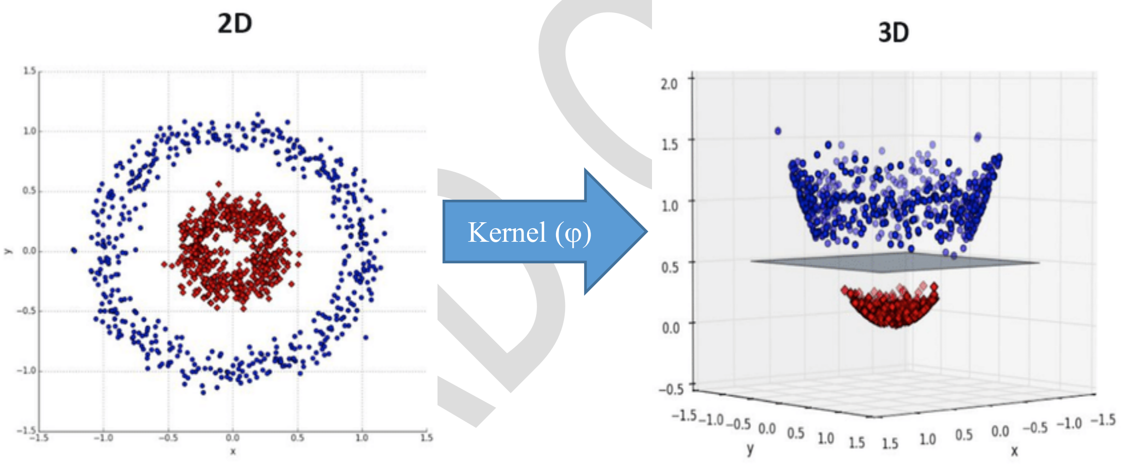


Figure 3 Kernel transformation of non-linear 2D feature space (left) to linearly separable 3D space(right). (Source: (José Luis et al, 2018)

A variety of machine learning problems such as text classification and image recognition fall into the category of what is known as non-convex optimisation problem and require iterative techniques such as *stochastic gradient descent* to find the optimums. Non-convex objective functions are difficult to treat due to the lack of closed form solutions and hence rely on numerical optimization methods most common in the implementations of deep neural networks (Bottou et al, 2018). In contrast, the classical SVMs like the one described above, require convex optimization techniques involving a quadratic problem. A further improvised version based on the least square formulation of SVMs (LS-SVMs), uses just a set of linear equations. This is achieved by changing the inequality type constraints to equality type thereby making them computationally efficient with excellent generalization performance (Suykens & Vandewalle, 1999).

This translation of SVMs to a set of linear equations underpins the source of speedup on quantum hardware in two ways. First is the calculation of kernel matrix through direct evaluation of inner product within the embedding quantum space, and second is the availability of a quantum algorithm for efficiently solving these linear equations through matrix inversion using HHL algorithm. (Harrow et al, 2009).

Chapter 2

Quantum Machine Learning

2.1 Quantum Computation

Quantum computers make use of quantum-mechanical phenomenon such as *superposition* and *entanglement* to perform computations, which can, for a certain class of problems be significantly faster than classical computers. Peter Shor's factoring algorithm (Shor, 1994), which solves the problem of finding prime factors of a given integer *N* in polynomial-time $O(n^k)$, and Grover's algorithm (Grover, 1996) for database search in sublinear polynomial-time $O\left(\sqrt{n}\right)$, are quintessential depictions of this quantum prowess.

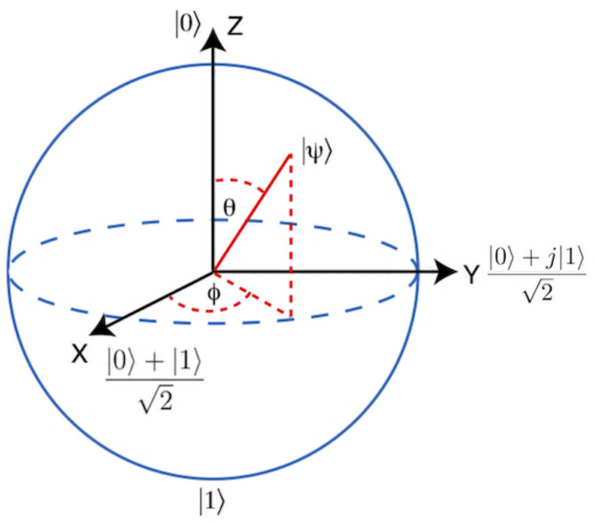
Similar to a *bit* in classical computing, the fundamental unit of processing in quantum computing (QC) is called a quantum bit or *qubit* (Schumacher, 1995) that describes the overall state of a quantum particle (electron, photon etc). Bizarrely, that quantum state is actually a superposition of the two different ground states 0 and 1, implying that the qubit is in both these states at the same time - which is our first step towards understanding the source of quantum speedups in computation. In the *ket-notation*[i] (Dirac, 1939) a single qubit is formally written as a linear combination of the ground states as :

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad where\ \alpha, \beta \in \mathbb{C} \tag{2.0}$$

$$and\ \ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.1}$$

The vectors $|0\rangle,\ |1\rangle$ themselves form an orthonormal basis of a 2-dimensional Hilbert space $(\mathcal{H})$, also called the computational basis as all transformation or computations are performed with respect to these two basis vectors. Since α and β are amplitudes of the wave function that describes the time evolution of a quantum particle via Schrödinger's equation, these are complex numbers ($\mathbb{C}$) and hence qubits are often visualized as vectors in a complex vector space. The use of Hilbert space allows generalized representation of multi qubit systems as finite-dimensional complex vector spaces and also provides an inner product structure to this representation which can be used to calculate the similarity between two qubit vectors. This similarity measure allows estimation of kernels in the quantum versions of *least-squares support vector machines* (Suykens & Vandewalle, 1999) that are based on the distance calculation between data points to derive the decision boundaries.

The ability to visualize how a single-qubit vector appears in a *unit* complex sphere (Fig. 2), via bloch sphere (Bloch, 1946), where the ground states are the two poles and qubit as a general point on its surface, is useful in understanding the probabilistic nature of quantum computation – in that – the squared norms of α and β are the probabilities of measuring the qubit in state $|0\rangle$ and $|1\rangle$ respectively, and necessarily sum up to 1, i.e. $\alpha^2 + \beta^2 = 1$.



$|0\rangle$ Z
$|\psi\rangle$
θ
ϕ
Y $\frac{|0\rangle + j|1\rangle}{\sqrt{2}}$
X $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$
$|1\rangle$

Figure 4 A geometric representation of qubit $|\psi\rangle$ inside a Bloch sphere. (Source: IBM Q Experience)

A multi-qubit system is usually represented as a tensor product of the states of its constituents, for e.g. the combined state of 2-qubits, $|\psi\rangle$ and $|\varphi\rangle$ is given as

$$|\phi\rangle = |\psi\rangle \otimes |\varphi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \qquad (2.2)$$

As can be seen, the combined states of two qubits (also notated as $|\psi\varphi\rangle$) is fully described by $4 = (2^2)$ dimensional complex vector $[\alpha, \beta, \gamma, \delta]^T$, and therefore, more generally for q qubits, the overall state can be denoted as a unit vector in a complex Hilbert space $(\mathcal{H})$ of dimension $\mathbb{C}^Q$, where $Q = 2^q$. Modelling an N-dimensional classical dataset, in the state space of a quantum system, can therefore be done using $n = log_2 N$ qubits only. Since this representation provides a straightforward exponential advantage it is termed as quantum-enhanced feature space or quantum kernels or quantum feature maps (Schuld & Killoran, 2019).

At quantum scales, another emergent property which transcends physical description is that of 'entanglement' between qubits. A quantum system is said to be *entangled* if individual qubits cannot be described independently of other qubits within it, i.e. they are correlated to each other. Consider a bipartite or 2-qubit system from (2.2) above with $\alpha = \delta = 0$ such that

$$|\phi\rangle = \beta|01\rangle + \gamma|10\rangle \tag{2.3}$$

It is easy to show that this new state cannot be decomposed as a tensor product of any other two quantum states i.e.

$$|\phi\rangle \neq |\psi\rangle \otimes |\varphi\rangle \quad \forall \ |\psi\rangle, |\varphi\rangle \tag{2.4}$$

What equation 2.3 describes is one of the maximally entangled states called 'Bell state', where if the first qubit is measured in state $|0\rangle$, the other will instantly and definitively measure to be in $|1\rangle$, and vice-versa. This was proved by Bell in his famous paper of 1964 – "*On the Einstein Podolsky Rosen paradox"* – where he argued by simple probability theory that quantum mechanics predicts perfect correlations (Bell, 1964) . In its simplest form entanglement gives rise to correlations between quantum states which is beyond or "somewhat more" than its classical analogue, and is a distinctive feature in many quantum information processing algorithms (Zeng et al, 2019) and in entanglement-assisted versions of SVM for data classification and principal component analyzer for data compression (Zhuang & Zhang, 2019) .

Just as in classical digital computation Boolean gates such as AND, OR and NOT perform operations on bits, *quantum gates* control and transform qubit(s) to perform quantum computations. A gate, mathematically interpreted as an invertible operation, rotates the unit state vectors (Fig. 2) of qubits in their Hilbert spaces to a new final state, and therefore are key towards exploring the computational space that grows exponentially – a critical issue when applying data mining and classical machine learning algorithms on high dimensional datasets (Li & Liu, 2017).

