University of Minnesota

Department of

Electrical and Computer Engineering

EE 4982V - Senior Honors Project II, Spring 2021

Quantum Perceptron Models and Applications to Traditional Machine Learning Problems

Author:

Andrew LaFortune

Advisor:

Keshab K. Parhi Ph.D

May 8, 2021

1 Abstract

The initial goal of this project was to create a model of the classical perceptron training model for binary classification in machine learning using a quantum algorithm. Ultimately, due to the limitations of open-source quantum computer simulations and algorithms for quantum arithmetic, data structures, etc, this was not realized. Instead, this paper serves as an overview of key concepts in quantum computation and algorithms. The key components explored are: Quantum Dictionaries, Grover's Search Algorithm, and construction of Oracle operators for Grover's Search. The assembly of different quantum circuit components to create two different models for perceptron learning and a classical implementation of a perceptron are also included, but full quantum circuit implementations were not realizable given the timeframe of the project and availability of algorithmic tools such as arrays and multiplication operators.

Contents

List of Figures

2 Introduction

Machine learning and quantum computing are both right at the cutting edge of Computer Science research, but have not yet been joined together effectively. A major question in quantum computer research is whether or not these computers will actually be able to outperform the so-called "classical" computers that are widely available today. While theoretical algorithms and improvements help the case for quantum computing, there are also benefits to be had from actually trying to implement and apply the different building blocks of algorithms that researchers have developed to problems that are fathomable on the classical computers of today.

So, the purpose of this project is to explore the feasibility of two proposed algorithms for a machine learning classification algorithm known as a "perceptron". Given a set of linearly separable data, the perceptron algorithm starts with an initial guess for a "weight" vector in an attempt to find a plane which will separate the data. By checking the known classification of a sample against the classification produced by the perceptron's weight vector, small shifts to the weight vector can be made until a satisfactory separating plane is found. The Quantum Perceptron Models explored in this paper rely on the quantum Grover's Search [3] algorithm to reduce the number of sample vectors that need to be checked for each update to the weight vector.

The end goal of the two fully functional Quantum Perceptron Models proposed by Wiebe, Kapoor, and Svore [10] was ultimately not achieved. However, the necessary pieces were brought together. The key limitation was in the algorithms that make up the smaller pieces, and the main benefit of this work is a proof of the difficulties in using quantum algorithms at any scale larger than toy problems and in working with systems of many qubits.

3 Background

3.1 Prior Work and Inspiration

3.1.1 Quantum Computing

Quantum vs. Classical Computing

As the limitations of classical computers are being approached, more research is going into the feasibility of quantum computers for use in daily life. The Church-Turing Thesis [4] proposes that any problem that could be solved with a classical computer could also be solved with a quantum computer, and vice versa. Problems exist that are theoretically solvable by classical computers, but not in any perceivable amount of time. This is where the concept of "quantum advantage" [7] provides motivation for continued exploration of quantum algorithms. The idea of "quantum advantage" is the ability of quantum computers to perform the same computations with an exponential decrease in time complexity.

The main difference between quantum and classical computers is the representation of data. In a classical computer, a bit is the base unit of data and multiple bits can be used to represent more complex data. Each bit may take a value of 0 or 1. In quantum computing, a qubit is the base unit of data. While qubits also take values of 0 or 1 whenever they are measured, they may also take a value in between when they are in a state of superposition. Figure 1 shows the Dirac Notation [11] for both classical bits and quantum qubits. In superposition, a qubit is represented as a linear combination of the two possible states multiplied by weights α and β with a combined magnitude of 1 so that the quantum state is always represented by a unit vector on the Bloch Sphere [6] shown in Figure 2. Whenever a qubit is observed or measured, it collapses to either a 0 or 1 state. Because of this quantum circuits are executed many times to determine the probability that a qubit will take on a certain value.

Quantum Decoherence is one of the main roadblocks to efficient quantum computing. As

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \quad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad |\psi\rangle := \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}; \quad |\alpha|^2 + |\beta|^2 = 1.$$

Figure 1: Dirac notation of a classical bit (left) and a quantum bit (right).

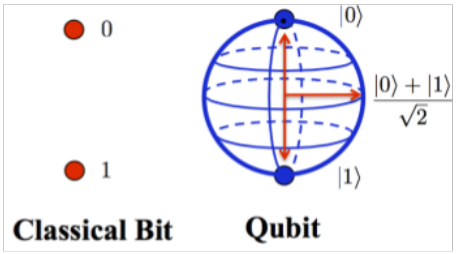


Figure 2: Visual representation of classical bit, and qubit on Bloch Sphere.

mentioned above, a qubit's state can be represented as a linear combination of two different states. The weight of each state acts as a probability that it will exist in that state. Relying on probabilities rather than deterministic states for computation inherently creates room for error. In addition to the error in a probabilistic model, error is introduced each time a qubit is measured because a quantum system is very delicately balanced, and measurement reduces the total coherence of the system. However, measurement of qubits is necessary to perform any meaningful calculations at this time, so the error created by the phenomenon is unavoidable.

Current Quantum Computers

The 2018 report from the Computing Community Consortium (CCC) [8] covers a variety of current research areas. One primary message that comes across is that a significant gap exists between quantum algorithms like Grover's Search and Shor's Factoring and the machines that exist to work with them. This gap can be boiled down to two issues that must be balanced against each other: size of systems and error probability. The quantum computers

that exist now are known as "Noisy Intermediate Scale Quantum Computers" (NISQ), and the name emphasizes the point that they are not large enough to do any major computing, and a significant amount of noise exists in systems of qubits which makes the computations that are performable relatively uncertain. Figure 3 from the CCC report outlines the gap between the systems of today and what is presumably the point of "quantum supremacy". That is, the point where theoretical quantum computers and their benefits will be realized in scalable physical systems which are able to outperform classical computers in certain problems. Google claimed to have reached this landmark in 2019 with a 53 qubit computer, but that claim has been debated heavily by their primary competitor, IBM, who boast a stable 27 qubit computer and simulation of up to a 5000 qubit system.

