Quantum Neural Computation and Associative Memory

Tamer Salman
Quantum Neural Computation and Associative Memory

Research Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Tamer Salman

Submitted to the Senate of the Technion — Israel Institute of Technology

Tamuz 5772 Haifa July 2012
The research was done under the supervision of Prof. Yoram Baram in the Department of Computer Science.

I am deeply grateful to my supervisor, Prof. Yoram Baram, for his devoted guidance and supervision over the years. This work would not have been possible without his help.

I would like to thank all the staff, in particular, Mrs. Yardena Kolet, for the support and encouragement.

I thank my colleagues Karine Even, Yossi Weinstein, and Yuval Elias for the fruitful discussions. Special thanks to my friend Usama Kadri for the insightful and valuable conversations, and his good company.

I owe my deepest gratitude to my loving and supporting family. To my parents, Laila and Milad, for the continuous unconditional support, and for inculcating in me the passion for knowledge and devotion to science. To my three brothers, Hanna, Zaher, and Eyas for showing me the way and encouraging me to cross the finish line.

Last, but not least, I thank my loving wife, Hanan, to whom this work is dedicated. For the love and friendship. For being by my side during the darkest hours of the sleepless nights. None of this work could have been carried out without you.

The generous financial help of the VATAT committee, the Ministry of Science, and the Technion Graduate School is gratefully acknowledged.
Dedicated to Hanan
# Contents

List of Figures ........................................... iii

List of Tables ........................................... vi

Abstract ................................................... viii

Abbreviations and Notations .............................. ix

1 Introduction .............................................. 1

2 Artificial Neural Networks ............................... 3
   2.1 The Artificial Neuron .................................. 3
   2.2 Hebbian Learning ....................................... 5
   2.3 Associative Memory Networks ......................... 6

3 Quantum Computation .................................... 9
   3.1 Basic Building Blocks of Quantum Computation ....... 10
       3.1.1 States and Qubits .................................. 10
       3.1.2 Measurement ....................................... 11
       3.1.3 Operators ......................................... 11
       3.1.4 Quantum Parallelism and Interference ............. 12
       3.1.5 Solving problems using Quantum Computation .... 16
   3.2 Grover’s Quantum Search Algorithm .................. 17
   3.3 Grover’s Quantum Search Algorithm for Multiple Marked States .... 21
   3.4 Adiabatic Quantum Computation ...................... 23
       3.4.1 The Quantum Adiabatic Theorem .................. 23
       3.4.2 Adiabatic Quantum Computation Algorithm ....... 24
       3.4.3 Quantum Hebbian Interactions through Adiabatic Computation ... 24
List of Figures

2.1 A general neuron. ......................................................... 4
2.2 The transfer functions of the linear-threshold neurons vs. the weighted sum of
inputs \( u \). .............................................................. 4
2.3 The 2-dimensional XOR problem. ........................................ 5
2.4 A Simple feed forward Hebbian network. ............................. 6
2.5 A Hopfield network architecture. ....................................... 7
3.1 The 2-qubit Controlled-Not (CNOT) quantum gate. ................. 12
3.2 The 3-qubit Toffoli quantum gate. ................................... 12
3.3 A classical reversible Oracle. ........................................... 13
3.4 A quantum Oracle. ...................................................... 13
3.5 Duetsch’s quantum circuit. .............................................. 14
3.6 Duetsch-Jozza’s quantum circuit. ...................................... 15
3.7 The effect of Grover’s iteration on all basis states. \( \tau \) is the solution. ................................. 18
3.8 The effect of Grover’s rotation on the state \( |\Psi\rangle \) for a single marked state. ............ 19
3.9 The angle between the final state of Grover’s algorithm and the marked state \( \delta \)
is upper bounded by the rotation angle \( w \). .......................... 21
3.10 The effect of Grover’s rotation on the state \( |\Psi\rangle \) for multiple marked states. ........ 22
4.1 The intersection oracle \( f_{K \cap M} \) created using \( f_M, f_K, \) and \( Toffoli \). ......................... 28
4.2 The probability of measuring a state in the intersection between M and K (solid
line) and the probability of measuring a state not in the intersection (dotted line) vs. the number of iterations performed by Algorithm 1 ......................... 33
4.3 The probability of measuring a state in the intersection between M and K vs.
the ratio \( |M|/|N| \) (in logarithmic scale), when \( |K \cap M| << N \) ......................... 35
4.4 Success probability of pattern completion vs. memory size divided by the maxi-
mal completion capacity \( v \). ........................................... 41
4.5 Pattern completion or correction probability vs. the memory size divided by the maximal capacity $v$. For $0 < v < 1$, the probability is above 75% and for $v < 2$ is above 50%. .................................................. 43

4.6 Memory size vs. success probability of Algorithm 4. Optimal results are achieved only when the memory size is close to $\frac{N}{4}$. .................................................. 44

4.7 Memory size vs. success probability in Algorithm 5. Satisfactory results are achieved only when the memory size exceeds $\frac{N}{4}$. .................................................. 45

4.8 Memory size vs. success probability in Algorithms 2 and 3. .................................................. 46

4.9 An implementation of a completion oracle with an up to $n + 1$ dimensional controlled-not operator. .................................................. 47

4.10 An implementation of a correction oracle with a $\lceil \log n \rceil$ additional bits, $n$ controlled operators that add 1, and a threshold operator. .................................................. 48

4.11 (a) A set of memory patterns $M$ (b) a set of possible completions $K_1$ to a partial pattern, and (c) the memory completion result in amplitudes. .................................................. 49

4.12 (a) A set of memory patterns $M$ (b) a set of possible completions $K_2$ to a partial pattern, and the memory completion result in amplitudes. .................................................. 50

4.13 Simulation of a series of iterations of the completion algorithm. The graph shows the different behavior of the different subgroups of basis states. $K$ is the completion set and $M$ is the memory set. The memory completions and the non completions or memories are amplified alternatingly, while the amplitudes of $K \setminus M$ and $M \setminus K$ subgroups stay close to zero. .................................................. 51

4.14 Success probability of measuring a desired memory completion vs. the $\log$ of the memory size (solid), completion query size (dashed), possible memory completions (dotted), and number of qubits (dash-dotted). .................................................. 52

5.1 A schematic description of the first model of a quantum neuron. ................................. 54

5.2 A model of a layer of a quantum neural network with $m$ neurons of the first kind. 55

5.3 A schematic description of the second model of a quantum neuron. ................................. 56

5.4 A quantum neuron solving the XOR problem. ................................................................. 56

5.5 Hebbian relation through two 2-qubit quantum gates. .................................................. 57

5.6 A model of a quantum neuron with a single output trained on an $n$-dimensional training set. ................................................................. 59
5.7 The output of the classical Hebbian network (dashed line) vs. the network inputs and the normalized Hebbian weighted sum (solid line) vs. the network inputs. The diamond shaped points represent the probability of the quantum neuron to fire for each input. ................................................................. 61

5.8 A model of a quantum neural network with \( m \) outputs trained on an \( n \)-dimensional training set. ................................................................. 63

5.9 The probability of measuring a computational basis state at the output of the network trained on the training set in Eq. 5.18 vs. the decimal representation of the states when the network is presented with \([+1, +1, +1, +1]\). ........................................ 64

5.10 The probability of measuring a computational basis state at the output of the network trained on the set in Eq. 5.18 vs. the decimal representation of the states when the network is presented with \([+1, +1, +1, +1]\) and when the square root of the Hebbian relations are embedded in the superposition. .................. 66

5.11 A 3 dimensional training set with binary classification. ......................... 67
List of Tables

4.1 Properties of the four simulations depicted in Figure 4.14. The $x$ axis in Figure 4.14 represents the varying set size, while the other set sizes are constant in each simulation. .................................................. 50

5.1 A 3-dimensional training example set. .................................................. 59
5.2 The results of the quantum Hebbian classification trained with the training set in Table 5.1. .................................................. 62
Abstract

This thesis presents quantum analogues of artificial neural networks. We analyze and compare their performance to known classical and previously proposed quantum models.

First we propose a model for associative memory based on a modification of Grover’s quantum search algorithm and prove that the capacity of the model is exponential in the number of bits. We present algorithms for pattern completion and correction and prove that the model does not suffer from spurious memories and has a controllable basin of attraction.

Then we define models of quantum neurons and devise an algorithm for Hebbian learning through quantum gates. We show that the algorithm implements probabilistic learning.
Abbreviations and Notations

$x$ : input
$y$ : output
$x^i$ : input vector
$x^i_j$ : $j$th element of the vector $x^i$
y$^i$ : output
$d^i$ : desired output
$M$ : Memory set
$n$ : input space dimension
$N$ : the size of the $n$ dimensional binary space
$l$ : Training set size
$\mathbb{R}$ : The set of real numbers
$\mathbb{R}^n$ : The $n$ dimensional vector space over $\mathbb{R}$
$U$ : A quantum operator
$H$ : Hadamard operator
$I$ : Identity matrix
$I_\tau$ : Phase flip operator
AQC : Adiabatic Quantum Computing
Chapter 1
Introduction

In this research we study quantum analogues to the classical artificial neural networks and discuss their advantages and their ability to improve classical learning techniques in neural networks. It is widely believed that quantum computation can help solving classical problems more efficiently than classical computation. This belief was strengthened by the introduction of Shor’s algorithm [1] for factoring large numbers in polynomial time and Grover’s algorithm [2] for searching an unsorted database in sub-exponential time. Similarly to other fields of classical computation, the field of artificial neural networks is seeking to benefit from the advantages of quantum computation. This idea was first suggested in [3], and has been the main subject of many subsequent work. However, a theory that establishes a substantial advantage over classical networks is still missing and no quantum model has been widely adopted yet. A more difficult issue concerning whether there would be any implications on models of biological neural networks remains an open question beyond the scope of the present work.

Classical artificial neural networks are interconnected parallel collection of processing units called neurons. A small perturbation in one neuron causes a spread of information through the network. The nature of these models bears resemblance to the properties of quantum computational systems. However, the analogy and connection between both fields are far from being fully understood.

The first attempt made by the neural networks community to benefit from quantum computation was made through the development of a quantum-inspired network [4], which later developed to a classical subfield of neural networks assisted by algorithms inspired by quantum mechanics [5, 6, 7, 8, 9]. A similar subfield emerged for quantum analogues of genetic and evolutionary algorithms [10, 11, 12, 13]. These works concern the development of classical algorithms that only resemble quantum computational algorithms and were, thus, bound to present limited advancement over previously known models and algorithms. Another development, also beyond the scope of this work, targeted the problem of learning quantum information [14, 15].
One of the major causes for the interest in quantum computation shown in the neural networks research is the simple solution of the XOR problem [16, 17, 18]. It is well known that a single linear threshold neuron is unable to solve the non-linearly separable XOR problem, while it is easily solved using the controlled-not operator.

In this thesis, we present models of quantum neurons and quantum neural networks. We show how they can be applied to classical learning problems with an emphasis on associative memory. We analyze these models and compare them to classical algorithms and to previously proposed quantum models [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42]. Previously proposed quantum analogues of artificial neural networks appear to lack certain essential features of learning and associative memory, such as pattern completion and correction. We compare our models to those previously proposed ones and to classical models and show that they achieve better performance and require less computation.

In chapters 2 and 3 we present a background of neural networks and quantum computation respectively. In chapter 4 we show how a quantum neural network can realize associative memory with capacity exponential in the number of qubits. We develop an algorithm based on a modification of Grover’s quantum search and prove its pattern completion and correction abilities. Furthermore, we prove that the model allows control over the basin of attraction and that it does not suffer from the emergence of spurious memories. In chapter 5 we present two models of a quantum neuron and present a classical Hebbian training algorithm for one of these models, then we compose neural networks that constitute probabilistic learning models. Finally, in chapter 6, we conclude with a discussion of our results and potential future work.
Chapter 2

Artificial Neural Networks

The study of artificial neural networks and their ability to learn complex (non-linear) behavior consists of the study of the basic neuron’s capabilities and the effect of dense interconnectivity between such elements on the overall modeling power [43, 44, 45, 46, 47, 48]. Different problems induced the use of both different neuron models and a variety of interconnectivity patterns. Among these models are the feed-forward network model [49, 50, 51] and recurrent network model [52]. In addition, many learning algorithms (adaptation and tuning models) were devised for each given structure, such as the Error Back-Propagation algorithm and the Hebbian learning algorithm. A learning rule can be viewed as a search algorithm for an appropriate assignment of the parameters of the network in some parameter space defined by the structure and the types of neurons in the network. The appropriate assignment seeks to optimize some measure of aptness usually called the error function or the cost function [51, 53, 54, 55, 56].

2.1 The Artificial Neuron

A neuron is a computation unit with \( n \) inputs, \( x = (x_1, ..., x_n)^T \), and a single output \( y \). Fig. 2.1 shows a schematic description of a commonly used neuron model. The output of the neuron is

\[
y = f(u) = f(\langle w, x \rangle - \theta) = f \left( \sum_{i=1}^{n} w_i x_i - \theta \right)
\]

(2.1)

where \( w \) is a vector of weights and \( \theta \) is an activation threshold, \( u \) is the weighted sum of inputs, and \( f \) is the neuron’s transfer function.

The transfer function embodies the different capabilities and limitations of different neurons. A linear neuron, is a neuron with a linear transfer function, for instance \( y = u \), performs a linear transformation of the input space. The linear-threshold neuron is a non-linear neuron with a linear threshold unit in which the transfer function (cf. Fig. 2.2) is the step function

\[
y = f(u) = \begin{cases} 
1 & , \ u \geq 0 \\
-1 & , \ u < 0 
\end{cases}
\]
which divides the \( n \) dimensional space with a hyperplane defined by \( w \) and \( \theta \) and causes the neuron either to fire (i.e. output +1) or to not fire (i.e. output -1).