A single qubit gate for e.g. a Hadamard gate H acts on a single qubit, usually in ground state $|0\rangle$, and rotates it by 90° in the bloch sphere, aligning it to the X-axis (Fig. 2) to create an equal superposition of the two ground states as follows:

$$|0\rangle \text{ ---------------------- } \boxed{H} \text{ ---------------------- } \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{2.2}$$

The corresponding matrix for H is the following, 2 x 2 unitary $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ which when multiplied by the vector $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ results in the above superposition. Similar to (2.1), for applications to n-qubit systems, the resultant gate $H^{\otimes n}$ is just the tensor product of H with itself n times.

$$H^{\otimes n} = H \otimes H \ldots n \; times \ldots \otimes H \tag{2.3}$$

This leads to a fascinating aspect of quantum gate application – its constant time complexity. Consider the operation – $H^{\otimes n}|0\rangle^{\otimes n}$ i.e. applying n Hadamard gates H to n qubits which are in ground state $|0\rangle$. This computation does not take $O(n)$ but is rather $O(1)$ in time – which means it does not depend on the input size (n). This theoretical quantum "*parallelism*" can heavily subsidize various compute intensive tasks – both in evaluation and optimization phase of machine learning – which suffer from the curse of high dimensional input datasets, thereby rendering classical computers unsuitable. Theoretically, it may seem the power of quantum computers doubles with addition of every new qubit in the system, however in current practical setups, due to hardware limitations of current quantum devices that are currently at our disposal, n cannot be made arbitrarily large to solve any real-world machine learning task more efficiently than it is already done by classical techniques.

Quantum gates (also referred to as Operators) are in general written as (U) in order to signify that their matrix formulation is *unitary*. Unitary matrices satisfy the invertibility requirement as they are equal to their complex-conjugate transpose, i.e. $U = (U^*)^T$. This means that applying a specific quantum gate twice cancels out each other and is therefore equivalent to an Identity matrix multiplication. Multitude of other single qubit and 2-qubit gates exist (see Appendix A) to perform specific computations and play a central role in quantum computation networks. It has been theoretically proven that a set comprising of all one-qubit gates, and two-qubit XOR (exclusive OR) gate is universal, in that any arbitrary unitary operation can be expressed as the composition of these gates (Barenco et al, 1995). From machine learning perspective this means that any computation that can be done on classical machine can in theory be done via composition of some elementary quantum gates, although in practical setups the very long compositions are inevitably affected by noise and residual heat from environment causing decoherence on quantum devices (Kauffman & Lomonaco, 2004). Of particular interest in this work is the quantum SWAP gate that is used to calculate the inner product between qubits for kernel matrix estimation in the quantum version of SVM algorithm. SWAP gate can be constructed using single qubit rotation and CNOT gates (Wilmott & Wild, 2014) and can be seen in the circuit diagram generated for Quantum SVM algorithm later in the implementation section.

In order to implement specific algorithms, these gates and qubits are wired together to compose – analogous to electronic circuits – what is called a Quantum Circuit (QCs). To represent connections, single wires for qubits and double wires for classical bits are used in circuit diagrams such as the one shown in (2.2) above, followed by one or more measurements at the end of the wire. A key deviation from the classical circuits is due to the reversible nature of the gates involved in QCs, which result in equal number of input and output qubits. Another key difference from classical electronic circuits, which have no upper bound on the number of components and inputs/outputs (I/O), is that most modern quantum devices are limited both in their number of qubits, which defines the *circuit width*, and their number of quantum gates, that define the *circuit depth*. For e.g. we use a 15-qubit device at the end of this work to implement a QML algorithm with only a handful number of gates before environmental noise causes decoherence of qubits rendering the computation useless. Parameterized versions of these quantum circuits called Parameterized Quantum Circuits (PQCs) (Sim et al, 2019), wherein constituent gates can be adjusted via certain parameters (θ), have been demonstrated as machine learning models with "remarkable expressive power" capable of generating highly non-trivial outputs that cannot be efficiently simulated by classical means (Benedetti et al, 2019).

In resembling how a classical neural network learns the weight parameters during training, PQCs iteratively update the gate parameters, such as the amount of rotation (θ) performed by an R_θ gate in the training phase. A detailed theoretical background on quantum computation and information can be found in Nielsen and Chuang's book (Nielsen & Chuang, 2010)

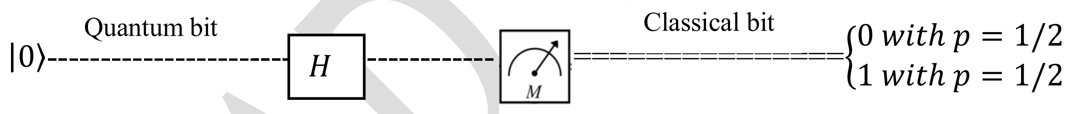
2.2 Measurements on Quantum Machines

Qubits are governed by probabilistic laws of quantum mechanics and measuring or observing them provides one sample as output from the underlying probability distribution. That sample value according to *Born rule*, is calculated as the absolute square of the qubit's amplitude which as we've seen earlier are complex valued vectors (Born, 1926). Essentially, the act of physical observation causes decoherence and collapses the qubit to one of its ground states with a probability proportional to the associated amplitude (α and β) of that ground state.

Take for instance the qubit in (Eq. 2.2) which is in a state of superposition after the application of Hadamard (H) gate. Upon subsequent measurement (M), the probability p of finding it in either state $|0\rangle$ or $|1\rangle$, is equal to the square of respective amplitudes i.e.

$$p = \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$$

In terms of the circuit representation this is often shown as below where the double line following the measurement (M) denotes output as a classical bit –



The most general approach of measuring finite dimensional quantum systems is given by POVM's which stands for *positive operator valued measure* (Steinberg, 2017). This is a type of projective measurement in which the operator acting on a given quantum state $|\psi\rangle$, projects that state onto the *i*'th basis vector with the probability that is proportional to the absolute value of its amplitude in that basis. In a way, it is an implementation of the Born Rule described above. To calculate the probability p in general case the operator M which is a positive Hermitian matrix M is multiplied with the state vectors on both sides

$$p = \langle\psi|\, M\, |\psi\rangle$$

where, $\langle\psi|$ is the row vector obtained by the conjugate transpose of the column vector $|\psi\rangle$

Consequently, full information about the underlying distribution is never revealed by performing a single measurement, which means if the overall state is in a superposition of more than two ground states, such as in Eq. 2.2, a single observation will result in only one of four possible amplitudes$[\alpha, \beta, \gamma, \delta]$. In the context of quantum machine learning, this probabilistic nature of measurement entails that the machine learning model has to be repeatedly executed a certain number of times to obtain specific level of confidence in its outcome. Together with quantum parallelism it can be shown that a squared-distance based binary classifier, implemented in a quantum setup, would require only one single computational operation and two simple measurements (repeated a couple of times) to obtain the classification results, and would at the same time be independent of the size of input vectors (Schuld & Petruccione, 2018b). As a complimentary strategy, research has also been done on supervised machine learning algorithms without the need for intermediate measurements on quantum systems by means of time-delayed dynamics and advanced implementations of unitary operations which are beyond the scope of this work as they require deeper understanding quantum mechanics involving feedback-induced effects (Alvarez-Rodriguez et al, 2017).

2.3 Learning on Quantum Machines

Quantum machine learning is an interdisciplinary field combining seemingly disparate disciplines, namely – machine learning, quantum computation, quantum information theory and quantum physics (Wittek, 2014). Under the hood, however, these fields share strong mathematical notions in theories of high dimensional vector spaces, linear algebra and probabilistic distributions that are foundational across both domains (Biamonte et al, 2017). The physics of quantum mechanics – which describes properties of subatomic particles – lends the phenomenon of *entanglement* and *superposition*, that are at the heart of quantum parallelism (Djordjevic, 2012), to be harnessed for computational speedups (in some cases) which would otherwise be impossible in classical domain. These similarities and potentials provide justifiable reason to postulate that quantum machines have the ability to outperform their classical counterparts in machine learning tasks.

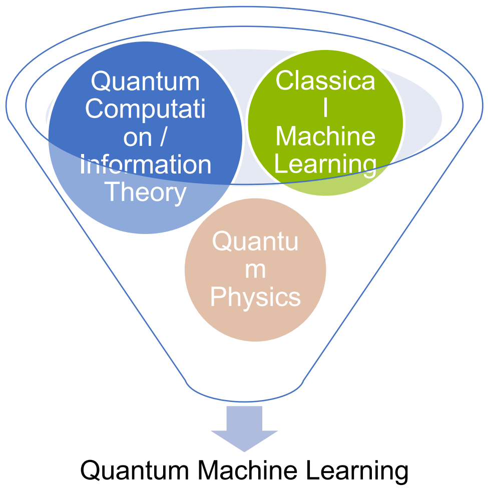

Quantum Computati on / Information Theory
Classica l Machine Learning
Quantu m Physics
Quantum Machine Learning

Figure 5 Interdisciplinary components in quantum machine learning.

Both Quantum computation and machine learning can be viewed essentially as a set of linear transformations of some input states to achieve desired outcomes be it – classification, regression or pattern recognition in unstructured data. The non-linearity induced by activation functions in the classical versions of neural networks is based on the similarity from biological neurons where signals activate the network if they surpass a certain threshold value, in the context of QML it has been the subject of study in quantum variational classifiers (Lubasch et al, 2020) and more recently with tailored non-linear quantum kernels (Blank et al, 2020) both of which are beyond the scope of this work.

Depending on the type of system the training data is generated and the type it is processed on – both of which can be either classical (*C*) or quantum (*Q*) – four different combinatorial approaches exist in terms of combining quantum computing and machine learning (Fig. 6). However, to narrow down the definition of QML in this work, only the *CQ* setting is considered, wherein classical (*C*) datasets (such as text, images) are processed using quantum (*Q*) computation. Other remaining approaches for e.g. *CC,* which is just traditional machine learning with classical dataset on classical computer; *QC,* that utilizes

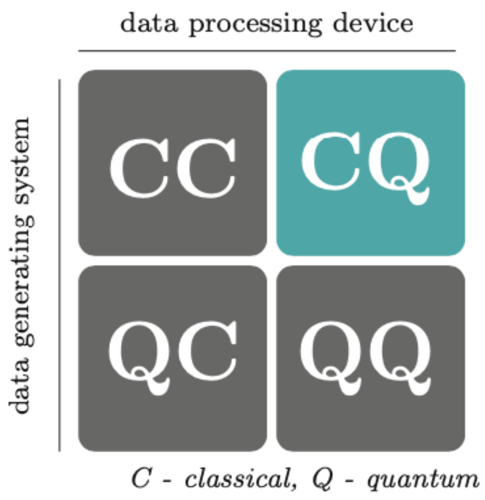


Figure 6 Four possible combinations of quantum computation and machine learning. (Source: (Schuld & Petruccione, 2018b)

machine learning to enhance quantum computing; and QQ relevant for physical simulations of quantum states on quantum computers are impertinent and out of scope here. Within CQ too, for the sake of simplicity, only supervised learning classification is considered and will be synonymous with our understanding of QML in this work.