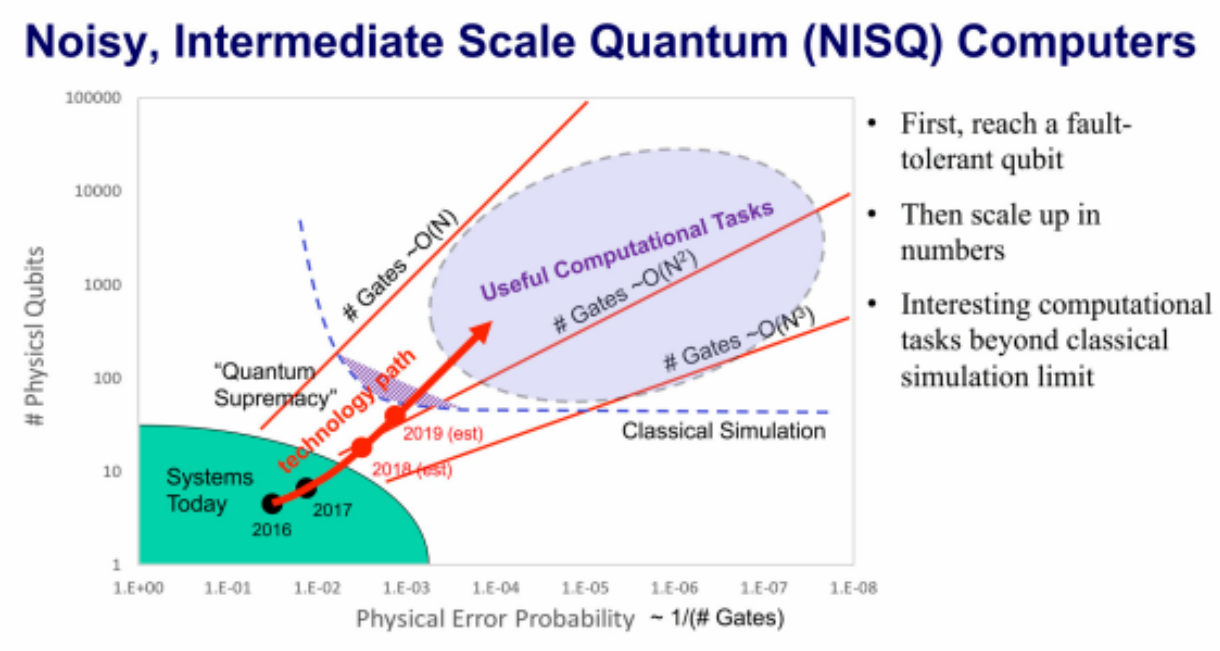


Figure 3: The current limitations of NISQ Computers and error/scale tradeoff.

Beyond the physical limitations of quantum computers, there is still much to be learned in terms of efficient algorithms. The same CCC report cites machine learning as a key area where quantum computers could succeed. However, that success is dependent on the ability of quantum computers to efficiently access classical data. A method of doing so efficiently and without errors does not yet exist, as will become evident later in this paper. Two other

major hurdles and areas of ongoing research are in error correction and state preparation. These are less relevant to this paper, since most of the experimentation is done with a classical simulation of a quantum computer.

Building Blocks of Quantum Computing

At this juncture, quantum algorithms are generally implemented as small-scale quantum circuits, which are in turn made up of quantum gates. These circuits are very similar to classical circuits which take in some series of input states, apply a series of operations via gates, and produce some output. There are, however, a few key differences to note. The first is that classical circuits are limited to the operators NOT, OR, AND, NAND, NOR, and XOR, but since a quantum bit can exist in an infinite number of states, there are also an infinite number of quantum gates that can be created. There still exists a common set of quantum gates, some of which are summarized in Figure 4 and in [14]. A few key ones to note are the Hadamard gate (H), which is responsible for superposition. When applied to a single qubit, the qubit enters an equal superposition between the states $|0\rangle$ and $|1\rangle$, and is often used on each variable bit at the beginning of a circuit to create a "uniform superposition" of all input states. The Pauli-X Gate (X) is exactly the same as a classical NOT gate, and in fact all of the classical operators can be expressed as quantum gates or combinations of quantum gates.

Another important aspect of quantum circuits is that they can be represented as square matrices as was just demonstrated. The dimension of the matrices is NXN, where $N = 2^n$ for n qubits. Furthermore, the square matrix representing a series of quantum operations or a quantum circuit must be *unitary*. This means that it is invertible, and provides a great amount of utility in circuits being reversible. One quirk of unitarity and invertibility is a concept known as "phase kickback", which allows for an unknown input to be discovered

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad C(U) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}$$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad R_y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

Figure 4: Matrix representations of common quantum gates.

from a known output. Figure 5 shows an example of a binary function which will set an output qubit to $|0\rangle$ or $|1\rangle$ based on the input x. Since the circuit must be invertible, it is necessary to be able to recover the input state from the output. So, if the output state is initialized to the "ket minus" state: $|-\rangle = \frac{|0\rangle + \frac{\pi}{2}||1\rangle}{\sqrt{2}}$. The actual mechanics of phase kickback are explained further in [9], but initializing the output qubit to the ket minus state allows inputs which would return a 1 in our initial function to be identified, and a negative phase shift is applied to those input states to "mark" them.

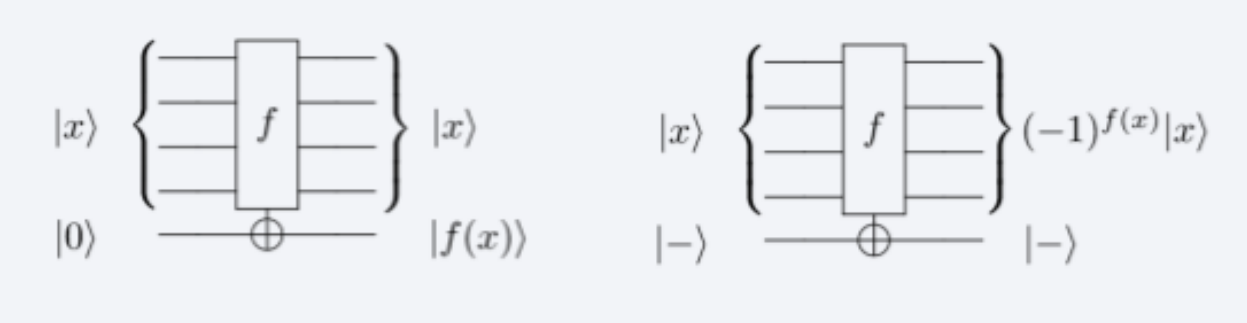


Figure 5: Converting binary operator (left) to an oracle (right) via phase kickback.