Equivalently, the linear-threshold neuron can be defined as a unit receiving \( n+1 \) input vector \( \tilde{x} = (x_1, ..., x_n, -1) \) and a weight vector \( \tilde{w} = (w_1, ..., w_n, \theta) \), thus, performing a homogenous weighted sum in an \( n+1 \) dimensional space. However, a single neuron is able to separate only sets that are linearly separable in the \( n \)-dimensional space. Thus, for instance, the XOR problem in Fig. 2.3 is unsolvable by a single neuron [45] and requires a network of multiple neurons. An appropriate combination (network) of such neurons can implement any linear division of the \( n \) dimensional space.
2.2 Hebbian Learning

Hebb’s rule (cf. [46, 47]) is one of the oldest learning rules and it states that if the activity in one end of a synapse repeatedly and persistently causes a similar activity in the other end of the synapse then the synaptic weight increases. In learning processes of artificial neural networks, it can be applied by the increase of the weight of a connection between two neurons that are activated simultaneously and the decrease (or elimination) of the weight between two neurons that are activated asynchronously.

In the context of a single neuron depicted in Fig. 2.1, Hebb’s rule simply states that if the correlation between input and output of the neuron is large, then the weight corresponding to that input should be large. A Hebbian neuron learning can be achieved for example by the following update rule:

\[ w_i(t + 1) = w_i(t) + \eta y(t)x_i(t) \tag{2.2} \]

where \( w_i(t) \) is the weight of the synapse connected to the \( i \)th input of the neuron at time \( t \), \( x_i(t) \) is the input on that synapse at time \( t \), \( y(t) \) is the output of the neuron, and \( \eta \) is some learning rate.

The Hebbian learning rule can be manifested in the context of the learning algorithm of a neural network. For each two connected neurons, where the axon of neuron \( A \) is connected to the synapse of neuron \( B \). If the firing of neuron \( A \) persistently causes the firing of neuron \( B \), then the algorithm increases the weight on the synapse connecting them. On the other hand, if there is no correlation between the firing of both neurons, then the algorithm decreases the weight on the synapse connecting them. Similarly to Eq. 2.2, this can be achieved with the following update rule:

\[ w_{ij}(t + 1) = w_{ij}(t) + \eta y_j(t)x_i(t) \]
where \( w_{ij} \) is the weight of the synapse connecting the output of neuron \( i \) to the input of neuron \( j \) and \( \eta \) is the learning rate.

2.3 Associative Memory Networks

Associative memory networks are networks that store sets of patterns \( \{(x^k, y^k)\}_{k=1}^l \) and retrieve a value \( y^k \) when presented with a partial or a noisy version of \( x^k \). Hebbian learning is widely used for training (i.e. tuning) such networks.

Consider for example the simple feed forward network in Fig. 2.4 with \( n \) inputs and \( m \) outputs. The network can store pairs of input output \( \{(x^k, y^k)\}_{k=1}^l \), where \( x^k \in \{-1, 1\}^n \) and \( y^k \in \{-1, 1\}^m \) using Hebb’s rule to determine the weights on the connections between them. The following matrix embodies Hebb’s rule, and thus is called a Hebbian weight matrix:

\[
W = \frac{1}{l} \sum_{k=1}^l y^k x^k T
\]

(2.3)

where \( y^k x^k T \) is the outer product between the vector \( y^k \) and the vector \( x^k \).

![Figure 2.4: A Simple feed forward Hebbian network.](image)

The Hebbian learning paradigm exists in dynamical recurrent networks as well. A dynamic recurrent network is a network in which the neurons’ states evolve over time according to their connectivity. Several models exist in which the time can be a discrete or a continuous process, and the evolution rule can be synchronous or asynchronous. In synchronous evolution, the states of all the neurons are updated simultaneously in such a way that the state of each neuron at time \( t \), \( x_i(t) \), is influenced by the states of the connected neurons at time \( t - 1 \). On the other hand, in asynchronous evolution, at each time step a single neuron is chosen, either randomly
or in an ordered fashion, and is updated similarly. Let \( n \) denote the number of neurons in the network, then the update of the neurons outputs can be expressed for neuron \( i \) as follows:

\[
x_i(t) \leftarrow f \left( w_{1i} x_1(t - 1) + \cdots + w_{ni} x_n(t - 1) - \theta_i \right)
\]  

(2.4)

where \( f \) is the transfer function of the neurons, \( w_{ji} \) is the synaptic weight from neuron \( j \) to neuron \( i \), and \( \theta_i \) is the activation threshold of neuron \( i \).

Hopfield network [52] is a dynamic recurrent network with discrete time and asynchronous evolution model. Each neuron in the network implements a threshold function of the weighted sum of its entries and outputs \( \pm 1 \) accordingly. The neurons are fully connected to each other with no reflexive connections \( (w_{ii} = 0, \forall i) \) and the weight matrix is symmetric \( (w_{ij} = w_{ji} \forall i, j) \). The network is depicted in Fig. 2.5.

![Figure 2.5](image)

Figure 2.5: A Hopfield network architecture.

The network realizes an auto-associative memory model and is able to store \( l \) memory patterns \( x^1, \ldots, x^l \) through the Hebbian learning rule formulated as follows:

\[
W = \frac{1}{l} \sum_{k=1}^{l} x^k x^k T
\]  

(2.5)

where \( x^k \) is an \( n \)-dimensional pattern and where the value in the \( i \)th row and the \( j \)th column of \( W \) is

\[
w_{ij} = \frac{1}{l} \sum_{k=1}^{l} x^k_i x^k_j
\]  

(2.6)

Once the network is presented with an input, the state is changed by a series of updates of the neurons, where at each time step a neuron is chosen randomly and updated. The Hopfield

7
network convergence theorem [52] states that the network initiated with any input converges to a fixed point of the network by minimizing the energy function:

\[ E = -\frac{1}{2} X^T W X + \Theta^T X = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} \theta_i x_i \]  

(2.7)

where \( X \) is the column state vector of the network and \( \Theta \) is the column vector of the neurons’ activation thresholds. The proof of the theorem relies on the fact that the energy function is bounded from below and that updates do not increase the value of the energy resulting in an asymptotic convergence of the probability of reaching a fixed point (also known as attractor) of the network to 1. It has been shown [57, 58] that the network converges very rapidly to one of its attractors.

A similar network using synchronous updates was proved [59] also to converge to a fixed point or to a repeating cycle of two patterns.

The convergence theorem does not state that the resulting fixed point is a memory pattern. In fact, associative memory networks are known to produce spurious memories, which are unmemorized fixed points. It is also easy to see that for each fixed point of the network, also the negation is a fixed point.

However, a sufficient condition for the memory patterns to be fixed points of the network states that each pattern \( x^j \) should be far apart from the rest of the memory patterns so that

\[ \forall j \in \{1, l\} \sum_{k=1, k\neq j}^{l} \left| 1 - 2\rho(x^k, x^j) \right| < 1 \]  

(2.8)

where \( \rho(x^k, x^j) \) is the normalized Hamming distance (i.e. radius) between \( x^k \) and \( x^j \).

Furthermore, the minimal radius of the memory patterns that allows pattern correction of a noisy pattern \( x \) to the memory pattern \( x^c \) is

\[ \rho(x, x^c) < \frac{1}{2} \left( 1 - \left| \sum_{k=1, k\neq c}^{l} (1 - 2\rho(x, x^k)) \right| \right) \]  

(2.9)

Associative memory networks have been extensively studied [52, 59, 60, 61, 62, 63, 64, 65]. It was shown [60, 66, 67] that when Hopfield network stores random memory patterns and the number of neurons \( n \) grows to infinity, the correct pattern will be revealed with probability close to 1 as long as the memory size is upper bounded by

\[ \frac{(1 - 2\rho)^2 n}{4 \log n} \]

where \( \rho \) is the signal to noise ratio of the input. Sparse encoding [62, 65] has been shown to increase the storage capacity considerably [63]. A capacity exponential in the input dimension has been shown to result from a network size also exponential in the input dimension [64].
Chapter 3

Quantum Computation

This chapter is an introduction to quantum computation and its basic building blocks. A deeper view of the various elements in this introduction can be found in [68].

The introduction of Shor’s algorithm for factoring numbers in polynomial time [1] has demonstrated the ability of quantum computation to solve certain problems more efficiently than classical computers. This perception was ratified two years later, when [2] introduced a sub-exponential algorithm for quantum searching a database.

The field of quantum computation is based on the combination of computation theory and quantum mechanics. Computation theory concerns the design of computational models and the study of their time and space complexities. Quantum mechanics, on the other hand, concerns the study of systems governed by the rules of quantum physics. The combination of the two fields addresses the nature of computation in the physical world. However, there is still no efficient reduction of quantum mechanical behavior to classical computation.

Quantum mechanics is a conceptual framework that mathematically describes physical systems. It is based on four postulates, known as the postulates of quantum mechanics [68], which provide a connection between the physical world and mathematical formalism. Through these postulates, it is possible to better understand the nature of physical computation and what can be physically computed.

The first postulate states that a physical system is completely described by a state in a Hilbert space known as the state space. The second states that the evolution of a closed quantum system is described by a unitary transformation. The third states that a quantum measurement is described by a collection of measurement operators that satisfy the completeness equality, i.e. the sum of all possible measurements adds up to 1. The forth states that the state space of a composite physical system is the tensor product of the state spaces of its components.

Next, we introduce the basic building blocks of quantum computation. The reader is referred
3.1 Basic Building Blocks of Quantum Computation

3.1.1 States and Qubits

The basic entity of classical computation is the classical bit. Each classical bit can have one of two values, 0 and 1. The state of any finite physical system that can be found in a finite number of states can be described by a string of bits. A string of \( n \) bits represents one of \( 2^n \) possible states of a system enumerated \( 0, \ldots, 2^n - 1 \).

In quantum computation, the basic entity is called a qubit (quantum bit). The qubit can have the analogue values \(|0\rangle\) and \(|1\rangle\), known as the computational basis states, where \(|\cdot\rangle\) is the Dirac notation. Yet, the qubit can also have any other value that is a linear combination of \(|0\rangle\) and \(|1\rangle\):

\[
|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle
\]

where \( \alpha \) and \( \beta \) are any complex numbers (called the amplitudes of the basis states 0 and 1, respectively), such that \(|\alpha|^2 + |\beta|^2 = 1\). Consequently, the qubit can be in any one of an infinite number of states described by unit vectors in a 2-dimensional complex vector space. The unary representation of a qubit can be given as a vector of two values

\[
|\Psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}
\]

Another convenient representation of the state of a qubit may be written as

\[
|\Psi\rangle = e^{i\gamma} \left( \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \right)
\]

where \( \gamma \), \( \phi \), and \( \theta \) are real numbers. The term \( e^{i\gamma} \), known as the global phase of the qubit, has no observable effects and is usually omitted and instead, the qubit state is represented by

\[
|\Psi\rangle = \left( \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \right)
\]

Analogously to the classical bit string, qubits strings (or quregisters) describe the state of a system. A two qubit system comprising two qubits \( |a\rangle = \alpha |0\rangle + \beta |1\rangle \) and \( |b\rangle = \gamma |0\rangle + \delta |1\rangle \) is described by the tensor product of the two qubits \( |a, b\rangle \equiv |a\rangle \otimes |b\rangle = \alpha \gamma |00\rangle + \alpha \delta |01\rangle + \beta \gamma |10\rangle + \beta \delta |11\rangle \).
3.1.2 Measurement

The measurement of a qubit reveals only one of two possible outcomes. The value of \(\alpha\) and \(\beta\) cannot be extracted from the measurement of a qubit. Instead, when measuring the qubit \(\alpha |0\rangle + \beta |1\rangle\) in the computational basis, the result can be either 0 or 1 with probabilities \(|\alpha|^2\) and \(|\beta|^2\) respectively. For example, the state \(\left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle\right)\), when measured, yields any of the two results 0 or 1 with probability 1/2. The measurement operation is not reversible and, once made, the qubit no longer exists in its state before the measurement. Measurements can be performed in different bases. For example, measuring the qubit \(\alpha |0\rangle + \beta |1\rangle\) in the Hadamard basis defined by the two basis states \(|+\rangle \equiv \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle\right)\) and \(|-\rangle \equiv \left(\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle\right)\) gives \(|+\rangle\) with probability \(\frac{(\alpha + \beta)^2}{2}\) and \(|-\rangle\) with probability \(\frac{(\alpha - \beta)^2}{2}\), since \(\alpha |0\rangle + \beta |1\rangle = \frac{\alpha + \beta}{\sqrt{2}} |+\rangle + \frac{\alpha - \beta}{\sqrt{2}} |-\rangle\).

An \(n\)-qubit system can be either measured completely or partially. When measured partially, the unmeasured subsystem can retain quantum superposition and further quantum manipulations can be performed upon it. However, any measurement can be delayed to the end of the computation process.

3.1.3 Operators

In quantum computation, a system changes its state under a unitary quantum operator \(U\) from \(|\Psi\rangle\) to \(U|\Psi\rangle\). An operator \(U\) can be described as a \(2^n \times 2^n\) matrix operating on the unary representation of the system state. A unitary operator satisfies \(UU^\dagger = I\), where \(U^\dagger\) is the conjugate transpose of \(U\) (transpose the matrix \(U\) then substitute the conjugate complex of each element in the matrix).

Quantum operators can be implemented using quantum gates, which are the analogue of the classical gates that compose classical electrical circuits. In this analogy, the wires of a circuit carry the information on the system’s state, while the quantum gates manipulate their contents to different states. For example, the Hadamard operator

\[
H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}
\]

transforms a qubit in the state \(|0\rangle\) into the state \(|+\rangle\) and the state \(|1\rangle\) into the state \(|-\rangle\).

Operators can also be quantum gates operating on multiple qubits. An \(n\)-qubit quantum operator has \(n\) inputs and \(n\) outputs. For example, the 2-qubit controlled-not (CNOT) gate depicted in Figure 3.1, flips the target (second) qubit if the control qubit (first) has value \(|1\rangle\) and leaves it unchanged if the control qubit has the value \(|0\rangle\). Specifically, the CNOT gate performs the following transformations on the four computational basis states: \(|00\rangle \rightarrow |00\rangle\),
\[ |01\rangle \rightarrow |01\rangle, \ |10\rangle \rightarrow |11\rangle, \text{ and } |11\rangle \rightarrow |10\rangle. \] It can be described as a unitary matrix operating on the unary representation of the state as follows:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix}
\]

Another gate is the 3-qubit controlled-controlled-not gate also known as the Toffoli gate depicted in Figure 3.2, which flips the target (third) qubit if the two control bits (first and second) both have the values \( |1\rangle \) and leaves it unchanged otherwise.

\[
CNOT = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix}
\]

The Hadamard operator can also be seen as operating on \( n \)-qubits by the tensor product of \( n \) single qubit Hadamard operators. Each qubit is then transformed according to the single qubit Hadamard transform, i.e.,

\[
H^\otimes n |x_{n-1}, \ldots, x_1, x_0\rangle = H |x_{n_1}\rangle \otimes \ldots \otimes H |x_1\rangle \otimes H |x_0\rangle
\]

3.1.4 Quantum Parallelism and Interference

Consider an oracle of an \( n \)-dimensional function \( f \) implemented by a classical circuit, which, for every input \( x \), produces \( f(x) \) at the output. The oracle can be made reversible by adding an additional bit to the input, initialized by 0, and letting the output be \( x, 0 \oplus f(x) \) as depicted in Figure 3.3, where \( \oplus \) is the addition modulo 2. Initializing the additional bit with 1 produces \( f(x) \) at the output.