Many proposals of Quantum Neural Networks (QNN) have existed, but none of them have been shown to successfully exploit any quantum advantage. This is largely due to the challenge of combining non-linear and dissipative dynamics of neural computation with the linear and unitary dynamics of quantum computing (Schuld et al, 2014) and therefore remains beyond the subject area and research topic in this work.

Broadly, any machine learning pipeline on quantum computers usually involves three steps (Fig. 7). The initiation happens with creation of a superposition state by applying the Hadamard gate (H) to input qubits (refer to Eq. 2.2). Next the input dataset gets mapped via entanglement onto this superposition to create a quantum feature space, thereby completing the pre-processing or state preparation protocol. Lastly, the training occurs via a parameterized quantum circuit (θ) which acts on the state prepared earlier to produce observable quantities estimated by performing measurements.

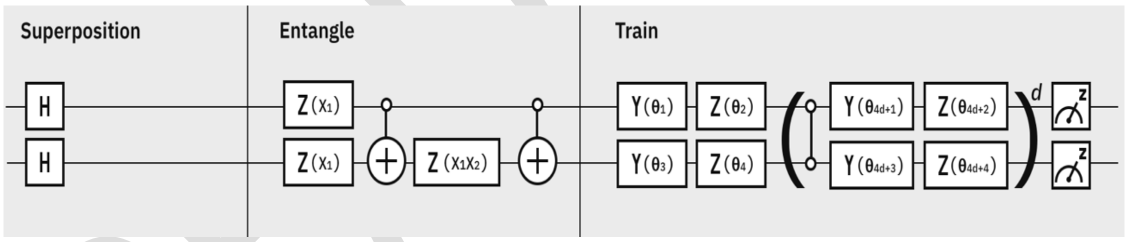


Figure 7 Circuit diagram of a quantum classifier. (Source: IBM Q Experience)

At this point it becomes imperative to highlight that even though most of the computational leg work has been accomplished by the quantum circuit, the pipeline is still reliant on classical computation for the optimization of quantum parameters. This is because classical optimization strategies such as gradient descent do not yet have their simpler quantum equivalents (Schuld & Petruccione, 2018a) and since it affects the overall performance of the algorithm, optimization has remained mostly classical, partly also because the convergence speeds of objective loss functions are both well understood and ubiquitous in classical realms. Another intuition behind this hybrid algorithmic approach is to significantly reduce the requirement of already constrained quantum resources such as number of qubits, circuit depth and coherence time, and utilize these noisy intermediate-scale quantum hardware primarily for solving the classically intractable part of the machine learning problem (Benedetti et al, 2019).

This amalgamation of what is termed as a variational or hybrid quantum-classical setup, has gained huge popularity amongst QML researchers in recent years where it has been employed not only in quantum versions of supervised (Mitarai et al, 2018), but also for unsupervised (Date et al, 2020), generative adversarial (Dallaire-Demers & Killoran, 2018) and variational autoencoders (Khoshaman et al, 2019).

A more specific example relevant to this dissertation is that of the quantum version of SVM (QSVM) which is essentially a hybrid quantum-classical algorithm. Because kernel estimation scales exponentially with the size of the training dataset, its estimation is not efficient via classical techniques more so as the dimensionality of the dataset become huge in practical real-world dataset. The QSVM algorithm solves this problem by a direct estimation of the kernel in quantum feature space during the training phase while the optimization of the loss function still happens classically. The algorithm

2.4 State Preparation Protocols

To avail the benefits of quantum computation in machine learning, the task of information encoding to represent classical features within quantum system becomes not only indispensable but is key in defining the overall performance of the algorithm, especially more so, for the *CQ* case defined in the last section, wherein quantum algorithms require classical data for training purposes. Two prominent strategies of *state preparation* or *quantum embedding* to that end exist – Basis encoding and Amplitude encoding – in both of which the data is encoded in amplitudes of either individual qubits or in the amplitudes of entangled qubits (Grant et al, 2018).

In *basis encoding*, classical data is mapped on the amplitude of corresponding basis vector of the *n*-qubit system. For e.g. to encode a classical decimal value of 3, each bit of its binary string representation i.e. $(011)_2$, is mapped to individual qubits of a 3-qubit system which by extension of eq. 2.2, has a total of $2^3 = 8$ basis vectors in its superposition state. Only the superposition corresponding to $|011\rangle$ basis vector is retained while all the remaining seven basis vectors of the superposition are marked with zero amplitudes.

Similarly, to encode two data points for e.g. 3 and 5 through their binary representation of $(011)_2$ and $(101)_2$, only two basis vectors are required to have non-zero amplitudes. All other amplitudes β, γ, δ ... of remaining basis vectors as shown below are set to zero while fulfilling the necessary requirement of sum of amplitude squares to be 1.

$$|\phi\rangle = \frac{1}{\sqrt{2}}|011\rangle + \frac{1}{\sqrt{2}}|101\rangle + \beta|000\rangle + \gamma|001\rangle + \delta|100\rangle + \cdots$$

In contrast to the above, *amplitude encoding* requires the classical data to be normalized for it be encoded by the amplitudes. For e.g. a two-dimensional classical vector [3,5] will first be normalized to $\left[\frac{3}{\sqrt{3^2+5^2}}, \frac{5}{\sqrt{3^2+5^2}}\right]$ and then associated with the amplitude vector $[\alpha_1, \alpha_2]$ of a qubit.

The encoding of classical data is the very first and crucial step in understanding the quantum advantages that can be leveraged in machine learning and it has been shown that such a 'data superposition' can be created in the time that is linear in *M* and *N*, where *M* is the number of data points in dataset and *N* is dimension of each data point (Ventura & Martinez, 2000). Since amplitudes of qubits are probabilistic distributions, these approaches of encoding information have been debated in light of the nonlinear quantum theories which pose severe limitations on the linear operations that can be executed on the encoded data through unitary transformations (Abrams & Lloyd, 1998).

2.5 Notions of Quantum Speedups – A literature review

It is helpful to visualize what a contemporary quantum computer (QC) might look like in reality, before embarking on the journey of performance boosts – either in theory or reality – that it can provide to current machine learning algorithms. Of the various forms of QCs available, from the ones ring-fenced within the research labs, to those available for public use through cloud platforms, the super-conducting electronic circuit-based implementation (Devoret & Schoelkopf, 2013), where qubits constructed out of cryogenically cooled superconducting material and contained at extremely low temperatures[ii] using dilution refrigerators, has become its distinctive face (Fig.7). As a predominant architecture adopted by the industry giants – such as IBM (Steffen et al, 2011), Google (Courtland, 2017), Intel and Rigetti (Castelvecchi, 2017) – who have all adopted and successfully developed their respective hardware's using this approach, superconducting QCs even with all its complexities have emerged at the forefront of quantum race.

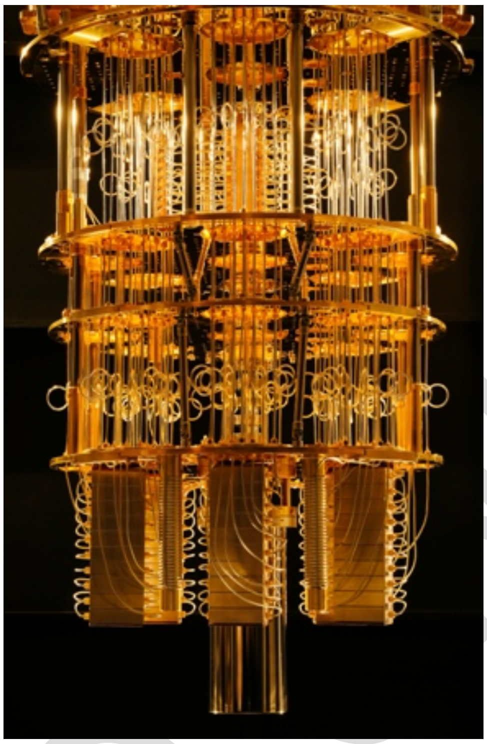


Figure 8 An IBMs superconducting Quantum Computer (Source: IBM Research Blog at ibm.com/blogs)

Alternate architectures do exist, like the one based on quantum annealing called *Adiabatic Quantum Computers*[iii] that have been used to solve certain optimization problems (Neukart et al, 2017) by finding global minima of the objective function. However, compared to the more general-purpose nature of superconducting QCs with gate arrays, the adiabatic versions are highly specialized quantum devices, suitable for optimization and sampling tasks and therefore somewhat restrictive in their scope (Table 1) in training machine learning models to only Boltzmann machines (Amin et al, 2018).

The idea of quantum speedup is embedded in the fact that for solving certain problems, quantum algorithms can in principal outperform their classical counterparts, even though its definition and measurement is debatable (Rønnow et al, 2014). Even still, real advances have been made with these 'unideal' Noisy Intermediate-Scale Quantum (NISQ) devices. The Basic Linear Algebra Subroutines (BLAS), such as solving system of linear equations using matrix inversion, finding eigenvectors and eigenvalues and Fourier transforms – at the very core of many machine learning frameworks (Li et al,

2019) – have been shown to perform exponentially better in quantum domains using HHL[iv] algorithm (Harrow et al, 2009). HHL has invariably become an indispensable ingredient in many of the subsequent QML algorithms and often cited as a milestone algorithm that kick-started the field of quantum machine learning(Childs, 2009).