Lastly, the Kronecker Product [13] is responsible for applying single and multiple qubit quantum gates to larger circuits. The basic formula is shown in Figure 6. This matrix representation of complex multi-qubit systems allows the introduction of *controlled* gates which consist of one or more target and control bits. One example is the $C(U)$ gate in

Figure 4, which will only perform the matrix operation U_{ij} on its target qubit if the control qubit is in a $|1\rangle$ state. This allows for conditional operations in otherwise simple circuits as well as the ability to represent multi-qubit circuits or operators as gates to be applied to other larger circuits.

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

Figure 6: Kronecker Product Formula.

3.1.2 The Perceptron

As mentioned in the introduction, the perceptron model is designed to find a hyperplane that will perfectly separate two classes of samples marked as {+1,-1}, assuming one exists. The algorithm begins with some arbitrary initialization of a weight vector w, often as the zero vector or a vector of random floats with the same dimension as the training samples x. The bias term b is not entirely necessary as it can be encoded into the w vector, so it won't be shown in the implementation later for simplicity. A learning rate η is also set so that each adjustment to the weight vector is not too big or too small. Lastly, the stopping criterion for the algorithm must be defined. This could be a certain number of iterations through the entire training set called "epochs", or a certain lower bound of samples misclassified or a percentage of the training set misclassified. Once these conditions are set, the algorithm starts looking at each training sample x and its pre-assigned class y. The decision function $y(w \cdot x) \geq 0$ determines whether a sample has been misclassified.

This can be seen graphically in Figure 7 where the dot product between the vector from 0 to x and the weight vector w is a measure of the similarity between the two. If w is more than 180° away from x, the dot product will be negative, and the perceptron would predict

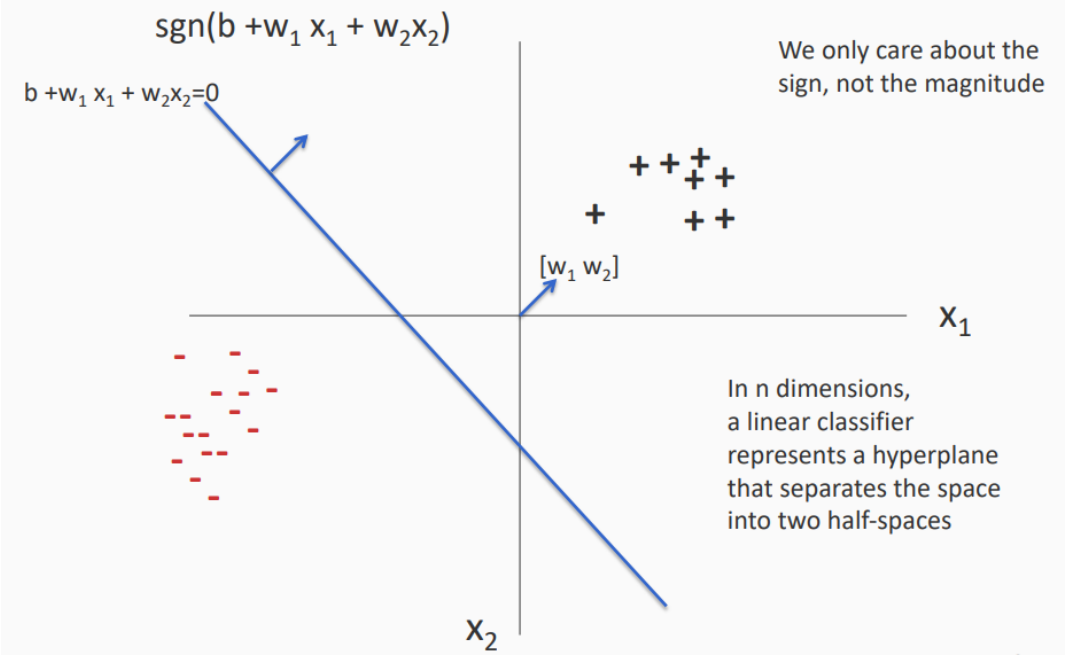


Figure 7: Geometric vizualization of decision boundary.

the class of x to be -1. This is where the y in the decision function comes in. If y actually is -1, the sign will reverse again, and the result will be positive indicating the sample has been correctly classified, so no adjustment is needed. However if the real class is +1, the decision function will be negative and an adjustment in the direction of x will be made to the decision boundary. [12] provides more detail on the history of the perceptron, and is the source for Figure 7. Algorithm 1 shows the steps described above in a concise fashion.

Algorithm 1: Perceptron Learning Algorithm

Result: w s.t. $y(w \cdot x) \geq 0 \; \forall \; x, y$
initialize weight vector w;
set learning rate η ;
while *not break condition* **do**
  for *x,y in training set* **do**
    if $y(w \cdot x) < 0$ **then**
      $w = w - y \cdot x \cdot \eta$;
    end
  end
end
return w

3.1.3 Quantum Perceptron Models

Now that a basic understanding of general quantum circuits and the basic operations within them has been established, a surface-level explanation of the two proposed perceptron models will be provided. The basis for these models is in "Quantum Perceptron Models" [10]. Both algorithms take the weight adjustment algorithm described above and reformulate it as a search problem, which can be exponentially sped up through the use of Grover's Search algorithm. Grover's Search makes use of superposition and an oracle operator to effectively represent all possible input states at once and mark the states satisfying some criteria using a phase shift operation. Then, through "amplitude amplification", the probability of finding those shifted states upon measurement is increased significantly. Through these operations, the time needed to find any target state is improved from the classical search complexity $O(N)$ to $O(\sqrt{N})$ The steps of that algorithm will be described in more detail in the "Methodology" section.

