QEML:
(Quantum Enhanced Machine Learning)

Using Quantum Computation to implement a novel K-nearest Neighbors Algorithm in a Quantum Feature Space

Siddharth Sharma

1. Abstract

Machine learning and quantum computing are two technologies which are causing a paradigm shift in the performance and behavior of certain algorithms, achieving previously unattainable results. Machine learning (kernel classification) has become ubiquitous as the forefront method for pattern recognition and has been shown to have numerous societal applications. While not yet fault-tolerant, Quantum computing is an entirely new method of computation due to its exploitation of quantum phenomena such as superposition and entanglement.

While current machine learning classifiers like the Support Vector Machine are seeing gradual improvements in performance, there are still severe limitations on the efficiency and scalability of such algorithms due to a limited feature space which makes the kernel functions computationally expensive to estimate. By integrating quantum circuits into traditional Machine Learning, we may solve this problem through the use of a quantum feature space, a technique which improves existing Machine Learning algorithms through the use of parallelization and the reduction of the storage space from exponential to linear. This research expands on this concept of the Hilbert space and applies it for classical machine learning by implementing the quantum-enhanced version of the K-nearest neighbors' algorithm (an existing lazy-learning deterministic classifier).

This paper first understands the mathematical intuition for the implementation of a quantum feature space and successfully simulates quantum properties and algorithms like *Fidelity* and *Grover's Algorithm* via the Qiskit python library and the IBM Quantum Experience platform. The primary experiment of this research is to build a noisy variational quantum circuit KNN (QKNN) which mimics the classification methods of a traditional K-nearest neighbors' classifier. The QKNN utilizes the distance metric of Hamming Distance and is able to outperform the existing KNN on a 10-dimensional Breast Cancer dataset.

Manuscript begins below

2. Introduction to Quantum Computing

Quantum Computing is a new paradigm of algorithmic study which extends quantum mechanical phenomena to the world of traditional computing. In 1982, Richard Feynman proposed an initial quantum computer, which would have the capacity to facilitate traditional algorithms with quantum circuits [1]. To understand systems of electrons and to navigate the multiple independent probabilities of electron location based on quantum phenomena, Feynman envisioned the concept of a quantum computer; he believed that quantum computers could ideally simulate quantum behavior as it would have occurred in nature. The quantum systems which Feynman wished to simulate could not be modeled by even a massively parallel classical computer. For example, let us consider the probability calculations of multiple particle systems. If we have two electrons constrained to being at two points (A and B), then there are 4 possible probabilities of their location (both at A, one A – one B, one B – one A, both at B, etc.). For 3 electrons, there are 8 probabilities, for 10 electrons, there are 1,024 probabilities, and at 20 electrons, there are 1,048,576 probabilities. Therefore, it is easy to see that measurements get out of hand for traditional physical systems with millions of electrons. Thus, research into quantum computers began and long-term goals for the field of quantum computing have arisen.

2.1 - Qubits

To begin an analysis of quantum computing, we must designate the operators and fundamentals of quantum circuits: quantum information. Quantum information is data for quantum states which is utilized for the study of computational manipulation. This quantum information is processed in the form of "qubits". Qubits can be thought of as the computational analog for bits and are the fundamental units of information for quantum computing. The qubit is described as a vector in the 2-dimensional Hilbert space. A traditional bit has 2 states (0 and 1). The state for a qubit also has 2 ground states - $|0\rangle$ and $|1\rangle$. The states for a qubit are modeled below:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (1)$$

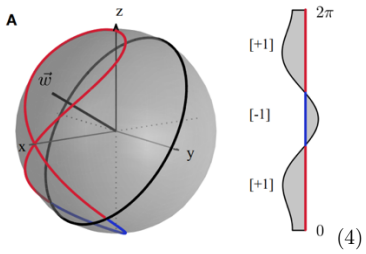
A qubit also has the ability to perform quantum superposition: it may hold multiple states at once. The superposition of a qubit can be represented as a probability function dependent upon the amplitudes of the qubit in its Hilbert space (a and b). A superposition of a qubit $|\phi\rangle$ can be represented as the following:

$$|\phi\rangle = a|0\rangle + b|1\rangle \quad (2)$$

Thus, we may represent each qubit as a linear combination of $|0\rangle$ and $|1\rangle$. a and b represent the amplitude probabilities and are typically complex numbers. When qubits are measured, they will follow the Born Rule:

$$|\alpha|^2 + |\beta|^2 = 1. \quad (3)$$

To visually understand a qubit, we can understand it as a Bloch Sphere. This sphere shows the probabilities (a and b) as being the determinate factors for the qubit and provides 3 degrees of freedom (originally 4 degrees of freedom but 1 is eliminated by the normalization constraint via the Born Rule). The two-dimensional Bloch Sphere visualization and feature map is shown below:



(4)

2.2 - Quantum Registers

Quantum registers are simply considered to be systems of multiple qubits. It is the quantum analog of a classical computing register. All quantum calculations are performed by manipulating qubits within the register. The quantum register has a size dependent on the number of qubits in its system. The Hilbert space of a quantum register expands with an increased number of qubits. We may mathematically represent a 2-qubit quantum register via the following:

$$|\phi\rangle = |\phi 1\rangle \otimes |\phi 2\rangle = |\phi 1\rangle|\phi 2\rangle = |\phi 1 \phi 2\rangle \tag{5}$$

The above mathematical description captures the concept of qubits via tensor products. The tensor product or Kronecker product combines the two quantum states of a qubit. The difference between a classical register and a quantum register is seen in the content of each respective environment. A classical register will store N "flip-flops" while a quantum register will store N qubits. Quantum registers also hold the ability to store data in a state of superposition. The states that a quantum register may access are known as Hilbert spaces with 2^n states. The Hilbert space is what gives a quantum computer the power to apply superposition on multiple qubits.

