National College of Ireland

Current Applicability of Quantum Machine Learning to Data Analytics

MSc Research Project
Data Analytics

Stephen McMullan
Student ID: x19139497

School of Computing
National College of Ireland

Supervisor: Dr. Majid Latifi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Stephen McMullan
Student ID:	x19139497
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Majid Latifi
Submission Due Date:	16/08/2021
Project Title:	Current Applicability of Quantum Machine Learning to Data Analytics
Word Count:	6773
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	19th September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Current Applicability of Quantum Machine Learning to Data Analytics

Stephen McMullan
x19139497

Abstract

In recent years, quantum computing and its application to machine learning have evolved to the point where the data analytics practitioner must ask whether the technology is ready to aid large scale data processing tasks. This research describes the state of the art along with the limitations of error prone current generation quantum processors in the so called NISQ (Noisy Intermediate Stage Quantum) era. An introduction to the cloud based IBM Quantum infrastructure and Qiskit software application development kit is given. The design and implementation in software of quantum classifiers based on the distance measurement between quantum states using the SWAP Test circuit is examined. A method of using quantum feature maps and quantum kernel estimation for use with SVM classifiers on conventional computers is described. This research illustrates the feasibility of application of current quantum hardware technology to multi-dimensional datasets with a reasonable runtime performance. Future directions for research into suitable datasets and novel parameterized quantum feature map circuits is advised to empirically demonstrate advantage for fast, accurate classifications over conventional ML methods. This process will be aided by the ongoing improvement in quantum hardware capabilities and larger scale access to quantum computing services.

1 Introduction

Quantum Machine Learning (QML) is a very recent application within the realm of quantum computing (Benioff; 1980). It is an area of research bringing together aspects of mathematics, physics, computer science and data analytics (Feynman; 1981). This introduction provides a concise description of quantum computing and continues through to the current application of QML techniques in data analytics.

1.1 Introduction to Quantum Computing

There are several physical fabrication methods being pursued for the production of quantum computing devices. The most prevalent of these are superconducting circuits maintained at very low temperatures to reduce the effects of environmental interference. Other methods include photonics and trapped ion systems. This current generation of non-error corrected quantum processors belong to the so called Noisy Intermediate Stage Quantum (NISQ) era of devices.

Quantum information can be described using the quantum bit (qubit) which is analogous to the binary digit (bit) representing the values 0 or 1. Unlike the n-bit registers of conventional computing which can represent one n-bit value at any one time as a bit string, an n-qubit system can represent 2^n values simultaneously through superposition in an n-dimension Hilbert space. The additional quantum effects of entanglement and interference allow a parallel processing effect with a probabilistic outcome to produce the most likely results of a processing operation. This new paradigm of computing allowed the development of algorithms like Grover's and Shor's which provided solutions for problems that would be considered intractable for conventional computers arousing great interest in the field (Shor; 1994) (Grover; 1996).

Another motivating factor is the slowdown in development of the capacity of integrated circuits for CPUs and even GPUs which are currently reaching physical limits. This incentive has not yet yielded a fully fault tolerant and error corrected quantum computing device to date.

Most experiments in quantum computing are therefore theoretical models executed on simulators or on small scale quantum hardware devices. These contain a very limited number of qubits with shallow circuit depths in terms of the number of quantum gates for processing the input quantum state and producing a coherent result. Experiments on physical quantum devices require repeated runs of the calculation firstly in order to determine the probability distribution of the result and secondly to compensate for the presence of noise or other environmental interference effects (Wittek; 2014).

1.2 Quantum Machine Learning

The potential advantage to the data analytics practitioner for using QML is the capability of representing huge amounts of information on a relatively small number of qubits compared to the capacity of a comparable amount of bits in a conventional computer. In addition, processing those quantum states representing the data inputs can be done in an inherently parallel fashion. The processing results can be presented as states collapsing to the most probabilistic outcomes. These quantum effects, notably superposition, entanglement and interference, lend themselves to allowing a highly significant speedup in processing large datasets over conventional computing devices.

1.3 Research Question

At this stage of the evolution of quantum computing it is desirable to consider the following research questions:

> **RQ:** How can the current state of the art of QML techniques be applied to practical data analysis problems via the availability of quantum computing cloud platforms?
>
> **Sub-RQ1:** How can fundamental distance based classification machine learning algorithms such as kNN and kernel methods such as SVM be applied in the context of quantum computers?
>
> **Sub-RQ2:** How can QML classifiers be applied to large cardinality and high dimension datasets?

1.4 Report Structure

The remainder of this document is structured as follows: Section 2 summarises the current state of the art with regard to quantum computing and QML. Through a review of notable literature, the validity of QML for the purpose of data analytics in the near term is established. Focus is given to establishing a foundation for the viability of distance based classification techniques such as kNN and SVM with kernels generated on a quantum processing device. Section 3 introduces the methodology by which the research into distance based classification problems using quantum devices is performed and how the results are evaluated. Section 4 describes the design of typical distance based QML algorithms for classification. Section 5 describes the implementation of the classifiers in the form of Juypter notebooks using IBM Qiskit quantum machine libraries. Section 6 evaluates the results of running these QML models to process the classic 4-dimensional Iris dataset in order to illustrate the principles yet also the current limitations of QML. Section 7 concludes the document and points to future directions for work in this area.

2 Related Work

The evidence from current literature and the state of the art supports the notion that the development of QML algorithms on quantum computers with quantum processors has been well established. However, the current implementations of physical quantum processors in the NISQ era limits their usability and the production of noise free or error corrected processors remains a significant challenge to the general availability of the first true general, error corrected quantum computer. Some solace is gained in the knowledge that conventional computers went through a similar evolution whereby great improvements were made in operating systems, computer languages and software development frameworks in tandem with the hardware being improved to the standards enjoyed today. It is anticipated that QML algorithms will continue to evolve despite the lack of suitable quantum computing hardware to actually allow them to run on any meaningful data. A theme of this project is to understand how far away the current state of the art is from quantum computers being able to perform typical useful data analysis.

2.1 Emergence of QML

The process of programming a quantum computer involves the construction of a circuit comprising one or more qubits and encoding the input data into an initial quantum state for the circuit. That quantum state is then transformed through a series of operations known as gates. The output of the circuit is measured which collapses the quantum state back to conventional bit values. The programmed circuit needs to run several times in order to generate a probability distribution for the results (Metawei et al.; 2020).

Despite the lack of current availability of fault tolerant quantum computing devices capable of processing large datasets, many algorithms and machine learning models have been proposed. These have been generally tested with simulators and at small scale with the current crop of limited physical quantum devices (Ramezani et al.; 2020).

Quantum computing is based on probabilistic programming in that it uses linear algebra and specifically matrix operations to describe quantum state transformations and their time evolution as per Schrodinger's equation. The quantum states can contain complex number values which allow the phenomenon of interference whereby the probability

of certain outcomes is reinforced by positive interference and other outcomes are reduced or cancelled by negative interference of probabilities (Nielsen and Chuang; 2002).