Standard ML algorithms such as least-squares (Schuld et al, 2016), gradient descent, principal component analysis (Lloyd et al, 2014) and SVMs (Rebentrost et al, 2014), have all, with the use of "quantized" version of BLAS and HHL, exhibited some form of computational speedups over what is achievable classically. However, analogous to the proclaimed "mileage" of automobiles under ideal conditions but never attained in real world use, these quantum speedups too occur under a set of assumptions and pre-conditions (Clader et al, 2013) – such as the existence of qRAM (quantum Random Access Memory) that allows storage of quantum states, access to quantum version of data, and a fast and efficient state preparation protocol for n copies of quantum state – and hence their realizations in pragmatic setups may remain a far cry on the near-term devices. Table 1 shows theoretical quantum speedups with respect to classical counterparts and whether qRAM is required for attaining such speedups where $O(\sqrt{N})$ denotes quadratic and $O(\log N)$ denotes exponential speedups.

Table 1 Quantum speedups in various machine learning algorithms. Source: (Biamonte et al, 2017)

Method	Speedup	AA	HHL	Adiabatic	qRAM
Bayesian Inference	$O(\sqrt{N})$	*Y*	*Y*	*N*	*N*
Online Perceptron	$O(\sqrt{N})$	*Y*	*N*	*N*	*Optional*
Least squares fitting	$O(\log N)$	*Y*	*Y*	*N*	*Y*
Classical BM	$O(\sqrt{N})$	*Y/N*	*Optional/N*	*N/Y*	*Optional*
Quantum BM	$O(\log N)$		*N*	*N/Y*	*N*
Quantum PCA	$O(\log N)$	*N*	*Y*	*N*	*Optional*
Quantum SVM	$O(\log N)$	*N*	*Y*	*N*	*Y*
Quantum reinforcement learning	$O(\sqrt{N})$	*Y*	*N*	*N*	*N*

Because quantum computers are based on qubits which encapsulate superposition of different states, they are often equated to non-deterministic Turing machines (NTM) which can take multiple actions based on their current state and input. Due to this quantum computers are often cited as having potential to solve NP-hard problems (Freedman, 1998). NP-hard or *non-deterministic polynomial time* problems are a class of problems, the solution to which can be found in polynomial time on NTMs. A relevant example in machine learning context is the problem of empirical risk minimization (see e.q. 1.4) for an SVM classifier which has been shown to be NP-hard (Ben-David et al, 2003) and although quantum computers have not yet been proven to mimic a non-deterministic Turing machines they remain the most likely candidates to solve such class of problems if at all.

Not all work in QML though has remained the subject of theoretical interest. Real world applications such as recommender systems in e-commerce have leveraged QML technique of efficiently sampling the correlation matrix of customers and their preferred products to reduce the time complexity from polynomial to poly-logarithmic[v], essentially giving an exponential boost (Kerenidis & Prakash, 2016). In another experimental demonstration of quantum version of SVM, the construction of optimal separating hyperplane for hand-written digit recognition task, was performed in poly-logarithmic times (Li et al, 2015). This utilizes matrix inversion techniques based on HHL algorithm and kernel-based method called Gaussian process regression (Zhao et al, 2019) to gain edge over classical gaussian processes which require $O(n^3)$.

On the hardware side with only a handful number of physical qubits and quantum gate operations, before errors and faults start degrading the algorithm, these NISQ machines generally with 50-100 qubits, are still nascent in their processing capacity and scaling capabilities. Often unable to meet the practical expectations and theoretical efficiencies cited in research work they nevertheless remain a useful tool and significant step towards their more accurate, scalable and fault-tolerant commercial versions of future and even though crossing the “quantum chasm” from a handful to millions of qubits is going to take time it remains an achievable goal (Preskill, 2018). It may not change the world overnight or even imminently, but the current rate of progress looks promising enough to substantively address the scientific and technical challenges of the field in the next ten years (Steffen et al, 2011).

Another formidable hardware challenge that can potentially impact QML is the fabrication of qRAM or quantum random access memories for encoding the classical information. Although simple prototype constructions demonstrating proof-of-principle exist, large scale arrays of robust quantum switches with low error rates required for big data problem still remain technologically out of scope (Arunachalam et al, 2015) .

Although quantum leaps have been made in several enabling technologies at NISQ level devices for execution of these algorithms, the question of quantum speedup remains somewhat un-answered to date. In his 2015 seminal paper – *"Read the fine print"*, Scott Aaronson, a leading quantum-computing researcher, argues that this excitement of exponential speedups needs to be balanced with a sober understanding of the quantum algorithm's 'fine print'. Contrasting quantum and classical machine learning algorithms as he puts it, is a *"simple question with a complicated answer"* – take away some of the stipulations and the quantum advantage may disappear or may theoretically be matched by an equally fast classical algorithm (Aaronson, 2015).

As the exponential growth curve in computation power begins to defect from Moore's law and towards its inevitable end (Waldrop, 2016), our digital future, only through its embracement of QML, can keep pace with the proverbial snowball of big data which is accelerating and growing by the day. That it may not appear to be of any practical use in NISQ-era, due to technological limitations, it still has the potential to revolutionise both current and future ways of intelligent and faster data processing, be it in real world domains of financial modelling (Orús et al, 2019), weather forecasting (Frolov, 2017), clinical diagnosis (Solenov et al, 2018) or research subjects in computational sciences, modern astronomy and drug research, all of which will require new paradigm and faster tools to deal with big data.

Amidst all the excitement that QML brings, several key research questions remain open. The problem of benchmarking to compare the performance of classical learning networks with that of quantum models (Nguyen et al, 2020) has not yet been definitely mapped onto known classical complexity theory. Another not so well researched question relates to how well the quantum learning models generalize. Generalization is a key characteristic of any machine learning model for the classification of unseen data. Intuitively based on the principle of *Occam's Razor*, the idea is to measure how simple or complex a model is. According to the principle a simpler model is more likely to generalize well, produce good accuracy and avoid over-fitting, compared to a complex one (Rasmussen & Ghahramani, 2000).

There is extensive research pertaining to the generalization theory of classical ML based classifiers (Bartlett & Shawe-Taylor, 1998) but similar analysis of determining the factors that guarantee good performance of QML models has mostly remained on the back-burner. Finding optimal quantum circuit architectures that are best suited for machine learning is yet another active research area to find whether such a general architecture even exists. The connections between deep learning and quantum theory, intrinsic noise leading to decoherence and limited qubit connectivity are some other challenges requiring systematic investigations (Perdomo-Ortiz et al, 2018) .

Research in QML so far has been two-fold. One aims to translate the existing classical ML models to their equivalent quantum analogues – the likes of binary classifiers (Ristè et al, 2017) , unsupervised learning (Aïmeur et al, 2013), quantum reinforcement learning (Lamata, 2017) and quantum neural networks (Benedetti et al, 2017) – are some examples, each trying to harvest the speedups offered in the world of quantum computations. The other explores the frontiers of pure quantum algorithms also called 'coherent algorithms' which can exploit physical characteristics and constraints of these devices for e.g. to develop new connections between quantum computing and machine learning for molecular simulations in computational chemistry (Choo et al, 2020). In an inverted approach, as Antonio Mezzacapo, an IBM quantum applications researcher shares, classical neural networks trained using measurement data generated on quantum computers have been used for simulation of molecular energies with extreme precision – something that remains classically prohibitive due to the number of measurements required (Torlai et al, 2020). Progress has even been made towards trialling a universal quantum operating system named *Deltaflow.OS*, cited as both a "technical breakthrough" and an enormous "commercial breakthrough" (Times, 2020) providing a full developmental stack for quantum programmers and machine learning application developers alike.

The focus is therefore not limited just on achieving exponential gains but additionally towards contributing innovative methods suitable for machine learning with wider implications. The current state of QML is no doubt in its infancy with research investments trying to attain the 'holy grail' of exponential speedups by careful choice of datasets and tailored algorithms scratching at the tip of machine learning iceberg, but with the increasing number of experimental demonstrations on real quantum devices the field is set to have a spectrum of real-world use cases.

This literature review would remain incomplete without a discussion on the contemporary mainstream machine learning architecture of neural networks. Over the last decade neural networks have gained some remarkable successes in learning from unstructured data. Be it there use in commercial applications like handwriting and speech recognition to generative models creating fake content, machine learning today has become synonymous with these networks. Expectedly, quantum analogues of the same have been tried and experimented with as early as end of last century (Purushothaman & Karayiannis, 1997). Starting from the fundamental building block of neural networks called *perceptron* (Torrontegui & García-Ripoll, 2019), to full scale deep neural networks such as Quanvolutional Neural Networks (Henderson et al, 2020) the research has mostly remained either theoretical, or simulation based (Behrman et al, 2000), or proof-of-concept translations of Convolutional Neural Networks (CNN) where individual neurons were simply replaced by qubits without any bearing to leverage quantum advantage.

Google's demonstration of 'quantum supremacy' in 2019 is a testimonial to how far the field has come in 40 years when Richard Feynman first proposed the idea of quantum computers for exponential gains to solve problems in physics and chemistry (Feynman, 1982). The 'supremacy' experiment was conducted on a 53-qubit quantum processor for computation of a sampling task in 200 seconds for which a classical supercomputer would have taken 10,000 years (Arute et al, 2019) and even though such speedups are highly tailored and engineered for specific outcomes to showcase quantum advantages, they nevertheless exhibit the potential of what quantum computing can bring to the compute intensive field of machine learning.

Chapter 3

Implementing a QML Algorithm

Here we aim to implement a quantum learning algorithm on a publicly available NISQ device via an online cloud platform called – ‘IBM Q Experience[vi]’. It provides a free quantum computing service for research in both business and science applications to general public through its prototype quantum processors for developing, running and testing quantum programs. As of this writing a 15-qubit quantum processor called *‘ibmq_16_melbourne’* located in Melbourne, Australia was generally available for public use through IBM Q Network (IBM Q, 2020). The following section is an attempt to run the code which implements a quantum version of the classical SVM algorithm on this platform using purely classical data (*CQ* approach). Arguably, even though NISQ quantum devices are not suitable for high data volume ML jobs – which modern classical computers can perform routinely – they still can be leveraged for proof-of-principle of proposed quantum algorithms on smaller datasets and therefore it is not the goal of this section to showcase quantum speed-ups or any like-for-like comparison with classical ML algorithm but rather an exploration into the possibilities of machine learning using contemporary quantum hardware.