2.3 - Quantum Gates

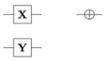
The most basic quantum circuits are known as quantum gates. They deal with small pre-ordained amounts of qubits. These quantum gates are analogous to classical logical gates and follow similar patterns of logic. Complex quantum circuits are built through stacking and utilization of these quantum gates, similar to how classical logic gates are the building blocks for traditional computational circuits. However, unlike classic logic gates, quantum gates are reversible. Each quantum gate can operate on the collective Hilbert space of several bits. Thus, quantum gates possess the ability to change or modify the state of a certain system or quantum register. Quantum gates are mathematically represented by unitary matrices where the number of qubits in the input and output must be equal. A quantum gate (U) is mathematically defined below:

$$UU^{\dagger} = I \tag{6}$$

A quantum state which may be manipulated by a quantum gate are typically shown as "kets" or a bra-ket ($|0\rangle$ and $|1\rangle$.). We can model the transformation applied by a quantum gate (U) on a state (ψ_1) as the following:

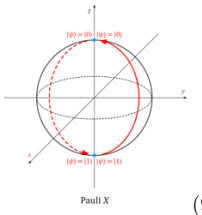
$$U|\psi_1\rangle = |\psi_2\rangle \tag{7}$$

This transformation multiplies the initial quantum state vector (ψ_1) by the unitary quantum gate operator (U) to achieve a new quantum state (ψ_2). Some common examples of gates are given below (Pauli-X, Pauli-Y, Pauli-Z, Hadamard, Phase, CNOT, CZ, SWAP, Toffoli):

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)			$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

(8)

These gates apply transformations on an initial system of qubits to map the basis state to a new value. For example, the Hadamard gate $(\boldsymbol{H})$ maps the basis state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ (rotation of π about bloch sphere $\hat{x}, \hat{z}$ axis). Similarly, other quantum gates like Pauli-X $(\boldsymbol{X})$, Pauli-Y $(\boldsymbol{Y})$, Pauli-Z $(\boldsymbol{R_\pi})$, and the Controlled Not $(\boldsymbol{CNOT})$ apply transformations that may be represented as unitary matrices. All of these transformations can also be visually recognized as shifts on the axis of a Bloch Sphere. For example, we can see the Pauli-X gate transformation mapped to the Bloch Sphere:



(9)

Another important quantum state which allows us to model quantum entanglement is the concept of the Bell States. The 4 Bell States (EPR pairs) represent 4 maximally entangled 2-qubits states:

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B)$$

$$|\psi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |1\rangle_B)$$

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B)$$

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |0\rangle_B) \tag{10}$$

Together, these 4 Bell States form the Bell Basis or a maximally entangled basis of the 4$^{\text{th}}$ dimensional Hilbert Space.

3. K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm is a simple example of supervised learning (input being mapped to output). KNN is a classification algorithm and classifies an input as a discrete or categorical output. Other examples of supervised learning classifiers include decision trees, naïve Bayes, and random forest models. Since KNN is an instance-based lazy learning algorithm, it is usually simple and intuitive to train. The fundamental assumption for the KNN algorithm is that datapoints with similar behavior exist in close proximity to each other. If this assumption does not remain fulfilled for a certain sample problem or dataset, the KNN model will not provide statistically significant results. In other words, KNN captures the concept of "closeness" or proximity. Thus, the KNN algorithm is successful in sample datasets where similar outputs cluster together in close proximity. The algorithm works by computing the distances from a specified example to other local examples. The algorithm is known as "K"-Nearest Neighbors since K nearby points are examined. K is a parameter which can be chosen and tuned via iterations of training a KNN algorithm. In most cases, K is an initial arbitrary point. Pseudocode for KNN is given below.

Algorithm 1 KNN algorithm

Input: $\mathbf{x}, S, d$
Output: class of $\mathbf{x}$
for $(\mathbf{x}', l') \in S$ **do**
  Compute the distance $\mathrm{d}(\mathbf{x}', \mathbf{x})$
end for
Sort the $|S|$ distances by increasing order
Count the number of occurrences of each class l_j among the k nearest neighbors
Assign to $\mathbf{x}$ the most frequent class

(11)

Since the KNN algorithm is based on a distance metric (typically Euclidean distance), we must evaluate this for two sample feature vectors ($\vec{v}, \vec{w}$). The goal is to identify the class of a new feature vector ($\vec{u}$) based on its distance to the two other feature vectors. The vector, $\vec{u}$, is assigned the class of the feature vector that it was closer to. The two parameters which affect the performance of KNN the most are the value of K and the number of dimensions (n). A smaller value of K could allow noise to have a major influence on the prediction (majority vote), whereas a large value of K makes the problem more computationally expensive. The common consensus is

to give K the value of $\sqrt{N}$ where N is the number of training data points. Thus, the value of K heavily affects performance of the classifier.

On the other hand, it is important to note that a KNN classifier suffers from the "*Curse of Dimensionality*". This is best illustrated in an example. If we have 1,000 training data points uniformly distributed across our vector space and our test data point is at the origin, the performance rapidly varies with dimension. In 1-D space, it takes about a distance of $5/1{,}000 = 0.005$ on average to get 5 nearest neighbors. In 2-dimensional space, it takes about a distance of $(0.005)^{1/2}$ and in n-dimensional space, it takes about a distance of $(0.005)^{1/n}$ in each direction as the training data points become sparsely distributed when the dimension of the space increases. The goal of a Quantum KNN (QKNN) approach would be to compute the distance via a quantum algorithm that is both efficient and simple to scale while minimizing both CPU time and cost of data point storage.