The advantage of quantum computing comes from the fact that a conventional computer utilises digital encoding, also known as binary encoding, in order to represent data values in terms of a string of 1s and 0s. Quantum computers can make use of amplitude encoding whereby the magnitude of each feature value of the n-dimensional vector representing the data point in Hilbert space can be applied. In Hilbert space a vector with dimension N needs n = log2(N) qubits to be encoded. Quantum superposition allows the storing of 2^n numbers on n qubits, thus reducing the number of bits required exponentially compared to conventional computers (Lloyd et al.; 2013). The current limitations and challenges regarding the capability of physical quantum devices have not hindered the development of algorithms and theoretical processing models of quantum computation. Interesting reusable template circuits have been developed along with algorithms and models that could potentially achieve a performance advantage over conventional computing platforms (Ciliberto et al.; 2018).

The role of machine learning is the derivation of a model from data in order to make predictions on unseen data based on the underlying probability distribution of the dataset. It is quite common to use heuristic methods to select algorithmic hyperparameters to minimise a loss function in order to improve the fit of the model. Quantum computing generalises the probability distribution of the data based on states represented by n-dimensional vectors in Hilbert space. Hilbert space itself is a generalization of the concept of Euclidean space whereby lengths and angles and operations such as inner products hold true based on the laws of linear algebra. With the quantum properties of superposition and entanglement of states, this allows potentially faster and novel solutions to data processing problems despite exponentially increasing state space with increases in data dimensionality which is a clear advantage over conventional computing (Aïmeur et al.; 2013).

The current issues surrounding the development of physical quantum computing hardware relate to its sensitivity to environmental interference. Quantum hardware requires the superconducting circuits to be maintained at near zero Kelvin temperatures. Its stability in terms of holding a steady state to allow processing declines with increases in scale in terms of number of qubits and the depth of the circuit. This degradation phenomenon is known as quantum decoherence. In light of this susceptibility to environmental interference, not only have quantum computing models been developed for pure quantum systems, they have also been inspired the hybridisation of conventional computing and quantum computing elements. In this case the quantum device provides acceleration of certain operation suited to the quantum computing model. One example of this would be quantum variational circuits inspired by deep learning networks where the loss function and hyperparameter tuning of a quantum model is performed by a conventional computing platform. In this case the use of a calculated loss function and training of the hyperparameters can mitigate the effects of environmental interference noise and quantum decoherence effects. Quantum computing has also inspired improvements and new ways of thinking in the design of algorithms and models for conventional computing platforms (Schuld et al.; 2020; Benedetti et al.; 2019).

2.2 QML for Supervised and Unsupervised Learning

One fundamental technique for classification of data points is to represent them as n-dimensional vectors in a n-dimensional space and calculate their similarity based on geometrical distance with respect to existing labelled or unlabelled data points in that space. The idea is that the closer the new test data point is to existing data points, the more similar they are in terms of classification (Schuld; 2018). SVM build on this idea of distance between data points and identifies a limited number of data points (known as support vectors) from the dataset whereby a hyperplane may be defined with a maximum margin between these support vectors to segregate the data points for classification purposes. The kernel trick inherent in these techniques allows the replacement of the kernel defining with distance measurement with alternates. It is possible to produce a quantum kernel calculating distance between training and test set data points in Hilbert space as a preliminary computation on a quantum computer yet use the quantum computed kernel on a conventional computer using SVM to classify future unseen data (Havlíček et al.; 2019).

There are various methods for calculating the distance between data points represented as vectors in n-dimensional space. Fidelity or cosine similarity are particularly suited to quantum computing. Quantum systems operate in Hilbert space and similarity between data points can be represented as being relative to the angle between the vectors. The smaller the angle between the vectors, the closer the data points are in terms of similarity. This can be represented by a cosine whose value tends to 1 as the angle tends to 0. Another method is that an orthogonal projection of one vector representing the data point onto the other vector calculates a measure of similarity. All vectors in Hilbert space are normalised so as the magnitude of the projection goes to value one, it indicates that the points are increasingly similar. This projection can be taken simply from the inner product of the two vectors which is a basic property of the Hilbert space and very suitable for calculation using a quantum computer. In either case the distance between the data points can be defined as (1 - fidelity) (Schuld and Killoran; 2019).

The algorithm for calculating kNN classification is to calculate the distance between the test point and all the data points for which there is an existing classification. The top k points are selected which have the lowest distance from the test point. A simple majority vote is performed amongst these k points to determine which classification is applied to the test point (Afham et al.; 2020). Note that as the number of training points (i.e. existing data points with classifications) increases, the effort in classifying the test point increases. Note also that as the number of dimensions of the vector representing the data points increases, the more effort is involved in calculating the distance. Indeed the effort required in calculating the distances from a test data point of D dimensions to N labelled data points and derive the kNN is O(D*N*k). In comparison the effort involved in calculating the kNN in a quantum system is O(log(D*N*k)) (Fastovets et al.; 2019; Schuld et al.; 2015).

Lloyd et al. (2013) describes a distance calculation based on cluster centroids thus the kNN algorithm is transformed into a k-means method, whereby each classification is defined by the centroid (or mean) of all data labelled with that class. Rather than calculating the distance to each data point from the test point, the distance can be calculated to the centroid. However, as new classified data is added the centroid must be recalculated. Shrivastava et al. (2020) discusses the quantum equivalent k-means and k-medians algorithms in detail with an analysis of the complexity compared with its conventional

equivalent and an indication of the speedup possible with quantum processing.

2.3 Recent Developments in Cloud Hosted Quantum Computers

IBM released their multi-tier development roadmap for quantum computing in September 2020. It is based around their Qiskit (pronounced kiss-kit) software development kit and IBM Quantum cloud based runtime environment comprising simulators and actual quantum processors. This roadmap covered the evolution of the IBM quantum hardware ecosystem towards 1000 qubit machine availability with support by developers of kernels, algorithms and models to provide a rich machine learning environment. The point was stated that although it took many decades for logic gate circuits to evolve to the computers of today, the hope is that this process can be accelerated in the case of quantum computing (*IBM's roadmap for building an open quantum software ecosystem*; 2021).

3 Methodology

In order to pursue the objectives of this project, it is necessary to choose a dataset suitable for the purpose of analysis with the current limitations of quantum computing in mind. The Iris data set from the scikit-learn Python package is used for this project. The Iris data set has 150 observations with a dimensionality of 4 and contains three classification classes. It consists of length and width measurements of petals and sepals from three different flowers from the Iris family: Iris Virginica, Iris Setosa and Iris Versicolour. The flowers are classified based on the sepal and petal measurements. Although the dataset is very small and of low dimension, it is an excellent starting point to demonstrate the concepts of QML. Figure 1 shows the scatter plots of the different features of the data set in combinations of two dimensions.

The methodology for researching the answer to the research question and sub-questions involves the following steps:

1 Use IBM Qiskit software development environment to produce quantum computing equivalents of distance based classification algorithms such as kNN and SVM.

2 Preprocess the Iris dataset into 120 training data points and 30 test data points, reduce their dimensionality, normalize and scale.

3 Train the classification models on the training set and record CPU and elapsed (wall clock) time for completion on an actual quantum processor hosted by IBM Quantum, an ideal noise-free simulator and a simulator that incorporates a noise model.