3.1 Quantum enhanced SVM

The motivation for this implementation is based on a recently published algorithm proposed for supervised learning using quantum “enhanced” feature space which experimentally demonstrated 100% success rates in binary classification of the dataset chosen for experiment (Havlíček et al, 2019). Arguably, the dataset was tailored specifically such that finding the kernel function was classical intractable or at-least hard and was therefore not necessarily representative of any real-world practical use. Compared to a previous proposal of quantum version of SVMs where exponential gains were shown under the assumption that the quantum feature map of the dataset is readily available in coherent superposition (Rebentrost et al, 2014), this algorithm achieved two quantum advantages – first by estimating the kernel function in quantum domain, that was conjectured to be classically hard, and second by encoding the feature space of classical data (ref. section 2.4) in the quantum state space – it established potential speedups that can be attained on error-corrected quantum devices of future with hundreds of thousands of qubits.

3.2 Quantum Programming Frameworks

Application programming interfaces (APIs) and frameworks used in formulating quantum algorithms and computations have existed for long. Quantum Computing Language (QCL) (Ömer, 2005), Microsoft's Q# (Q Sharp), *Quipper* (Green et al, 2013), *Cirq*, IBM Research's *Qiskit* and very recently *Silq* developed at ETH Zürich (Bichsel et al, 2020) are some prominent examples of domain-specific programming tools used across academia and business. With the advent of quantum machine learning, however, many of these tools have extended their scope to become full-stack quantum machine learning frameworks. Most notably in this regard are the software development kits (SDK) – Qiskit, TensorFlow Quantum (Broughton et al, 2020) and PennyLane (Bergholm et al, 2018) – all of which are open-source and written in Python programming language for rapid implementation of hybrid quantum-classical models on real quantum devices. It is note-worthy that these libraries are still evolving and in active phase of being developed, often leading to teething issues with users who need to constantly work towards their code's upkeep.

To implement the quantum version of SVM, Qiskit SDK provides a machine learning module called `'qiskit.ml'` containing a set of helper functions and some of the popular classification training sets such as breast cancer, iris and wine datasets. Another module `'qiskit.aqua.algorithms'` contains algorithmic functions for quantum classifiers such as `QSVM` (Quantum SVM) and `VQC` (Variational Quantum Classifier) which encapsulate bulk of the intricacies that we've discussed in this work, such as quantum state preparation protocol, quantum circuit generation, compilation and scheduling the execution of code on the IBM Q network, all behind a clean interface thereby leaving its users to focus on machine learning pipeline and not so much on the quantum aspects of the implementation. As a consequence, a complete implementation of QSVM classification using Qiskit can be written in roughly 20 lines of code, something that would otherwise take hundreds if not thousands of lines to implement each quantum gate and the classical optimization algorithm.

3.3 Walkthrough of Code

Here some salient points explaining certain key functions used in the code attached in Appendix B are given. The code has been adapted from a legacy tutorial available on the Qiskit GitHub page the reference to which is provided in the code. As with any machine learning code the first step is to load and split the available data into training and test datasets. Since this is a supervised learning algorithm the labels are accordingly split. The bult-in breast cancer dataset is used for a binary classification in this demonstration and loaded by calling the `breast_cancer()` function to split it in training and test sets of sizes 20 and 10 respectively (Fig. 9) with each data point having two feature dimensions. Since the original dataset has more than two features the function also reduces the dimension down to 2 per data point using principal component analysis (PCA). For the purpose of this implementation we can consider this dataset as some ad-hoc randomly generated data points belonging to two different classes as shown colour coded in the plot below.

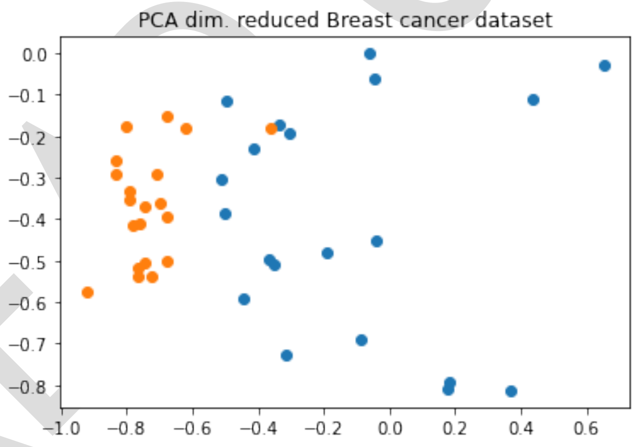


Figure 9 Dataset used for QSVM binary classification

Qiskit aqua provides a pre-built data-encoding function called `ZZFeatureMap()` that can construct a quantum circuit $\mathcal{U}_{\emptyset(x)}$ for translating this classical dataset to a quantum feature space where $\emptyset(x)$ is some non-linear classical function that maps the original non-linear space of the dataset to a higher dimensional feature space where data is linearly separable (refer Fig. 3). Additionally, the function `ZZFeatureMap` takes two arguments:

`feature_dimension` - equal to the dimension of input data, which in our case is 2, and `reps` - which defines the number of times that quantum circuit is repeated.

The core of this algorithm is however concentrated in one single function – `QSVM()` that implicitly performs two different tasks – the quantum kernel estimation and the classical optimization. The transformed feature maps of training and test data obtained above through `ZZFeatureMap`(), and the instance of the quantum hardware to execute the training phase on, are passed as the parameters to this function. In this implementation we provide the IBM Q 15-qubit device as the backend on which to run the QSVM algorithm. Although it is possible to use training data with 15 feature dimensions on such a device, the resultant quantum circuit for QSVM become far too deep than what is currently supported on the device and hence is not a viable option in this case. The problem with deep circuits on NISQ devices also poses issues of lower accuracy, due to noise and decoherence during computations, on the test dataset.

A full quantum circuit representation of the algorithm is attached for reference in the Appendix C that details various quantum logic gates and the measurements occurring on the quantum hardware. The generated circuit diagram primarily comprises of three main quantum gates namely – U1, U2 and CNOT (controlled NOT) which form a set of basis gates for any quantum algorithm that is run on the IBM Q devices. As mentioned earlier the SWAP gate required for computing the inner product of qubits for kernel matrix estimation is implemented via this standard gate set of single qubit rotation (U1 and U2) and CNOT gates in Qiskit. The reason behind this is the efficiency with which these gates can be implemented on the available hardware. The U1 gate is executed in software via a frame change which means that the gate time for this is practically zero, leading to faster runtimes for machine learning applications. U2 on the other hand is a $\pi/2$ rotation around the x-axis (ref to Fig. 4) with a gate time of one unit. Since both these gates have an error rate which are immeasurable, it results in a circuit with practically zero error and better fidelity during actual gate executions (McKay et al, 2017).

It is necessary to re-iterate that quantum backend is only used for the kernel estimation and that the optimization of the training error still happens classically. In addition, due to the probabilistic nature of the quantum measurements, the algorithm also requires the number of times each circuit needs to be executed and measured. This value is provided by the `shots` parameter and instructs the final probability values to be calculated based on that many trials of the circuit. The accuracy of the final classification results will rise with the number of `shots` although at a cost of longer execution times on the hardware.

The output of the function `QSVM()` are the test accuracy and kernel matrix (also known as *Gram matrix*). Since the total number of data point used in the training is 40 (20 for each class), the kernel obtained is a square matrix of the size 40 x 40 and the same has been plotted and attached in the Appendix C. The accuracy on the test dataset obtained in this implementation only was 60%, which is not high enough and is down to some key factors chosen for this run (Fig.10). The accuracy is calculated as the ratio of total number of correct predictions to the total number of predictions in the test set. In this case we have a total of 20 class labels (10 in each class) out of which 9 from the first class and only 3 from the other class are predicted correctly so the accuracy is determined as [(9+3)/20 = 0.6].

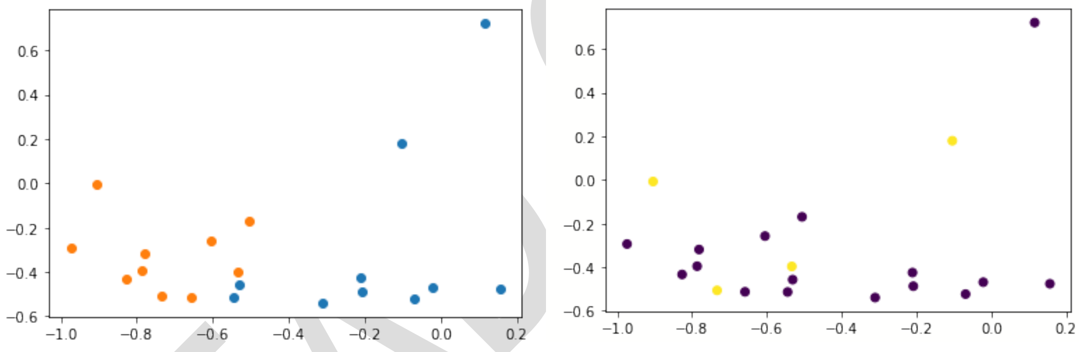


Figure 10 Actual class labels (left) vs predicted classes (right)

There are multiple reasons behind this low accuracy obtained in the run. Firstly, the data set was reduced down to just two dimensions which means some information about the features is completely lost during this transformation resulting in low accuracy. The limitations imposed by fair-share usage policy of the available quantum cloud services means the jobs submitted for performing the computations are often queued up and cannot be performed with arbitrary precisions and high number of shots. In practical setups such as this, the time taken to run the circuit is proportional to the depth of circuit (no. of gates involved) resulting in a trade-off between the accuracy of the classification task and the probability of it completing and returning the results. For these reasons, rapid prototyping is often performed on the quantum simulators, such as *ibm_qasm_simulator*, which are implementations of the gates on the classical computers and have no limitations other than the number of qubits that can be utilized.