3.1 - Quantum-Enhanced Machine Learning

Quantum-Enhanced Machine Learning (QEML) involves taking supervised learning algorithms and making them more efficient through the use of quantum gates and orthogonal transformations to achieve more meaningful results. QEML strives to also offer solutions to the challenges of both data storage and slower execution. Before constructing a Quantum K-Nearest Neighbors Algorithm, it is necessary to explore the limitations of previous quantum algorithms while understanding the advantageous nature of quantum supervised learning. The primary properties of QEML can be expressed in the following aspects: improved representation space and acceleration of algorithm execution due to use of quantum heuristics. The storage space can be exponentially reduced through the use of *quantum superposition*. As discussed earlier, superposition is a fundamental quantum property which allows qubits to hold multiple states at once. For example, a N qubit state $|\phi_1,\ \phi_2, \dots, \phi_3\rangle$ can also be written as:

$$|\phi_1\phi_2\cdots\phi_n\rangle = \sum_{i=0}^{2^n-1} c_i\,|i\rangle \tag{12}$$

In a quantum computer, all binary numbers in the set $\{0, 1, \dots,\ 2^{n-1}\}$ exist in an N qubit quantum register. In a classic computer, only 1 binary number in the set $\{0, 1, \dots,\ 2^{n-1}\}$ can be stored in an N bit register. This example shows the benefit that quantum computers can provide in terms of reducing the size of the feature space (storage scale). Quantum computers achieve this through the aforementioned property of superposition which allows a N qubit register to hold exponentially more binary numbers without issues in size and scale. Thus, this property promotes the idea of an *enhanced feature space* in QEML classifiers.

The second benefit that quantum computing offers to machine learning is that of acceleration during execution of machine learning algorithms. This property of QEML is also known as

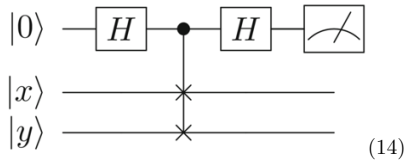
"*quantum parallelism*". "Quantum parallelism" similarly arises from the ability of a quantum register to exist in a state of superposition. Each component of superposition can be represented as a function. Each component of the superposition is evaluated by its respective function within the quantum register. Since the number of possible states is 2^n where n represents the number of qubits, it would take a classical computer an exponential number of operations to perform a task. However, a quantum computer can mitigate this issue through superposition and quantum parallelism, performing a similar task in one operation. The following example mathematically demonstrates this effect. Assuming a quantum environment, if a unitary operator U_f is transformed by the function $f(x)$, U_f must accomplish the task by inputting x from $|00 \ldots 0\rangle$ to $|11 \ldots 1\rangle$. For a classic computing environment, computing $f(x)$ from x inputs would take 2^n cycles or would require 2^n CPUs working in parallel. The relationship between $U_f, f(x)$ and the x inputs is outlined in the equation below:

$$U_f\left(\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}U_f|x\rangle = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|f(x)\rangle \tag{13}$$

In summary, quantum algorithms offer an elegant solution to problems faced in classical learning since quantum algorithms can store all training data points of exponentially large size as a linear size due to superposition; due to entanglement and interference, they can also compute distances near-simultaneously.

3.2 - Fidelity and Distance Computation

Even with the benefit of quantum parallelization, retrieving information from a quantum state with high performance is a difficult task. This is because during the process of measurement of a quantum state, there tends to be partial collapse of the quantum state and loss of previous information attained by quantum algorithms. To get the computing function $f(x)$ requires an innovative approach. Buhrmann's technique (quantum fingerprinting) is a simple solution to the problem of loss of quantum state and helps to calculate the distance of two vectors with high accuracy and fast execution. His process is detailed as the following: The auxiliary qubit $|0\rangle$ is first transformed through the left Hadamard gate to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$. The circuit then employs a ***SWAP*** gate to switch the two vectors $|x\rangle$ and $|y\rangle$ (i.e. $|xy\rangle \rightarrow |yx\rangle$). This process has also been illustrated in the diagram below:



(14)

The above quantum circuit is also known as a Controlled Swap ($\boldsymbol{C-SWAP}$) test and is essential to QEML since it provides for the property of *fidelity* (analogous to *cosine similarity* in classical machine learning). Fidelity measures the similarity of two quantum states ($|\phi\rangle$, $|\psi\rangle$). Fidelity can be represented as $|\langle x|y\rangle|$. If the two quantum states are orthogonal, the fidelity is 0; when the quantum states are identical, the fidelity is at its maximum which is 1. Another important note

about fidelity concerns the efficiency of the quantum solution; when the dimension of the quantum state vectors is higher, the quantum solution that uses fidelity becomes more efficient. For developing a quantum analog to the K-Nearest Neighbors algorithm, a quantum solution for the calculation of distance must be carried out. In classical machine learning, the distance between labeled examples in the vector space is typically calculated by the *Euclidean Distance* ($\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$). Using the earlier discussed trick of fidelity, we can represent the concept of Euclidean Distance in a quantum space: $\sqrt{2 - 2|\langle x|y\rangle|}$.

Since most supervised machine learning classifiers are based on concepts of class similarity and distance measurement, this trick (representation of distance via fidelity in a quantum space) is foundational to the performance of QEML algorithms. Some examples of algorithms and metrics used for computation of the distance in Quantum Machine Learning include Grover's Algorithm, the Hamming Distance, Lloyd's Algorithm, and Schor's Algorithm. These are all closely related to *quantum amplitude estimation* and attempt to measure distances between labeled examples in a quantum space. Another key breakthrough in quantum amplitude estimation is seen in the development of the quantum minimum search algorithm (similar to QESA).