A quantum oracle is a reversible oracle that accepts a superposition of inputs and produces a superposition of outputs as depicted in Figure 3.4. When the additional qubit is initialized
by $|0\rangle$, the oracle performs the following transformation: $|x\rangle|0\rangle \rightarrow |x\rangle|0 \oplus f(x)\rangle$. When the additional qubit is initialized by $|−\rangle$ the oracle is called a quantum phase oracle that gives $f(x)$ in the phase of the state $|x\rangle$ as follows: $|x\rangle|−\rangle \rightarrow (-1)^{f(x)}|x\rangle|−\rangle$.

Suppose that we have constructed a quantum circuit $U_f$ that implements a function $f : \{0,1\}^n \rightarrow \{0,1\}$, such that when introduced with an input $|x\rangle|y\rangle$, the output of the circuit would be $|x\rangle|y \oplus f(x)\rangle$. Quantum parallelism is the ability of the quantum circuit to process many inputs simultaneously and receive all the outcomes at the output. Consider the case where $|y\rangle = |0\rangle$ and $|x\rangle = H^\otimes n |0\rangle^\otimes n$. Applying the $n$-qubit Hadamard operator to the $|0\rangle$ state yields a superposition of all basis states $\frac{1}{\sqrt{2^n}} \sum_i 0^{2^n} |i\rangle$. The superposition will be maintained through the quantum circuit and the resulting output would be $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n} |i\rangle |f(i)\rangle$. This can be computed as follows:

$$U_f^\otimes n+1 ((H^\otimes n \otimes I) |0\rangle^\otimes n |0\rangle) = U_f^\otimes n+1 \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n} |i\rangle |0\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n} |i\rangle |f(i)\rangle$$
Although this computes $f$ for all values of $i$ simultaneously there is no immediate way of seeing them all together, because once the output state is measured, only one value of $f(i)$ is revealed and the rest vanish. However, further quantum computation allows the different values to interfere together and reveal some information concerning the function $f$.

It has been shown [69, 70, 71] that the revealed information can be considerably more than what classical computation would achieve after one query of the circuit that implements the function $f$. Deutsch and Josza [70] showed that if a binary $n$-dimensional function is guaranteed to be either constant or balanced, then determination can be done using only one query to a quantum oracle implementing it, while a classical solution would require an exponential number of queries. Another example of the advantage of quantum algorithms over classical ones was presented by [71], in which a function is known to have the property that there exists some $s \in \{0, 1\}^n$ for which for all $x, y \in \{0, 1\}^n$ it hold that $f(x) = f(y)$ if and only if $x = y$ or $x \oplus y = s$. [71] proved that using quantum computations, $s$ can be found exponentially faster than with any classical algorithm, including probabilistic algorithms.

**Deutsch’s and Deutsch-Josza’s Algorithm**

The algorithm shows that it is possible to determine for a one dimensional function $f$ whether $f(0) = f(1)$ or not. Obviously, classically this requires two queries of the function. However, exploiting quantum parallelism and interference, this is achievable with only one query. The circuit is depicted in Fig. 3.5 and the output of the circuit can be evaluated as follows:

![Figure 3.5: Duetsch’s quantum circuit.](image-url)
\[(H \otimes I)U_f(H \otimes H)|0\rangle|1\rangle = \frac{1}{2^n}(\pm|+\rangle|+\rangle - |0\rangle|0\rangle)\]

Measuring the first qubit of the output of the circuit yields the value \(f(0) \oplus f(1)\), which is 0 if \(f(0) = f(1)\) and 1 otherwise. Hence, using a quantum circuit, it is possible to determine whether \(f\) is constant or not using only one query of the quantum circuit implementing \(f\).

A similar, yet more general algorithm, can determine whether an \(n\)-dimensional function \(f : \{0, 1\}^n \rightarrow \{0, 1\}\) is balanced (outputs 1 for exactly half of the possible inputs and 0 for the other half) or constant. The computational process is similar to Duetsch’s algorithm except that the input \(x\) is \(n\)-dimensional. The circuit is depicted in Fig. 3.6 and the computational process is described as follows:

\[
\left(\frac{1}{2^n}\sum_{i=1}^{2^n} \frac{1}{\sqrt{2^n}} (-1)^{i \cdot j + f(j)} |j\rangle |i\rangle \right) = \left(\frac{1}{2^n}\sum_{i=1}^{2^n} (-1)^{i \cdot j} |i\rangle |i\rangle \right)
\]

Figure 3.6: Duetsch-Joza’s quantum circuit.

Measuring the \(n\)-dimensional register gives the answer to the question at hand. If the measured value is 0 then the function \(f\) is constant, any other measured value means that \(f\) is balanced.
This example problem shows an advantage of quantum computation over classical computation, since it is possible to decide whether $f$ is constant or balanced using only one query of the function, whereas in classical computation, making the decision without any possibility of error requires $\frac{2^n}{2} + 1$ queries of $f$ in the worst case.

3.1.5 Solving problems using Quantum Computation

According to the above definitions of a system state, measurement, and operators, a quantum computer drives the dynamics of the quantum system through operators that change its state and measures the final state to reveal classical information. In the general case, one might think of the process of quantum computation as a multi-phase procedure, which performs some classical computation on the data at hand, creates a quantum state describing the system, drives it through quantum operators, which might depend on the data, to the target state, measures the outcome, and performs some more classical computation to receive a result.

The above two examples of Duetsch’s algorithm and Duetsch-Josza’s algorithm show a simple case of solving a problem, in which the given data is a function $f$ and the requested output is a classical binary decision that encodes one of two possible solutions. In these cases no classical computation is required before the creation of the quantum system, instead, a quantum state is created initially to be some constant value. A quantum computation procedure using quantum operators partly dependant on the given data, in this case $U_f$ is applied to the system followed by a measurement and a simple classical processing to give the appropriate answer based on the measured value.

Consequently, we can describe a schematic process for solving problems using quantum computation as follows:

A general solution scheme using quantum computations

Given:

1. Preliminary classical computation of the input
2. Create initial quantum state
3. Apply quantum circuit to the initial quantum state
4. Measure the resulting state
5. Apply classical computation to the measured state
3.2 Grover’s Quantum Search Algorithm

Given a database of \( N \equiv 2^n \) unsorted elements of \( n \) bits each, any classical search would require \( O(N) \) queries to find a desired element. In 1996, Grover presented a quantum computational algorithm that searches an unsorted database with \( O(\sqrt{N}) \) operations \([2, 72]\). The algorithm performs a series of \( O(\sqrt{N}) \) unitary operations on the superposition of all basis states that amplify the solution state causing the probability of measuring the solution at the end of the computation to be close to 1.

Suppose that the search problem has a single solution \( \tau \) and that we own an oracle function \( f_\tau \) that identifies the solution according to the following:

\[
f_\tau(x) = \begin{cases} 1, & x = \tau \\ 0, & x \neq \tau \end{cases}
\]

Any classical algorithm that attempts to find the solution clearly needs to query the oracle \( N - 1 \) times in the worst case. Grover’s algorithm shows that we can find the solution with the help of the oracle by querying it only \( O(\sqrt{N}) \) times.

The quantum phase oracle of the function \( f_\tau \) flips (rotates by \( \pi \)) the amplitude of the solution \( \tau \), while leaving all other states unchanged. This is done by the operator \( I_\tau = I - 2 |\tau \rangle \langle \tau| \). In matrix formulation, \( I_\tau \) is similar to the identity matrix \( I \) except it has \( -1 \) on the \( \tau \)th element of the diagonal.

Grover’s algorithm starts with the superposition of all basis states created by applying the Hadamard operator on the zero state, \( H^\otimes n |0\rangle^\otimes n \) (shortened by \( H |0\rangle \)), and goes about performing multiple iterations, in which each iteration consists of applying the phase oracle followed by the operator \( HI_0H \), where \( I_0 \) flips the phase of the state \( |0\rangle^\otimes n \). Grover’s iterator is thus defined as

\[
Q = -HI_0HI_\tau
\]

where the sign "\(-\)" stands for the global phase flip that has no physical meaning and is performed only for analytical convenience.

The operator \( HI_0H \) flips the phase of all basis states about the average amplitude \([72, 73]\). As a result, amplitudes above the average become below the average and vise versa as shown in Fig. 3.7. Fig. 3.7 (a) shows the initial superposition of all basis states with equal amplitudes. Applying the phase oracle \( I_\tau \) on the superposition, flips the amplitude of the state \( \tau \) (the marked state), resulting in the superposition in (b). The average amplitude of all basis states is the dashed line in (b'). Applying \( HI_0H \) flips all amplitudes about the average amplitude, resulting in the superposition depicted in (c), and in which the probability of measuring the state \( \tau \) is higher than any other basis state.
The operator in Eq. 3.1 can be viewed in the space defined by the marked state, the superposition of all other computational basis states, and some orthonormal extension of states. Formally, define

$$|l_1\rangle \equiv |\tau\rangle$$

and

$$|l_2\rangle \equiv \frac{1}{\sqrt{N-1}} \sum_{i \neq \tau} |i\rangle$$

then the basis is the set \{\(|l_1\rangle, |l_2\rangle, \ldots\)\}, where and the rest of the basis states are any orthonormal extension. The initial state is the superposition of the first two basis states

$$|\Psi\rangle = H |0\rangle = \frac{1}{\sqrt{N}} |l_1\rangle + \sqrt{\frac{N-1}{N}} |l_2\rangle$$

and the application of Grover’s operator on \(|\Psi\rangle\) results in a state in the same 2-dimensional space spanned by \(|l_1\rangle\) and \(|l_2\rangle\). \(I_{\tau}\) rotates \(|\Psi\rangle\) about \(|l_2\rangle\) as follows

$$I_{\tau} |\Psi\rangle = (I - 2 |\tau\rangle \langle \tau|) |\Psi\rangle = |\Psi\rangle - 2 |\tau\rangle \langle \tau| |\Psi\rangle = |\Psi\rangle - 2\alpha |\tau\rangle = -\alpha |l_1\rangle + \beta |l_2\rangle$$  \hspace{1cm} (3.2)

and \(-HI_0H\) is a rotation about \(|\Psi\rangle\)

$$-HI_0H = -H(I - 2 |0\rangle \langle 0|)H = 2 |\Psi\rangle \langle \Psi| - I$$  \hspace{1cm} (3.3)

which means that Grover’s operator is a product of two reflections, and thus, a rotation in that space. The rotation is depicted in Fig. 3.8, where \(\phi\) is the angle between the initial state and
\(|l_2\rangle \)

\[ \phi = \arctan \left( \frac{1}{\sqrt{N} - 1} \right) \approx \frac{1}{\sqrt{N}} \]  

(3.4)

and \( w \) is the rotation angle

\[ w = \arccos \left( 1 - \frac{2}{N} \right) \]

which, for large \( N \), can be approximated to

\[ w \approx \frac{2}{\sqrt{N}} \]  

(3.5)

Figure 3.8: The effect of Grover’s rotation on the state \(|\Psi\rangle\) for a single marked state.

Grover’s operator can be described in matrix form as

\[
\begin{pmatrix}
1 - \frac{2}{N} & 2\sqrt{\frac{N-1}{N}} \\
-2\sqrt{\frac{N-1}{N}} & 1 - \frac{2}{N}
\end{pmatrix}
\]

Consequently, applying the operator for a certain number of times brings the state \(|\Psi\rangle\) close to \(|l_1\rangle\). Measuring the state then, yields the marked state \(|l_1\rangle\) with high probability. Let \(|\Psi(t)\rangle\) be the state after applying \(Q\) for \(t\) times and \(|\Psi(0)\rangle = |\Psi\rangle = H |0\rangle\), then

\[ |\Psi(t)\rangle = Q^{t} H |0\rangle = \sin(wt + \phi) |l_1\rangle + \cos(wt + \phi) |l_2\rangle \]

and the probability of measuring a marked state from \(|\Psi(t)\rangle\) is

\[ Pr(t) = \sin^2 (wt + \phi) \]

Therefore, the optimal \(t\) for measuring, \(t_{opt}\), satisfies

\[ \sin^2 (wt_{opt} + \phi) = \frac{1}{2} - \frac{1}{2} \cos (2wt_{opt} + 2\phi) = 1 \]
which yields
\[ 2\omega t_{opt} + 2\phi = \pi \] (3.6)
Substituting Eq. 3.4 and Eq. 3.5 in Eq. 3.6 results in
\[ t_{opt} = \frac{\pi - 2\phi}{2\omega} \approx \frac{\pi - \frac{2}{\sqrt{N}}}{\frac{1}{\sqrt{N}}} = \frac{\pi}{4} \sqrt{N} - \frac{1}{2} \] (3.7)
But since we have to perform a complete number of iterations, then the algorithm measures the state after
\[ T = \left[ \frac{\pi}{4} \sqrt{N} - \frac{1}{2} \right] \] (3.8)
iterations. Suppose that the algorithm stops with the state in angle \( \delta \) from the marked state after \( T \) iterations as shown in Fig. 3.9, then the probability of measuring the marked state is
\[ Pr(T) = \sin^2 (wT + \phi) = \sin^2 (w(t_{opt} - \Delta T) + \phi) = \sin^2 (w t_{opt} + \phi - w \Delta T) \]
where
\[ \Delta T = t_{opt} - T < 1 \] (3.9)
Using trigonometric identities and Eq. 3.9, we obtain
\[
Pr(T) = [\sin (w t_{opt} + \phi) \cos (w \Delta T) - \cos (w t_{opt} + \phi) \sin (w \Delta T)]^2 \\
= \sin^2 (w t_{opt} + \phi) \cos^2 (w \Delta T) \\
- 2 \sin (w t_{opt} + \phi) \cos (w \Delta T) \cos (w t_{opt} + \phi) \sin (w \Delta T) \\
+ \cos^2 (w t_{opt} + \phi) \sin^2 (w \Delta T) \\
= \cos^2 (w \Delta T) \\
> \cos^2 (w) = \left(1 - \frac{2}{N}\right)^2 \] (3.10)
The above lower bound on the probability of success is valid for any number of qubits. For instance, for a 4-qubits register and 1 marked state, Grover’s algorithm finds the marked state with probability higher than \( \left(1 - \frac{2}{16}\right)^2 = \frac{49}{64} \approx 77\% \) and for a 7-qubits register with 1 marked state, the probability is higher than \( \left(1 - \frac{2}{64}\right)^2 = 92\% \).
Moreover, for large \( N \), Grover’s algorithm finds the marked state with probability asymptotically close to 1 after \( O\left(\sqrt{N}\right) \) iterations [2, 72]. A few repetitions of the algorithm brings the probability of finding the marked state to 1, which presents a quadratic speedup over the best classical algorithm.
3.3 Grover’s Quantum Search Algorithm for Multiple Marked States

A generalization of Grover’s quantum search algorithm for multiple marked states was shown in [72]. The algorithm is able to find one of a set of marked states. Suppose that the search problem has a set \( X \) of \( r \) solutions and that we own an oracle function \( f_X \) that identifies the solution \( x \in X \) according to the following:

\[
f_X(x) = \begin{cases} 
1, & x \in X \\ 
0, & x \notin X 
\end{cases}
\]

Let \( I_X \) be the quantum phase oracle of the function \( f_X \) that flips (rotates by \( \pi \)) the amplitude of the states of \( X \), while leaving all other states unchanged

\[
I_X = I - 2 \sum_{x \in X} |x\rangle \langle x| 
\]  

(3.11)

which, in matrix formulation, is similar to the identity matrix \( I \) except it has \(-1\) on the \( x \)th elements of the diagonal.