Conclusion

By a way of introduction to this emerging and interdisciplinary field of QML we have attempted to not only lay out – through comprehensive literature review – the shifting perceptions from the classical to the quantum, but also demonstrated in a tangible attempt this new machine learning paradigm is realizable on a real device and thereby evidenced that it is indeed a quantum leap for the digital-future, which we can contemplate is not going to be digital alone.

This exposition commenced with the question of whether the use of quantum resources in machine learning exhibit any advantage. In answering that, we've presented – the shared mathematical grounds between the two fields such as high dimensional vector spaces, the central tenets of 'superposition' and 'entanglement' in quantum mechanics which give rise to exponential speedups, the volume and pace of on-going research leading to efficient algorithms, and the rapidly increasing capabilities of quantum hardware for practical implementations – as some of the notable arguments to qualify our stand of meaningful quantum advantage in machine learning.

The idea behind present work was not only to showcase QML in limelight but also to scientifically argue that the possibility of experimental realizations of long existing theoretical proposals – however limited that may be on current devices – should be adhered to with well-grounded expectations. Clearly a successful realization of commercial ML application on NISQ device is a milestone that needs to be pursued, but by not losing sight of the open problems in the field we can hope to remain grounded amid exhilarating expectations of this rapidly growing field which is poised to bring a change nothing less than what the digital computers have brought in last 50 years.

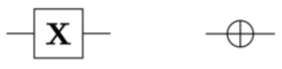
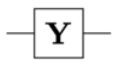
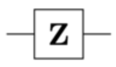
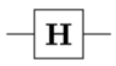
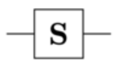
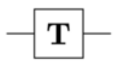
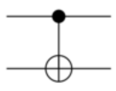
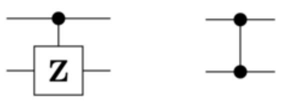
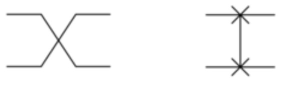
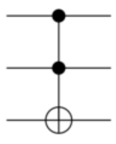
Much of the inherent limitations of quantum computers is due to their constant interaction with the environment. Recent advancements in 'hot qubits' that can remain coherent for longer duration at higher temperatures and topological qubits that can mimic macroscopic properties will allow researchers to create quantum circuits with greater depth. This will lead to quantum algorithms with more expressive powers and enhance their model capacity for commercial and industrial machine learning applications. Computational processing hardware such CPUs, GPUs (Graphical Processing Units), TPUs (Tensor

Processing Units), have already greatly improved the performance of computationally intensive machine learning applications in recent years. Quantum Processing Units (QPU) are inevitably the next logical step in that line and perhaps in future will be as commonly accessible as its counterparts, marking a new shift in how machine learning applications are deployed and executed. Just as in classical computing where existing generation of processors were leveraged in designing faster and efficient next generation of processors, perhaps quantum machine learning can be put to task of solving problems such as design of quantum processors that are more resilient to noise and errors creating a symbiotic feedback loop of each enhancing the other.

In summary, we hope to have expressed enough and necessary arguments that quantum computing and machine learning are not seemingly orthogonal to each other but can greatly benefit in their collaboration together. How this collaboration ultimately fares in the face of future big-data problems with the discovery of new applications of hybrid quantum-classical machine learning algorithms remains to be seen.

Appendix

A) List of commonly used quantum logic gates with their circuit representation and corresponding unitary matrices. The gates U1 and U2 used in the implementation of QSVM algorithm are specific cases of the Pauli-Z (Z) gate (Source: Wikipedia.org)

Operator	**Gate(s)**	**Matrix**
Pauli-X (X)		$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S, P)		$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & -1\end{bmatrix}$
SWAP		$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 1 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$

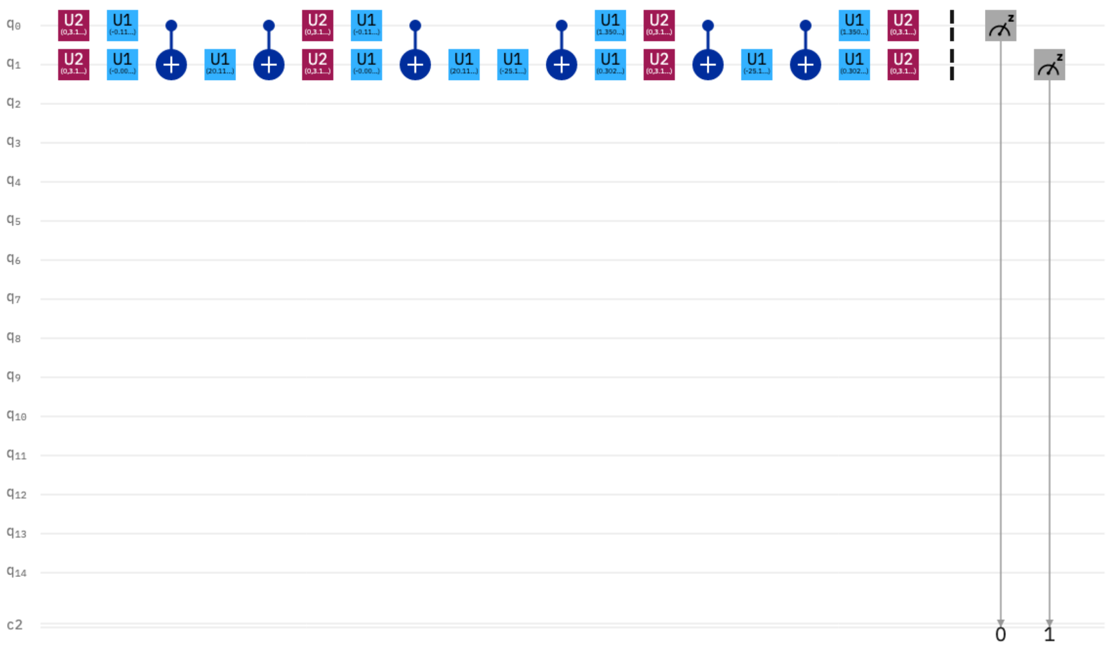
B) The Qiskit code listing used for performing the binary classification. It has been adapted from the legacy tutorials available on Qiskit github page here - https://github.com/Qiskit/qiskit-tutorials/blob/ff90e6d74d3c5f7590bdf5cd341c848adddc596d/legacy_tutorials/aqua/machine_learning/qsvm_classification.ipynb

```
#If this is being run on a jupyter notebook, install the qiskit
package library first.

!pip install qiskit.aqua==0.7.1;
!pip install qiskit-aer;
!pip install qiskit-ibmq-provider;

# Import required libraries
from qiskit import BasicAer
from qiskit import IBMQ
from qiskit.aqua.utils import split_dataset_to_data_and_labels,
map_label_to_class_name
from qiskit.aqua import QuantumInstance
from qiskit.aqua.algorithms import QSVM
from qiskit.ml.datasets import breast_cancer
from qiskit.circuit.library import ZZFeatureMap

import matplotlib.pyplot as plt
import numpy as np

# Note that the token below is only used for IM906 dissertation
purposes and will not work post submission of the same. New tokens
can be obtained by registering for a new account and replacing the
token value from: https://quantum-computing.ibm.com/

IBMQ.enable_account('d7240f0ccbd55371c19b47210b0c614678bd3798dffc3e
90567a8ac9381eea3de5ca8f7786da547d2b334b675dccf51e6cd4708b747d4741d
79d05c137b7dd08')

# Get IBM Q backend provider
ibmq_provider = IBMQ.get_provider()
ibmq_provider.backends()
```

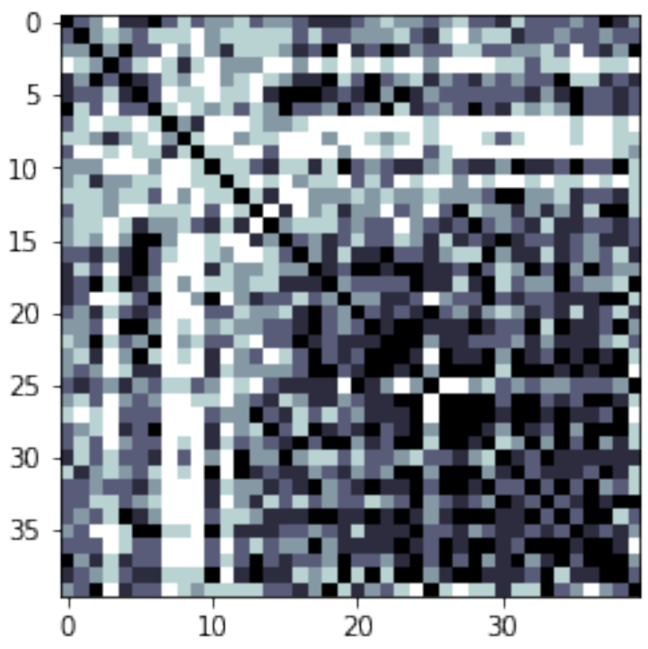
```
# To use the simulator instead of real quantum device for quick
prototyping use the simulator backend uncomment the line
# ibmq_sim_backend = BasicAer.get_backend('qasm_simulator')

# IBM 16 qubit backend
#ibmq_backend = ibmq_provider.get_backend('ibmq_16_melbourne')
print("IBM Q 16 qubit backend available !")

# Five qubit backends
#ibmq_backend5 = ibmq_provider.get_backend('ibmq_vigo')
ibmq_backend5 = ibmq_provider.get_backend('ibmq_santiago')
print("IBM Q 5 qubit backend available !")

# Load the breast cancer dataset. We'll use 20 data points for
training and 10 data points for test purposes.
feature_dim = 2
sample_total,training_data,test_data,labels=
breast_cancer(training_size=20,test_size=10,n=feature_dim,
plot_data=True)

# Seed enables results to be reproducible again in subsequent runs.
seed_value = 1880563

feature_map  =  ZZFeatureMap(feature_dimension=feature_dim,  reps=2,
entanglement='linear')
qsvm = QSVM(feature_map, training_data, test_data)

ibmq_instance     =     QuantumInstance(ibmq_backend5,     shots=5,
seed_simulator=seed_value,seed_transpiler=seed_value,
skip_qobj_validation=False)

ibmq_result = qsvm.run(ibmq_instance)

print("All jobs finished. QSVM Run complete. Results are available.")
print(f"Testing Accuracy: {ibmq_result['testing_accuracy'] * 100}%")

# This will show a plot of the kernel matrix estimated on the quantum
device
```

```
print("Estimated Kernel Matrix:")
kernel_matrix = result_bc['kernel_matrix_training']
img=plt.imshow(np.asmatrix(kernel_matrix),interpolation='nearest',o
rigin='upper',cmap='bone_r')
plt.show()
# Plot the actual and predicted class labels for the test set
test_set = np.concatenate((test_data['B'], test_data['A']))
y_test = qsvm.predict(test_set, ibmq_instance)
plt.scatter(test_set[:, 0], test_set[:,1], c=y_test)
plt.show()

plt.scatter(test_data['A'][:,0], test_data['A'][:,1])
plt.scatter(test_data['B'][:,0], test_data['B'][:,1])
plt.show()
```

C) The circuit diagram of the QSVM algorithm generated through Qiskit. This is the circuit that was run on the IBM Q backend on the real quantum device. Various gates such as U2, U1 and CNOT are shown with the measurement happening at the very end.