Online Perceptron Model

The first algorithm seeks to improve training efficiency by reducing the number of total samples that need to be passed through the algorithm. Doing so relies on the quadratic speedup of a quantum search versus a classical search and the ability to represent the entire space of training samples/vectors as a single sample/vector in a state of superposition of all training samples. This saves quite a bit of memory compared to a classical computer because an exponentially large quantum state vector representing the total set of N training vectors does not take up an exponentially large amount of memory space. Instead, only enough space to represent one address vector and one input state vector is needed. The benefit of quantum search in this case is that the entire space of vectors can be searched and tested for proper classification using an oracle. Since the weight used to shift the classification

boundary only needs to be updated in the event of a misclassification, only misclassified training vectors should be accessed from the data structure. The classical perceptron model accesses every single vector and checks if it is misclassified individually, but this quantum perceptron model can determine which vectors are misclassified without accessing each one in $O(\sqrt{N})$ steps, and only access those that are misclassified to make an update. Because the speedup of the quantum search algorithm is quadratic, a quadratic reduction in the total time complexity of the algorithm can be seen. A simplified version of Algorithm 1 from [10] is shown in 2.

Algorithm 2: Online Perceptron Model

Result: w s.t. $y(w \cdot x) \geq 0 \; \forall \; x, y$
initialize weight vector w;
while *not break condition* **do**
    initialize superposition state Ψ of all training vectors;
    Ψ = Grover's Search(Ψ) ;
    measure Ψ to find target vector $x \in x_{misclassified}$;
    if $y(w \cdot x) < 0$ **then**
        $w = w - y \cdot x \cdot \eta$;
    end
end
return w

One major drawback of this algorithm that can be seen is the need to re-initialize the state of superposition after each update. This is because upon measuring a quantum circuit the state collapses and all probabilistic values of qubits are lost. To make matters worse, as mentioned in 3.1.1, efficient state initialization is one of the major hurdles still facing quantum algorithms, so the only real attainable speedup here is in the reduction of how many database accesses are needed.

Quantum Version Space Perceptron

The second algorithm also relies on the quadratic advantage of quantum search but instead

uses the separation between each class in the training data, γ, as a measure of the probability that a randomly selected hyperplane within the vector/sample space would perfectly separate the data. Using this formulation with a classical search algorithm and a separation between samples of γ, otherwise known as the resolution of the weight vector needed, the expected number of randomly selected weight vectors tested is $O(\frac{1}{\gamma})$. Alternatively, using a quantum implementation where all candidate weight vectors are represented simultaneously via superposition, the expected number of Grover's Search operations to find a hyperplane that perfectly classifies the data will scale as $O(\frac{1}{\sqrt{\gamma}})$. The simplified version of Algorithm 2 from [10] is shown in 3.

Algorithm 3: Quantum Version Space Perceptron Model

Result: w s.t. $y(w \cdot x) \geq 0 \; \forall \; x, y$
initialize superposition state Ψ of possible weight vectors;
Ψ = Grover's Search(Ψ) ;
measure Ψ to find target vector $w \in w_{separating}$;
if $y(w \cdot x) \geq 0 \; \forall \; x, y$ **then**
  return w
end
return $w = 0$

One major difference between the algorithms to note is that the Online Perceptron Model will return a w which is a product of the training samples even if the training data is not linearly separable. On the other hand, the Quantum Version Space Perceptron will return $w = 0$ if there is not a satisfying hyperplane. Both of these models are very broad and will take some time to fully realize as quantum circuits, but the basic concept of quadratic speedup via Grover's Search is the most important piece of each model.

4 Methodology

4.1 Proposed Online Perceptron Circuit

The circuit shown in figure 8 provides a high-level overview of the different components needed. From top to bottom, it begins with the initialization of a quantum register to hold each bit of the binary representation of a training sample's index, a quantum register to hold the binary representation of each training sample, a quantum register for ancillary bits used in computations, a quantum weight register, and two classical registers corresponding to the quantum index and value registers. Each bit in the index and value register has a Hadamard (H) gate applied to it to create a state of equal superposition. The circuit blocks from left to right are defined in the next paragraph. First, a Quantum Dictionary [2] which takes the superposition of indexes as an input and outputs an equal superposition of the indexes matched with their values. Then, the Grover's Search circuit works with the value and weight registers to mark and amplify training samples that are misclassified. Finally, measurement operations are applied to the index and value registers which should now have a high probability of being in a state corresponding to a misclassified training sample. The index measurement is not entirely necessary, but since there may be a number of indices that do not correspond to a value in the training set, it can be useful to have the index for error checking. This entire circuit corresponds to one iteration of the Online Quantum Perceptron Model and must be re-initialized until the stopping criteria for the algorithm is reached. The components of each block in the circuit are outlined in the following sections.

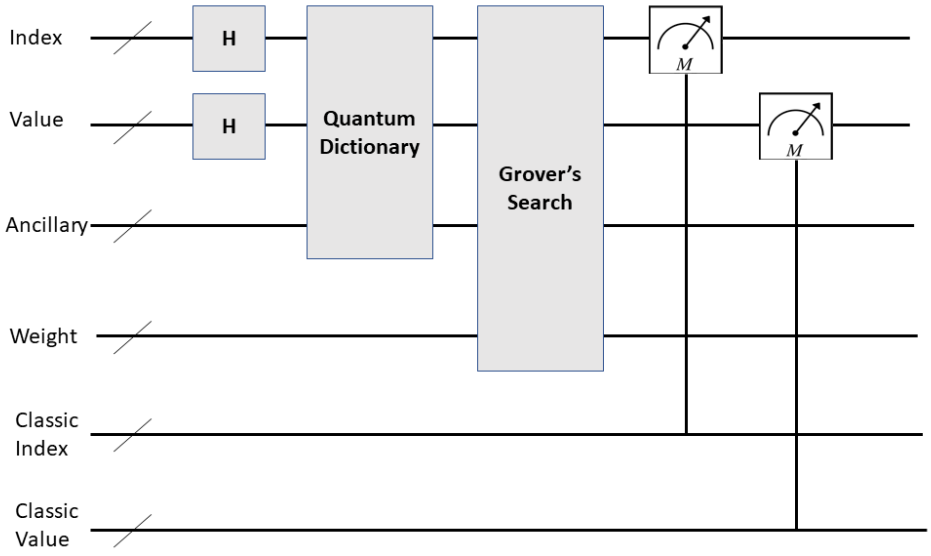


Figure 8: Full Perceptron Circuit.