Similarly to the case of a single marked state, the operator

\[
Q = -HI_0HI_X
\]

can be viewed in the space defined by the two basis states

\[
|l_1\rangle = \frac{1}{\sqrt{r}} \sum_{i \in X} |i\rangle
\]  

(3.12)
and

\[ |l_2⟩ = \frac{1}{\sqrt{N-r}} \sum_{i \in X} |i⟩ \]  

(3.13)

where the initial state can be represented in this basis as a superposition of Eq. 3.12 and Eq. 3.13:

\[ |Ψ⟩ = H |0⟩ = \sqrt{\frac{r}{N}} |l_1⟩ + \sqrt{\frac{N-r}{N}} |l_2⟩ \]  

Consequently, the angle of the initial state in this space with \( |l_2⟩ \) is

\[ \phi = \arctan \left( \sqrt{\frac{r}{N-r}} \right) \]

and the rotation angle of the operator \( Q \) is

\[ w = \arccos \left( 1 - \frac{2r}{N} \right) \]  

(3.14)

as depicted in Figure 3.10. The matrix form of the operator is

\[
\begin{pmatrix}
1 - \frac{2r}{N} & 2\sqrt{\frac{r(N-r)}{N}} \\
-2\sqrt{\frac{r(N-r)}{N}} & 1 - \frac{2r}{N}
\end{pmatrix}
\]

Figure 3.10: The effect of Grover's rotation on the state \( |Ψ⟩ \) for multiple marked states.

The initial angle and the rotation angle can be approximated for \( r \ll N \) as

\[ \phi = \sqrt{\frac{r}{N}} \]

and

\[ w = 2 \sqrt{\frac{r}{N}} \]
respectively. Similar analysis to the case of a single marked state shows that the optimal number of iterations of the algorithm is

\[ T = \left\lfloor \frac{\pi}{4} \sqrt{\frac{N}{r}} - \frac{1}{2} \right\rfloor \]  

(3.15)

The algorithm performs \( O\left(\sqrt{\frac{N}{r}}\right) \) iterations, where \( r = |X| \) is the number of marked states. Additional improvements were made in [72] and [74] coping with an unknown number of marked states in complexity \( O\left(\sqrt{\frac{N}{r}}\right) \). [73] introduced a third improvement that outputs a marked state when initiated with a state having an arbitrary amplitude distribution.

### 3.4 Adiabatic Quantum Computation

Adiabatic quantum computation [75, 76, 77, 78] is a method for problem solving through evolutionary computation that relies on the adiabatic theorem [79]. It is based on the adiabatic approximation method of quantum mechanics for the solution of Schrödinger equation. Adiabatic approximation states that, if a quantum system is initiated in its ground state and its Hamiltonian varies slowly, it remains close to the ground state. In the next two subsections we introduce the quantum adiabatic theorem and the basics of the adiabatic quantum computation algorithm.

#### 3.4.1 The Quantum Adiabatic Theorem

A quantum mechanical system’s evolution in time is specified by its Hamiltonian. Let \( |\Psi(t)\rangle \) denote the state of a system at time \( t \) under the influence of a time-dependant Hamiltonian \( H(t) \), then the time evolution of the system can be described by the time-dependant Schrödinger equation:

\[ i\hbar \frac{d}{dt} |\Psi(t)\rangle = H(t) |\Psi(t)\rangle \]  

(3.16)

where

\[ \hbar = \frac{h}{2\pi} \]  

(3.17)

and \( h \) is Planck’s constant.

The Hamiltonian can be viewed as a Hermitian matrix, whose eigenvectors are the different eigenstates of the system and its eigenvalues are their corresponding energies. The ground state of the system is the eigenvector that corresponds to the lowest eigenvalue of the Hamiltonian.

The evolution of the Hamiltonian from its initial value \( H(0) \) to its final value \( H(T) \) describes the transformation acting on the system. The quantum adiabatic theorem states that a system that is initially in its ground state remains in the lowest energy state if its Hamiltonian varies
'slowly enough'. Consider the parameterizing of the Hamiltonian over time by $H(s)$, where $0 \leq s \leq 1$, then the system evolves from the ground state $|\Psi(0)\rangle$ of $H(0)$ to the ground state $|\Psi(1)\rangle$ of $H(1)$, through the Hamiltonians $H(t/T)$ at time $t$.

In order to guarantee the system evolves slowly enough one should consider

$$T \gg \left\| \frac{d}{dt} H(s) \right\|_2 \frac{g^2(s)}{g^2(s)}$$

where $g(s)$ is the gap between the two smallest eigenvalues of $H(s)$. Eq. 3.18 can be replaced by an upper bound of the right hand side of the inequality, yielding

$$T \gg \max_s \left\| \frac{d}{dt} H(s) \right\|_2 \frac{g^2(s)}{\min_s g^2(s)}$$

3.4.2 Adiabatic Quantum Computation Algorithm

Adiabatic quantum computation is the evolutionary process of a system from its initial state described by the Hamiltonian $H_i$ to its final state described by $H_f$, where the eigenstates of $H_i$ are easy to compute and the ground state of $H_f$ is the solution of the computation problem [77].

According to [77, 78], one can define the time-dependant Hamiltonian of the system as the linear combination

$$H(t) = (1 - \frac{t}{T})H_i + \frac{t}{T}H_f$$

for $0 \leq t \leq T$. The system maps the ground state of $H_i$, $|\Psi(0)\rangle$, to the solution of the computation problem if $T$ satisfies Eq. 3.18.

In general, it was shown [80] that the power of adiabatic quantum computation is equivalent to the power of digital quantum computations.

3.4.3 Quantum Hebbian Interactions through Adiabatic Computation

Quantum Hebbian neural networks were suggested in [37, 38, 39, 40, 41, 42, 81, 82] using the adiabatic quantum computation algorithm [75, 76, 77, 78]. Farhi et al. [77] presented an adiabatic computation algorithm that can realize excitatory and inhibitory interactions (cf. [77], bit agree and bit disagree) between two qubits in constant time. Furthermore, an $n$-qubits Hebbian network with inhibitory and excitatory interactions was shown [77, 37] to evolve to the solution in polynomial time. The Hopfield energy function was used in [37, 38, 39, 40, 41, 42] to encode the final Hamiltonian of the system in quadratic time [77]. An easily constructed initial Hamiltonian is smoothly changed to a final Hamiltonian that encodes the Hebbian relations.
between the neurons of the network. The solution is the ground state of the final Hamiltonian and is a superposition of assignments that yield minimal values of the Hopfield energy. The evolution should be slower than \( \min_s g^{-2}(s) \) according to Eq. 3.19, where \( g(s) \) is the energy gap between the two lowest energy levels at time \( s \).

The adiabatic quantum algorithm coincides with the optimization problem solved by the Hopfield network [57, 83] by minimizing the Hopfield energy function embedding the Hebbian weight matrix. However, this quantum model does not present any improvement over the classical Hopfield network. The evolution time of the adiabatic quantum model is polynomial and is, thus, not faster than the classical model, which is known to be linear [57, 58]. Furthermore, it is not clear how to use the adiabatic quantum computation algorithm with Hebbian interactions as a model of associative memory with pattern completion and correction abilities. The initial state should always be the ground state of the initial Hamiltonian and no pattern retrieval can be accomplished. Therefore, the quantum Hebbian adiabatic model can realize the Hopfield network for optimization problems [57, 83], not for associative memory, and presents no computational advantage over it.
Chapter 4
Quantum Associative Memory Based on Quantum Search

Associative memory stores and retrieves patterns with error correction or pattern completion of the input. The task can be defined as memorizing $m$ pairs of data $(x^i, y^i)$, where $x^i$ is an $n$ dimensional vector and $y^i$ is a $q$ dimensional vector, and outputting $y^i$ when presented with $\tilde{x}$, which is a faulty or a partial version of $x^i$. A specific case of associative memory is the auto-associative memory, in which $y^i \equiv x^i, \forall i \in \{1, \ldots, m\}$. Associative memory can be defined over a continuum, where $(x^i, y^i) \in \mathbb{R}^n \times \mathbb{R}^q$, or over a binary space, where $(x^i, y^i) \in \{0, 1\}^n \times \{0, 1\}^q$ or $(x^i, y^i) \in \{\pm 1\}^n \times \{\pm 1\}^q$. In this work, we concentrate on the binary model.

Algorithms for implementation of associative memory [52, 65] have been extensively studied in the neural networks literature [52, 60, 84, 63, 85, 65, 47, 46, 86, 87]. The Hopfield model [52] consisting of $n$ threshold neurons stores $n$-dimensional patterns $x \in \{\pm 1\}^n$ by the sum of outer products and retrieves a stored pattern when presented with a partial or a noisy version of the pattern. The maximal storage capacity for which the stored patterns will be retrieved correctly with high probability [60] is

$$M_{\text{max}} = \frac{n}{2 \ln n}$$

giving $N \equiv 2^n$ possible $n$-bit patterns. A similar version of the asynchronous Hopfield model with synchronous updates is the Little model [88, 89]. In this model, all neurons are updated simultaneously in the retrieval phase. The model then converges either to a stable point or to a limit cycle of length 2 [90, 91].

Sparse encoding has been shown to increase the storage capacity considerably [63]. A capacity exponential in the input dimension has been shown to result from a network size also exponential in the input dimension [64].

In this chapter we present a model of associative memory that relies on Grover’s quantum search algorithm [2], which has been shown to possess advantages over the best possible classical
search algorithm. We define each quantum neuron in the network as a single qubit. The model requires a number of qubits that equals the number of bits in a pattern. We show that the model stores and retrieves a number of patterns that is exponential in the number of qubits, as well as, complete patterns with missing bits and correct patterns with faulty bits. The model presents the concept of memory as a quantum operator rather than a state of superposition of the memory patterns. In addition, there is no requirement for a minimal distance between stored patterns, since the quantum model chooses any of a number of closest memory patterns unlike classical models that might produce spurious memories. All stored patterns are equilibrium points of the model and spurious memories arise with very low probability, which asymptotically vanishes as the number of bits of a pattern grows.

In the rest of the chapter we present our algorithm for quantum set intersection based on the quantum search algorithm and then present the quantum associative memory and show the retrieval, pattern completion, and pattern correction algorithms and prove them.

4.1 Quantum Intersection

Given a set of marked states $K$ of size $k$, Grover’s quantum search algorithm for multiple marked states yields any member of the set with probability $O(1/k)$ when given a phase version $I_K$ of an oracle $f_K$ of the form

$$f_K(x) = \begin{cases} 1, & x \in K \\ 0, & x \notin K \end{cases} \quad (4.1)$$

Consequently, given a phase oracle $I_K$, Grover’s algorithm chooses a member of the subset $K$. Suppose that, in addition to the oracle $f_K$, we have an oracle $f_M$, such that

$$f_M(x) = \begin{cases} 1, & x \in M \\ 0, & x \notin M \end{cases} \quad (4.2)$$

where $M$ is another set of size $m$ of marked states.

We define the problem of quantum intersection as the choice of any member of the intersection set $K \cap M$ of size $r$ with probability $O(1/r)$.

A straightforward algorithm for finding a member of the intersection between two sets of marked states $K$ and $M$, based on the oracles $f_K$ and $f_M$ involves the use of the intersection oracle, comprising the oracles in Eq. 4.1, Eq. 4.2 in sequence, and a Toffoli gate as depicted in Fig. 4.1.

When the input to the oracle is $|x\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$, then the output is $|x\rangle \otimes |f_K(x)\rangle \otimes |f_M(x)\rangle \otimes |f_K \cap_M(x)\rangle$. The intersection phase oracle is realized when $|b\rangle = |\rangle$, then, the input $|x\rangle \otimes |0\rangle \otimes |0\rangle \otimes |\rangle$ will cause the output to be $|x\rangle \otimes |f_K(x)\rangle \otimes |f_M(x)\rangle \otimes \left[ (-1)^{f_K \cap_M(x)} |\rangle \right]$. 

27
Figure 4.1: The intersection oracle $f_{K \cap M}$ created using $f_M$, $f_K$, and Toffoli.

Retrieving a state in the intersection between the two sets is accomplished by using Grover’s quantum search algorithm with the phase version $I_{K \cap M}$ of the oracle $f_{M \cap K}$.