Kernel of (Gram) matrix of size 40 x 40 estimated on the quantum hardware by running QSVM algorithm.


0
5
10
15
20
25
30
35
0
10
20
30

References

Aaronson, S. (2015) Read the fine print. *Nature Physics*, 11(4), 291-293.

Abrams, D. S. & Lloyd, S. (1998) Nonlinear Quantum Mechanics Implies Polynomial-Time Solution for NP-Complete and # P Problems. *Physical Review Letters*, 81(18), 3992-3995.

Alvarez-Rodriguez, U., Lamata, L., Escandell-Montero, P., Martín-Guerrero, J. D. & Solano, E. (2017) Supervised Quantum Learning without Measurements. *Scientific Reports*, 7(1), 13645.

Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B. & Melko, R. (2018) Quantum Boltzmann Machine. *Physical Review X*, 8(2), 021050.

Arunachalam, S., Gheorghiu, V., Jochym-O'Connor, T., Mosca, M. & Srinivasan, P. V. (2015) On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12), 123010.

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G. S. L., Buell, D. A., Burkett, B., Chen, Y., Chen, Z., Chiaro, B., Collins, R., Courtney, W., Dunsworth, A., Farhi, E., Foxen, B., Fowler, A., Gidney, C., Giustina, M., Graff, R., Guerin, K., Habegger, S., Harrigan, M. P., Hartmann, M. J., Ho, A., Hoffmann, M., Huang, T., Humble, T. S., Isakov, S. V., Jeffrey, E., Jiang, Z., Kafri, D., Kechedzhi, K., Kelly, J., Klimov, P. V., Knysh, S., Korotkov, A., Kostritsa, F., Landhuis, D., Lindmark, M., Lucero, E., Lyakh, D., Mandrà, S., McClean, J. R., McEwen, M., Megrant, A., Mi, X., Michielsen, K., Mohseni, M., Mutus, J., Naaman, O., Neeley, M., Neill, C., Niu, M. Y., Ostby, E., Petukhov, A., Platt, J. C., Quintana, C., Rieffel, E. G., Roushan, P., Rubin, N. C., Sank, D., Satzinger, K. J., Smelyanskiy, V., Sung, K. J., Trevithick, M. D., Vainsencher, A., Villalonga, B., White, T., Yao, Z. J., Yeh, P., Zalcman, A., Neven, H. & Martinis, J. M. (2019) Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779), 505-510.

Aïmeur, E., Brassard, G. & Gambs, S. (2013) Quantum speed-up for unsupervised learning. *Machine Learning*, 90(2), 261-287.

Balabin, R. M. & Lomakina, E. I. (2011) Support vector machine regression (SVR/LS-SVM)—an alternative to neural networks (ANN) for analytical chemistry? Comparison of nonlinear methods on near infrared (NIR) spectroscopy data. *Analyst*, 136(8), 1703-1712.

Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P., Sleator, T., Smolin, J. A. & Weinfurter, H. (1995) Elementary gates for quantum computation. *Physical Review A*, 52(5), 3457-3467.

Bartlett, P. & Shawe-Taylor, J. (1998) Generalization Performance of Support Vector Machines and Other Pattern Classifiers.

Behrman, E. C., Nash, L. R., Steck, J. E., Chandrashekar, V. G. & Skinner, S. R. (2000) Simulations of quantum neural networks. *Information Sciences*, 128(3), 257-269.

Bell, J. S. (1964) On the Einstein Podolsky Rosen paradox. *Physics Physique Fizika*, 1(3), 195-200.

Bellman, R. (1961) Adaptive control processes a guided tour.

Ben-David, S., Eiron, N. & Long, P. M. (2003) On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3), 496-514.

Benedetti, M., Lloyd, E., Sack, S. & Fiorentini, M. (2019) Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4), 043001.

Benedetti, M., Realpe-Gómez, J. & Perdomo-Ortiz, A. (2017) Quantum-assisted Helmholtz machines: A quantum-classical deep learning framework for industrial datasets in near-term devices. *ArXiv*, abs/1708.09784.

Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Sohaib Alam, M., Ahmed, S., Arrazola, J. M., Blank, C., Delgado, A., Jahangiri, S., McKiernan, K., Meyer, J. J., Niu, Z., Száva, A. & Killoran, N. (2018) PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv e-prints*, arXiv:1811.04968.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N. & Lloyd, S. (2017) Quantum machine learning. *Nature*, 549, 195.

Bichsel, B., Baader, M., Gehr, T. & Vechev, M. (2020) Silq: a high-level quantum language with safe uncomputation and intuitive semantics, *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation.* London, UK: Association for Computing Machinery, 286–300.

Blank, C., Park, D. K., Rhee, J.-K. K. & Petruccione, F. (2020) Quantum classifier with tailored quantum kernel. *npj Quantum Information*, 6(1), 41.

Bloch, F. (1946) Nuclear Induction. *Physical Review*, 70(7-8), 460-474.

Born, M. (1926) Zur Quantenmechanik der Stoßvorgänge. *Zeitschrift fur Physik*, 37, 863-867.

Boser, B. E., Guyon, I. M. & Vapnik, V. N. (1992) A training algorithm for optimal margin classifiers, *Proceedings of the fifth annual workshop on Computational learning theory.* Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 144–152.

Bottou, L., Curtis, F. E. & Nocedal, J. (2018) Optimization Methods for Large-Scale Machine Learning. *SIAM Review*, 60(2), 223-311.

Breiman, L. (2001) Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statist. Sci.*, 16(3), 199-231.

Broughton, M., Verdon, G., McCourt, T., Martinez, A. J., Yoo, J. H., Isakov, S. V., Massey, P., Yuezhen Niu, M., Halavati, R., Peters, E., Leib, M., Skolik, A., Streif, M., Von Dollen, D., McClean, J. R., Boixo, S., Bacon, D., Ho, A. K., Neven, H. & Mohseni, M. (2020) TensorFlow Quantum: A Software Framework for Quantum Machine Learning. *arXiv e-prints*, arXiv:2003.02989.

Castelvecchi, D. (2017) Quantum computers ready to leap out of the lab in 2017. *Nature*, 541(7635), 9-10.

Childs, A. M. (2009) Quantum algorithms: Equation solving by simulation. *Nature Physics*, 5, 861.

Choo, K., Mezzacapo, A. & Carleo, G. (2020) Fermionic neural-network states for ab-initio electronic structure. *Nature Communications*, 11(1), 2368.

Clader, B. D., Jacobs, B. C. & Sprouse, C. R. (2013) Preconditioned Quantum Linear System Algorithm. *Physical Review Letters*, 110(25), 250504.

Cotter, A., Keshet, J. & Srebro, N. (2011) Explicit Approximations of the Gaussian Kernel. *arXiv e-prints*, arXiv:1109.4603.

Courtland, R. (2017) Google aims for quantum computing supremacy [News]. *IEEE Spectrum*, 54(6), 9-10.

Dallaire-Demers, P.-L. & Killoran, N. (2018) Quantum generative adversarial networks. *Physical Review A*, 98(1), 012324.

Date, P., Schuman, C., Patton, R. & Potok, T. (2020) A Classical-Quantum Hybrid Approach for Unsupervised Probabilistic Machine Learning, *Advances in Information and Communication*. Cham, 2020//. Springer International Publishing.

Devoret, M. H. & Schoelkopf, R. J. (2013) Superconducting Circuits for Quantum Information: An Outlook. *Science*, 339, 1169 - 1174.

Dirac, P. A. M. (1939) A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35(3), 416-418.

Djordjevic, I. (2012) Chapter 1 - Introduction, in Djordjevic, I. (ed), *Quantum Information Processing and Quantum Error Correction*. Oxford: Academic Press, 1-27.

Domingos, P. (2012) A few useful things to know about machine learning. *Commun. ACM*, 55(10), 78–87.

Feynman, R. P. (1982) Simulating physics with computers. *Int. J. Theor. Phys*, 21(6/7).

Freedman, M. H. (1998) P/NP, and the quantum field computer. *Proceedings of the National Academy of Sciences of the United States of America*, 95(1), 98-101.

Frolov, A. V. (2017) Can a quantum computer be applied for numerical weather prediction? *Russian Meteorology and Hydrology*, 42(9), 545-553.

Grant, E., Benedetti, M., Cao, S., Hallam, A., Lockhart, J., Stojevic, V., Green, A. G. & Severini, S. (2018) Hierarchical quantum classifiers. *npj Quantum Information*, 4(1), 65.

Green, A. S., Lumsdaine, P. L., Ross, N. J., Selinger, P. & Valiron, B. (2013) An Introduction to Quantum Programming in Quipper, *Reversible Computation*. Berlin, Heidelberg, 2013//. Springer Berlin Heidelberg.

Grinblat, G. L., Izetta, J. & Granitto, P. M. (2010) SVM Based Feature Selection: Why Are We Using the Dual?, *Advances in Artificial Intelligence – IBERAMIA 2010*. Berlin, Heidelberg, 2010//. Springer Berlin Heidelberg.