4 Run the classification models on the test set and record CPU and elapsed (wall clock) time for completion on an actual quantum processor hosted by IBM Quantum, an ideal noise-free simulator and a simulator that incorporates a noise model.

5 Calculate a classification accuracy score for all the classification models for the three quantum runtimes and compare with their counterparts running on a conventional computer with scikit-learn.

6 Evaluate the results from the perspective of technical feasibility of application to larger datasets in terms of greater cardinality and dimensions.

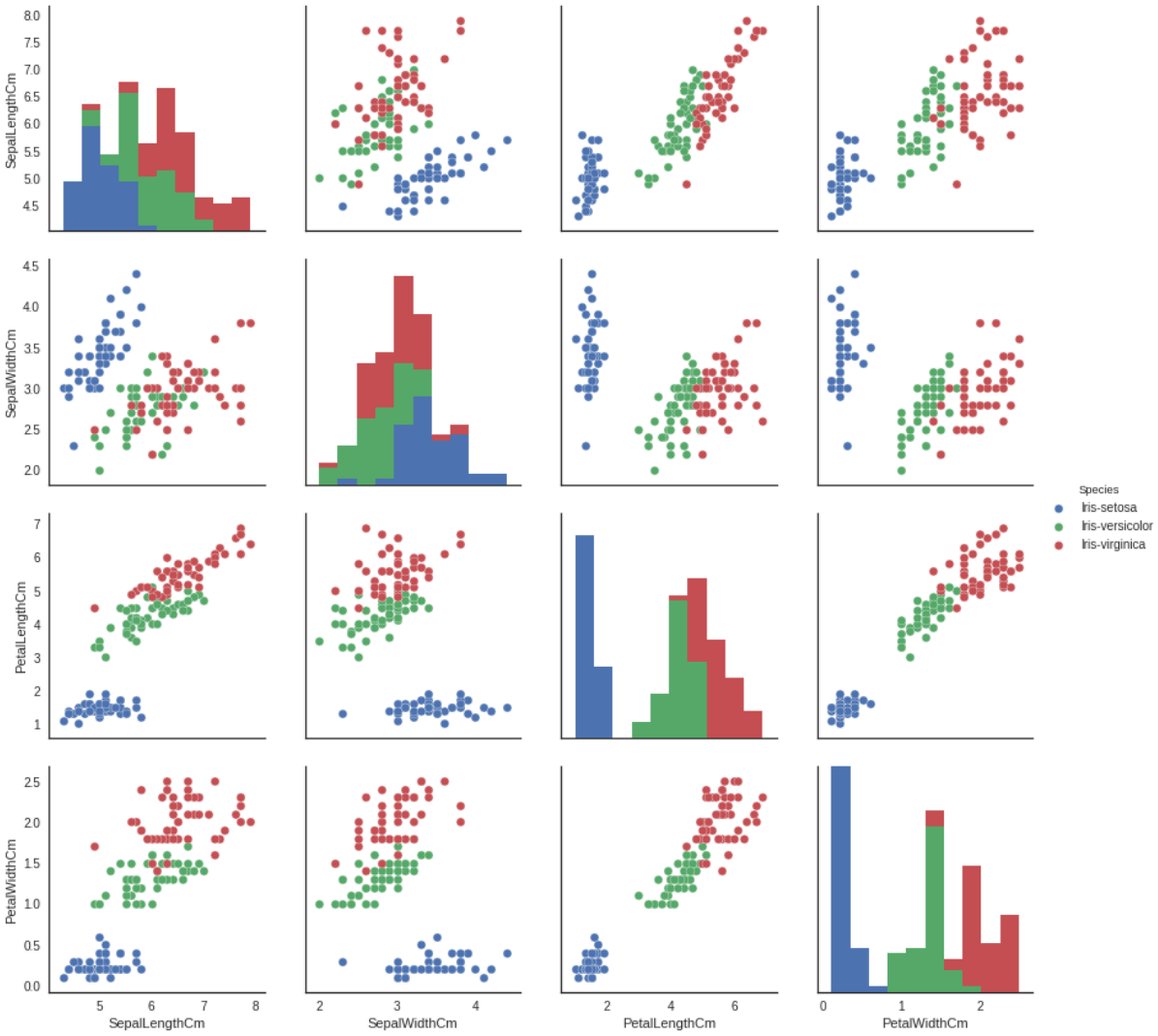

Figure 1: Iris dataset scatter plots produced by Pandas scatter_matrix function

A specific example of the methodology applied to analysing the Iris dataset using QML with SVM kernel methods is shown in Figure 2.

4 Design Specification

Quantum circuits consist of sets of qubits whose quantum states are initialised and then transformed through a series of operations via quantum gates (analogous to logic gates in digital circuits) and the eventual quantum state is then measured. This collapses the quantum state of the qubit to a 0 or 1 reading. By stochastic method and repeating the circuit and measuring many times, a probability distribution of the qubits final quantum state may be achieved leading to interpretation of results.

4.1 Quantum Encoding

As a prerequisite to applying the quantum classification algorithms, the dataset must be encoded into quantum states within the Hilbert space. This process in itself can be very

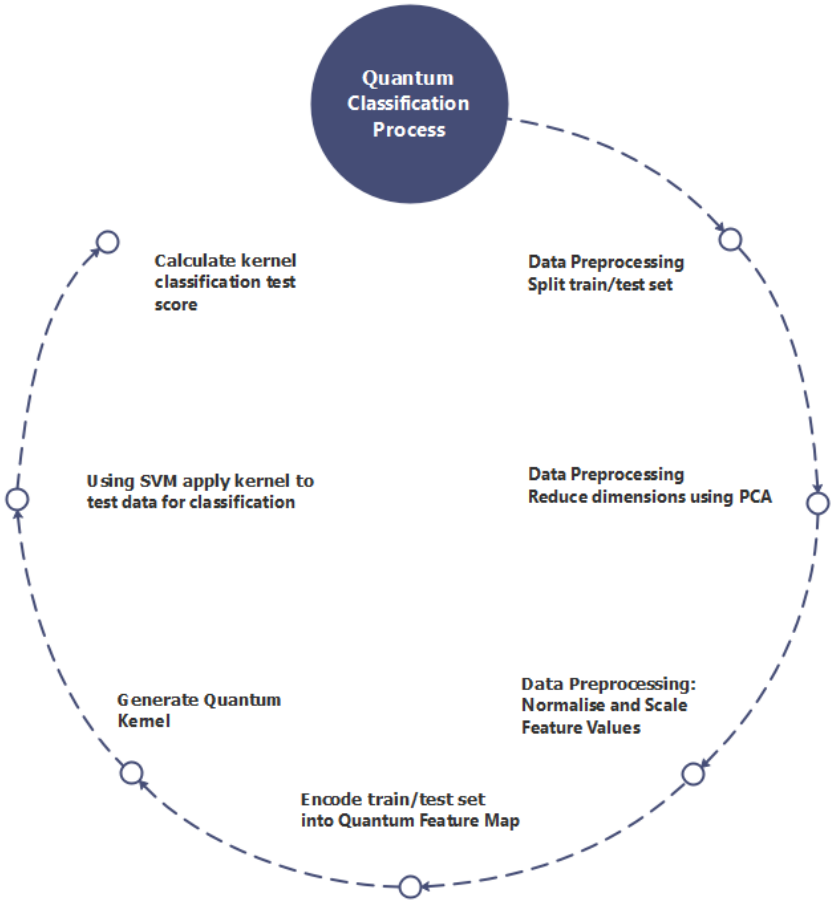

Figure 2: Quantum Classification Process using Kernel methods

compute resource intensive and can negate any speedups that the quantum algorithm may achieve. There are several possible encoding mechanisms. The simplest method is basis encoding whereby the binary string representation of the data value is encoded to a corresponding number of qubits e.g. the value 9 is represented as the binary string 1001 which can then be represented as the 4 qubit state $|1001\rangle$. Given that current quantum devices have limited number of qubits, this is not an efficient encoding method.

