Project Report - Quantum Machine Learning

Amrit Singhal
amrits@iitk.ac.in
Indian Institute of Technology Kanpur - Quantum Computing (CS682)

Abstract

The aim of the project is to study two of the most widely used machine learning strategies, namely K-Nearest Neighbours algorithm and Perceptron Learning algorithm, in a quantum setting, and study the speed-ups that the quantum modules allow over the classical counterparts. The study is primarily based on the following 3 papers:

1. Quantum Perceptron Models, by N. Wiebe, A. Kapoor and K. M. Svore.
2. Quantum Algorithm for K-Nearest Neighbors Classification Based on the Metric of Hamming Distance, by Y. Ruan, X. Xue, H. Liu, J. Tan, and X. Li.
3. Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning, by N. Wiebe, A. Kapoor and K. M. Svore.

Keywords – Quantum Machine Learning, Perceptron, Nearest Neighbours, Hamming Distance, Inner Product via Swap test

Introduction

Motivation Machine Learning is one of the fastest developing fields in computer science in today's time. Problems in machine learning frequently require manipulation of large number of high dimensional vectors. Quantum Computers are pretty good at handling multiple large dimensional vectors simultaneously, because of the inherent superposition property of quantum systems. This allows to "seemingly" operate on large number of vectors (implemented as the superposition state) simultaneously. Although, it must be noted that only one of these vectors can be seen in output, and all other "computation" done cannot be used, as the state will in principle be lost on measurement. So, we need cleaver techniques to utilize this property of quantum systems to get a better implementation of machine learning techniques than in classical case.

The project This project demonstrates how certain classical machine learning algorithms can be implemented faster on a quantum computer using its special properties, making only slight modifications. Specifically, two machine learning techniques have been studied:

1. The K-Nearest Neighbours Learning Algorithm (using various similarity metrics)
2. The Perceptron Learning Model

The rest of the report will sequentially describe these techniques.

Quantum Nearest Neighbor Learning Classification algorithms

What is meant by nearest neighbor learning? To speak plainly, nearest neighbor learning aims to classify a given new data point *i.e. test vector* into one of the predetermined classes. To do this, we compare the similarity of the *test vector* with all the training data points, and we assign to the test sample the class of the most similar training data point.
More formally, we are given a set of N training examples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \in \mathbb{R}^D$, with each of them belonging to a class $\{y_1, y_2, \cdots, y_N\}$, out of a total of l classes such that $y_i \in \{1, 2, \cdots, l\}, \forall i \in [N]$. Now, given a test vector $\{\mathbf{v}_0\} \in \mathbb{R}^D$, we use some similarity metric S to find the most similar vector to $\mathbf{v}_0$. Say $\mathbf{v}_m$ be the vector such that $S(\mathbf{v}_m, \mathbf{v}_0) \geq S(\mathbf{v}_i, \mathbf{v}_0), \forall i \in [N]$, then we assign to $\mathbf{v}_0$ the class y_m.

NN algorithms prove to be very accurate if they are given a big enough amount of training data. But, this accuracy comes at a computation cost of $O(N)$, which is very high is big data scenarios where NN techniques are generally very accurate. Using quantum superposition property, and some fast quantum modules for search and minimum finding, we can perform this estimation in $O(\sqrt{N} \log N)$, as done by *Weibe et al (1)* and can even be made independent of the number of training data points, as shown by *Ruan et al (2)*.

These two bounds are achieved by taking two widely different similarity metrics. The former uses inner product value between vectors as the similarity metric, whereas the latter utilizes the Hamming Distance between two bit-strings to produce the desired output.Thus, clearly depending on the choice of the metric S, we can have different variants of Quantum-KNN, with varying accuracies, and complexities.

Amplitude Estimation

Given a state $|\Psi\rangle + |\Psi_1\rangle + |\Psi_2\rangle$, where $|\Psi_1\rangle$ is the superposition of all the "good" states that satisfy any required property, and $|\Psi_2\rangle$ is the superposition of all the "bad" states that do not satisfy that property. Amplitude Estimation is the problem of determining $a = \langle\Psi_1, \Psi_1\rangle$, i.e. the probability that $|\Psi\rangle$ yields a "good" state on measurement.(3)

Theorem 1. (Amplitude Estimation) For any positive integers k and L, the amplitude estimation algorithm outputs $\tilde{a}(0 \leq \tilde{a} \leq 1)$ such that

$$|\tilde{a} - a| \leq 2\pi k \frac{\sqrt{a(1-a)}}{L} + \left(\frac{\pi k}{L}\right)^2$$

with probabilty $\geq 8/\pi^2$ when $k = 1$ and with probability $\geq 1 - 1/(2(k-1))$ for $k \geq 2$. It uses *exactly* L iterations of the Grover's algorithm in the procedure.

So, amplitude estimation allows us to estimate the amplitude squared of a marked component of a state.

Complexity If we try to determine the amplitude of the component via statistical sampling, then we require $O(\frac{1}{\gamma^2})$ queries to determine the result with a error tolerance of γ. Amplitude Estimation reduces the scaling with γ to $O(1/\gamma)$, providing quadratic speed-up.