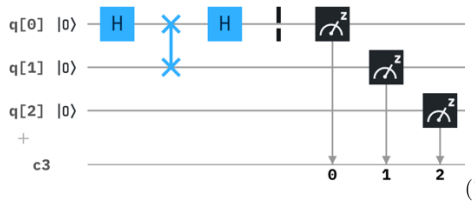
4. Initial Simulation of Fidelity with Qiskit

To build a Quantum K-Nearest Neighbors algorithm and to compare it to its machine learning counterpart, the distance will be measured via the concept of fidelity and the Controlled Swap gate. For measuring distance with a metric, Hamming Distance is optimal for discrete values whereas Euclidean distance is a strong indicator for continuous values. To conduct an initial simulation of fidelity using a quantum computer, we must use a python library known as *Qiskit*. Qiskit is a popular framework which allows users to simulate quantum circuits on a classical computer. To designate the earlier defined ($\boldsymbol{C-SWAP}$) gate, we must first build the circuit in the IBM Quantum Experience simulator. Since Qiskit is an open-source framework which provides access to building circuits on noisy quantum computers, this is not a challenging task.

Alongside its main function of facilitating quantum research in open areas in quantum computation, Qiskit also consists of four smaller libraries which allow developers to build full-stack quantum circuits: *Aqua*, *Aer*, *Terra*, and *Ignis*. Aer is designed to accelerate development through use of simulators and debuggers. Aqua is for building algorithms and larger quantum circuits. Terra is the library which propagates the code foundation of quantum circuits. Lastly, Ignis addresses issues with noise and interference in quantum circuits. To build a circuit to prove the concept of fidelity, we will use Aer and Terra in conjunction to both build the quantum circuit and to analyze results and produce the state-vectors.

4.1 – Results of Simulation with IBMQ

First, we build the variational quantum circuit and its respective quantum registers within the IBM quantum experience platform. To illustrate fidelity and to build the $\boldsymbol{C-SWAP}$ gate, we first instantiate our register with 3 qubits (q_0, q_1, q_2) and 3 bits (c_0, c_1, c_2). The complete quantum circuit is shown below:



(15)

To transfer our first qubit (q_0) into a state of superposition, we first apply a Hadamard (***H***) gate. The other two qubit registers (q_1, q_2) store the transformed state of the first qubit. We then apply a Fredkin gate (the gate in-between) to conduct the ***C − SWAP*** operation. The final signal passes through the second Hadamard gate and is then measured via the Z-measurement gate. If the initial states were orthogonal, the probability is 0.5, while if they were identical, the probability is then 1. To physically simulate the circuit, two backends were used. The *state-vector* and *BasicAer* (unitary) backends were used. The unitary result was a matrix of all possible state-vectors. If the state-vectors are roughly equally distributed (after noise) in a ***C − SWAP*** circuit, then we have reaffirmed the concept of fidelity. The results from the aforementioned circuit were promising when simulated with the backends in the IBMQ Experience. The unitary state-vectors are illustrated below:

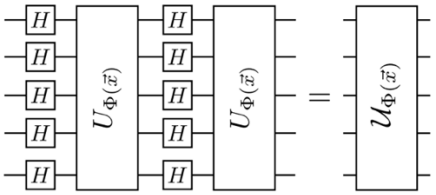
```
[ 0.5+0j, 0.5+0j, 0.5+0j, 0.5+0j,
0+0j, 0+0j, 0+0j, 0+0j ]
```
(16)

This unitary state-vector reaffirms the concept of fidelity since the value is usually 0.5 (both the qubits are orthogonal) or 0. This is very similar to cosine similarity and reflects the instrumental trick which was discussed earlier. Now that fidelity has been illustrated, it may be employed to calculate the distance. Since the main goal of this paper is to build a quantum variant of the K-nearest neighbors, this simulation is a promising start to calculating distance in the QKNN model.

4.2 – Analysis of Quantum SVM

Prior to designing the QKNN, it is important to understand the development of quantum kernel methods and the implementation of the enhanced quantum feature space, one of the central tenets of QEML. We will now analyze the quantum variant of the Support Vector Machine (QSVM), another popular kernel method for classification. The benefit of QSVM is that it provides for an enhanced feature space and thus, improved performance and storage capabilities. The classic SVM is known as a kernel method, a ubiquitous technique in pattern recognition. As described in the paper [1] by Havlicek et al., classical information is mapped to a quantum state. For background, we are given data from a training set T and a test set S of a subset. Both are assumed to be ground truth labeled by a map m: $T \cup S \rightarrow \{+1, -1\}$ unknown to the algorithm. The training algorithm only receives the labels of the training data T. The goal is to infer an approximate map on the test set S, where we are able to map the output from a given input to the set, $\{+1, -1\}$. This output should ideally closely correlate with the ground truth labeled map ($T \cup S \rightarrow \{+1, -1\}$). In a classical Support Vector Machine (SVM), the data is mapped non-linearly to a higher dimensional space where it is later separated into distinctive clusters via a defined segmentation, also known as a *hyperplane*. Havlicek et al. were able to generate a defining hyperplane in the

quantum feature space as expressed in the following: $\Phi: \vec{x} \epsilon \Omega \rightarrow |\langle\Phi(\vec{x})| \quad |\Phi(\vec{x})\rangle|$. Their mapping of a hyperplane in the quantum feature space is modeled below:



(17)

Havlicek et al.'s second goal was to implement a variational circuit which is normally tremendously difficult to operate on a classical computing system. To be able to design the QSVM, they initially defined the feature map on n-qubits generated by the following unitary ($\mathcal{U}_\Phi(\vec{x}) = U_{\Phi(\vec{x})}H^{\otimes n}U_{\Phi(\vec{x})}H^{\otimes n}$). They then generate the hyperplane through the aforementioned algorithm and equation below:

$$U_{\Phi(\vec{x})} = \exp\left(i \sum_{S\subseteq[n]} \phi_S(\vec{x}) \prod_{i\in S} Z_i\right) \tag{18}$$