An alternative to basis encoding is amplitude encoding (also known as analog encoding). In this case the data dimension values are used to adjust the probability amplitudes of the quantum basis states. In this way each quantum state can represent all dimensional values of a data observation. The following worked example illustrates the principle.

The Iris dataset contains 150 observations (50 in each of the three classes) with the features comprising sepal length, sepal width, petal length and petal width. The feature values are all defined in units of centimeters. Finally the three classes are defined as target labels comprising Setosa, Versicolour and Virginica. A single observation from the dataset is shown in Equation 1 :

$$\text{data observation} = 5.1, 3.5, 1.4, 0.2, \text{Iris-setosa} \tag{1}$$

This data observation could be represented as a four dimensional vector as shown in

Equation 2:

$$\text{feature vector} = \begin{bmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{bmatrix} \tag{2}$$

In quantum computing, the quantum bit or qubit is the fundamental unit of information. A qubit is a two state system where the 0 state is represented as $|0\rangle$ which is a "ket" in the Dirac notation and the 1 state is represented as $|1\rangle$. A qubit represents a two dimensional Hilbert space where the basis vectors of the space are $|0\rangle$ and $|1\rangle$. All quantum states including superpositions of $|0\rangle$ and $|1\rangle$ can be defined as linear combinations of $|0\rangle$ and $|1\rangle$ e.g.

$$\phi = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3}$$

where ϕ represents the general quantum state of a qubit, the coefficients α and β are the probability amplitudes of the state being measured as $|0\rangle$ or $|1\rangle$ respectively. One method of representing data observation feature vectors as quantum states is by putting the qubit state into superposition and embedding the feature values of the data observation into the probability amplitudes of the basis states of the qubit. This is known as amplitude encoding. The four dimensional data observation from the Iris dataset above can be represented as a two qubit system with a quantum state in Dirac notation as:

$$\phi = 5.1\,|00\rangle + 3.5\,|01\rangle + 1.4\,|10\rangle + 0.2\,|11\rangle \tag{4}$$

This vector can then be normalized to unit length in Hilbert space as:

$$\frac{1}{\sqrt{5.1^2 + 3.5^2 + 1.4^2 + 0.2^2}}(5.1\,|00\rangle + 3.5\,|01\rangle + 1.4\,|10\rangle + 0.2\,|11\rangle) \tag{5}$$

Higher order data i.e. data with more dimensions/features can be represented by multi-qubit array known as quantum registers.

4.2 Quantum Distance Measurement

The preprocessing of the data involves representing the data points as quantum states. The Iris dataset consists of four features or dimensions and thus are represented by 4-dimensional vectors in Hilbert space. All the classification algorithms under consideration e.g. kNN and SVM make their classifications based on distance between data points. The underlying assumption is that feature vectors containing the data observations of similarly classed observations will lie close together within the feature space. In the case of quantum computing this feature space is known as Hilbert space.

There are several definitions of distance that can be applied to judge similarity of vectors in Hilbert space. The vectors are a representation of a quantum state whereby the data observation feature values are encoded into the vector dimensions. A property of Hilbert space is that inner products between vectors are easily calculable. Cosine similarity is a measure of similarity between vectors. It is expressed as the cosine of the angle between the vectors in n-dimensional space. It can be calculated by taking the inner product of the two vectors and then dividing by the product of their magnitudes in order to normalise the result.

$$\text{cosine similarity} = \frac{\langle A, B\rangle}{|A| \cdot |B|} \tag{6}$$

$$\text{cosine distance} = 1 - \text{cosine similarity} \tag{7}$$

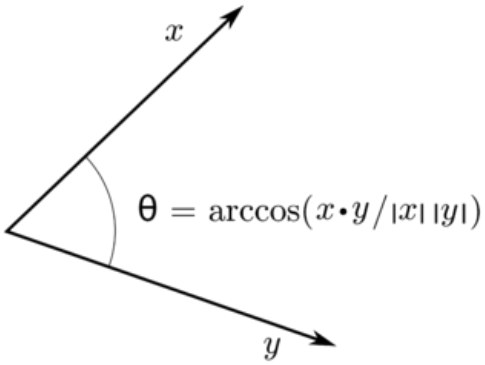

Figure 3: Angle between vectors defined using an inner product (Afham et al.; 2020)

From Figure 3 it can be seen that vectors that are close have a relatively small angle between them. The cosine of the angle as it approaches zero (hence the vectors are closer) tends to the value one and likewise if the vectors are orthogonal i.e. the angle between them is 90 degrees then the cosine value is value zero. Note that quantum states in Hilbert space are by convention normalised to have length one. Note also that the vectors representing the quantum states can have complex values therefore the convention is to use the general term "inner product" rather than "dot product" to represent the operation. For pure quantum states i.e. those that can be represented as vectors in Hilbert space, the fidelity is a measure of the "closeness" and it is defined as the squared overlap between them

$$\text{fidelity} = |\langle\alpha|\beta\rangle|^2 \tag{8}$$

It can be seen that the fidelity is simply the square of the cosine similarity between two unit length vectors representing quantum states in Hilbert space. Another definition of distance is:

$$\text{distance} = 1 - \text{fidelity} \tag{9}$$

4.3 SWAP Test Circuit

The fidelity between quantum state inputs can be measured using a SWAP Test quantum circuit (Fredkin and Toffoli; 1982). This consists of the quantum circuit as shown in Figure 4. The SWAP Test quantum circuit consists of a control qubit initialised to the $|0\rangle$ state and multi-qubit registers representing the two quantum states ϕ and ψ for which the distance is to be measured. The control qubit is put through a Hadamard gate that puts the initial $|0\rangle$ state into a superposition:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{10}$$

The overall state of the quantum system at this point is:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,|\phi\rangle\,|\psi\rangle \tag{11}$$

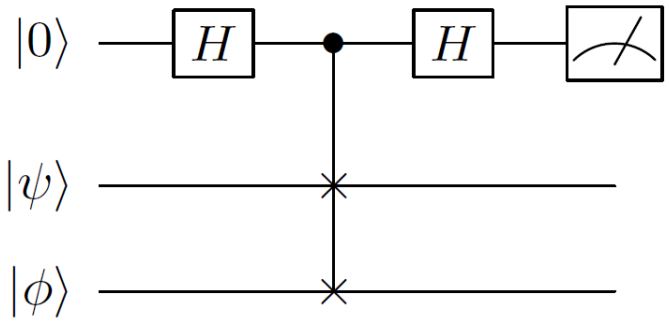

Figure 4: SWAP Test circuit (Afham et al.; 2020)