Durr-Hoyer Minimum Finding Algorithm

Consider the problem where we are given an unsorted list L of N elements. Each of these elements is an element from a known ordered set S. We need to find the element in the L which is the minimum in the S ordering. Clearly, this task cannot be accomplished classically in less than $O(N)$ steps, as we definitely will have to look at all the elements once before concluding the minimum.

However, Durr and Hoyer presented a simple quantum algorithm (4) which can do the aforementioned task in just $O(\sqrt{N})$ queries to the oracle of L.

The Algorithm The exact algorithm for task goes as follows:

Uniformly randomly choose an index $y \in \{0, 1, \cdots, N-1\}$;
while *Running Time* $\leq 22.5\sqrt{N} + 1.4\lg^2 N$ **do**
  Initialize state as

$$\sum_{i=1}^{N} \frac{1}{\sqrt{N}} |j\rangle |y\rangle$$

  Mark every item i for which $L[i] < L[y]$;
  Search for a marked state using quantum exponential searching;
  Measure the first register, say outcome is z ;
  if $L[z] < L[y]$ **then**
    $y \leftarrow z$;
  end
end
Output y

Algorithm 1: Quantum Minimum Finding(4)

Theorem 2. The expected number of Grover's iterations required to get the minimum element in a given set $\{y_1, y_2, \cdots, y_N\}$ is bounded by

$$\frac{45}{2}\sqrt{N}$$

Now, having mentioned the required quantum modules, we now proceed on to mention the KNN algorithm using inner product value between vectors as the similarity metric.

NN using Inner Product

Problem Setup

We have a set of N training vectors $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_N\}$. The test vector, which is to be classified, is taken to be $\mathbf{v}_0$. All of the training vectors as well as the test vectors are *d-sparse*, which means $\forall j \in \{0, 1, \cdots, N\}, \mathbf{v}_j$ has atmost d non-zero entries. Let $v_{ji} = r_{ji}e^{i\phi_{ji}}$ denote the i-th element of the $\mathbf{v}_j$ vector, and $f(j, l)$ be the location of the l-th non-zero entry in $\mathbf{v}_j$. We have the following quantum oracles provided to us:

$$\mathcal{O} : |j\rangle |i\rangle |0\rangle \rightarrow |j\rangle |i\rangle |\mathbf{v}_{ji}\rangle$$

$$\mathcal{F} : |j\rangle |l\rangle \rightarrow |j\rangle |f(j, l)\rangle$$

Also, it is known that $\forall j, i,\ \mathbf{v}_{ji} \leq r_{max}$

Lemma 3. The state $|\Psi_j\rangle$ as

$$\frac{1}{\sqrt{d}} |j\rangle \sum_{i=1}^{d} |f(j,i)\rangle \left(\sqrt{1 - \frac{r^2_{jf(j,i)}}{r_j^2\ max}} e^{-i\phi_{jf(j,i)}} |0\rangle + \frac{r_{jf(j,i)}}{r_{j\ max}} e^{i\phi_{jf(j,i)}} |1\rangle \right)$$

can be prepared using 3 oracle calls to $\mathcal{O}$ and $\mathcal{F}$.

Proof. The procedure is as follows:

1. Start with the state $|j\rangle|0\rangle|0\rangle|0\rangle$.
2. Apply Hadamard ($H^{\otimes[\log d]}$) on second register to obtain $\frac{1}{\sqrt{d}}\sum_{i=1}^{d}|j\rangle|i\rangle|0\rangle|0\rangle$.
3. Apply oracle $\mathcal{F}$ to obtain $\frac{1}{\sqrt{d}}\sum_{i=1}^{d}|j\rangle|f(j,i)\rangle|0\rangle|0\rangle$.
4. Query from $\mathcal{O}$ to obtain $\frac{1}{\sqrt{d}}\sum_{i=1}^{d}|j\rangle|f(j,i)\rangle\left|v_{jf(j,i)}\right\rangle|0\rangle$.
5. Apply $R_y(2sin^{-1}(r_{jf(j,i)}/r_{j\ max}))$ on the last qubit to obtain

$$\frac{1}{\sqrt{d}}\sum_{i=1}^{d}|j\rangle|f(j,i)\rangle\left|v_{jf(j,i)}\right\rangle\left(\sqrt{1-\frac{r_{jf(j,i)}^2}{r_j^2\ max}}|0\rangle + \frac{r_{jf(j,i)}}{r_{j\ max}}|1\rangle\right)$$

6. Apply $R_z(2\phi_{jf(j,i)})$ to the last qubit, followed by $\mathcal{O}^\dagger$ for cleaning the ancilla register, to obtain the required state.

Clearly, only 3 queries to $\mathcal{O}$ and $\mathcal{F}$ were enough to prepare the state. □

How to get to inner product? To get the inner product of $\mathbf{v}_0$ and $\mathbf{v}_j$, prepare the states $|\Psi_0\rangle$ and $|\Psi_j\rangle$, as shown in Lemma 4. Then, add an ancilla register $|1\rangle$ in both states, but at different locations to get the following two states:

$$|\psi_j\rangle = \frac{1}{\sqrt{d}}|j\rangle\sum_{i=1}^{d}|f(j,i)\rangle\left(\sqrt{1-\frac{r_{jf(j,i)}^2}{r_j^2\ max}}e^{-i\phi_{jf(j,i)}}|0\rangle + \frac{r_{jf(j,i)}}{r_{j\ max}}e^{i\phi_{jf(j,i)}}|1\rangle\right)$$

and

$$|\phi\rangle = \frac{1}{\sqrt{d}}\sum_{i=1}^{d}|f(0,i)\rangle|1\rangle\left(\sqrt{1-\frac{r_{0f(0,i)}^2}{r_0^2\ max}}e^{-i\phi_{0f(0,i)}}|0\rangle + \frac{r_{0f(0,i)}}{r_{0\ max}}e^{i\phi_{0f(0,i)}}|1\rangle\right)$$