Following the generation of the hyperplane to minimize the margin, they then optimize the algorithm for the parameters $(\vec{\theta}, b)$ in the noisy experimental setting via *Spall's SPSA* (Stochastic Gradient Descent Algorithm). Havlicek et al.'s final variational quantum classifier $W(\vec{\theta})$ is able to perform both binary and multiclass classification with high accuracy as exemplified in their experiments with a 5-qubit processor. The efficacy of their results (a successful variational quantum classifier that is able exploit quantum feature space) improves with dimension n. By analyzing their adoption of quantum-supported kernel methods like the QSVM, we gain useful intuition on the process behind building a QEML algorithm like the QKNN. This process has also been simplified due to pre-implementation of practical quantum algorithms on the Qiskit Aqua library. Some practical algorithms already available in Aqua include Grover's algorithm, quadratic estimator, the Fourier transform, eigen_solver, etc.

5. Building a QKNN and computing Hamming Distance

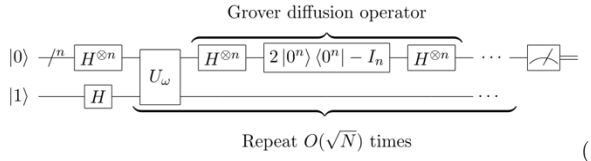
We will now lay down the foundation for distance calculation based on the earlier discussion of Schuld's trick and Havlicek et al.'s use of an enhanced quantum feature space. Based on analysis of metrics of quantum amplitude estimation, the Hamming Distance was found to be most optimal for higher dimension categorical learning problems. Hamming Distance was selected since it solves the "*Curse of Dimensionality*" for typical KNN. In a KNN algorithm, we usually have N of D patterns and access to the training set. The complexity of rating of one neighbor is $O(D)$. To rate all neighbors, we achieve complexity $O(ND)$. To find an additional "K" nearest neighbors on top of the already defined examples, we have complexity $O(KN)$. Thus, our total algorithmic

complexity for KNN is shown to be $O(ND + KN)$. Thus, despite current technology and polynomial factorization, it is still a computationally expensive task to run the KNN algorithm for large datasets (especially in higher dimensions).

Thus, we can solve this problem by adopting a quantized metric for computing distance (rating) between sample data points. This paper uses the *Hamming Distance* as the primary metric for assessing the distance between sequential data in a quantum vector space. We may now define the Hamming Distance: it is the number of positions at which the corresponding symbols of two-bit vectors of equal length are different. The operator for a Hamming distance is $\leftrightarrow$. This is best illustrated with a few examples: $00101 \leftrightarrow 00101 = 0$, $00101 \leftrightarrow 00111 = 1$, $00101 \leftrightarrow 10111 = 2$. The Hamming distance metric always meets 3 criteria: 1.) non-negative, 2.) symmetric in nature, 3.) satisfies the triangle inequality. It is not immediately obvious how the Hamming distance is useful for determining the distance between two feature vectors. However, Hamming distance is used for a variety of practical applications (ex. text classification, image classification). Even a simple QKNN algorithm in Hamming space is competitive with the highest performing discriminative models. Hamming distance also allows us to skip time-consuming operations in manipulating quantum state such as phase estimation and tomography.

5.1 – Simulating Grover's Algorithm

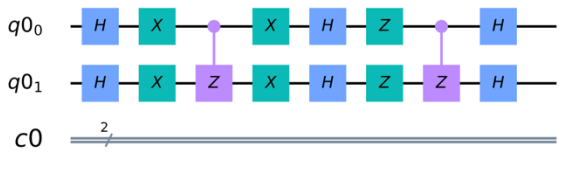
The next algorithm which can be used to model a QKNN is Grover's Algorithm. Now that we have identified that we will find the distances between labeled examples in a quantum feature space using Hamming Distance, we can now use Grover's Algorithm to find the minimum spanning distance (find the nearest neighbors). Grover's Algorithm is illustrated below:



(19)

Grover's Algorithm is one of the best examples to demonstrate the physical speed-up that quantum computers provide. Alongside Shor's Algorithm, it is one of the most prominent algorithms used in quantum computation and provides direct benefit.

If we are searching through a dataset of dimension n, the time it takes to locate a certain data point takes $O(N)$ on a classical computer. However, if we use Grover's Algorithm and a quantum computer, this operation only takes $O(\sqrt{N})$ time in terms of its complexity. Grover's Algorithm is also known as an Oracle algorithm and it applies both unitary operators and amplify operator probability amplitudes (together, these denote the *Grover Diffusion Operator*). We can use the IBM Quantum Experience platform and Qiskit library to model the Grover Algorithm. An example of Grover's Algorithm for 2 qubits in a closed quantum space is given below:

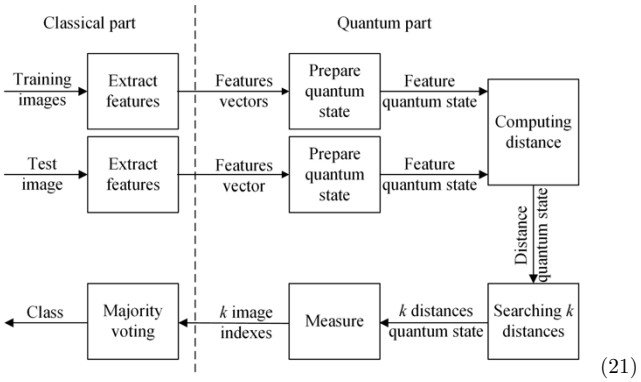


(20)

We have thus demonstrated that is both feasible and practical to implement Grover's Algorithm for the purpose of calculating the distance between points in a quantum feature map.