However, as we consider the dynamics of a neural network for associative memory in the next sections, we would like the activity in the network in each step to relate either to the input pattern or to the stored memory set. Therefore, we present an algorithm that applies a series of computations carried out by the two owners of the oracles in an alternating fashion and prove that it also computes the intersection of the two given sets.

**Algorithm 1 : Quantum Set Intersection**

*Given: Phase oracles $I_M$ and $I_K$*

*Denote: $Q_M \equiv -HI_0HI_M$, $Q_K \equiv -HI_0HI_K*

1. Let $|\Psi\rangle = H|0\rangle^\otimes n$

2. Repeat

   $|\Psi\rangle = Q_K |\Psi\rangle$

   $|\Psi\rangle = Q_M |\Psi\rangle$

   $T = O\left(\sqrt{\frac{N}{|K \cap M|}}\right)$ times

3. Measure $|\Psi\rangle$

Algorithm 1 assumes that the size of the intersection set $|K \cap M|$ is known in order to determine the number of iterations. In the more general case where $|K \cap M|$ is unknown we apply the modification for an unknown number of marked states [72]. Alternatively, we can use the quantum counting algorithm [74] for the apriori estimation of the number of marked states. In both cases the time complexity of the set intersection algorithm is sub-exponential.
Next, we present two theorems that prove that Algorithm 1 measures a member of the intersection set with high probability in sub-exponential time.

**Theorem 1** Let $I_K$ and $I_M$ be phase oracles that mark two sets of $n$-qubit states $K$ and $M$ with $|K|, |M| < N/2$. Let us denote $|K| = k$, $|M| = m$, $|K \cap M| = r$,

$$Q \equiv Q_M Q_K = (HI_0 HI_M HI_0 HI_K) \quad (4.3)$$

and

$$|\Psi(t)\rangle = Q^T H |0\rangle$$

Then, the maximal probability of measuring a state in the intersection $K \cap M$ is achieved at

$$T = \arg \max_t \sum_{x \in K \cap M} |\langle x|\Psi(t)\rangle|^2 = \left| \frac{\pi/2 - \arctan \left( \sqrt{\frac{r}{N-r}} \right)}{\arccos \left( \frac{4km}{N^2} - \frac{4r}{N} - \Gamma \right)} \right| \quad (4.4)$$

where

$$\Gamma = \sqrt{1 - \frac{8rN^3 + 8kmN^2 - 16rkN^2 - 16rmN^2 + 32rkmN - 16k^2m^2}{N^4}} \quad (4.5)$$

**Proof of Theorem 1** The Hilbert space spanned by all computational basis states of the $n$-qubit register can be divided into four different subspaces: the subspace spanned by the states in $K \cap M$, the subspace spanned by the states in $K \setminus M$, the subspace spanned by the states in $M \setminus K$, and the subspace spanned by the states in $K \cup M$. Accordingly, we select an orthonormal basis for representing the operator, in which the first four basis states are:

$$|l_1\rangle \equiv \frac{1}{\sqrt{|K \cap M|}} \sum_{i \in K \cap M} |i\rangle \quad (4.6)$$

$$|l_2\rangle \equiv \frac{1}{\sqrt{|K \setminus M|}} \sum_{i \notin K \cap M} |i\rangle \quad (4.7)$$

$$|l_3\rangle \equiv \frac{1}{\sqrt{|M \setminus K|}} \sum_{i \notin K \setminus M} |i\rangle \quad (4.8)$$

$$|l_4\rangle \equiv \frac{1}{\sqrt{|K \cup M|}} \sum_{i \notin K \cup M} |i\rangle \quad (4.9)$$

and the rest of the basis consists of orthonormal extensions of these states, then

$$H |0\rangle = \sqrt{\frac{r}{N}} |l_1\rangle + \frac{k-r}{N} |l_2\rangle + \sqrt{\frac{m-r}{N}} |l_3\rangle + \frac{N-m-k+r}{N} |l_4\rangle$$

The operator $Q_K$ affects only the four states ($|l_1\rangle, |l_2\rangle, |l_3\rangle, |l_4\rangle$) as follows:
\[ Q_K |l_1\rangle = -HI_0H|l_1\rangle = -H \left( I - 2 |0\rangle \langle 0| \right) H \left( I - 2 \sum_{i \in K} |i\rangle \langle i| \right) |l_1\rangle = \\
\left( 1 - \frac{2r}{N} \right) |l_1\rangle + \left( -\frac{2\sqrt{r(k - r)}}{N} \right) |l_2\rangle + \\
\left( -\frac{2\sqrt{(m - r)(m - r)}}{N} \right) |l_3\rangle + \left( -\frac{2\sqrt{(N - m - k + r)}}{N} \right) |l_4\rangle \]

\[ Q_K |l_2\rangle = -HI_0H|l_2\rangle = -H \left( I - 2 |0\rangle \langle 0| \right) H \left( I - 2 \sum_{i \in K} |i\rangle \langle i| \right) |l_2\rangle = \\
\left( -\frac{2\sqrt{r(k - r)}}{N} \right) |l_1\rangle + \left( 1 - \frac{2(k - r)}{N} \right) |l_2\rangle + \\
\left( -\frac{2\sqrt{(k - r)(m - r)}}{N} \right) |l_3\rangle + \left( -\frac{2\sqrt{(k - r)(N - m - k + r)}}{N} \right) |l_4\rangle \]

\[ Q_K |l_3\rangle = -HI_0H|l_3\rangle = -H \left( I - 2 |0\rangle \langle 0| \right) H \left( I - 2 \sum_{i \in K} |i\rangle \langle i| \right) |l_3\rangle = \\
\left( \frac{2\sqrt{r(m - r)}}{N} \right) |l_1\rangle + \left( \frac{2\sqrt{(k - r)(m - r)}}{N} \right) |l_2\rangle + \\
\left( \frac{2(m - r)}{N} - 1 \right) |l_3\rangle + \left( \frac{2\sqrt{(m - r)(N - m - k + r)}}{N} \right) |l_4\rangle \]

\[ Q_K |l_4\rangle = -HI_0H|l_4\rangle = -H \left( I - 2 |0\rangle \langle 0| \right) H \left( I - 2 \sum_{i \in K} |i\rangle \langle i| \right) |l_4\rangle = \\
\left( \frac{2\sqrt{r(N - m - k + r)}}{N} \right) |l_1\rangle + \left( \frac{2\sqrt{(k - r)(N - m - k + r)}}{N} \right) |l_2\rangle + \\
\left( \frac{2\sqrt{(m - r)(N - m - k + r)}}{N} \right) |l_3\rangle + \left( \frac{2(N - m - k + r)}{N} - 1 \right) |l_4\rangle \]

yielding \( Q_K \) in matrix form

\[
Q_K = \begin{pmatrix}
1 - \frac{2r}{N} & -2\sqrt{\frac{r(k - r)}{N}} & 2\sqrt{\frac{r(m - r)}{N}} & 2\sqrt{\frac{r(N - k - m + r)}{N}} \\
-2\sqrt{\frac{(k - r)(m - r)}{N}} & 1 - \frac{2(k - r)}{N} & 2\sqrt{\frac{(k - r)(N - m - k + r)}{N}} & 2\sqrt{\frac{k - r}{N}} \\
-2\sqrt{\frac{(m - r)(N - m - k + r)}{N}} & -2\sqrt{\frac{(k - r)(m - r)}{N}} & 2\sqrt{\frac{(m - r)}{N}} - 1 & 2\sqrt{\frac{m - r}{N}} \\
-2\sqrt{\frac{(N - m - k + r)}{N}} & 2\sqrt{\frac{(k - r)(m - r)}{N}} & 2\sqrt{\frac{(N - k - m + r)}{N}} - 1 & 1 - \frac{2(r)}{N}
\end{pmatrix} \tag{4.10}
\]

Similarly, we obtain the matrix form of $Q_M$

$$Q_M = \begin{pmatrix}
1 - \frac{2r}{N}
& 2\sqrt{\tau}k - r
& -2\sqrt{\tau}m + r
& 2\sqrt{\tau}N - k + m + r \\
-2\sqrt{\tau}k - r
& \frac{2(k+r)}{N} - 1
& -2\sqrt{\tau}m - r
& 2\sqrt{\tau}N - k + m + r \\
-2\sqrt{\tau}m - r
& \frac{2(k-r)}{N} - 1
& -2\sqrt{\tau}m - r
& 2\sqrt{\tau}N - k + m + r \\
-2\sqrt{\tau}N - k + m + r
& 2\sqrt{\tau}N - k + m + r
& -2\sqrt{\tau}N - k + m + r
& 2\sqrt{\tau}N - k + m + r
\end{pmatrix} \quad (4.11)$$

Substituting Eq. 4.10 and Eq. 4.11 into Eq. 4.3 yields

$$Q = \begin{pmatrix}
1 - \frac{8r(N-m)}{N^2}
& -4\sqrt{\tau}k - r(N-2m)
& 4\sqrt{\tau}N - k + m + r(N-2m)
& 0 \\
-4\sqrt{\tau}k - r(N-2m)
& \frac{8m(k-r)}{N^2} - 1
& -8m\sqrt{\tau}N - k + m + r(N-2m)
& 0 \\
-8\sqrt{\tau}m - r(N-m)
& -4\sqrt{\tau}N - k + m + r(N-2m)
& 8m\sqrt{\tau}N - k + m + r(N-2m)
& 0 \\
-4\sqrt{\tau}N - k + m + r(N-2m)
& 0
& 0
& 1 - \frac{8m(N-k-m+r)}{N^2}
\end{pmatrix}$$

The compound operator $Q$ is a rotation in the 4-dimensional space spanned by Eq. 4.6 - Eq. 4.9. Finding the rotation angles requires the diagonalized matrix of $Q$. Let $V$ be the matrix whose columns are the eigenvectors of $Q$, then $Q^D = V^{-1}QV$ is a diagonal matrix whose diagonal components are the eigenvalues of $Q$ as follows:

$$Q^D = \begin{pmatrix}
e^{-iw_1}
& 0
& 0
& 0 \\
e^{iw_1}
& 0
& 0
& 0 \\
0
& 0
& e^{-iw_2}
& 0 \\
0
& 0
& 0
& e^{iw_2}
\end{pmatrix}$$

where

$$e^{-iw_1} = \frac{4km}{N^2} - \frac{4r}{N} - \Gamma - \Delta$$
$$e^{iw_1} = \frac{4km}{N^2} - \frac{4r}{N} - \Gamma + \Delta$$
$$e^{-iw_2} = \frac{4km}{N^2} - \frac{4r}{N} + \Gamma - \Delta$$
$$e^{iw_2} = \frac{4km}{N^2} - \frac{4r}{N} + \Gamma + \Delta$$

and $\Gamma$ and $\Delta$ are given by

$$\Gamma = \sqrt{1 - \frac{8rN^3 + 8kmN^2 - 16krN^2 - 16kmN^2 + 32rkmN - 16k^2m^2}{N^4}}$$
$$\Delta = 2\sqrt{\frac{2N^2r(r + k + m) - N^3r(1 + \Gamma) + N^2km\Gamma - N^2km - 8Nkmr + 4k^2m^2}{N^4}}$$

31
which implies

\[ w_1 = \arccos \left( \frac{4km}{N^2} - \frac{4r}{N} - \Gamma \right) \] (4.12)

\[ w_2 = \arccos \left( \frac{4km}{N^2} - \frac{4r}{N} + \Gamma \right) \]

The amplitude of \(|l_1\rangle\) is given by

\[ a(t) = A \sin (w_1 t + \phi) \] (4.13)

where \(A\) is the maximal amplitude, to be found, and \(\phi\) is the angle between the initial state \(H |0\rangle^\otimes n\) and \(|l_1\rangle\):

\[ \phi = \arctan \sqrt{\frac{r}{N-r}} \] (4.14)

The maximal probability \(A^2\) that a measurement of the system will produce a state in \(K \cap M\) will be obtained at time

\[ T = \arg \max_t A^2 \sin^2 (w_1 t + \phi) \]

yielding

\[ T = \left[ \frac{\pi/2 - \arctan \left( \sqrt{\frac{r}{N-r}} \right)}{\arccos \left( \frac{4km}{N^2} - \frac{4r}{N} - \Gamma \right)} \right] \] (4.15)

as asserted (Eq. 4.4).

Fig. 4.2 shows the probability of measuring a state in the intersection between \(M\) and \(K\) (solid line) and the probability of measuring a state not in the intersection (dotted line) vs. the number of iterations. The maximal probability is achieved at iteration \(T\) given by Eq. 4.15.

Theorem 1 suggests a way for approximating the time complexity of the algorithm when \(m, k < N/2\) and \(r << N\). Since \(r << N\) we can approximate \(\Gamma\) as follows

\[ \Gamma = \sqrt{4 \left( \frac{2km}{N^2} - \frac{1}{2} \right)^2 - \frac{8r}{N} \left( 1 - \frac{2k}{N} \right) \left( 1 - \frac{2m}{N} \right)} \approx \frac{4km}{N^2} - 1 \] (4.16)

We substitute Eq. 4.16 in Eq. 4.12 to get the angle of the rotation

\[ w_1 \approx \arccos \left( 1 - \frac{4r}{N} \right) \] (4.17)

Employing the Taylor series, the second order approximation of the rotation angle is

\[ w_1 = O \left( \sqrt{\frac{r}{N}} \right) \] (4.18)
and the number of iterations can be approximated by
\[ T \approx O \left( \frac{\pi/2 - \sqrt{\frac{N-r}{N}}}{\sqrt{\frac{r}{N}}} \right) = O \left( \sqrt{\frac{N}{r}} \right) \quad (4.19) \]

In the following, we prove a lower bound on the probability of measuring a state in the intersection when \( K \) and \( M \) are considerably large. Different lower bounds can be achieved similarly.

**Theorem 2** The maximal probability, \( A^2 \) from Eq. 4.13, of measuring a marked state in \( K \cap M \) is greater than
\[ \left( 3 - \frac{8r}{N} \right)^2 \quad (4.20) \]
when \( |K|, |M| \leq N/4, r \ll N, \) and \( N \) is large.