Adding these extra $|1\rangle$ qubits helped us get rid of the unwanted terms from he amplitde of $|0\rangle$ in the inner product of the states. So, the inner product between $|\psi_j\rangle$ and $|\phi\rangle$ comes out to be:

$$\langle\psi_j,\phi\rangle = \frac{1}{d}\sum_i\frac{v_{ji}v_{0i}^*}{r_{j\ max}r_{0\ max}} = \frac{\langle\mathbf{v}_0,\mathbf{v}_j\rangle}{dr_{j\ max}r_{0\ max}}$$

So, we need to get the value of $\langle\psi_j,\phi\rangle$ which will enable us to get the inner product between the vectors as well.

Swap test

Lemma 4. There exists a quantum SWAP gate F that swaps two given qubits.

Proof. The circuit in Figure 1 implements the SWAP gate.
$|\psi\rangle|\phi\rangle \rightarrow |\psi\rangle|\phi\oplus\psi\rangle \rightarrow |\psi\oplus\phi\oplus\psi\rangle|\phi\oplus\psi\rangle = |\phi\rangle|\phi\oplus\psi\rangle \rightarrow |\phi\rangle|\phi\oplus\psi\oplus\phi\rangle = |\phi\rangle|\psi\rangle$ □

[1]

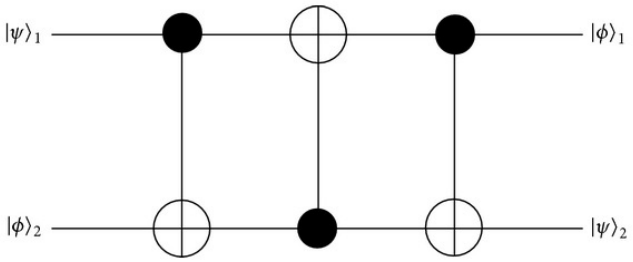

Figure 1: **SWAP Gate**

Theorem 5. (SWAP Test) Using the SWAP gate, we can have a efficient quantum circuit that can allow us to estimate the inner product of the two input vectors.

Proof. Follow the steps given below:

1. Start with the state $|0\rangle|\psi\rangle$ $\quad(|\psi\rangle = |x\rangle|y\rangle)$
2. Apply Hadamard on the first qubit to obtain $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)|\psi\rangle$.
3. Apply controlled SWAP gate to obtain $\frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle F(|\psi\rangle))$
4. Apply Hadamard on the first qubit again to obtain

$$|\vartheta\rangle = \left(\frac{I+F}{2}\right)|0\rangle|\psi\rangle + \left(\frac{I-F}{2}\right)|1\rangle|\psi\rangle$$
$$= P_0|0\rangle|\psi\rangle + P_1|1\rangle|\psi\rangle$$

Here, P_0 and P_1 form a projection as $P_0+P_1=I$ and $P_0\cdot P_1=0$.

[1] https://www.researchgate.net/figure/263776912_fig2_The-swap-gate-cascades-three-quantum-Controlled-Not-gates

Therefore,

$$Pr(0) = \langle\psi|P_0|\psi\rangle = \frac{1}{2}\langle xy, xy + yx\rangle$$
$$= \frac{1}{2}\left(1 + (\langle x, y\rangle)^2\right)$$

So,

$$|\langle x, y\rangle|^2 = (2P(0) - 1)$$

□

So, in out KNN algorithm, perform SWAP test on $|\psi_j\rangle$ and $|\phi\rangle$ to obtain the state $|\vartheta_j\rangle$, such that the probability of measuring the first qubit of $|\vartheta_j\rangle$ to be **0** will allow us to get the inner product between $\mathbf{v}_j$ and $\mathbf{v}_0$. We can use statistical sampling to get this probability, but due to reasons already mentioned earlier, we will use amplitude estimation instead.

Getting the $P(0)$ from amplitude estimation, we get the inner product between $\mathbf{v}_j$ and $\mathbf{v}_0$ for any $j \in [N]$. Now, to get the maximum inner product, we can either do a linear check over all possible values of j as done in the classical case, which would require $O(N)$ time. But using the Durr-Hoyer algorithm for finding minimum(or maximum, in our case), we can get the maximum value of the inner product in just $O(\sqrt{N})$ queries.

Note Amplitude Estimation is a irreversible process, as it involves a measurement of the final state. So, to be able to use the Durr-Hoyer algorithm, we need to perform amplitude estimation in a reversible manner. The reversible version of the AE algorithm is know as *coherent amplitude estimation*.

Coherent Amplitude Estimation

Normal amplitude estimation outputs the state $\sqrt{a}\,|y\rangle + \sqrt{1-|a|}\,\left|y^\perp\right\rangle$, where $|y\rangle$ is a bit-string that encodes the desired amplitude.
Let $\mathcal{A}$ be the unitary operator that produces the state $\sqrt{a}\,|y\rangle + \sqrt{1-|a|}\,\left|y^\perp\right\rangle$ from the state $\left|0^{\otimes nk}\right\rangle$, for $1/2 \leq |a_0| \leq |a| \leq 1$.