5.2 – Application towards QKNN

We can summarize the background research in two approaches: the implicit and explicit techniques for building a quantum classifier. In the implicit approach, the kernel (quantum feature space) can be evaluated using traditional computation and can estimate the inner products via $\Phi: \vec{x} \, \epsilon \, \Omega \rightarrow ||\Phi(\vec{x})\rangle \quad |\Phi(\vec{x})\rangle|$. The explicit approach involves finding the physical distances between data points in the "feature Hilbert space" of the quantum system. Techniques like the Hamming Distance and Grover's Algorithm allow us to conduct the explicit approach for quantum classification. We may now summarize the overall explicit approach for an image classification QKNN classifier in the following flow-chart:

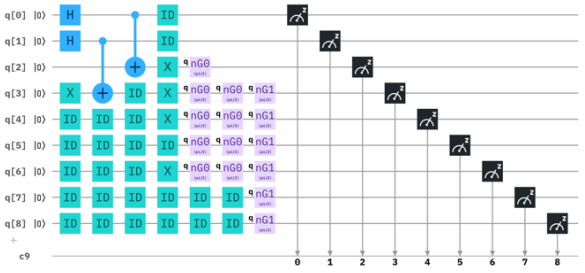


(21)

In this explicit approach, the model is solely trained by the quantum computer (no outside computation) and is able to output the prediction directly without the need for intermediate computation of the kernel. This approach corresponds with the recognized goals of QEML. It promotes both faster execution due to quantum algorithms and more storage capabilities due to an enhanced feature space in a quantum register. One technique to create a desirable quantum Hilbert space is through squeezing as proposed by Schuld et al. with their example of the Fock space.

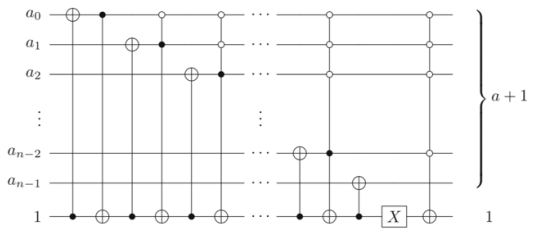
6. Variational Circuit for QKNN

As earlier mentioned, quantum enhanced KNN (QKNN) offers a few key benefits over its classical learning counterpart. Below is the final QKNN circuit that was built based on the intuition of Fidelity and the Hamming Distance metric:



(22)

This complete circuit calculates Hamming distance and evaluates the quantum state of each qubit via special subroutines. In a QEML algorithm, if we map the features of our data to the ground quantum states in Hilbert space, it is then easier to select the K nearest neighbors. The variational circuit shown above employs Schuld's trick of using fidelity and later calculates the Hamming distance, while also manipulating quantum parallelism with a 9-qubit register. The core module for the above circuit was suggested by Kaye and is illustrated below:



(23)

This circuit by Kaye is also known as the *Quantum a + 1 circuit* and uses incrementation. Prior to understanding this circuit, we must describe the setup. All bit vectors are mapped to their quantum ground state. ($0 \rightarrow |0\rangle$ and $1 \rightarrow |1\rangle$). The training set with N feature vectors is represented in the following training set superposition:

$$|\mathcal{T}\rangle = \frac{1}{\sqrt{N}} \sum_{p} |v_1^p ... v_n^p, c^p\rangle \tag{24}$$

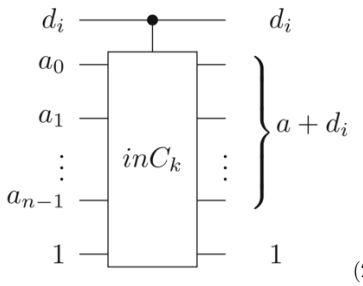
This circuit's underlying principle is to use addition between ancillary qubits. If a qubit (a_0) is "flipped" by this circuit, the circuit switches to the next least significant qubit. (i.e. $a+1$). The number a is within the bounds $[0, n-1]$. If a qubit $a[i]$ is "flipped" from 1 to 0, the addition continues. On the other hand, if $a[i]$ is flipped from 0 to 1, the addition stops and $a[i]$ is reset to 1, prompting the circuit to then continue. The workflow of this core circuit is shown below:

```
i = 0;
Do
    if a[i]==1 then {
        a[i]: 1 → 0;
        i++;
    }
    else
        a[i]: 0 → 1;
Until (a[i]: 0 → 1)
```
(25)

The physical calculation of the Hamming Distance is now undertaken by this circuit and we must expand from the Quantum $a+1$ circuit to a "$a+d$" circuit. We first apply a CNOT gate (to overwrite the first entry a as 0 if a = b) and later an X gate (to reverse the value). We now record the distances between all training points ($|x_1, \ldots, x_n\rangle$) and the new examples in the training set, ($|v_1, \ldots, v_n\rangle$)). We store the vector of computed distances ($|d^p{}_1, \ldots, d^p{}_n\rangle$). Since we had already constructed the quantum state ($|\phi_0\rangle$) in the first register, the training set ($|\mathcal{T}\rangle$) in the second register, and an ancillary qubit in the last register ($|0\rangle$), we can represent the labeling of neighbors (modification of the ancillary qubit $|0\rangle \rightarrow |1\rangle$. in the following unitary operation:

$$|\phi_3\rangle = U|\phi_2\rangle = \frac{1}{\sqrt{N}}\left(\sum_{p\in\Omega}|d_1^p \ldots d_n^p; v_1^p \ldots v_n^p, c^p; 1\rangle + \sum_{p\notin\Omega}|d_1^p \ldots d_n^p; v_1^p \ldots v_n^p, c^p; 0\rangle\right) \quad (26)$$