**Proof of Theorem 2** The amplitude \( a(t) \) of \( |l_1 \rangle \) behaves as in Eq. 4.13 and the maximal amplitude \( A \) can be obtained from
\[ a(t) = A \sin(w_1 t + \phi) = (Q^t H |0\rangle)_1 \quad (4.21) \]
where \((Q^t H |0\rangle)_1\) is the first element in \( Q^t H |0\rangle \). Substituting \( t = 1 \), we get
\[ A = \frac{(Q H |0\rangle)_1}{\sin(w_1 + \phi)} \quad (4.22) \]
The nominator of the right hand side of Eq. 4.22 can be bounded from below by

$$
(QH|0)_{1} = \sqrt{\frac{r}{N}} \left( 1 - \frac{8r(N - m)}{N^2} \right) + \sqrt{\frac{k - r}{N}} \left( -4\sqrt{r} \sqrt{k - r(N - 2m)} \right) + \sqrt{\frac{m - r}{N}} \left( 8\sqrt{r} \sqrt{m - r(N - m)} \right) + \sqrt{\frac{N - m - k + r}{N}} \left( 4\sqrt{r} \sqrt{N - m - k + r(N - 2m)} \right)
$$

$$
= \frac{\sqrt{r}}{\sqrt{NN^2}} (5N^2 - 4Nm - 8Nk - 8Nr + 16mk)
$$

$$
> \sqrt{\frac{r}{N}} \left( 3 - 8 \frac{r}{N} \right)
$$

and the denominator can be estimated by

$$
w + \phi = \arccos \left( \frac{\Gamma + 4km}{N^2} - \frac{4r}{N} \right) + \frac{r}{N - r}
$$

$$
\approx \arccos \left( 1 - \frac{4r}{N} \right) + \frac{r}{N - r}
$$

$$
\approx \sqrt{8} \sqrt{\frac{r}{N}} + \sqrt{\frac{r}{N - r}} \approx (\sqrt{8} + 1) \sqrt{\frac{r}{N}}
$$

Substituting Eq. 4.23 and Eq. 4.24 into Eq. 4.22 we get

$$
A > 3 - \frac{8r}{\sqrt{8} + 1}
$$

and the probability of measuring a marked state is, therefore, bounded from below by

$$
A^2 > \left( 3 - \frac{8r}{\sqrt{8} + 1} \right)^2
$$

(4.26)

Theorem 2 gives the lower bound

$$
A^2 > \frac{9}{9 + 2\sqrt{8}} \approx 61.4\%
$$

(4.27)

on the probability of measuring a state in the intersection between $M$ and $K$ when $|M|, |K| \leq N/4$. Fig. 4.3 shows the success probability vs. the ratio $|M|/N$ for $|K| = |M|$. It can be seen from the figure that the success probability is asymptotically 1 when $|M| = |K| << N$ and remains above the lover bound given in Eq. 4.27 for $|M| = |K| = N/4$.

Based on the quantum set intersection (Algorithm 1 or Fig. 4.1) we now introduce our associative memory model and the retrieval procedures with pattern completion and correction abilities.
Let $I_M$ be a phase oracle on a set $M$, called the memory set, of $m$ $n$-qubit patterns and let $x'$ be a version of a memory pattern $x \in M$ with $d$ missing bits. We are required to output the pattern $x$ based on $I_M$ and $x'$. The partial pattern is given as a string of binary values 0 and 1 and some unknown bits marked '?'. Denoting the set of possible completions of the partial pattern $K$ and its size $k$, the completion problem can be reduced to the problem of retrieving a member $x$ of the intersection between two sets $K$ and $M$, $x \in K \cap M$. For example, let $M = \{0101010, 0110100, 1001001, 1111000, 1101100, 1010101, 0000111, 0010010\}$ be a 7-bit memory set of size 8 and let ”0110?0?” be a partial pattern with 2 missing bits, so the completion set is $K = \{0110000, 0110001, 0110100, 0110101\}$. Pattern completion is the computation of the intersection between $K$ and $M$, which is the memory pattern 0110100.

Pattern completion can use either the intersection oracle presented in Fig. 4.1 or the quantum intersection presented in Algorithm 1. In either case, we need to create the completion oracle $f_K$, which can be implemented by checking whether a state is a completion of the partial pattern $x'$. The algorithm for pattern completion through the quantum intersection algorithm is

Algorithm 2 : Quantum Pattern Completion

*Given: A memory operator $I_M$ and a pattern $x' \in \{0,1\}^n$, which is a partial version*
of some memory pattern with up to \( d \) missing bits

1. Create the completion phase oracle \( I_K \).
2. Apply Algorithm 1 with \( I_M \) and \( I_K \)

A generalization to the case of an unknown number of possible completions is straight forward using quantum search for an unknown number of marked states [72] or quantum counting [74].

4.3 Pattern Correction

Let \( I_M \) be a phase oracle of a memory set \( M \) of size \( m \) and let \( x' \) be a version of a memory pattern \( x \in M \) with up to \( d \) faulty bits. We are required to output the pattern \( x \) based on \( I_M \) and \( x' \). The set \( K \) of possible corrections of the faulty pattern consists of all patterns in Hamming distance up to \( d \) from \( x' \). The correction problem can be reduced to the problem of retrieving a member \( x \) of the intersection between two sets \( K \) and \( M \), \( x \in K \cap M \). For example, let \( M = \{0101010, 0110100, 1001001, 1111000, 1101100, 1010101, 0000111, 0010010\} \) be a 7-bit memory set and let “0110001” be the input pattern with 2 possible errors. The correction set \( K \) consists of all patterns that are in Hamming distance up to 2 from \( x' \). Pattern correction should then retrieve the memory pattern 0110100.

Pattern correction can be solved using the quantum intersection algorithm. It requires the creation of the correction oracle \( f_K \), which can be implemented by checking whether the number of different bits between any given state and the faulty pattern \( x' \) is less or equal to \( d \).

Algorithm 3 : Quantum Pattern Correction

Given: A memory operator \( I_M \) and a pattern \( x' \in \{0,1\}^n \), which is a faulty version of some memory pattern with up to \( d \) faulty bits

1. Create the correction phase oracle \( I_K \).
2. Apply Algorithm 1 with \( I_M \) and \( I_K \)

A generalization of Algorithm 3 for the case of unknown number of possible corrections is straight forward using quantum search for an unknown number of marked states [72] or quantum counting [74].

Algorithm 3 finds a memory pattern that is in Hamming distance up to \( d \) from \( x' \). If the memory is within the correction capacity bounds, Algorithm 3 finds the correct pattern \( x \) with high probability, as will be proved in Section 4.4. However, if we are interested in ensuring that
we find the closest memory pattern to \(x',\) with no dependence on the capacity bound, then we can apply Algorithm 3 for \(i = 0\) bits and increase it up to \(i = d\) bits. In this case we ensure that we find a pattern \(x,\) such that
\[
\{x : (x \in M) \land (\forall x'' \in M : \text{dist}(x'', x) \geq \text{dist}(x', x))\}
\]
where \(\text{dist}(\cdot, \cdot)\) is the Hamming distance.

### 4.4 Analysis of the Quantum Associative Memory

In this section we analyze the time complexity and memory capacity of the proposed quantum associative memory. We first show that the time complexity of retrieval operations is sub-exponential in the number of bits. Then we show that the number of memory patterns that can be stored while the model retains its correction and completion abilities is exponential in the number of bits.

#### 4.4.1 Time Complexity Analysis

The time complexity of the retrieval procedure with either pattern completion or correction ability is determined by the complexity of the quantum intersection algorithm and the complexity of the completion and correction oracles. The two oracles can be implemented by a number of operations which is linear in \(n\) or in \(n \log n\), as shown in Sec. 4.8. According to Eq. 4.19, the completion and correction operations are performed in
\[
T \approx n \log n + O \left( \sqrt{\frac{N}{|K \cap M|}} \right) = O \left( \sqrt{\frac{N}{|K \cap M|}} \right)
\]
operations, which is sub-exponential in number of bits.

#### 4.4.2 Capacity Analysis

We consider three different capacity measures. The first is the equilibrium capacity \(M_{eq}\), which is the maximal memory size that ensures that all memory patterns are equilibrium points of the model. An Equilibrium point is a pattern that, when presented to the model as an input, is also retrieved as an output with high probability. The second is the pattern completion capacity \(M_{com}\), which is the maximal memory size that allows the completion of any partial pattern with up to \(d\) missing bits with high probability. The third is the pattern correction capacity \(M_{cor}\), which is the maximal memory size that allows the correction of any pattern with up to \(d\) faulty bits with high probability.
Equilibrium is a special case of completion and correction with neither missing nor faulty bits. Therefore, the equilibrium capacity should be equal to the completion and correction capacities for \( d = 0 \).

**Equilibrium Capacity**

The following theorem concerns the equilibrium capacity and states that any number of patterns can be stored and retrieved. The number of \( n \)-bit patterns is \( N = 2^n \), therefore, \( M_{eq} = N \).

**Theorem 3** Every memory state of an \( n \)-bit associative memory \( M \) of any size \( m \leq N \) is an equilibrium state, i.e., If \( Q_x = -(HI_0HI_x) \), \( T = \left\lfloor \frac{\pi}{4} \sqrt{N} \right\rfloor \), and \( |\Psi(T)\rangle = Q_x^T H |0\rangle \), then

\[
\forall x \in M : |\langle x |\Psi(T)\rangle|^2 \to 1 \text{ as } n \to \infty
\]

**Proof of Theorem 3** This is a direct consequence of Grover’s algorithm [2] and the results obtained by [72] concerning the ability to find any member of the size \( N \) database with probability close to 1.

**Completion Capacity**

Given a pattern \( x' \), which is a partial version of some memory pattern \( x_c \) with \( d \) missing bits, we seek the maximal memory size, for which the pattern can be completed with high probability from a random uniformly distributed memory set.

The completion capacity is bounded from above by two different bounds. The first is a result of Grover’s quantum search algorithm limitations and the second is a result of the probability of correct completion.

**A Bound on Memory Size due to Grover’s Quantum Search Limitations**

Grover’s operator flips the marked states around the zero amplitude (negating their amplitudes) then flips all amplitudes around the average of all amplitudes [73]. Amplification of the desired amplitudes occurs only when the average of all amplitudes is closer to the amplitudes of the non-marked states than to the marked states. This imposes the following upper bound on the memory size:

\[
m < N/2
\]

This can be observed in the first iteration of the quantum search algorithm on a number of marked states. The initial amplitude of all basis states in \( H |0\rangle \) is \( 1/\sqrt{N} \). Flipping the phase of \( m \) marked states by \( I_M \) to \(-1/\sqrt{N}\) yields an average amplitude of
\[(N - m) \ast \left(\frac{1}{\sqrt{N}}\right) - m \ast \left(\frac{1}{\sqrt{N}}\right) = \frac{N - 2m}{\sqrt{N}}\]

Then, flipping the phases of all basis states around this average by \(HI_0H\) yields two values of amplitudes. The amplitudes of marked and unmarked states become

\[2 \left(\frac{N - 2m}{\sqrt{N}}\right) \pm \frac{1}{\sqrt{N}} = \left(\frac{2N - 4m \pm 1}{\sqrt{N}}\right)\]  

(4.28)

where \(\pm\) correspond to marked and unmarked states respectively.

A necessary condition for the amplification of marked states is that the absolute value of their amplitudes after an iteration of the algorithm is higher than the absolute value of the amplitude of unmarked states. The condition is satisfied if and only if the two equations given in Eq. 4.28 satisfy

\[\left|\left(\frac{2N - 4m + 1}{\sqrt{N}}\right)\right| > \left|\left(\frac{2N - 4m - 1}{\sqrt{N}}\right)\right|\]

which holds true if and only if \(m < N/2\). Therefore, if \(m \geq N/2\) the amplitudes of the marked states will not increase, which gives the following upper bound on the completion capacity:

\[M_{\text{com}} < \frac{N}{2}\]  

(4.29)

However, this is a very loose bound and the success probability of the completion will impose a tighter bound.

**A Bound on Memory Size due to Pattern Completion**

The bound on memory size that ensures a high probability of correct completion depends on the definition of the pattern completion procedure. If one defines pattern completion as the process of outputting any of a number of possible memory patterns when given a partial input, then the capacity bound of our memory is the amplification bound given in Eq. 4.29. However, this is not always the case. Pattern completion capacity is usually defined as the maximal size of a random uniformly distributed memory set that, given a partial version \(x'\) of a memory \(x_c \in M\) with \(d\) missing bits, outputs \(x_c\). The following theorem gives an upper bound on the capacity for pattern completion with high probability:

**Theorem 4** An \(n\)-bit associative memory with \(m\) random patterns can complete up to \(d\) missing bits on average when

\[m \leq 2^{n - d}\]

with probability higher than

\[
\frac{v}{e^v - 1} \sum_{i=1}^{m} \frac{v^{i-1}}{i^i}
\]

(4.30)
as \( n \) grows to infinity, where

\[
v = \frac{m}{2^{n-d}}
\]

**Proof of Theorem 4** Let \( M \) be a random uniformly distributed memory set of size \( m = v2^{n-d} \), where \( 0 < v < 1 \). Let \( x' \) be a partial pattern of \( x_c \in M \) with \( d \) missing bits. \( x' \) induces the set

\[
K = \{x|x \text{ is a completion of } x'\}
\]

where \( |K| = 2^d \). Let \( Z_i \) be random indicator variables representing the existence of the \( i \)th member of \( M \) in \( K \). Denoting \( p = Pr(Z_i = 1) = \frac{2^d}{2^n} = \frac{1}{2^{n-d}} \), we have

\[
m = \frac{v}{p}
\]

Let us denote \( |\Psi(T)\rangle = Q^T H |0\rangle \) and \( S = \sum_{i=1}^{m} Z_i \). If there is only one possible memory completion, then it is \( x_c \), and if there are two, then \( x_c \) is one of the two, and so on. Therefore, the probability of successfully outputting \( x_c \) from the partial pattern \( x' \) is the sum of the conditional probabilities that there are \( i \) memory completions divided by \( i \):

\[
|\langle x_c | \Psi(T) \rangle|^2 = \sum_{i=1}^{m} \frac{Pr(S = i) Pr(S \geq 1)}{i}
\]

\[
= \sum_{i=1}^{m} \frac{\frac{2^i}{2^n}}{i Pr(S \geq 1)}
\]

\[
= \sum_{i=1}^{m} \frac{\left(\frac{m}{2^n}\right)^i \left(1 - p\right)^{m-i}}{i \left(1 - \left(1 - p\right)^m\right)}
\]

(4.32)

Now, this probability can be lower-bounded by

\[
|\langle x_c | \Psi(T) \rangle|^2 \geq \sum_{i=1}^{m} \frac{\left(\frac{m}{2^n}\right)^i \left(1 - p\right)^{m-i}}{i \left(1 - \left(1 - p\right)^m\right)}
\]

(4.33)

Substituting Eq. 4.31 in Eq. 4.33 we have

\[
|\langle x_c | \Psi(T) \rangle|^2 \geq \sum_{i=1}^{m} \frac{\left(\frac{v}{n}\right)^i \left(1 - p\right)^{v/p-i}}{i \left(1 - \left(1 - p\right)^v\right)}
\]

\[
\geq \sum_{i=1}^{m} \frac{\left(\frac{v}{n}\right)^i e^{-v} \frac{1}{(1-p)^v}}{i \left(1 - e^{-v}\right)}
\]

\[
\geq \frac{1}{e^v - 1} \sum_{i=1}^{m} \frac{v^i}{i}
\]

\[
= \frac{v}{e^v - 1} \sum_{i=1}^{m} \frac{v^{i-1}}{i}
\]

(4.34)
Fig. 4.4 shows that the lower bound with approximation by only three terms of the sum in Eq. 4.30 as a function of $v$ is higher than 75% for all possible sizes of memory within the capacity limits.