Lemma 6. (Hoeffding's Inequality)[2]
For a Bernoulli's Disrtibution:

$$\mathbb{P}(H(n) \leq (p - \epsilon)n) \leq exp(-2\epsilon^2 n)$$

Theorem 7. Given $\mathcal{A}$, $\exists$ a algorithm such that for every $\Delta > 0, \exists k$ so that we can find a state $|\xi\rangle$ such that $\left|\left|\,|\xi\rangle - \left|0^{\otimes nk}\right\rangle|y\rangle\,\right|\right|_2 \leq \sqrt{2\Delta}$.

[2]https://en.wikipedia.org/wiki/Hoeffding%27s_inequality

Proof. Consider a unitary $\mathcal{M}$ that can find the median of k states, as follows:

$$\mathcal{M} : |y_1\rangle \cdots |y_k\rangle\,|0\rangle \to |y_1\rangle \cdots |y_k\rangle\,|\tilde{y}\rangle$$

This can be implemented by simply sorting the k states, and getting the middle element in $O(kn \log k)$ operations.

The algorithm goes as follows:

1. Prepare the start state $\left(\sqrt{a}\,|y\rangle + \sqrt{1-|a|}\,\left|y^\perp\right\rangle\right)^{\otimes k}$ (Independently prepare k copies through $\mathcal{A}$).
   We can partition this state as:
   $\left(\sqrt{a}\,|y\rangle + \sqrt{1-|a|}\,\left|y^\perp\right\rangle\right)^{\otimes k} = A\,|\Psi\rangle + \sqrt{1-|A|^2}\,|\Phi\rangle$
   where $|\Psi\rangle$ is the uniform superposition over states with median $= y$, and $|\Phi\rangle$, is the uniform superposition over states with median not equal to y.
2. Apply $\mathcal{M}$ to measure the median.
$$\mathcal{M}\left(\sqrt{a}\,|y\rangle + \sqrt{1-|a|}\,\left|y^\perp\right\rangle\right)^{\otimes k}|0\rangle$$
$$= A\,|\Psi\rangle\,|y\rangle + \sqrt{1-|A|^2}\,|\Phi\rangle\,\left|y^\perp\right\rangle$$
3. Apply $\mathcal{A}^{\dagger\otimes k}$ to the first register to obtain:
$|\xi\rangle = \mathcal{A}^{\dagger\otimes k}\left(A\,|\Psi\rangle\,|y\rangle + \sqrt{1-|A|^2}\,|\Phi\rangle\,\left|y^\perp\right\rangle\right)$
$= \mathcal{A}^{\dagger\otimes k}\left(A\,|\Psi\rangle\,|y\rangle + \sqrt{1-|A|^2}\,|\Phi\rangle\,|y\rangle\right)$
$+ \mathcal{A}^{\dagger\otimes k}\left(\sqrt{1-|A|^2}\,|\Phi\rangle\,|y\rangle - |\Psi\rangle\,|y\rangle\right)$
$= \left|0^{\otimes nk}\right\rangle|y\rangle + \mathcal{A}^{\dagger\otimes k}\left(\sqrt{1-|A|^2}\,|\Phi\rangle\,|y\rangle - |\Psi\rangle\,|y\rangle\right)$

Observe:

$$\left|\,|\xi\rangle - \left|0^{\otimes nk}\right\rangle|y\rangle\,\right| \leq \sqrt{2(1-|A|^2)}$$

So, we need $\mathbb{P}(y^\perp) = 1 - |A|^2 \leq \Delta$.

Claim 8. $\mathbb{P}(y^\perp) \leq \Delta$ if $k \leq \frac{\ln\left(\frac{1}{\Delta}\right)}{2\left(|a_0| - \frac{1}{2}\right)^2}$

Proof. In any sequence of measurements that contain more than $k/2$ y-outcomes, the median must be y.

$$\mathbb{P}(y^\perp) \leq \mathbb{P}(\text{no more than } k/2 \text{ } y\text{-outcomes})$$
$$= \sum_{i=0}^{k/2} \binom{k}{p} |a|^p |1 - |a||^{k-p}$$

Also, we know that $|a| > |a_0| > 1/2$. Thus,

$$\mathbb{P}(y^\perp) \leq exp\left(-2k\left(|a_0| - \frac{1}{2}\right)^2\right) \leq \Delta$$

$$\implies k \leq \frac{\ln\left(\frac{1}{\Delta}\right)}{2\left(|a_0|-\frac{1}{2}\right)^2}$$

□

So, for appropriate values of k, we have the relation $\mathbb{P}(y^{\perp}) \leq \Delta \implies \left| |\xi\rangle - \left|0^{\otimes nk}\right\rangle |y\rangle \right| \leq \sqrt{2\Delta}$. □

Theorem 8 proves that coherent majority voting can be used to remove the measurements used in algorithms such as amplitude estimation at the price of introducing a small amount of error in the resultant state.