4.2 Implementation Attempt

4.2.1 Data Representation and Generation

Quantum circuits are inherently noisy and require nearly every piece of logic to be built into them manually since there are not yet high level programming languages like Python to handle arithmetic and data structures. For this reason, the attempted problem was relatively straightforward for a classical perceptron to solve and the dataset is not particularly interesting. Due to the limits of how many qubits can be simulated in a single circuit by the IBM Qiskit Quantum Composer environment used, the data was intentionally limited to datasets which could be represented by less than 30 qubits. This led to the dataset shown in Figure 9 which consisted of all points in the xy plane which could be represented by a vector of the form: $\phi = [x_2, x_1, x_0, y_2, y_1, y_0, c]^T$ where x and y are Two's Complement [5] integers and $c \in \{+1, -1\}$ corresponds to the class of a sample. The weight vector w was randomly selected as one of the (x, y) points in this space, and the c component of each

sample was set according to the decision function $sign(w \cdot \phi)$. To keep the vector in binary, c was then set to $frac(1+c)2$ which maps -1 to 0 and 1 to 1. Binary vectors were chosen because they can be encoded in both quantum and classical systems easily. Note that there is no bias term included in the weight vector since it goes through the origin $(0, 0)$. This was done to reduce the complexity of the decision function and reduce the number of total qubits needed for the system. The weight vector in this case is $w = (-2, -1)$, but the slope of the decision boundary is actually the inverse of the weight vector since the decision function is the similarity between the weight vector and a given sample as was explained in 3.1.2.

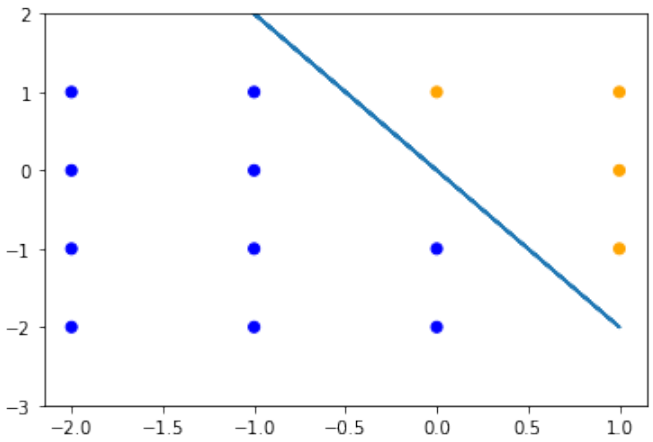


Figure 9: Generated Dataset.

4.2.2 Classic Perceptron

Given the dataset, a training set made up of 80% of the samples was used to train a classical implementation of the online perceptron in Python. The weight vector was initialized to $(0, 0)$, the step size for training was set to 1 for simplicity since there would not be an easy way to encode floating point numbers into the quantum circuit. Finally, the stopping criteria was defined as 500 epochs or iterations through the entire dataset. Realistically, 500 epochs is far more than was needed for this small problem, but the initial goal was to generalize this project to much larger and more complex datasets. The result of the training is shown in Figure 10 below, where the recovered decision boundary is exactly the same as the ground

truth from before. Training samples are colored green and red, and the test samples are colored blue and orange. Since all samples were classified correctly, the accuracy for this implementation was 100%.

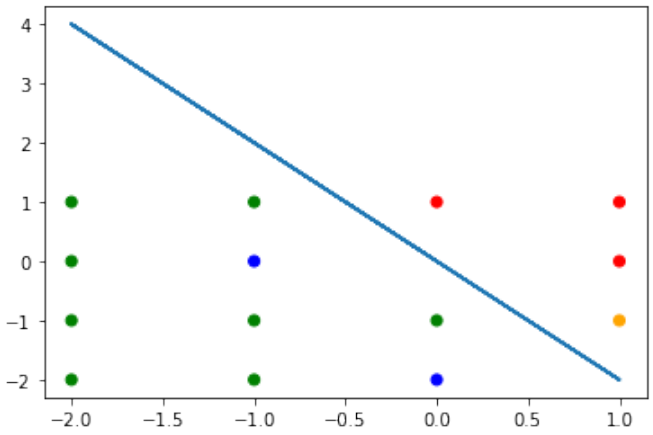


Figure 10: Recovered Classification Boundary with Train and Test Samples.

4.2.3 Quantum Dictionary

This portion of the Quantum Implementation turned out to be the most challenging and ultimately caused the final implementation to fail. Classical computers come equipped with Random Access Memory and hardware to store values at specific memory addresses to be quickly accessed later. In the quantum realm, no such luxuries exist. The proposed method for a quantum database relied on encoding an integer value as a geometric sequence of powers of a specific root of unity for a pre-defined angle $\theta = \frac{2\pi}{M}$ where $M = 2^m$ is the number of possible output states for the m qubits in the values register. Once this geometric sequence has been encoded in the quantum values register using a series of rotation gates controlled by the index qubits, an inverse Fourier transform is applied to estimate the nearest integer root of unity to the sum of the geometric sequence of rotations applied to the ancilla qubit. Figure 11 shows how a 3 bit value register's 8 possible outputs can be mapped to a unit circle given a base angle, and Figure 12 shows the realization of the quantum dictionary as a Quantum circuit along with the resulting measurement probabilities in Figure 13. Notice

that although the 4 possible input states shown as the rightmost two bits on the histogram began in equal superposition due to the H gate superposition, the output measurements are not exactly equal. This is because the inverse Fourier transform is an *estimation* of the root of unity, so there is some uncertainty and the probabilities of measuring the encoded states is not perfect. The mechanics of the circuit below and the theory behind it are explained further in [2].