![Graph showing success probability vs. memory size](image)

Figure 4.4: Success probability of pattern completion vs. memory size divided by the maximal completion capacity $v$.

Theorem 4 implies that $M_{com}(d) = 2^{n-d}$, which agrees with the result of Theorem 3 concerning the equilibrium capacity, since $M_{com}(0) = 2^{n-0} = 2^n = N = M_{eq}$.

**Correction Capacity**

A bound on the correction capacity of Algorithm 3 is given by the following theorem:

**Theorem 5** An $n$-bit associative memory with $m$ random patterns can correct up to $d$ faulty bits on average when

$$m \leq 2^{n-d}/\binom{n}{d}$$

with probability higher than

$$\frac{v}{e^v - 1} \sum_{i=1}^{m} \frac{v^{i-1}}{i^i} \quad \text{as } n \text{ grows to infinity, where}$$

$$v = \frac{m}{2^{n-d}}$$

**Proof of Theorem 5** Let $M$ be a random uniformly distributed memory set of size

$$m = v2^{n-d}/\binom{n}{d}$$
where $0 < v < 1$. Let $x'$ be a pattern $x_c \in M$ with $d$ faulty bits. $x'$ induces a set

$$D = \{x|\text{dist}(x, x') \leq d\}$$

(4.35)

where $|D| = \binom{n}{d}2^d$. Let $Z_i$ be random indicator variables representing the existence of the $i$th member of $M$ in $D$. Denoting $p = \Pr(Z_i = 1) = \binom{n}{d}2^{d-n}$, we have

$$m = \frac{v}{p}$$

Let us denote $|\Psi(T)\rangle = Q^T H |0\rangle$ and $S = \sum_{i=1}^{m} Z_i$. The probability of successfully retrieving $x_c$ from the pattern $x'$ is then

$$|\langle x_c | \Psi(T) \rangle|^2 = \sum_{i=1}^{m} \frac{\Pr(S = i / S \geq 1)}{i}$$

which, according to Eq. 4.32 - Eq. 4.34, also satisfies

$$|\langle x_c | \Psi(T) \rangle|^2 \geq \frac{v}{e^v - 1} \sum_{i=1}^{m} \frac{v^{i-1}}{i^i}$$

Theorem 5 yields $M_{\text{cor}}(d) = \binom{n}{d}2^{n-d}$, which also agrees with the result of Theorem 3 concerning the equilibrium capacity, since $M_{\text{cor}}(0) = \binom{n}{0}2^{n-0} = 2^n = N = M_{\text{eq}}$.

For example, let $M$ be a memory set over $\{0, 1\}^{100}$, then, as long as $|M| < 2^{80}$, we can complete up to $d = 100 - \log |M| = 20$ bits and correct up to $d = 13$ bits.

**Increasing Memory Size Beyond the Capacity Bounds**

The various capacities presented above are exponential in $n$ under the assumption $d << n$. However, an increase of $m$ beyond the capacity bound results in a decay of the correct completion probability as depicted in Fig. 4.5. It can be seen that it is more likely to find the correct completion than not to find it as long as $v < 2$.

In addition, the model can also output a superposition of a number of possible outputs, by skipping the measurement operation in Algorithm 1. This is not true for most classical memory models where spurious memories arise and the output is usually not a memorized pattern, but, rather, some spurious combination of multiple memory patterns [52, 90, 91].

### 4.5 Controlled Basin of Attraction

The quantum correction algorithm that was presented in Sec. 4.3 differs from other models of associative memory, such as the Hopfield model [52] in the fact that the basin of attraction
Figure 4.5: Pattern completion or correction probability vs. the memory size divided by the maximal capacity $v$. For $0 < v < 1$, the probability is above 75% and for $v < 2$ is above 50%.

for each memory pattern is actually controlled by a parameter of the algorithm. Instead of a large basin of attraction, for which any input is corrected to a memory pattern, the quantum correction algorithm sets the attraction basin to patterns with up to $d$ bits. If the input is not in radius $d$ of any memory pattern then no correction will be given in the output. Since $d$ is a parameter of the algorithm to be determined upon usage, one can define the maximal radius size for which retrieval would be performed. Increasing the radius $d$ until successful retrieval results in a model similar to the Hopfield model, with more collisions of basins. The quantum interpretation of these collisions are a probabilistic conversion to one of the centers of the colliding basins.

4.6 Comparison to Previous Quantum Models

Quantum computation was previously applied to associative memory by [30], [32], [31], and, subsequently, by others. An algorithm based on the model developed by [30] was proposed by [34]. It was further developed by [35] and analyzed by [36]. We analyze the two main algorithms [30, 35] and show their differences with respect to our algorithm. These algorithms are given below.

Algorithm 4 The algorithm proposed by [30]

Given: Phase oracles $I_M$ and $I_K$
1. Denote $Q_M = -HI_0HI_M$ and $Q_K = -HI_0HI_K$
2. Let $|\Psi\rangle = \frac{1}{m} \sum_{i=1}^{m} |i\rangle$.
3. Apply $Q_M Q_K$ on $|\Psi\rangle$.
4. Apply $Q_K$ on $|\Psi\rangle$ for $T = \left\lfloor \pi/4\sqrt{N/|K \cap M|} \right\rfloor - 2$ times.
5. Measure $|\Psi\rangle$.

Algorithm 5 The algorithm proposed by [35]

Given: Phase oracles $I_M$ and $I_K$
1. Denote $Q_M = -HI_0HI_M$ and $Q_K = -HI_0HI_K$
2. Let $|\Psi\rangle = \frac{1}{m} \sum_{i=1}^{m} |i\rangle$.
3. Apply $Q_M Q_K$ on $|\Psi\rangle$ for $T$ times. ($T$ was not found by [35])
4. Measure $|\Psi\rangle$.

Algorithm 4 can find only a single marked state with high probability when the memory size $m$ is close to $\frac{N}{4} - 2$, as shown by the solid line in Fig. 4.6. The probability of measuring this state reduces by a half when there are two marked states and only one of them is a memory pattern, as shown by the dashed line in Fig. 4.6, and so on.

![Figure 4.6: Memory size vs. success probability of Algorithm 4. Optimal results are achieved only when the memory size is close to $\frac{N}{4}$](image-url)
Algorithm 5 gives satisfying results only when the memory size exceeds $\frac{N}{4}$. The success probability of Algorithm 5 vs. the memory size is depicted in Fig. 4.7. [36] added a control parameter to tune the algorithm, changing the memory size for which the maximal amplitude is achieved. The algorithm is presented only for one marked state with no completion and correction abilities. The time complexity and stopping criteria were not stated by [34] and were later found to be $O\left(\sqrt{N}\right)$ [35, 36].

![Figure 4.7: Memory size vs. success probability in Algorithm 5. Satisfactory results are achieved only when the memory size exceeds $\frac{N}{4}$.](image)

Our algorithm, on the other hand achieves high success probability up to memory size $\frac{N}{4}$, as depicted in Fig. 4.8.

Furthermore, both Algorithm 4 and Algorithm 5 need to initialize the system at a superposition of the memory states:

$$|\Psi\rangle = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} |i\rangle$$

which is a shortcoming for two reasons: the time complexity of initialization when the memory size is large and the need for repeated initialization upon every application of the memory. The latter is important as it adds an exponential factor to the query time, for either completion or correction, and an exponential addition to the single query time when amplitude amplification is needed. Amplitude amplification ensures that we pick the correct pattern with probability 1 by performing the algorithm a multiple number of times. Our algorithm’s initialization, on the other hand, does not depend on the memory patterns.
4.7 Computational Advantage over Classical Set Intersection

The classical equivalent of our quantum computational model is required to find an $n$ bit pattern, which is a member in the intersection between two set $K$ and $M$ using classical oracles $f_K$ and $f_M$ given in Eq. 4.1 and Eq. 4.2. These oracles mark the two sets, where the size of each set can be of order $O(N)$. Performing such task requires a classical search over all the possible $n$ bit patterns in the space of size $N$. For each pattern, the oracles are queried to verify whether the pattern belongs to both sets $K$ and $M$ that can both be of size $O(N)$. Therefore, in the classical model, the number of oracle queries has to be of order $O(N)$ in the worst case. The quantum model, on the other hand, requires only order of $O(\sqrt{N})$ queries to the oracles according to Eq. 4.19.

4.8 Implementations of the Completion and Correction Oracles

In this section, we consider possible implementations of the completion and correction oracles. We show that these oracles can be implemented by $O(n)$ and $O(n \log n)$ simple quantum gates. For a given partial or noisy pattern $x'$, the completion and corrections oracles are

$$f_{K}^{\text{com}}(x) = \begin{cases} 1, & x \text{ is a possible completion of } x' \\ 0, & \text{otherwise} \end{cases}$$

(4.36)
and

\[ f^\text{com}_K(x) = \begin{cases} 
1, & \text{dist}(x, x') \leq d \\
0, & \text{otherwise}
\end{cases} \]  \hspace{1cm} (4.37)

where \( \text{dist}(x, x') \) is the hamming distance function between \( x \) and \( x' \) (i.e. the number of different bits between \( x \) and \( x' \)).

Implementing \( f^\text{com}_K \) for \( n \) qubits requires an additional qubit initiated to \( |0\rangle \) for the regular quantum oracle and to \( |-\rangle \) for the phase version of the oracle. The oracle performs one controlled not operation on the additional qubit with \( n - d \) controls. For any partial pattern \( x' \), the controlled not operator involves the qubits corresponding to the given bits in the pattern as controls and the additional qubit as target. This is shown in Fig. 4.9, where for each given bit in the pattern an appropriate control is added to the corresponding qubit. The dark circle means that the control is activated when the bit value is 1 and the empty circle means that the control is activated when the bit value is 0. For each missing bit, no control is added for the corresponding qubit. A single \( n \) qubit operator can be implemented by \( O(n) \) 2 and 3-qubit operators [92].

\begin{figure}[h]
\centering
\includegraphics[width=0.7\textwidth]{figure4.9}
\caption{An implementation of a completion oracle with an up to \( n+1 \) dimensional controlled-not operator.}
\end{figure}

Similarly, implementing the correction oracle for a noisy pattern \( x' \) should apply a linear threshold on the hamming distance between \( x' \) and any input \( x \). One implementation of such an oracle consists of \( n \) input qubits, \( \lceil \log n \rceil \) qubits for the hamming distance of \( x \) from \( x' \), and an additional qubit for the output, as shown in Fig. 4.10. The oracle adds 1 for each bit in \( x \) that is not equal to the corresponding bit in \( x' \), then applies a controlled not operator if the
hamming distance does not exceed the threshold \( d \). The oracle can, thus, be implemented using \( O(n \log n) \) 2 and 3-qubit operators.

Figure 4.10: An implementation of a correction oracle with a \( \lceil \log n \rceil \) additional bits, \( n \) controlled operators that add 1, and a threshold operator.

### 4.9 Numerical Examples and Simulations

Let us first consider an associative memory of 10 qubits. We have randomly chosen a set of 50 patterns \( M \) out of the possible 1024 to be stored in memory. We also chose two partial patterns, each with 4 missing qubits, yielding two completion sets \( K_1 \) and \( K_2 \) of 16 possible completions each. We chose \( K_1 \) and \( K_2 \) such that they have one and two completions in memory respectively. Fig. 4.11(a) shows the memory set, where each vertical line represents a memory pattern, and Fig. 4.11(b) shows the completion set \( K_1 \) in the same manner. The amplitudes of the final state of the completion algorithm are shown in Fig. 4.11(c), where the only possible memory completion has amplitude close to 1. Fig. 4.12 shows the high amplitudes of the two possible memory completions when the completion set is \( K_2 \).

As can be seen, applying our algorithm to both completion sets amplified the states that are possible completions in memory. The amplitudes of the desired states reached up to 0.9676% in the first case and 0.6844% in the second case for each one of the two high amplitudes. Therefore, the probability of measuring the correct completion in the first case is 93.62% and the probability of measuring one of the two correct completions in the second case is 93.67%.
Another simulation was carried out on a 10 qubits associative memory with $2^7$ memory patterns and completion queries with 3 missing bits. The behavior of the different subgroups of the basis states is schematically described in Fig. 4.13 for a series of iterations with the completion operator $Q$ of Eq. 4.3. Each amplitude value indicated represents the amplitudes of all the basis states that belong to the corresponding subgroup. It shows the amplification of states in the intersection group $K \cap M$ and in $NOT(K \cup M)$ alternatingly, while the amplitudes of states in $K \setminus M$ and $M \setminus K$ stay close to zero.