Lemma 9. If $\mathcal{A}$ requires Q queries to the oracles, then $|\xi\rangle$ can be prepared by number of queries bounded above by $2Q\left(\frac{\ln\left(\frac{1}{\Delta}\right)}{2\left(|a_0|-\frac{1}{2}\right)^2}\right)$.

Proof. We apply amplitude estimation k times, and each of those requires Q queries, totaling to Qk queries. Also, we apply $\mathcal{A}^{\dagger\otimes k}$ which would require Qk further queries. So, total number of queries $= 2Qk$. □

Theorem 10. (1)
The final task of obtaining $\max_{j\in[N]}|\langle v_j, v_0\rangle|^2$ within error ϵ can be done with success probability $1-\delta_0$ requiring an expected number of queries bounded above by

$$1080\sqrt{N}\left\lceil\frac{4\pi(\pi+1)d^2r_{max}^2}{\epsilon}\right\rceil\left\lceil\frac{\ln\left(\frac{81N(\ln(N)+\gamma)}{\delta_0}\right)}{2(8/\pi^2-0.5)^2}\right\rceil$$

where $\gamma \approx 0.5772$ is Euler's constant.

KNN using Hamming Distance

What is Hamming Distance? Hamming Distance between two n-bit-strings A and B is defined as number of positions at which the strings A and B differ at the corresponding bits. Note that A and B must be of the same length for Hamming distance between them to be defined.

Why? As *Ruan et al* state(2): "Mapping a natural vector to a bit vector by well-defined hash functions, simple KNN classifiers in Hamming space are competitive with sophisticated discriminative classifiers, including SVMs and neural networks." We will see that this method will prove to be much faster for larger data, and are competitive with some of the more complicated and established classifiers.

Problem Setup

We are provided with feature vectors $|\mathbf{v}^p\rangle, p \in 1, \cdots, N$ and their corresponding classes $c^p \in \{1, \cdots, l\}$. If the feature vectors are not provided as bit-vectors, then we will have to use some hashing function such as linear mapping, or kernelized mapping, neural network based mapping or any other mapping to convert them into bit vectors. Henceforth, all $\mathbf{v}_j$'s have been taken to be bit vectors.
The testing example $|x_1, \cdots, x_n\rangle$ has been provided to be classified.
We are also provided with a Hamming Distance threshold t.
Our aim is to find all those training datapoints, whose Hamming Distance from the test sample is $\leq t$.

Quantum Circuit for $a+1$ (2)

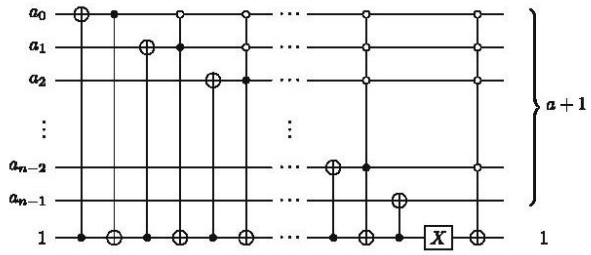

Fig. 3 Quantum $a+1$ circuit

Figure 2: **Quantum Circuit for** $a+1$

Construct the training superposition

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

Algorithm

1. Prepare the state

$$|\phi_1\rangle = \frac{1}{\sqrt{N}}\sum_{p=1}^{N}|x_1, \cdots, x_n; v_1^p, \cdots, v_n^p, c^p; 0\rangle$$

2. Record the difference between training and test vector, and store the result reversed in first register.

$$|\phi_2\rangle = \prod_k X(x_k)CNOT(x_k, v_k^p)\,|\phi_1\rangle$$
$$= \frac{1}{\sqrt{N}}\sum_{p=1}^{N}|d_1, \cdots, d_n; v_1^p, \cdots, v_n^p, c^p; 0\rangle$$

(CNOT(a,b) overwrites a with 0 if $a = b$, else 1.)

Now, Hamming distance between $|x\rangle$ and $|v^p\rangle = n - \sum_{i=1}^{n} d_i^p$.

$$\implies n - \sum_{i=1}^{n} d_i^p \leq t \text{ for all "good" training examples.}$$

Suppose $2^{k-1} \leq n \leq 2^k$, then define $m = 2^k - n$.

$$\implies \sum_{i=1}^{n} d_i^p + m + t \geq 2^k$$

So, set initial $a = m + t$. Then the condition $HD \leq t$ can be determined by whether the addition of $\sum_{i=1}^{n} d_i^p + a$ overflows or not.
We can implement $a + d_i$ as controlled $a + 1$, which only adds 1 to a if $d_i = 1$.
We get $a + \sum_{i=1}^{n} d_i$ using repeated controlled $a+1$ gates. Picking the most significant $\lceil \log t \rceil$ bits from that, we can say that if any of them is **1**, then the value computed is greater than 2^k and hence that training sample is a "good" sample. So, we simply apply an OR gate the these qubits, to get the output as $COND^p$. The quantum OR gate is shown in Figure 3, and the final circuit for computing $COND^p$ has been shown in Figure 4.