The Hamming distances can then be represented in the summation $\sum_i d^p{}_i$ and can then be introduced into the previous Quantum $a+1$ circuit to achieve an "$a+\ d_i$" quantum circuit. This new circuit is modeled below:



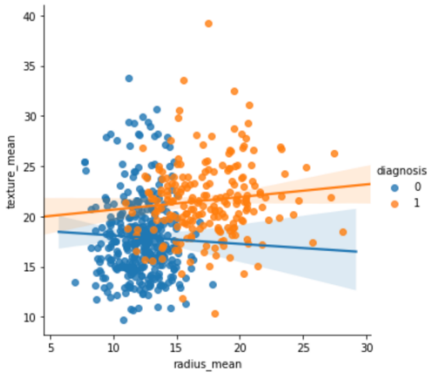
(27)

The quantum ***OR*** gate is now applied on top of the $a+\ d_i$ to achieve the final QKNN circuit (as modeled earlier). The majority voting in the classic KNN is best modeled in the QKNN as the probability of getting the final classification result. The IBM Quantum Experience platform allows us to simulate the QKNN circuit with quantum gates like ***OR***, ***ID*** (null gate), ***H***, ***X***, etc.

6.1 – Comparison to KNN with Breast Cancer Dataset

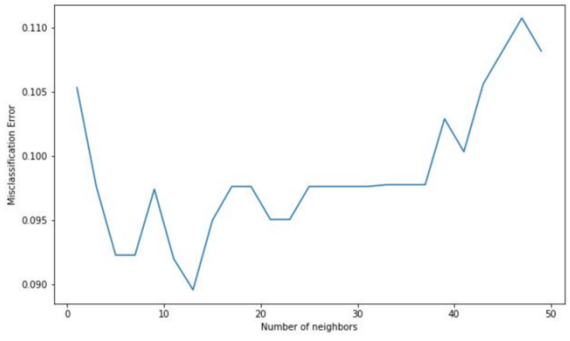
Since the main goal of this paper was to offer a side-by-side comparison of both classic ML and QEML, it would be incomplete without running tests to compare both the traditional KNN to its quantum variant (QKNN). The previously discussed benefits of QEML can be restated here: 1.) more storage capabilities and smaller solution space (due to enhanced feature space via superposition) 2.) faster execution of quantum algorithms due to quantum parallelization. We can now put the QKNN algorithm to the test by comparing its performance on a sample dataset against the classic lazy-learning KNN machine learning classifier. The dataset chosen for this problem was the *Wisconsin Breast Cancer Dataset* since it has $n = 10$ dimensions (multivariate) and is primarily useful for binary classification. 10 dimensions is considered optimal for gauging the performance of QEML against traditional machine learning since it is enough dimensions such that QEML should provide a tangible benefit but is not too large in that both ML and QEML will not become computationally expensive. We can first implement a KNN algorithm for the classification of tumors as either malignant (1) or benign (0). The dataset has the following

attributes: diagnosis, radius_mean, texture_mean, perimeter_mean, area_mean, smoothness_mean, compactness_mean, concavity_mean, concave points_mean, symmetry_mean, etc. We can implement the KNN algorithm in a Jupyter Notebook. The first step is to input data as a *.csv* file and to analyze it for any missing cases or attributes. We then split into features and labels (X and y). The data can be summarized in the following graph:



(28)

We can then use `sklearn` modules like `model_selection.train_test_split` to split the data into a reasonable train-test split (0.65 to 0.35 in this case). We can then use the Scikit-Learn `KNeighborsClassifier` class to fit a pre-made KNN algorithm to the breast cancer dataset. This specific KNN algorithm utilizes Euclidean Distance as its distance calculation metric and is thus comparable to the quantum approach. This version of the KNN algorithm is also unique in that it is designed solely for discrete classification. The inputs for classes for the algorithm were *NumPy* arrays and they were split using the aforementioned ratio. A nearest Centroid classifier was considered but was not substituted in place of a K neighbors classifier since there were no major benefits in the Centroid approach. The chosen KNN algorithm is also special in that it does not require traditional hyperparameter tuning. Rather, we must adjust the value for K from its initial arbitrary value (usually 3). In this case, the optimal K (that produces minimal loss and misclassification) was determined to be $K = 13$ as seen below:



(29)

The final performance for the KNN algorithm was evaluated via the `accuracy score` metric of Scikit-Learn (average accuracy for the data with a certain number of neighbors during a training run). The QKNN circuit (fig. 22) was also used to do training on the IBM QASM simulator, providing similar results for accuracy. As earlier mentioned, this QKNN algorithm uses Hamming

distance as the metric for calculating distance between neighbors in a quantum feature space. The final mean result for average performance/accuracy is shown below:

Algorithm	Performance
QKNN	0.9718
KNN	0.9627

These numbers come close to validating the original theory for why QKNN would outperform KNN in both performance and execution time due to quantum superposition. Although these results do not prove that QKNN has a statistically significant performance boost over KNN in terms of accuracy/performance on the same data, they provide a foundation for testing of quantum algorithms on much larger datasets. For example, one question that may arise is how is the performance difference between KNN and QKNN affected when there are $n = 100$ dimensions or $n = 1000$ dimensions? Answering these queries will require future analysis.

7. Conclusions and Cost Analysis

Although this paper has proved that quantum computing applied to machine learning (QEML) does provide some intermediate benefit, we cannot definitively argue that QEML is superior to machine learning and will be the future of big data computation. These results were somewhat randomized and were computed on a limited dataset of $n = 10$ dimensions, so the conclusions can only be taken with a grain of salt. However, we were definitively able to simulate both the property of fidelity (analogous to cosine similarity in classic machine learning) and the path of Grover's algorithm for 2 qubits (analogous to minimum path search in classic machine learning). Despite the fact that QKNN could provide slightly more accurate results in a fraction of the time, the results are not statistically significant and that means that QEML has only been partially validated.