This state then acts as the control on a controlled SWAP (or Fredkin gate) which operates as follows (Fredkin and Toffoli; 1982):

$$|0\rangle |a\rangle |b\rangle -> |0\rangle |a\rangle |b\rangle \tag{12}$$

$$|1\rangle |a\rangle |b\rangle -> |0\rangle |b\rangle |a\rangle \tag{13}$$

The overall quantum state at this point is (Afham et al.; 2020):

$$\frac{1}{\sqrt{2}}(|0\rangle |\phi\rangle |\psi\rangle + |1\rangle |\psi\rangle |\phi\rangle) \tag{14}$$

The control qubit is put through another Hadamard gate before being measured and collapsing its quantum state to either a 0 or 1 conventional bit measurement. The probability of measuring 0 and 1 on the control qubit is:

$$P(0) = \frac{1}{2} + \frac{1}{2}| \langle\phi|\psi\rangle |^2 \tag{15}$$

$$P(1) = \frac{1}{2} - \frac{1}{2}| \langle\phi|\psi\rangle |^2 \tag{16}$$

$$P(0) - P(1) = | \langle\phi|\psi\rangle |^2 \tag{17}$$

By running the circuit many times, it is possible to estimate P(0) and P(1) and thus calculate the fidelity by taking their difference(Afham et al.; 2020).

4.4 Quantum Classification with Interference

An extension to the SWAP Test circuit involves using a construct called an Oracle which can bring ALL the quantum states representing ALL the data points in the data set into superposition. This allows the fidelity of the test data point to be measured with respect to the entire dataset instead of one vector / quantum state at a time (Buhrman et al.; 2001).

In addition, quantum interference can be used to reinforce the probability of obtaining certain desired results and and reduce or remove less probable results via constructive or destructive interference. This effect can be used as a way of deriving the kNN from

the total set of distances calculated via the Oracle and superposition of all the quantum states representing the entire dataset in one operation. (Brassard et al.; 1998) (Deutsch; 1985).

In order to represent all the observations in the data set simultaneously in superposition as quantum states, an Oracle W is required as shown in Figure 5. The function performed by the Oracle W is:

$$W\left|i\right\rangle\left|0\right\rangle = \left|i\right\rangle\left|\phi_i\right\rangle \tag{18}$$

where ϕ_i is the i-th observation from the dataset.

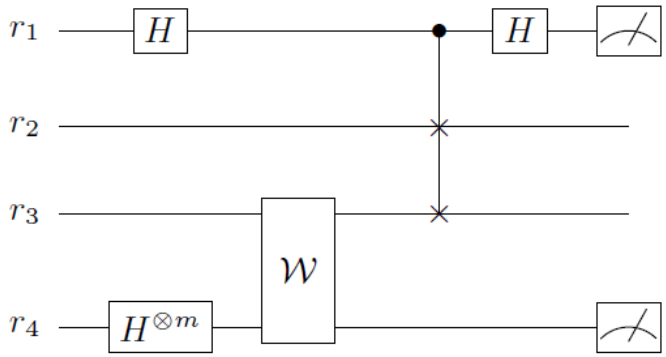

Figure 5: SWAP Test circuit including Oracle W (Afham et al.; 2020)

This circuit allows the measurement of the fidelity of the full data set in superposition against the test data observation in one operation rather than N operations i.e. one vector (representing a classified data observation) at a time (Afham et al.; 2020).

In the Figure 5, r_1 represents the control qubit which is a single qubit, r_2 represents the quantum register (consisting of several qubits) initialised to the amplitude encoded state of the test data observation, r_3 represents the quantum register containing the superposition of states representing the classified dataset and r_4 represents the computational basis. r_2 and r_3 consists of $n = log_2(N)$ qubits where N represents the dimensionality of the vector representing the quantum state of the data.

In order to measure the output of the circuit, a measurement must first be taken of r_1 which collapses the quantum state to 0 or 1. Additionally a measurement must be taken of the r_4 register which collapses the quantum state to a value i which represents the i-th observation from the dataset. $P_i(0)$ is the probability of getting 0 measurement in the control qubit for observation i whereas $P_i(1)$ is the probability of getting 1 measurement in the control qubit for observation i. Over many measurements Q_i is directly proportional to the fidelity:

$$Q_i = P_i(0) - P_i(1) \tag{19}$$

This quantity Q_i can be determined stochastically over many iterations of the experiment. Due to the quantum parallelism effect and its associated probabilistic nature, only those states representing data observations which have high fidelity with respect to the test data state have high probability of getting detected upon measurement with a limited number of iterations. Moreover no sorting is required to determine the nearest data points to the test observation (Afham et al.; 2020).

4.5 Quantum Kernel Estimation

An alternative classification mechanism is examined whereby SVM classifiers are run using kernels generated on quantum computers. Four different feature maps are examined which encode the dataset into a higher dimensional Hilbert space and then apply a quantum kernel estimation circuit to deduce a decision boundary hyperplane to segregate the data for classification. This involves quite a complex mathematical process which is described in Havlíček et al. (2019).

5 Implementation

IBM Qiskit is an open source software development kit (SDK) for developing applications for quantum computers. Qiskit allows the construction of quantum computing circuits with gates and measurements. It also offers some higher level modules that encapsulate domain specific functions in the areas of Machine Learning, Finance, Optimizations and modelling processes for Physics, Chemistry and Biology.

The Qiskit SDK also incorporates an interface to the physical quantum devices offered on the cloud based IBM Quantum platform. This platform offers a free service for visually composing quantum circuits called IBM Quantum Composer along with a Python notebook environment called IBM Quantum Lab. IBM Quantum Services provides a set of quantum computers from 1 up to 65 qubits at the time of writing. Not all of these systems are publicly available. The larger systems are reserved for use by IBM and their academic research partners. The lower specification open systems can have quite a backlog of jobs waiting to run and a potential wait time of several hours before results are available. This is very reminiscent of the batch job submission mode of computing on time shared mainframe computers.

The physical quantum devices do incorporate all the real world imperfections associated with non-error correcting and environmentally influenced quantum processors. However, a range of cloud based simulator devices is also offered along with local simulator devices that can be created in the IBM Quantum Lab notebook environment for experimental purposes. This latter mode of processing is ideal for exploring the data analytics implications of constructing QML models. This includes the preprocessing step of encoding datasets as quantum states and the post processing step of measuring the computed outputs. The results from the simulators can then be compared with those obtained from the actual quantum processors in terms of the accuracy of classification test metric which may differ due to introduced environmental noise and decoherence effects in the physical circuit. The runtimes between the simulators and the shared resources of the IBM Quantum physical devices are also of interest to the data analytics practitioner.

The process of implementing the design of the quantum circuits in order to classify the Iris dataset is as follows:

1 Use IBM Qiskit software development environment running within a Juypter notebook on a local Anaconda environment or on IBM Quantum cloud environment.