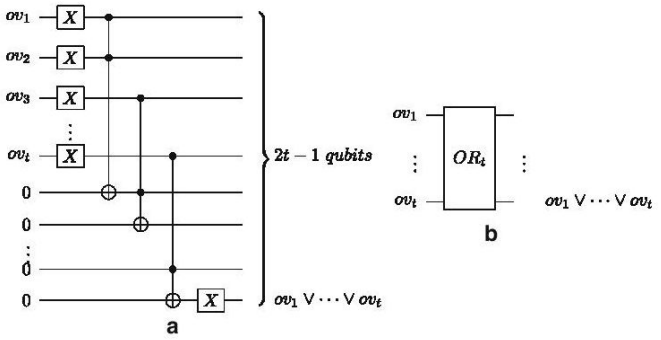

Fig. 5 The quantum OR circuit

Figure 3: **Quantum OR gate**

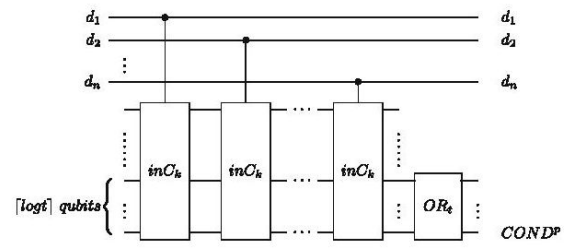

Fig. 6 The quantum circuit to generate $COND^p$

Figure 4: **Quantum Circuit for generating** $COND^p$

3. Define

$$\Omega = \{p| \text{ Hamming distance}(|x\rangle, |v^p\rangle) < t\}$$

Then, let U be the unitary such that

$$|\phi_3\rangle = U |\phi_2\rangle$$

$$= \frac{1}{\sqrt{N}} \Big(\sum_{p \in \Omega} |d_1, \cdots, d_n; v_1^p, \cdots, v_n^p, c^p; 1\rangle + \sum_{p \notin \Omega} |d_1, \cdots, d_n; v_1^p, \cdots, v_n^p, c^p; 0\rangle \Big)$$

4. Define $\Gamma = \mathbb{I} \otimes |1\rangle\langle 1|$. Obtain:

$$|\phi_4\rangle = \Gamma |\phi_3\rangle = \alpha \sum_{p \in \Omega} |d_1, \cdots, d_n; v_1^p, \cdots, v_n^p, c^p; 1\rangle$$

such that $\sum_{i=1}^{|\Omega|^2} |\alpha|^2 = 1$
α is the renormalized amplitude of each component of ϕ_4.

5. $|\phi_4\rangle$ is composed of $|v_p\rangle$ whose distance are no more than t to the testing sample. Measure c^p alone to get the category of the test sample $|x\rangle$.

Performance Analysis

The cost of this circuit is measured by the number of elementary gates { NOT, CNOT, Toffoli } required.
Constructing the training vector $\mathcal{T}$ takes $O(N)$ time, but this step is a preliminary step that needs to be done only once before the start of the algorithm as it will not change during the algorithm, and can be used for all subsequent runs of the algorithm without the need to create it again.
Among the main algorithm steps, first we have the $a+1$ gate.

$$\text{Cost of } a+1 \text{ gate} = \begin{cases} 1 & \text{if } k = 1 \\ 10 & \text{if } k = 2 \\ 2k^2 + k - 5 & \text{if } k \geq 3 \end{cases}$$

The total circuit to compute $COND^p$ contains this gate n times. Additionally, it also contains $\lceil \log t \rceil - 1$ NOT gates and $\lceil \log t \rceil + 1$ Toffoli gates. Hence,

$$\text{total cost } = n \cdot (2n^2 + n - 5) + (\lceil \log t \rceil + 1) + (\lceil \log t \rceil - 1)$$

$\implies$ the total cost of this algorithm is $O(n^3)$.

Note The complexity of this algorithm is independent of the number of training examples. This is because we work with a uniform superposition of all training examples, thus eliminating the need of repeated computation. Hence, in the case of big data scenarios, Hamming Distance method of nearest neighbor classification outperforms all other nearest neighbor methods. Also, the classification accuracy of the Hamming Distance

method is better than the one achieved by the other methods, as seen from Figure 5.(2)

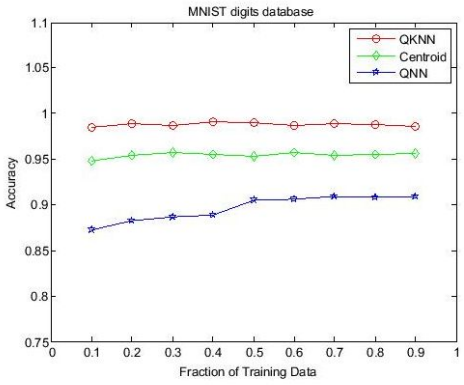

Fig. 7 The classification accuracy of QKNN, Centroid and QNN

Figure 5: **Hamming Distance QKNN vs other NN methods**

Quantum Perceptron Models

What is Perceptron Learning We are provided with a set of N training examples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \in \mathbb{R}^D$, with each of them having a label $\{y_1, y_2, \cdots, y_N\}$, such that $y_i \in \{+1, -1\}, \forall i \in [N]$. The perceptron model to be constructed is just a hyperplane in the space, to be parametrized as $\mathbf{w} \in \mathbb{R}^D$ such that the hyperplane forms a perfect barrier between the two sets of labeled data. Mathematically, we need to find such a $\mathbf{w}$ so that $y^i \cdot \mathbf{w}^T \mathbf{x}_i > 0$ for all $i \in [N]$.
Such a $\mathbf{w}$ is learned by initializing $\mathbf{w}$ to some random vector, and then updating it suitably every time a training example is misclassified by the model. The update rule that will be considered here is $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$, where $(\mathbf{x}, y)$ is the vector that has been misclassified.