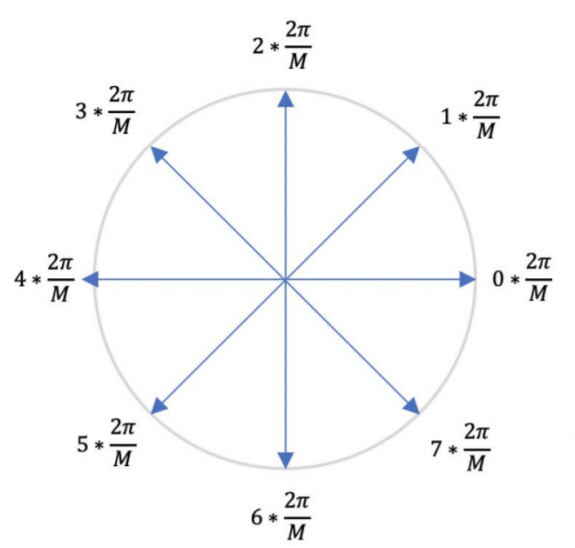


Figure 11: Encoding of a 3 Qubit value register as rotations about the unit circle.

Figure 12: 3 Qubit value register and 2 Qubit index register with the index 0 mapped to state $2 = |010\rangle$.

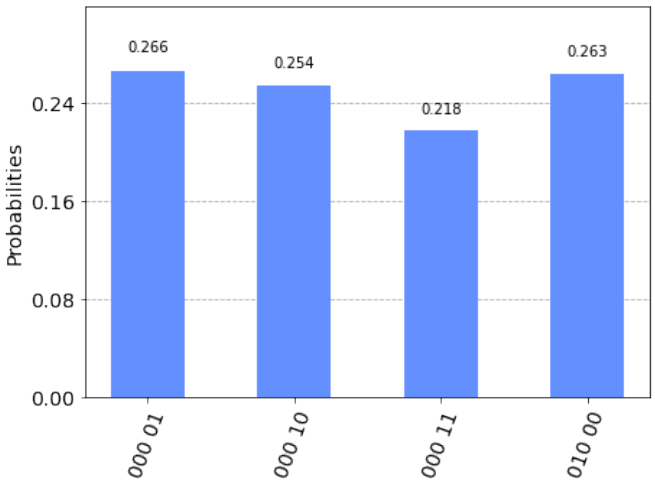
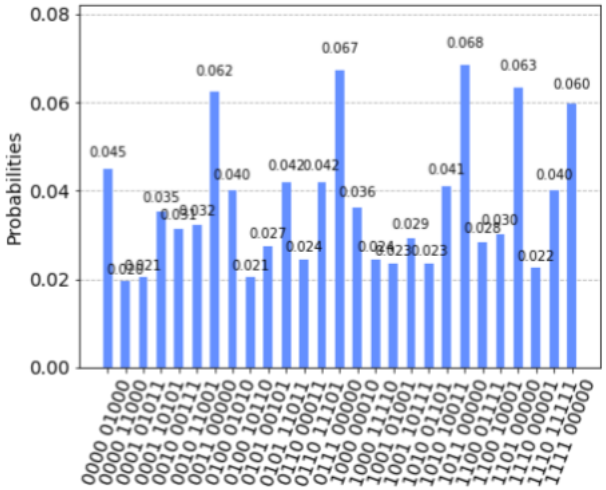


Figure 13: Results of measuring the circuit in Figure 12.

The attempted circuit for the main training data was too large to be shown in a comprehensible way in this paper, but the measurement results of the circuit reveal the trouble with using the inverse Fourier transform to estimate phase angle. Given the main circuit with 10 training samples, 4 qubits to represent their indexes, and 5 qubits to represent their binary values, the expected probability of measuring each index-value pair from an initial state of superposition should be $\frac{1}{N}$ where $N = 2^n$ is the total number of indexes that could be represented. However, the graph and table in Figure 14 show that there are many values not included in the dataset that are possible values for the circuit to return (The table only shows probabilities of encoded values for clarity). Additionally, the probability of measuring one of the encoded values is at most 72% of the expected probability, and at worst less than 50% of the expected probability. For this reason, the rest of the circuit could not be implemented as a cohesive system, but the individual pieces could still be explored.

4.2.4 Grover's Search

Grover's Search is the main workhorse of both quantum perceptron models proposed. Abandoning the idea of a cohesive perceptron circuit for the time being, this segment assumes that an equal superposition of all index-value states is received at the input (Figure 15). Then,



Expected Probability Per Sample = 0.0625

	Index	Value	Probability	In Dict?
0	0000	01000	0.045	True
2	0001	10101	0.035	True
5	0010	00111	0.031	True
7	0100	01010	0.040	True
9	0101	11011	0.042	True
11	0110	11101	0.042	True
14	1000	00010	0.036	True
16	1001	10111	0.029	True
18	1010	10011	0.041	True
21	1100	10001	0.030	True
24	1110	11111	0.040	True

Figure 14: Results of measuring the full database circuit.

an oracle operator can verify if a sample is part of the set of items being searched for, and it can be applied to all samples at once when they exist in superposition. If an item is part of the search set, its phase is flipped so that the quantum state representing it faces away from all the samples not in the set (Figure 16). Then, a "diffusion" operation is used to reflect the state of all possible quantum states so that after both operations have been performed, the amplitude of all valid output states is greater than invalid ones and the probability that the circuit will be in a valid state upon measurement is higher (Figure 17). This can be shown geometrically in two dimensions where the state $|w\rangle$ represents all samples in the search set (misclassified training vectors), $|s\rangle$ represents the set of all possible training vectors in an equal superposition, and $|s\prime\rangle$ is the state $|s\rangle$ with the vectors representing $|w\rangle$ removed from it so that $|s\prime\rangle$ is orthogonal to $|w\rangle$. The results of performing one full iteration of Grover's Search are shown in Figures 15, 16, 17, and of course multiple iterations can be done to maximize the amplitude of $|w\rangle$ until it is almost certain that a valid answer will be found by the search operation.