2 Design a quantum circuit that represents the problem under consideration, in this case classification of the Iris dataset.

3 Compile the circuit for a specific quantum service i.e. quantum hardware processor or simulator.

4 Encode the training and test data points from the dataset into quantum states to be processed through the circuit. This process is known as quantum embedding.

5 Run the compiled circuit on the specified quantum system or simulator bearing in mind that the results are stochastic and will require numerous runs to create a probability distribution of the results.

6 Compute summary statistics and visualize the results of the experiments. Compare the results of the experiments with their counterparts in conventional machine learning via the Python scikit-learn machine learning library.

5.1 Implementation of kNN

The kNN algorithm has no inbuilt support in IBM Qiskit and must be implemented from first principles using quantum circuitry, specifically the SWAP Test circuit with an Oracle implementation to put the training dataset into superposition. The kNN algorithm consists of the following broad steps:

1 Calculate the "distance" of the new test data observation from all classified points in the dataset

2 Choose the k points from the dataset that are closest in "distance" to the test data observation

3 By majority vote, assign the most common classification from the k Neighbours to the test data observation

Figure 6 shows an example of a k=3 neighborhood where circles and squares represent two different classes. The star represents a test data observation. Based on k=3 the majority class of the neighbours is square and this forms the classification of the test data.

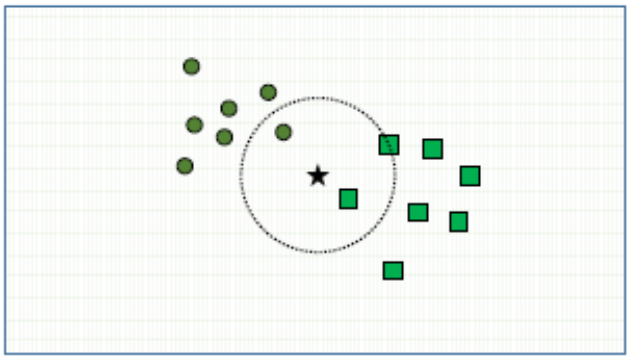

Figure 6: An example of kNN where k=3 (Afham et al.; 2020)

5.2 Implementation of SVM with Quantum Kernel Estimation

Havlíček et al. (2019), Schuld and Killoran (2019) and Schuld et al. (2020) describes the foundation of using kernel feature maps for classification. In this experiment, the **QuantumIris.ipynb** notebook was used to generate an SVM kernel using 4 different quantum feature map variants as provided by the IBM Qiskit SDK: ZFeatureMap without qubit entanglement, ZZFeatureMap with Linear entanglement between qubits, ZZFeatureMap with circular entanglement between qubits and PauliFeatureMap containing repetitions of the Pauli expansion circuit. These feature maps and the general technique of Quantum Kernel Estimation is described in Havlíček et al. (2019).

Feature maps project the dataset into a higher dimensional Hilbert space which allows for generating a hyperplane with a maximal margin between class defining boundary points known as support vectors. The hope in pursing this method is that by using quantum effects like qubit entanglement and parameterized quantum state rotation gates in the feature map to detect structure in the data in a manner reminiscent of the hidden layers of a neural network. This may lead to a machine learning model that could be hard to compute by conventional means. These models may perform better on certain datasets where convention classification methods have failed to perform to a high level.

Havlicek's work was implemented in the IBM Qiskit machine learning library and made generally available in Aug 2020. This was convenient as I am very much trying to treat this area from the perspective of the Data Analytics practitioner rather than getting deeply involved in the design of QML algorithms and implementation circuits from first principles. Leveraging the support of routines from a high-level machine learning library like Qiskit seems a more sensible approach with that objective in mind. In this manner I limited the scope of my contribution to identification of quantum kernel estimation for SVM as a promising area of practical QML application for the Data Analytics practitioner with an example implementation involving classification of a small but realistic dataset using quantum computing resources from the IBM Quantum environment.

Feature maps to project multi-dimensional feature values to higher dimensional spaces are a standard technique from classical machine learning. The idea is that it may not be possible to segregate the data points of, for example, a two-dimensional dataset for the purposes of classification or clustering. However, it may be possible to segregate them if the data points are projected into a 3 dimensional or higher space where the data points spread out to a certain degree in the higher dimension state space which allows a hyperplane to be produced to act as a decision boundary i.e., points to one side of the hyperplane are deemed to belong to class A and those to the other side are class B. This concept is frequently used in Support Vector Machine (SVM) classification.

A simple example of projecting the feature values of a two-dimensional data observation (x1, x2) into three dimensions is to synthesis the third feature value (x1, x2, x1 * x2) by simply multiplying the two feature values x1 and x2. The question of whether this trivial example of a feature map would allow the production of a segregating hyperplane in three dimensions would depend on the dataset and experimentation.

In the case of quantum kernels, the feature map is represented by a matrix. A matrix represents an operation or transformation of a state in a state space. A matrix operation can put a feature vector into a higher dimensional space by first zero-valuing the additional dimensions and then applying the matrix transformation. In the case of quantum feature estimation, the feature map matrix represents the inner products or fidelities of all the pairs of feature vectors representing the dataset. As demonstrated in the report the Swap

Test circuit can produce this inner product or fidelity measurement. However, Havlicek et al produced a more efficient way of producing this matrix through quantum circuits such as ZFeatureMap and ZZFeatureMap circuit implementations.

The second part of the operation is to calculate an estimated quantum kernel which segregates the data projected datapoints in Hilbert space using the maximum margin between support vectors.

6 Evaluation

6.1 Experimentation using kNN

Attempts were made to implement the kNN classification algorithm as described in Afham et al. (2020). Through the use of the SWAP Test circuit implemented in the **QiskitIntro.ipynb** Juypter notebook provided, it is possible to calculate the distance between a test data point and all other training data points in the Iris dataset. This is feasible due to the small number of observation in the dataset. The algorithm, especially as developed as a quantum circuit, does not scale well to larger number of observations or a greater number of features/dimensions.

Afham et al. (2020), Schuld et al. (2017) mentions the use of an Oracle to represent all the training data and allows a test data point distance to be measured with respect to all training data points simultaneously and the closest training points to be presented in one quantum circuit operation. The implementation of the Oracle also proved to be the defining point of this experiment. There is no clear guidance on how to implement this Oracle as a quantum circuit and Qiskit does not provide a specific Oracle implementation of this case. The implementation of such an Oracle would surely involve the use of Quantum Random Access Memory (QRAM) and using quantum interference to reduce the probabilities of all but the closest data points but this remains to date a theoretical construct. Schuld (2018) describes a QRAM as having the device attributes whereby data is loaded in parallel into a quantum register. The Oracle is queried using an index register and which loads the indexed data into the register. The index register can be put into a superposition of all states and thus the QRAM can load all data into the quantum register simultaneously. This quantum parallelism and production of a QRAM device is still the subject of active research.