Theorem 11. (Novikoff) (5)
Let $\mathbf{x}_1, \cdots, \mathbf{x}_T \in \mathbb{R}^N$ be a sequence of T points with $||\mathbf{x}_t|| \leq r, \forall t \in [1, \cdots, T]$, for some $r > 0$. Assume that there exists $\rho > 0$ and $v \in \mathbb{R}^N, v \neq 0$, such that $\forall t \in [1, T], \rho \leq \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{||\mathbf{v}||}$. Then, the number of updates made by the Perceptron algorithm when processing $x_1, \cdots, x_T$ is bounded by r^2/ρ^2.

Using Theorem 1, it is clear that if the training data is separated by a margin of γ, then atmost $O\left(\frac{1}{\gamma^2}\right)$ updates are need to be made on $\mathbf{w}$.

Classical Complexity We cannot do better than processing once at each training point in the classical case, because all points are independent. So, the complexity cannot be reduced beyond $O(N)$ classically.

Note We will see that this complexity can be quadratically improved and brought down to $O(\sqrt{N})$ in the quantum setting.

Grover's Search

Consider the problem where we are required to find the unique input to a blackbox function $f(x)$ that produces a particular output value. This cannot be done in better than $O(n)$ evaluations of f classically, where n is the total number of possible inputs to f. This is because in the worst case we will need to try out all the possible inputs to f. On the other hand, the given task can be accomplished in only $O(\sqrt{n})$ queries to oracle of f on a quantum computer, using the novel Grover's Search algorithm(6).[3]

Important points regarding Grover's Search

1. If the initial probability of getting a desired outcome is $sin^2(\theta)$, then this probability after j iterations of Grover's algorithm is $sin^2((2j+1)\theta)$.
2. If j drawn randomly from $\{0, ..., M-1\}$, then if $M \geq M_0 := \frac{1}{sin(2\theta)}$, then average probability is atleast 1/4.
3. But if no lower bound known on θ, then we use exponential searching.
4. For step i, $M = c^i$, for some $c \in (1, 2)$. After logarithmic number of such steps, $M \geq M_0$ with high probability.

Problem Setup

The training examples are sampled uniformly from the training set $\{v_1, v_2, \cdots, v_N\}$, with corresponding labels being $\{y_1, y_2, \cdots, y_N\}$. Define $\mathbf{v}_i = (v_i, y_i)$. This sampling procedure is different from the classical perceptron learning where training examples are provided one by one as input. In our case of quantum perceptron, a training example may even be picked multiple times. We can access the training examples by means of an oracle U provided to us.

$$U |\mathbf{u}_j\rangle |\mathbf{v}_0\rangle = |\mathbf{u}_j\rangle |\mathbf{v}_j\rangle$$

where $|\mathbf{v}_0\rangle$ is a blank register.

Let $f_{\mathbf{w}}(v_i, y_i)$ be a function which is 1 if and only if the perceptron $\mathbf{w}$ misclassifies the training vector v_i. This function is basically the quantum implementation of the

[3] Details of the Grover's Search algorithm have been skipped from the report as this algorithm was later discussed in the class itself. So, only some main results regarding the same have been reported.

classification algorithm being used by the perceptron. Define:

$$\mathcal{F}_{\mathbf{w}} |\mathbf{v}_j\rangle = (-1)^{f_{\mathbf{w}}(v_i, y_i)} |\mathbf{v}_j\rangle$$

$\mathcal{F}_{\mathbf{w}}$ is easily implementable through a multiply controlled phase gate. Now, define

$$F_{\mathbf{w}} = U^{\dagger}(1 \otimes \mathcal{F}_{\mathbf{w}})U$$

Now, we perform the Grover's search algorithm taking the Grover's iterate as

$$U_{\text{grover}} = (2 |\mathbf{s}\rangle \langle \mathbf{s}| - 1) F_{\mathbf{w}}$$

where $|\mathbf{s}\rangle = \frac{1}{N} \sum_{i=1}^{N} |\mathbf{u}_i\rangle$

Algorithm

for $k = 1, ..., \lceil \log_{3/4} \epsilon\gamma^2 \rceil$ **do**
  for $j = 1, ..., \lceil \log_c(1/sin(2sin^{-1}(1/\sqrt{N}))) \rceil$ **do**
    Draw m uniformly from $\{0, ..., \lceil c^j \rceil\}$;
    Prepare quantum state $|\Psi\rangle = |\mathbf{s}\rangle\Phi_0$;
    $\Psi = (U_{\text{grover}})^m \Psi$;
    Measure Ψ. Say, outcome $= u_a$;
    $(\mathbf{v}, y) = U_c(a)$;
    if $f_{\mathbf{w}}(\mathbf{v}, y) = 1$ **then**
      Return $\mathbf{w}' \leftarrow \mathbf{w} + y\mathbf{v}$.;
    end
  end
end
Return $\mathbf{w}'$;

Algorithm 2: Online Quantum perceptron training

Analysis

Lemma 12. Given the ability to uniformly sample from training vectors, number of queries to f_w classically required to find a misclassified vector, or to conclude perfect classification, with probablity $1 - \epsilon\gamma^2$, is atmost $O(N \log(1/\epsilon\gamma^2))$.