The graphics show just one iteration of the Grover's Search algorithm, but for systems with a large number of possible states the Oracle and Diffusion operators are applied $O(\sqrt{N})$ times

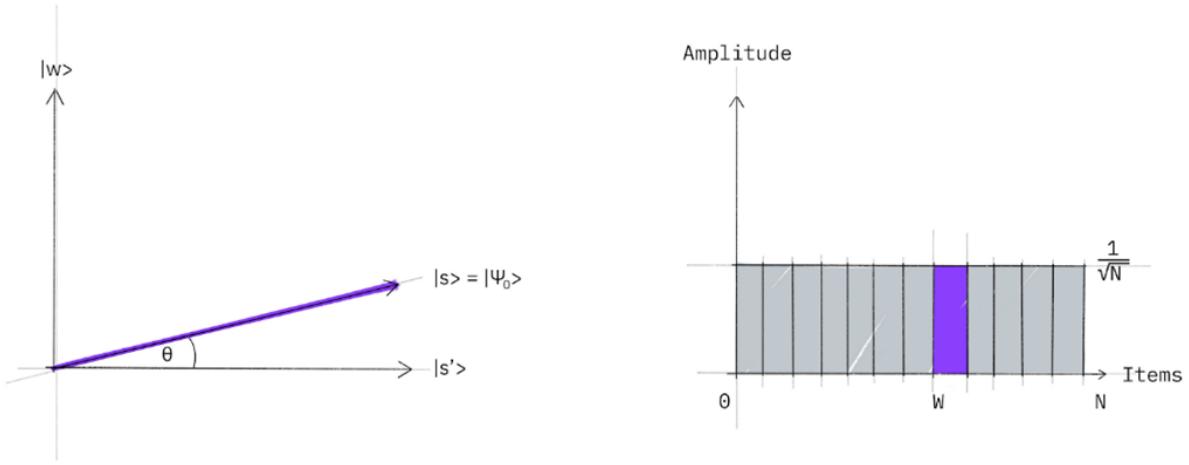


Figure 15: Initial superposition state in Grover's Algorithm[3].

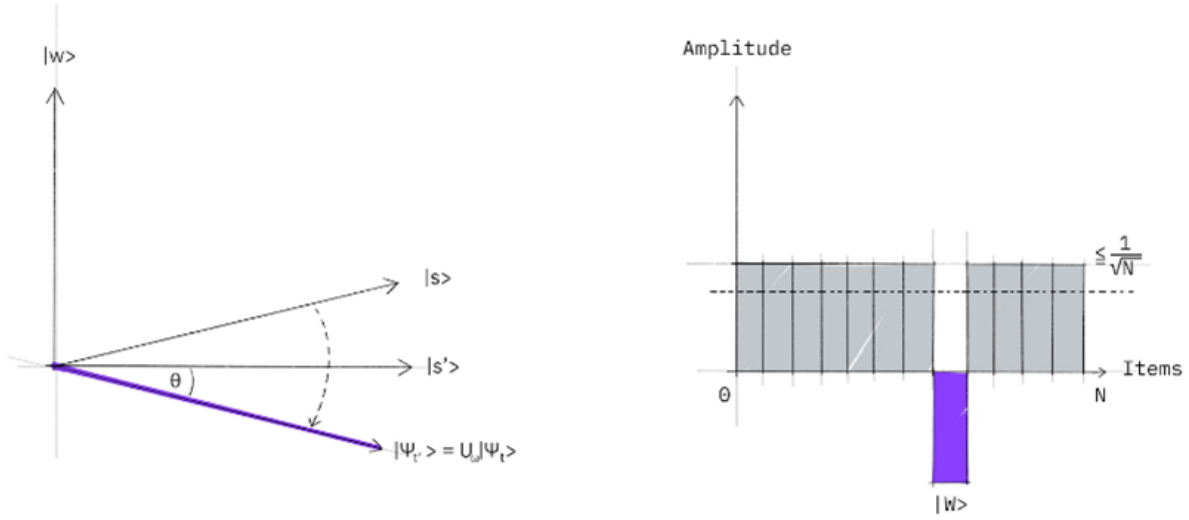


Figure 16: Oracle operator applying phase shift to target w[3].

to get a sufficiently confident measurement of a target value. The last piece of Grover's Search relevant to the Online Perceptron model is the Oracle operator. Quantum arithmetic is another underdeveloped area algorithmically, and requires more qubits than were available to be implemented as proposed in [1], so a verbal explanation will have to suffice. Using the concept of phase kickback shown in 5, the definition of an oracle is as simple as defining the classical perceptron decision function $c(w \cdot x) > 0)$ as a binary function. To do so, the Two's

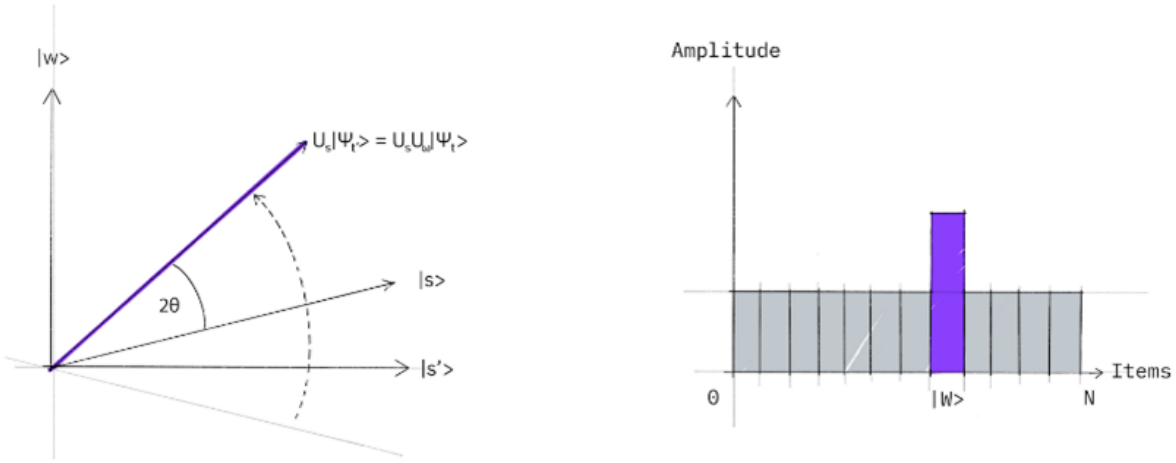


Figure 17: Amplitude amplification of target w [3].