Apart from some initial investigation of the construction of the SWAP Test circuit and experimentation with some sample data points, a full implementation of the kNN algorithm was not performed. This was due to the apparent lack of runtime performance and the inability to construct an Oracle circuit to allow, even in simulation, the superposition of all states from the training set. Focus shifted to the development of quantum generated kernels for SVM running on conventional computer hardware as described below.

6.2 Runtimes of SVM Quantum Kernel Classifiers

Table 1 records the CPU time and elapsed wall-clock time for generating the kernel feature maps for the four feature map circuit variants for an actual quantum processor and two simulators, one ideal and the other incorporating a noise model.

In this case the quantum circuit qubits are initialised into the $|0\rangle$ state and rotation gates applied using the feature vector values as angles by way of hyperparameters. The

Table 1: Quantum kernel generation runtimes by FeatureMap and Processor/Simulator

Quantum FeatureMap	**ibmqx Quantum Processor**	**StateVector Simulator (Ideal)**	**QASM Simulator (Noisy)**
ZFeatureMap	CPU times: user 1min, sys: 2.15 s, total: 1min 2s Wall time: 50min 1s	CPU times: user 901 ms, sys: 194 ms, total: 1.1 s Wall time: 2.2 s	CPU times: user 1min 11s, sys: 1.31 s, total: 1min 12s Wall time: 1min 58s
ZZFeatureMap (Linear Entanglement)	CPU times: user 2min 7s, sys: 2.18 s, total: 2min 9s Wall time: 51min 44s	CPU times: user 1.16 s, sys: 48.3 ms, total: 1.21 s Wall time: 2.45 s	CPU times: user 2min 19s, sys: 2.13 s, total: 2min 22s Wall time: 3min 50s
ZZFeatureMap (Circular Entanglement)	CPU times: user 1min 59s, sys: 2.07 s, total: 2min 1s Wall time: 56min 6s	CPU times: user 1.2 s, sys: 43.8 ms, total: 1.24 s Wall time: 2.36 s	CPU times: user 2min 53s, sys: 1.74 s, total: 2min 55s Wall time: 4min 16s
PauliFeatureMap	CPU times: user 2min 3s, sys: 2.11 s, total: 2min 5s Wall time: 53min 47s	CPU times: user 1.43 s, sys: 98.5 ms, total: 1.53 s Wall time: 2.94 s	CPU times: user 4min 15s, sys: 2.2 s, total: 4min 17s Wall time: 5min 55s

different feature map circuits allow repetitions of the circuit as a way to increase the parameterization of the classification process in a way similar to the parameterized layering of neural networks with configurable weights and biases as training hyperparameters. Just as in neural network design there is no absolute definition of good and bad design. To a large degree trial and error (and error measurement) is required until the circuit is performing adequately well for purpose.

6.3 Accuracy of SVM Quantum Kernel Classifiers

Table 2 records the classification accuracies achieved by the four feature map circuit variants for an actual quantum processor and two simulators, one ideal (StateVector) and the other incorporating a noise model (QASM).

Table 2: Kernel classification test scores by Feature Map and Processor/Simulator

Quantum FeatureMap	**ibmqx Quantum Processor**	**StateVector Simulator**	**QASM Simulator**
ZFeatureMap	0.97	0.97	0.97
ZZFeatureMap (Linear Entanglement)	0.9	0.93	0.97
ZZFeatureMap (Circular Entanglement)	0.93	0.93	0.93
PauliFeatureMap	0.97	0.97	0.93

6.4 Accuracy of SVM Conventional Kernel Classifiers

Table 3 records the classification accuracies achieved by conventional SVM kernels on the Iris dataset and may be compared with the classification accuracy scores achieved by the quantum generated kernel classifier as shown in Table 2.

Table 3: SVM classification test scores by Kernel

SVM Kernel	**Classification test score**
Linear	0.97
Poly	0.93
Rbf	0.97
Sigmoid	0.93

6.5 Discussion

In Table 2 I show the classification test scores produced by three different SVM kernels pre-calculated on an actual quantum computing device along with two quantum device simulators, one ideal / noise free (StateVector) and the other noisy (QasmSimulator). The metric refers to the number of correct predictions of class as a percentage for the test set of 30 observations chosen from the overall Iris dataset of 150 data observations. The quantum computer calculated kernels are produced using a FeatureMap which consists of a quantum circuit incorporating different ways of arranging the operator gates and using qubit state rotations to encode the training values as hyperparameters.

The FeatureMap circuits allow the repetition of the basic circuit constructs to allow more layers and more hyperparameters and greater "trainability" of the circuit to model the statistical properties of the dataset allowing better predictions on unseen data. For the sake of simplicity, only one layer was chosen for each of the FeatureMaps due to the additional quantum computing resources required in terms of circuit depth which brings a longer runtime and is more prone to noise.

Regardless of this, all experiments yielded high accuracy results with marginal differences as did the classical experiments using conventionally generated kernels. The point of this is that there is no credible advantage in using QML in this context. The limitations of the quantum computing devices available to me prevent the processing of a larger or more complex (in terms of numbers of features or structure within the data) dataset.

Given these constraints, it is easy for classical methods to perform just as well or indeed better than the quantum equivalent from a prediction accuracy point of view as demonstrated in Table 3. From a runtime performance point of view there is no contest. The current runtimes required for using physical quantum devices in the IBM Quantum environment with the job submission queue and shared runtime execution can be punishing where SVM kernel calculation using scikit-learn can be done in milliseconds. However, this may change as the data sets grow exponentially larger and more complex. At some point it may become computationally infeasible for a kernel to be generated using conventional and available computing resources and that is the opportunity for QML and quantum computing. Unfortunately, we are quite far from that point presently.

The runtime results for the SVM quantum kernel classifiers in Table 1 demonstrate that the quantum circuit running on the IBM Quantum hardware takes of the order of

1-2 minutes approximately of CPU time. Unfortunately due to the job queuing and time sharing execution mode for the public access quantum processors, the elapsed wall-clock time can be up to an hour. Considering the low dimensionality and cardinality of the dataset, this represents a serious performance deterrent to any data analytics practitioner. The simulator times are more reasonable and ideal for model development purposes.

The accuracy of classification for the quantum processor holds up well in comparison to both the simulators and the conventional SVM kernels. However this is largely down to the dataset and its attributes which are rather simplistic in this case. SVM with conventional kernels is able to generate both the kernel and classification results for the test set based on the training set in a matter of milliseconds for this dataset.

However the true advantage of generating a kernel on a quantum system is to develop a trained model that can potentially very accurately represent a hard to train classifier in comparison with conventional ML. QML can potentially achieve this through gate parameterization using the training data as hyperparameters and quantum effects such as entanglement between qubits which are not feasible to represent on a conventional computer. This may allow the quantum trained model to learn a feature set in a similar manner to hidden layers within an artificial neural network and possibly come up with a SVM decision boundary hyperplane that cannot be computed using a conventional computer.

There is scope for this mode of operation where the kernels can be generated offline on a quantum computer for datasets that are not classifiable using conventional methods and then the precompiled kernel can be brought back to a conventional computer for classification usage using an SVM in the normal manner. However this would require a dataset whose classification accuracy metric is low for conventional SVM kernels or of a scale large enough to be prohibitive in terms of conventional computing cost. Under these circumstances and with adequate access to large scale quantum computing resources in terms of number of qubits, circuit depth and time allocations, it might be possible to construct a quantum kernel with a significantly better classification accuracy. It is worth noting that no such dataset has to date been identified other than contrived artificial examples and there are still overbearing restrictions on stable quantum circuits in terms of number of qubits and circuit depth to allow this type of processing.