Proof. Given that we draw k samples, probability that all of them fail to detect a misclassification(if present) is atmost $(1 - \frac{1}{N})^k \leq \exp(-k/N)$.
If we want this error to be atmost δ, then

$$k_{min} = \lceil N \log(1/\delta) \rceil$$

So, if we draw $N\lceil \log(1/\epsilon\gamma^2) \rceil$ samples, then:

- If there exists a mistake, then probablity that it goes undetected is atmost $\epsilon\gamma^2$, i.e. algorithm finds a mistake with probability atleast $1 - \epsilon\gamma^2$.
- If no mistake occurs, then algorithm concludes w as the correct hyperplane, with probability atleast $1 - \epsilon\gamma^2$.

□

Lemma 13. Assuming training vectors are unit vectors, and are drawn from two classes separated by margin of γ in feature space, the algorithm will either update percptron weights, or conclude that the current w is a separating hyperplane between the two classes, using atmost $O(\sqrt{N} \log(1/\epsilon\gamma^2))$ queries to F_w, with probabilty of failure atmost $\epsilon\gamma^2$.

Proof. From Grover's search algorithm, starting with initial success probability $sin^2(\theta)$, after m updates the probability becomes $sin^2((2m + 1)\theta)$. So, the inner loop performs a perceptron update with probability $sin^2((2m + 1)\theta)$, using m queries to $F_{\mathbf{w}}$.

$$sin^2(\theta) \geq 1/N \implies \theta \geq sin^{-1}(\sqrt{1/N}) \in \Omega(1/\sqrt{N})$$

If misclassified vector exists, then the middle loop repeated at least $\log_c M_0$ times, so final iteration has $m \geq M_0$, and thus middle loop updates perceptron weights with probability atleast $1/4$. Given that a misclassification exists, the loop fails to find it with probability atmost $3/4$.
Each iteration of the middle loop requires total number of queries atmost proporational to $\sum_{i=1}^{\lceil \log_c M_0 \rceil} \lceil c^i \rceil \leq \frac{c^{\lceil \log_c M_0 \rceil + 1}}{c-1} + \lceil \log_c M_0 \rceil \in O(M_0)$. Now, we have

$$M_0 = 1/sin(2\theta) \leq \sqrt{N}$$

$\implies O(\sqrt{N})$ queries to F_w are required per inner loop iteration.
If middle loop repeated k times, then probability of failing to find that element all k times is at most $\left(\frac{3}{4}\right)^k$. So, to upper bound the error probability by δ: $k_{min} = \lceil \log_{3/4} \delta \rceil$ number of iterations of the inner loop are required.
$\implies$ So, total number of queries needed is $O(\sqrt{N} \log_{3/4} \delta) \in O(\sqrt{N} \log(1/\delta)$.
Hence, the lemma holds with $\delta = \epsilon\gamma^2$. □

Theorem 14. (7) Given a training set separated by a margin of γ in feature space, the number of queries needed to infer a perceptron model w such that $\mathbb{P}(\exists j : f_w(\phi_j) = 1) \leq \epsilon$ using quantum computer is N_q such that

$$\Omega(\sqrt{N}) \ni N_q \in O\left(\frac{\sqrt{N}}{\gamma^2} \log\left(\frac{1}{\epsilon\gamma^2}\right)\right)$$

and using a classical computer is N_c such that

$$\Omega(N) \ni N_c \in O\left(\frac{N}{\gamma^2} \log\left(\frac{1}{\epsilon\gamma^2}\right)\right)$$

Conclusion

Through means of this project, we have presented quantum algorithms for perceptron learning, nearest

neighbor learning through inner products and K-nearest neighbor learning through Hamming Distance. All these methods provide significant reductions in the query complexity as compared to the corresponding classical counterparts.
These algorithms surely are a step towards in world where machine learning techniques blend completely with the faster quantum modules. Future extensions from this project are studies into Quantum Neural Networks and search for exponential speed-ups over the classical cases.

References

[1] N. Wiebe, A. Kapoor, and K. M. Svore, "Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning," *Quantum Info. Comput.*, vol. 15, pp. 316–356, Mar. 2015.

[2] Y. Ruan, X. Xue, H. Liu, J. Tan, and X. Li, "Quantum algorithm for k-nearest neighbors classification based on the metric of hamming distance," *International Journal of Theoretical Physics*, vol. 56, pp. 3496–3507, Nov 2017.

[3] G. Brassard, U. of Waterloo. Department of Combinatorics, Optimization, and U. of Waterloo. Faculty of Mathematics, *Quantum Amplitude Amplification and Estimation*. Faculty of Mathematics, University of Waterloo, 2000.

[4] C. Durr and P. Hoyer, "A Quantum Algorithm for Finding the Minimum," Jan. 1999.

[5] A. B. J. Novikoff, "On convergence proofs on perceptrons," *Proceedings of the Symposium on the Mathematical Theory of Automata*, vol. 12, pp. 615–622, 1962.

[6] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, (New York, NY, USA), pp. 212–219, ACM, 1996.

[7] A. Kapoor, N. Wiebe, and K. M. Svore, "Quantum perceptron models," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3999–4007, 2016.