These experiments to simulate fidelity, the Grover's algorithm, and a QKNN were only possible due to open-source frameworks like the Qiskit python library and the IBM Quantum Experience platform. In the future when quantum computing is more accessible, we may have to conduct a cost-analysis to conclude whether the benefit is worth-it or not. Currently, physical quantum computers are not accessible to the wide population and are only accessible for researchers and collaborators at recognized institutions. Another important issue to recognize is that current quantum computers are quite susceptible to noise and physical perturbations during calculations. Moreover, decoherence is another key issue since loss/collapse of quantum information is still a common occurrence for today's quantum computers, and this will not be practical if physical quantum computing will one day be accessible to the majority of researchers. The ongoing "quest" to build a fault-tolerant quantum computer is still under development and will continue to require many more layers of abstraction and multiple paradigm shifts in the relationship between software and hardware. However, the success of the above experiments is a positive step for the field of quantum computing since it represents the synchronization of machine learning in quantum computing to benefit society as a whole. To summarize, quantum computing can benefit machine learning due to the idea of an enhanced quantum feature space. In essence, while fault-tolerant

quantum computers are still decades away, we can still harness the power of quantum computing to improve both the efficiency and scalability of machine learning algorithms.

8. References

[1] Feynman, R.P.: Simulating physics with computers. Int. J. Theor. Phys. 21(6), 467-488 (1982)

[2] Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. IEEE Computer Society, 124-134 (1994)

[3] Grover, L.K.: A fast quantum mechanical algorithm for database search. ACM. 212-219 (1996)

[4] Le, P.Q., Dong F., Hirota, K.: A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. Quantum Inf. Process. 10(1), 63 (2011)

[5] Vojtech Havlicek, Antonio D. Corcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum enhanced feature spaces. arXiv preprint arXiv: 1804.11326, 2018.

[6] Zhang, Y., Lu, K., Gao, Y.H., Wang, M.: NEQR: a novel enhanced quantum representation of digital images. Quantum Inf. Process. 12(8), 2833-2860 (2013)

[7] Jiang, N., Wang, J., Mu, Y.: Quantum image scaling up based on nearest-neighbor interpolation with integer scaling ratio. Quantum Inf. Process. 14(11), 1-26 (2015)

[8] Esma Amieur, Gilles Brassard, and Sebastien Gambs. Machine learning in a quantum world. In Advances in Artificial Intelligence, pages 431–442. Springer, 2006.

[9] Daoyi Dong, Chunlin Chen, Hanxiong Li, and Tzyh-Jong Tarn. Quantum reinforcement learning. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 38(5):1207–1220, 2008.

[10] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. arXiv preprint arXiv:1307.0411, 2013.

[11] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big feature and big data classification. arXiv preprint arXiv:1307.0471, 2013.

[12] Ilya Levner. Feature selection and nearest centroid classification for protein mass spectrometry. BMC bioinformatics, 6(1):68, 2005.

[13] Wisconsin Breast Cancer Dataset. (n.d.). Retrieved from https://archive.ics.uci.edu/ml/datasets/Breast Cancer Wisconsin (Diagnostic)

[14] Maria Schuld, Nathan Killoran. Quantum machine learning in feature Hilbert spaces. arXiv preprint arXiv: 1803.07128, 2018

[15] Dongnan Daxue Xuebao (Ziran Kexue Ban)/Journal of Southeast University (Natural Science Edition). Quantum K-nearest neighbor algorithm. 2015. 45(4):647-651 DOI: 10.3969/j.issn.1001-0505.2015.04.006

[16] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, Seth Lloyd. Quantum Machine Learning. arXiv preprint arXiv: 1611.09347, 2016, updated 2018.

[17] Yue Ran, Xiling Xue, Heng Liu, Jianing Tan, Xi Li. Quantum Algorithm for K-Nearest Neighbors Classification Based on the Metric of Hamming Distance. International Journal of Theoretical Physics 56(4) DOI: 10.1007/s10773-017-3514-4. 2017.

[18] Hu, W. (2018) Comparison of Two Quantum Nearest Neighbor Classifiers on IBM's Quantum Simulator. Natural Science, 10, 87-98. DOI: 10.4236/ns.2018.103010.

[19] Ning Yang. KNN Algorithm Simulation Based on Quantum Information. Proceedings of Student-Faculty Research Day Conference, CSIS, Pace University, May 3rd, 2019.

[20] Paul I. H., and Ioannis G. K., "Quantum Computers: Registers, Gates and Algorithms", Proc. 28th International Conference on Microelectronics (MIEL 2012), pp. 15-21, 2012.

[21] Grover, L. K." Quantum Mechanism Helps in Searching for a Needle in a Haystack", Physical Revies Letters, 79, 325-8, 1997.

[22] Yijie Dang, Nan Jiang, Hao Hu, Zhuoxiao Ji, Wenyin Zhang. Image Classification Based on Quantum KNN Algorithm. arXiv preprint arXiv: 1805.06260, 2018.

[23] Bravo, J.M. Calculating Hamming Distance with the IBM Q Experience. Preprints 2018, 2018040164 (DOI: 10.20944/preprints201804. 0164.v2).

9. Acknowledgements

1. The Qiskit Team and the IBM Quantum Experience development team for their fantastic work and mission to make quantum computing more accessible for researchers

2. Kevyn Adams of BASIS Independent Silicon Valley for being my mentor

3. My family for endearing me to continue my work