7 Conclusion and Future Work

To address the objectives of this research and assess the current application of QML to large scale data analytics problems, the small scale Iris dataset was processed to highlight any limitations of QML. kNN and quantum kernel estimation for use in conjunction with conventional SVM classifiers were considered. Three Python based Jupyter notebooks were provided to explore the dataset and introduce the IBM Quantum cloud computing environment and the Qiskit machine learning libraries. These notebooks are a very good programming entry point and template for anyone wanting to explore QML in the IBM Quantum environment with processing on multi-dimensional datasets.

The most significant outcome of this research is guiding the reader to the concept of quantum computation of feature maps and kernels to be subsequently used by SVM classifiers on conventional computers and demonstration of this concept by providing example notebook implementations. The advantage of this precomputed quantum kernel method is the use of gate parameterization and quantum effects like qubit entanglement

in the feature map to detect structure in the data that could be hard to detect by conventional ML. An efficient method of generating the kernel representing the decision boundary hyperplane can be done due to the inner product properties of Hilbert space in order to calculate distance between feature vectors. Although SVM performs very well with conventional computers in terms of accuracy and fast runtimes as shown by the use of scikit-learn, there may be datasets to which the quantum kernel estimation technique lends itself in terms of intractability of calculating a hyperplane using conventional ML. Unfortunately at this point in time no datasets possessing these characteristics have been identified by research apart from a contrived example by Havlíček et al. (2019).

For the data analytics practitioner, QML remains a theoretical curiosity in terms of practical applications to large scale data processing and analytics. Active research is ongoing to identify quantum circuits and models to aid useful practical applications in machine learning. The current issues with the lack of fault tolerant quantum hardware and the relatively scarce small scale systems available on a time shared basis illustrates an absence of clear advantage over conventional machine learning. Regardless of this, the ideas brought forward by QML allow re-examination of how one looks at data and how one implements machine learning processing algorithms and thus may well inspire improvements to the conventional machine learning techniques.

References

Afham, Basheer, A. and Goyal, S. K. (2020). Quantum k-nearest neighbor machine learning algorithm, *arXiv:2003.09187 [quant-ph]* . arXiv: 2003.09187.
URL: *http://arxiv.org/abs/2003.09187*

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

Benedetti, M., Lloyd, E., Sack, S. and Fiorentini, M. (2019). Parameterized quantum circuits as machine learning models, *Quantum Science and Technology* **4**(4): 043001. Publisher: IOP Publishing.
URL: *https://doi.org/10.1088/2058-9565/ab4eb5*

Benioff, P. (1980). The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines, *Journal of Statistical Physics* **22**(5): 563–591.
URL: *https://doi.org/10.1007/BF01011339*

Brassard, G., Chuang, I., Lloyd, S. and Monroe, C. (1998). Quantum computing, *Proceedings of the National Academy of Sciences* **95**(19): 11032–11033. Publisher: National Acad Sciences.

Buhrman, H., Cleve, R., Watrous, J. and De Wolf, R. (2001). Quantum fingerprinting, *Physical Review Letters* **87**(16): 167902. Publisher: APS.

Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S. and Wossnig, L. (2018). Quantum machine learning: a classical perspective, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **474**(2209): 20170551. Publisher: Royal Society.
URL: *https://royalsocietypublishing.org/doi/10.1098/rspa.2017.0551*

Deutsch, D. (1985). Quantum theory, the Church–Turing principle and the universal quantum computer, *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* **400**(1818): 97–117. Publisher: The Royal Society London.

Fastovets, D. V., Bogdanov, Y. I., Bantysh, B. I. and Lukichev, V. F. (2019). Machine learning methods in quantum computing theory, *International Conference on Micro- and Nano-Electronics 2018*, Vol. 11022, International Society for Optics and Photonics, p. 110222S.
URL: *https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11022/110222S/Machine-learning-methods-in-quantum-computing-theory/10.1117/12.2522427.short*

Feynman, R. P. (1981). Simulating physics with computers, 1981, *International Journal of Theoretical Physics* **21**(6/7).

Fredkin, E. and Toffoli, T. (1982). Conservative logic, *International Journal of theoretical physics* **21**(3): 219–253. Publisher: Springer.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search, *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M. and Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces, *Nature* **567**(7747): 209–212. Number: 7747 Publisher: Nature Publishing Group.
URL: *https://www.nature.com/articles/s41586-019-0980-2*

IBM's roadmap for building an open quantum software ecosystem (2021).
URL: *https://www.ibm.com/blogs/research/2021/02/quantum-development-roadmap/*

Lloyd, S., Mohseni, M. and Rebentrost, P. (2013). Quantum algorithms for supervised and unsupervised machine learning, *arXiv preprint arXiv:1307.0411* .

Metawei, M. A., Said, H., Taher, M., Eldeib, H. and Nassar, S. M. (2020). Survey on Hybrid Classical-Quantum Machine Learning Models, *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*, pp. 1–6.

Nielsen, M. A. and Chuang, I. (2002). *Quantum computation and quantum information*, American Association of Physics Teachers.

Ramezani, S. B., Sommers, A., Manchukonda, H. K., Rahimi, S. and Amirlatifi, A. (2020). Machine Learning Algorithms in Quantum Computing: A Survey, *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. ISSN: 2161-4407.

Schuld, M. (2018). *Supervised learning with quantum computers*, Springer.

Schuld, M., Bocharov, A., Svore, K. and Wiebe, N. (2020). Circuit-centric quantum classifiers, *Physical Review A* **101**(3): 032308. arXiv: 1804.00633.
URL: *http://arxiv.org/abs/1804.00633*

Schuld, M., Fingerhuth, M. and Petruccione, F. (2017). Implementing a distance-based classifier with a quantum interference circuit, *EPL (Europhysics Letters)* **119**(6): 60002. arXiv: 1703.10793.
URL: *http://arxiv.org/abs/1703.10793*

Schuld, M. and Killoran, N. (2019). Quantum Machine Learning in Feature Hilbert Spaces, *Physical Review Letters* **122**(4): 040504. Publisher: American Physical Society.
URL: *https://link.aps.org/doi/10.1103/PhysRevLett.122.040504*

Schuld, M., Sinayskiy, I. and Petruccione, F. (2015). An introduction to quantum machine learning, *Contemporary Physics* **56**(2): 172–185.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring, *Proceedings 35th annual symposium on foundations of computer science*, Ieee, pp. 124–134.

Shrivastava, P., Soni, K. and Rasool, A. (2020). Classical Equivalent Quantum Unsupervised Learning Algorithms, Vol. 167, pp. 1849–1860.

Wittek, P. (2014). *Quantum Machine Learning: What Quantum Computing Means to Data Mining*, Quantum Machine Learning: What Quantum Computing Means to Data Mining. Pages: 163.