Complement notation is again useful. Using the methods for addition and multiplication in [1], Algorithm 4 outlines the steps needed to produce an output where the sign bit of the dot product of x and w is the output. Then, simply initializing the output qubit to the ket minus ($|-\rangle$) state converts the binary decision function into an oracle operator for the superposition of all possible input states. A detailed explanation of the Grover's Search and the Oracle construction can be found in [3].

Algorithm 4: Perceptron Oracle

Result: Binary value answering: $c(w \cdot x) > 0)$
given weight vector w, sample x, and class label c;
split w and x into their x and y components;
multiply individual components;
$s1 = w_x \cdot x_x$;
$s2 = w_y \cdot x_y$;
add the result of multiplications;
$sum = s1 + s2$;
return Most Significant Bit of sum;
return $MSB(sum)$

4.2.5 Verification and Update

Once the Grover's Search has finished, the circuit is measured collapsing into a single index-value state. From here the update is done classically just the same as in the Classical implementation. First, due to the uncertainty introduced by the Fourier Transform-based Quantum Dictionary, the index-value pair is searched for in the training set. If it is in the set, the weight vector is updated by $w = w + c \cdot x$. Then, the circuit is reinitialized with a new value for w and the process is repeated until the stopping criteria are achieved.

4.3 Proposed Quantum Version Space Perceptron Circuit

The Quantum Version Space Perceptron is very similar in the components it has to the Online Quantum Perceptron. The main difference is in the state that is created initially. There is no need to keep registers for the indexes or values of the database. Rather, they are all encoded into the oracle operator which can be expressed again as a boolean function with the output qubit initialized to the negative ket state. Logically, a function $f_{x_i}(w)$ is encoded as the decision function for a single training sample for an input weight vector w, which will return a 1 if the sample is misclassified. The full binary circuit takes the form $f_{x_1}(w) \vee f_{x_2}(w) \vee \cdots f_{x_n}(w)$, which will return 1 if any sample is misclassified by the current weight vector w, indicating that a negative phase shift should be applied to the state vector representing it. Once the oracle is defined, the process is the exact same as before where it is repeated a certain number of times until a measurement can be made confidently. The classical register holding the measurement is then unloaded and checked against every sample to verify its validity as a solution.

5 Results and Conclusions

While this project did not offer what was hoped for in terms of a working model for high dimensional perceptron training, it does provide a good overview of the building blocks of full quantum circuits and the limitations that come with them. To review, a relatively new method for interpreting classical data in a quantum circuit was explored and found to be too unstable for systems of many qubits. The Grover's Search algorithm was explained and tested on a small scale, and Oracle operations for both the Online Quantum Perceptron and Quantum Version Space Perceptron were outlined. For future work, the only pieces missing are an actual implementation of the quantum multiplication and addition which the Online Quantum Perceptron Oracle relies on as outlined in [1], and an alternative method to implement a Quantum Dictionary or database. The perceptron and machine learning is just one of many possible applications for quantum computing which should be explored long before the physical quantum computers available evolve beyond the current NISQ systems. However, in hindsight it may have been more beneficial to explore solutions to the low level problems like data representation and arithmetic which were ultimately the roadblock to a successful implementation of this project.

References

[1] J J Alvarez-Sanchez et al. *A quantum architecture for multiplying signed integers*. 2008. URL: https://iopscience.iop.org/article/10.1088/1742-6596/128/1/012013/pdf.

[2] Austin Gilliam et. al. *Foundational Patterns for Efficient Quantum Computing*. JP-Morgan Chase. URL: https://arxiv.org/pdf/1907.11513.pdf (visited on 26th Jan. 2021).

[3] Jupyter Book Community. *Learn Quantum Computation Using IBM Qiskit*. URL: https://qiskit.org/textbook/ch-algorithms/grover.html (visited on 5th July 2021).

[4] Jack Copeland. *The Church Turing Thesis*. The Stanford Encyclopedia of Philosophy. URL: https://plato.stanford.edu/entries/church-turing/ (visited on 10th Nov. 2017).

[5] Thomas Finley. *Two's Complement*. 2000. URL: https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html.

[6] Ian Glendining. *The Bloch Sphere*. European Center for Parallel Computing at Vienna. URL: http://www.vcpc.univie.ac.at/~ian/hotlist/qc/talks/bloch-sphere.pdf (visited on 16th Feb. 2005).

[7] Jack Krupansky. *What is Quantum Advantage and What is Quantum Supremacy?* URL: https://jackkrupansky.medium.com/what-is-quantum-advantage-and-what-is-quantum-supremacy-3e63d7c18f5b (visited on 14th June 2019).

[8] Margaret Martonosi and Martin Roetteler. *Next Steps in Quantum Computing: Computer Science's Role*. Tech. rep. 2018. URL: https://cra.org/ccc/wp-content/uploads/sites/2/2018/11/Next-Steps-in-Quantum-Computing.pdf.

[9] Giacomo Nannicini. *Quantum Computing, Lecture 3*. URL: https://researcher.watson.ibm.com/researcher/files/us-nannicini/8100_lecture_3.pdf (visited on 18th Sept. 2019).

[10] Kyrsta M. Svore Nathan Wiebe Ashish Kapoor. *Quantum Perceptron Models*. URL: https://arxiv.org/pdf/1602.04799.pdf (visited on 15th Feb. 2016).

[11] David Sherrill. *Dirac Notation*. URL: http://vergil.chemistry.gatech.edu/notes/intro_estruc/node5.html (visited on 8th July 2003).

[12] Vivek Srikumar. *The Perceptron Algorithm*. URL: https://www.cs.utah.edu/~zhe/pdf/lec-10-perceptron-upload.pdf.

[13] Eric W. Weissten. *Kronecker Product*. Mathworld - a Wolfram Web Resource. URL: https://mathworld.wolfram.com/KroneckerProduct.html.

[14] C.P. Williams. 'Explorations in Quantum Computing'. In: Springer-Verlag London Limited, 2011. Chap. Quantum Gates, pp. 51–122. URL: https://www.asc.ohio-state.edu/perry.6/p5501_sp17/articles/quantum_gates.pdf.