Grover, L. K. (1996) A fast quantum mechanical algorithm for database search. *arXiv e-prints*, quant-ph/9605043.

Harrow, A. W., Hassidim, A. & Lloyd, S. (2009) Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15), 150502.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M. & Gambetta, J. M. (2019) Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747), 209-212.

Henderson, M., Shakya, S., Pradhan, S. & Cook, T. (2020) Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1), 1-9.

Hofmann, T., Scholkopf, B. & Smola, A. J. (2008) Kernel methods in machine learning. *Ann. Statist.*, 36(3), 1171-1220.

José Luis, R.-Á., Manel, M.-R., Jordi, M., ntilde, oz, M., iacute & Gustau, C.-V. (2018) Support Vector Machine and Kernel Classification Algorithms, *Digital Signal Processing with Kernel Methods*IEEE, 433-502.

Kauffman, L. H. & Lomonaco, S. J. (2004) Braiding operators are universal quantum gates. *New Journal of Physics*, 6, 134-134.

Kerenidis, I. & Prakash, A. (2016) Quantum Recommendation Systems. *arXiv e-prints*, arXiv:1603.08675.

Khoshaman, A., Vinci, W., Denis, B., Andriyash, E., Sadeghi, H. & Amin, M. H. (2019) Quantum variational autoencoder. *Quantum Science and Technology*, 4, 014001.

Lamata, L. (2017) Basic protocols in quantum reinforcement learning with superconducting circuits. *Scientific Reports*, 7(1), 1609.

Li, F., Ye, Y., Tian, Z. & Zhang, X. (2019) CPU versus GPU: which can perform matrix computation faster—performance comparison for basic linear algebra subprograms. *Neural Computing and Applications*, 31(8), 4353-4365.

Li, J. & Liu, H. (2017) Challenges of Feature Selection for Big Data Analytics. *IEEE Intelligent Systems*, 32(2), 9-15.

Li, Z., Liu, X., Xu, N. & Du, J. (2015) Experimental Realization of a Quantum Support Vector Machine. *Physical Review Letters*, 114(14), 140504.

Lloyd, S., Mohseni, M. & Rebentrost, P. (2014) Quantum principal component analysis. *Nature Physics*, 10(9), 631-633.

Lubasch, M., Joo, J., Moinier, P., Kiffner, M. & Jaksch, D. (2020) Variational quantum algorithms for nonlinear problems. *Physical Review A*, 101(1), 010301.

McKay, D. C., Wood, C. J., Sheldon, S., Chow, J. M. & Gambetta, J. M. (2017) Efficient Z gates for quantum computing. *Physical Review A*, 96(2), 022330.

Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. (2018) Quantum circuit learning. *Physical Review A*, 98(3), 032309.

Neukart, F., Compostella, G., Seidel, C., von Dollen, D., Yarkoni, S. & Parney, B. (2017) Traffic flow optimization using a quantum annealer. *arXiv e-prints*, arXiv:1708.01625.

Nguyen, N. H., Behrman, E. C., Moustafa, M. A. & Steck, J. E. (2020) Benchmarking Neural Networks For Quantum Computations. *IEEE Transactions on Neural Networks and Learning Systems*, 31(7), 2522-2531.

Nielsen, M. A. & Chuang, I. L. (2010) Fundamental concepts, in Chuang, I. L. & Nielsen, M. A. (eds), *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge: Cambridge University Press.

Orús, R., Mugel, S. & Lizaso, E. (2019) Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4, 100028.

Perdomo-Ortiz, A., Benedetti, M., Realpe-Gómez, J. & Biswas, R. (2018) Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Science and Technology*, 3(3), 030502.

Preskill, J. (2018) Quantum Computing in the NISQ era and beyond. *arXiv e-prints*, arXiv:1801.00862.

Purushothaman, G. & Karayiannis, N. B. (1997) Quantum neural networks (QNNs): inherently fuzzy feedforward neural networks. *IEEE Transactions on Neural Networks*, 8(3), 679-693.

Rasmussen, C. E. & Ghahramani, Z. (2000) Occam's Razor, *Proceedings of the 13th International Conference on Neural Information Processing Systems.* Denver, CO: MIT Press, 276–282.

Rebentrost, P., Mohseni, M. & Lloyd, S. (2014) Quantum Support Vector Machine for Big Data Classification. *Physical Review Letters*, 113(13), 130503.

Ristè, D., da Silva, M. P., Ryan, C. A., Cross, A. W., Córcoles, A. D., Smolin, J. A., Gambetta, J. M., Chow, J. M. & Johnson, B. R. (2017) Demonstration of quantum advantage in machine learning. *npj Quantum Information*, 3(1), 16.

Russell, S. (1996) Chapter 4 - Machine Learning, in Boden, M. A. (ed), *Artificial Intelligence*. San Diego: Academic Press, 89-133.

Rønnow, T. F., Wang, Z., Job, J., Boixo, S., Isakov, S. V., Wecker, D., Martinis, J. M., Lidar, D. A. & Troyer, M. (2014) Defining and detecting quantum speedup. *Science*, 345(6195), 420.

Schuld, M. & Killoran, N. (2019) Quantum Machine Learning in Feature Hilbert Spaces. *Physical Review Letters*, 122(4), 040504.

Schuld, M. & Petruccione, F. (2018a) Information Encoding, in Schuld, M. & Petruccione, F. (eds), *Supervised Learning with Quantum Computers*. Cham: Springer International Publishing, 139-171.

Schuld, M. & Petruccione, F. (2018b) Introduction, in Schuld, M. & Petruccione, F. (eds), *Supervised Learning with Quantum Computers*. Cham: Springer International Publishing, 1-19.

Schuld, M., Sinayskiy, I. & Petruccione, F. (2014) The quest for a Quantum Neural Network. *Quantum Information Processing*, 13(11), 2567-2586.

Schuld, M., Sinayskiy, I. & Petruccione, F. (2016) Prediction by linear regression on a quantum computer. *Physical Review A*, 94(2), 022342.

Schumacher, B. (1995) Quantum coding. *Physical Review A*, 51(4), 2738-2747.

Schölkopf, B. (2000) The kernel trick for distances, *Proceedings of the 13th International Conference on Neural Information Processing Systems.* Denver, CO: MIT Press, 283–289.

Shor, P. W. (1994) Algorithms for quantum computation: discrete logarithms and factoring, *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 20-22 Nov. 1994.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. & Hassabis, D. (2016) Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 484-489.

Sim, S., Johnson, P. D. & Aspuru-Guzik, A. (2019) Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. *Advanced Quantum Technologies*, 2(12), 1900070.

Solenov, D., Brieler, J. & Scherrer, J. F. (2018) The Potential of Quantum Computing and Machine Learning to Advance Clinical Research and Change the Practice of Medicine. *Missouri medicine*, 115(5), 463-467.

Steffen, M., DiVincenzo, D. P., Chow, J. M., Theis, T. N. & Ketchen, M. B. (2011) Quantum computing: An IBM perspective. *IBM Journal of Research and Development*, 55(5), 13:1-13:11.

Steinberg, A. M. (2017) Quantum measurements: a modern view for quantum optics experimentalists, *Quantum Optics and Nanophotonics*. Oxford: Oxford University Press.

Suykens, J. A. K. & Vandewalle, J. (1999) Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, 9(3), 293-300.

Times, F. (2020) *Quantum operating system trialled successfully*, 2020. Available online: https://www.ft.com/content/2309b2cb-f122-4366-8cc5-7d186e8cd949 [Accessed: 01-Sep-2020].

Torlai, G., Mazzola, G., Carleo, G. & Mezzacapo, A. (2020) Precise measurement of quantum observables with neural-network estimators. *Physical Review Research*, 2(2), 022060.

Torrontegui, E. & García-Ripoll, J. J. (2019) Unitary quantum perceptron as efficient universal approximator. *EPL (Europhysics Letters)*, 125(3), 30004.

Vapnik, V. & Chervonenkis, A. (1974) Theory of pattern recognition. Nauka, Moscow.

Ventura, D. & Martinez, T. (2000) Quantum associative memory. *Information Sciences*, 124(1), 273-296.

Waldrop, M. M. (2016) The chips are down for Moore's law, *Nature*. England, 144-7.

Wilmott, C. & Wild, P. R. (2014) Towards an optimal swap gate. *Quantum Information Processing*, 13(6), 1467-1482.

Wittek, P. (2014) 1 - Introduction, in Wittek, P. (ed), *Quantum Machine Learning*. Boston: Academic Press, 3-10.

Zeng, B., Chen, X., Zhou, D.-L. & Wen, X.-G. (2019) Correlation and Entanglement, in Zeng, B., Chen, X., Zhou, D.-L. & Wen, X.-G. (eds), *Quantum Information Meets Quantum Matter: From Quantum Entanglement to Topological Phases of Many-Body Systems*. New York, NY: Springer New York, 3-35.

Zhao, Z., Fitzsimons, J. K. & Fitzsimons, J. F. (2019) Quantum-assisted Gaussian process regression. *Physical Review A*, 99(5), 052331.

Zhuang, Q. & Zhang, Z. (2019) Physical-Layer Supervised Learning Assisted by an Entangled Sensor Network. *Physical Review X*, 9(4), 041023.

Ömer, B. (2005) Classical Concepts in Quantum Programming. *International Journal of Theoretical Physics*, 44, 943-955.

15-qubit backend: IBM Q team. IBM Q 15 Melbourne backend specification v1.0.1. https://quantum-computing.ibm.com (2020).

[i] Ket-notation is part of *bra-ket* notation, introduced by Dirac in 1939, for representation of vectors in a complex Hilbert space using angled bracket symbols '⟨' and '⟩'
[ii] Of the order of 10 milli-Kelvins.
[iii] D-Wave is a well-known industry leader in this field having commercialized their adiabatic quantum computers as early as 2011.
[iv] Named after their inventors – Harrow, Hassidim and Lloyd.
[v] Polylogarithmic times are polynomials in log scales of input size i.e. poly(log N).
[vi] https://quantum-computing.ibm.com/