We have also tested our algorithms with a larger number of qubits in order to verify that the success rate of retrieval grows asymptotically to 1 as the number of qubits grows. For instance, we tested a 30 qubit system with $2^{25}$ memory patterns and a completion query that has 3 missing bits. We tested different completions of 3 missing bits so that the intersection set size varied from 1 to 10 patterns. Our algorithm found a member of the memory completion set with probability \(96.8\%\). Increasing the memory size to $2^{26}$ and $2^{27}$, and thereby bringing the capacity close to its limit resulted in completion probabilities of \(93.5\%\) and \(86.7\%\) respectively. Fig. 4.14 depicts the success rates of pattern completion in a 30 qubit system. An explanation of the different graphs can be found in Table 4.1. The solid line in Fig. 4.14 depicts the success probability vs. the logarithm of the size of memory with completion queries set to 3 missing bits and the number of possible memory completions set to 1. The dashed line depicts the success probability vs. the logarithm of the completion query size when the memory size is set to $2^{25}$ patterns and the number of possible memory completions set to 1. The dotted line
Figure 4.12: (a) A set of memory patterns $M$ (b) a set of possible completions $K_2$ to a partial pattern, and the memory completion result in amplitudes.

Table 4.1: Properties of the four simulations depicted in Figure 4.14. The $x$ axis in Figure 4.14 represents the varying set size, while the other set sizes are constant in each simulation.

| Graph     | $|N|$   | $|M|$     | $|K|$    | $|K \cap M|$ |
|-----------|--------|----------|---------|------------|
| Solid     | constant | varying  | constant | constant   |
| Dashed    | constant | constant | varying  | constant   |
| Dotted    | constant | constant | constant | varying    |
| Dash-dotted | varying | constant | constant | constant   |

depicts the success probability vs. the logarithm of the number of possible memory completions when both the memory size and the completion query size are set to $2^{25}$. The dash-dotted line depicts the success probability vs. the number of qubits in the system (growing from 5 to 30 qubits) when the memory size, the completion query size, and the number of possible memory completions are small constants.

Fig. 4.14 shows that the deterioration of the success probability vs. the memory size or the completion query size is very slow. For instance, deterioration starts at memory size $2^{26}$. Furthermore, the success probability increases when the number of possible memory completions (the size of the intersection set) grows towards the sizes of the completion query and the memory, which indicates that choosing a member of the intersection becomes easy (by randomly...
choosing a possible completion). Finally, the figure also shows that, as the number of qubits in the system grows, the success probability becomes asymptotically 1, which indicates that, practically, our algorithm produces the intersection when \( n >> 1 \).

### 4.10 Conclusions

In this chapter we have presented a quantum computational algorithm that computes the intersection between two subsets of \( n \)-bit strings. The algorithm is based on a modification of Grover’s quantum search and can mimic the behavior of a network whose dynamics at each step amplify either the memory patterns or the completion / correction of the input pattern. Using the intersection algorithm, we have presented a set of algorithms that implement a model of associative memory via quantum computation. We introduced the notion of memory as a quantum operator, thus avoiding the dependence of the initial state of the system on the memory set. We have shown that our algorithms have capacity advantage with respect to classical associative memory models. The model is able to store a number of memory patterns which is exponential in the number of bits and retrieve them in sub-exponential time. Pattern retrieval algorithms with completion and correction abilities were presented. Bounds relating memory capacity to the maximal allowed signal to noise ratio were found.
Figure 4.14: Success probability of measuring a desired memory completion vs. the $\log$ of the memory size (solid), completion query size (dashed), possible memory completions (dotted), and number of qubits (dash-dotted).
Chapter 5

Hebbian Learning in Quantum Neurons

In this chapter we define two models of a quantum neuron, then develop a training process in which the quantum neuron embodies Hebbian learning based on a set of classical training examples. The training phase of the neuron determines the incorporation of appropriate quantum gates into the neuron’s circuit based on the Hebbian interaction between each input bit and the output on the training set.

Our model of the quantum neuron is trained classically by determining the quantum gates involved in the neuron’s circuit. In the testing phase, the input determines how the quantum circuit is applied on a constant initial input to the circuit, which is initialized as a superposition of all basis states. We show that the quantum neuron embeds the Hebbian relation between its inputs and output and accordingly fires probabilistically.

Based on the quantum neuron model with the Hebbian properties, we propose a quantum neural network for Hebbian learning. We show that the network implements associative memory and demonstrate the ability of the associative memory to retrieve and correct stored memory patterns. We also present an example of a probabilistic classification task with the proposed neuron that realizes the Hebbian relations acquired during the training phase as a quantum superposition.

In the following sections we present the basic model of the neuron followed by a set of quantum gates that are used by the quantum neuron as building blocks and realize both inhibitory and excitatory interactions between an input bit and the output bit. Then we present the model of a quantum neuron that realizes a quantum neural network and show that it implements a probabilistic model of associative memory as well as classification.
5.1 A Quantum Neuron

The classical neuron shown in Fig. 2.1 receives $n$ input values and outputs a single value according to a transfer function (cf. Eq. 2.1) acting on the weighted sum of its inputs. In a quantum mechanical system, we would like the neuron to output a quantum state that depends on the inputs. The transfer function is, therefore, a quantum operator. Let the quantum neuron be some quantum operation applied to a qubit. We define two models of a quantum neuron. The first is a quantum operator with $n+1$ inputs and $n+1$ outputs, as described schematically in Fig. 5.1. The neuron gets the $n$ dimensional input qubits and an additional qubit in some initial state and outputs the $n+1$ qubits with the last qubit changed according to some transfer function. A layer of $m$ neurons of this model in a network is described by in Fig. 5.2.

![Figure 5.1: A schematic description of the first model of a quantum neuron.](image)

The second model is described schematically by Fig. 5.3. The $n$ dimensional input is classical. The inputs determine the quantum operator applied by the neuron (i.e. the transfer function) on its initial state. The initial state corresponds to the prior knowledge about the Hebbian relation between the inputs and the output and the final state corresponds to the acquired knowledge from the training set and the test input.

The second definition is quite different from other definitions suggested in the literature [21, 22, 23, 24, 20], where the quantum neuron receives an $n$-qubits input and applies an operator on them. Our neuron has only one qubit initial state, and the $n$ dimensional input is used to construct the quantum operator that the neuron performs. This model learns classically.

Both models can be trained in various ways and the transfer function $U$ may exploit quantum properties that give a superposition of the possible classical outputs. For example, the outcome of training the first model to solve the XOR problem can consist of 2 controlled-not gates and a qubit initialized to the state $|0\rangle$, as shown in Fig. 5.4. Hence, the remaining question is how
Figure 5.2: A model of a layer of a quantum neural network with $n$ neurons of the first kind.

to determine the neuron’s transfer function, or how to train the neuron. In the rest of the chapter we consider the second model and devise a training process for it. In the following sections, we apply the Hebbian learning method to train the quantum neuron. We first present the basic Hebbian building blocks for the construction of the operator $U$.

## 5.2 Hebbian Quantum Gates

Let $|x_i\rangle$ be an input qubit holding one of two values $|0\rangle$ or $|1\rangle$ and $|y\rangle$ an output qubit initialized with $H|0\rangle \equiv |+\rangle$. An excitatory interaction between $|x_i\rangle$ and $|y\rangle$ should perform one of the following two rotations on the second qubit:

\[
|x_i = 0\rangle |y = +\rangle \longrightarrow |x_i = 0\rangle |y = 0\rangle \tag{5.1}
\]

and

\[
|x_i = 1\rangle |y = +\rangle \longrightarrow |x_i = 1\rangle |y = 1\rangle \tag{5.2}
\]

while an inhibitory interaction should perform one of the following two rotations

\[
|x_i = 0\rangle |y = +\rangle \longrightarrow |x_i = 0\rangle |y = 1\rangle \tag{5.3}
\]
These interactions can be implemented by two controlled quantum gates each. We denote the excitatory operator of the first kind $U_0^E$, and the excitatory operator of the second kind $U_1^E$, where the subscript, either 0 or 1, determines whether the use of the gate is when the control bit is 0 or 1 respectively and the superscript $E$ stands for excitatory interaction. Similarly, the inhibitory operators are $U_0^I$ and $U_1^I$, where $I$ stands for inhibitory interaction. Consider the parameterized rotation operator

$$R(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

The set of basic quantum gates can be formulated as:
The Hebbian input-output relation can then be realized by two 2-qubits controlled quantum gates, where the input qubit is the control qubit and the output is the target qubit as depicted in Fig. 5.5, where the dark circle applies the gate on the target bit when the control bit is $|1\rangle$ and the light circle applies the gate when the control bit is $|0\rangle$. The matrix representation of the circuit for the excitatory interaction can be formulated as follows:

$$
(U^E_0 \bar{0}, U^E_1) = 
\begin{pmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
0 & 0 \\
0 & 0
\end{pmatrix}
$$

(5.8)

and similarly, for the inhibitory interaction as

$$
(U^I_0 \bar{0}, U^I_1) = 
\begin{pmatrix}
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
0 & 0 \\
0 & 0
\end{pmatrix}
$$

(5.9)

Figure 5.5: Hebbian relation through two 2-qubit quantum gates.

Note that when applying any gate more than once the excitation and inhibition relations do not hold. Therefore, in the case of $n$ input qubits interacting with the output qubit, the excitation or inhibition applied by each input qubit on the output must take $n$ into account so that the total rotation of the output qubit reaches the desired state based on the appropriate Hebbian relation. In the following sections we consider quantum neurons for which the output
bit reaches the desired state with certainty only when all inputs have maximal agreement with that output on the training set.

Furthermore, since the qubit $|x_i\rangle$ is known to be classical in our model, we prefer, for convenience, not to use the controlled versions of the quantum gates in the next sections. Instead, we consider the $n$ dimensional training points as classical inputs that determine which quantum gates are chosen by the neuron. Thus, the neuron applies some $k$ rotation gates $R(\theta)$

$$|+\rangle \rightarrow R(\theta)^k |+\rangle$$

where $k$ and $\theta$ are determined classically.

5.3 A Quantum Hebbian Neuron

We define a quantum Hebbian neuron based on the quantum Hebbian gates presented in Sec. 5.2. The neuron has 1 input qubit and one output qubit and can be trained on a training set $\{(x^i, y^i)\}_{i=1}^l$, where $x^i \in \{\pm 1\}^n$ and $y^i \in \{\pm 1\}$. The classical value +1 is represented by the qubit $|1\rangle$ and the classical value −1 is represented by $|0\rangle$. The neuron’s training phase determines the quantum rotation gate $R(\theta)$, while the test pattern determines the number of times $k$ it is applied. Consider the classical Hebbian neuron that is given by the Hebbian weight vector

$$W = Y \ast X^T$$

(5.10)

where $X$ is a matrix whose columns are the training inputs defined over $\{\pm 1\}^n$ and $Y$ is the row vector of corresponding training outputs defined over $\{\pm 1\}$. The classical neuron classifies a new point $x$ according to the result of

$$\text{sgn} \left( W^T x \right)$$

where $\text{sgn}(\cdot)$ is the signum function. The sign of the $i$th element in the weight vector represents either excitatory or inhibitory interaction between the $i$th input bit $x_i$ and the output bit $y$.

One can achieve the results of the classical Hebbian neuron by applying the operator

$$R \left( \frac{\pi}{4} \right)^{\text{sgn}(W^T x)}$$

(5.11)

on $H |0\rangle = |+\rangle$. However, this does not present any advancement over the classical Hebbian solution. Both cases, the quantum as well as the classical, apply a threshold on the weighted sum of inputs. On the other hand, utilizing quantum superposition, we can express the level of agreement or disagreement between an input bit superposition and the output to produce a quantum
Table 5.1: A 3-dimensional training example set.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
</tbody>
</table>

superposition which, upon measurement, yields probabilistic outputs. This is shown in Fig. 5.6, where the training set determines the angle of the rotation and the test point determines its sign and the number of times it is applied.

Consider for example the 3-dimensional input and 1-dimensional output training example set given in Table. 5.1. The training set in Table. 5.1 yields the weights vector $W = [2, 2, -4]$.

The Hebbian relation between each input bit $x_i$ and the output bit $y$ is measured by the level of agreement between the two bits on the training set. A positive value in the weights vector represents an excitatory interaction, while a negative value represents an inhibitory interaction. The absolute values of the Hebbian weights vector represent the amount of confidence in the Hebbian relation between input and output, whether it is excitatory on inhibitory. Our proposal consists of a quantum operator that outputs a quantum state, which, upon measurement, yields any of the possible outputs with probability representing the confidence in the output according
to the Hebbian relations acquired from the training set. Define the basic rotation angle

$$\theta = \frac{\pi}{4} \frac{1}{\sum_{i=1}^{n} |W_i|}$$

(5.12)

which for the case of the given example equals $\frac{\pi}{32}$. This rotation angle represents the basic rotational step corresponding to a unit of excitation $(+\theta)$ or inhibition $(-\theta)$ between an input and the output. A rotation by this angle shall be used as the basic building block of the quantum neuron.

The output of the quantum neuron for an input vector $x$ is defined by applying the rotation

$$\prod_{i=1}^{n} R(\theta)^{W_i x_i} = R(\theta)^{\sum_{i=1}^{n} W_i x_i}$$

(5.13)

to the output qubit that is initialized with $H |0\rangle = |+\rangle$, where $W_i$ is the $i$th element of $W$, $x_i$ is the $i$th element of the input vector $x$, and $\prod$ stands for the product of the rotations induced by the different input elements. To better understand the suggested neuron, we substitute Eq. 5.12 into Eq. 5.13 and eliminate the dependance on the Hebbian weight vector $W$ from the rotation operator as follows:

$$R(\theta)^{\sum_{i=1}^{n} W_i x_i} = R\left(\frac{\pi}{4} \frac{\sum_{i=1}^{n} W_i x_i}{\sum_{i=1}^{n} |W_i|}\right)$$

(5.14)

Eq. 5.14 shows that the neuron applies the rotation $R(\pm \frac{\pi}{4})$ in cases where all the inputs agree on the desired output for all training points leading the neuron to either fire with certainty or not to fire with certainty. For other cases, a partial rotation that embeds the level of agreement is applied, which in turn leads the neuron to a quantum superposition of firing and not firing. Measuring the output of the neuron, then, results in a model of probabilistic firing. Fig. 5.7 shows a comparison between the classical Hebbian neuron and the quantum Hebbian neuron for the given example. The dashed line shows the output of the classical neuron for all inputs after being trained on the training set given in Table. 5.1. The solid line, on the other hand, depicts the difference between the confidence levels in each classical answer. The diamond marks represent the probability that the quantum neuron will fire upon measurement. When the Hebbian confidence is high, the probability of firing is close to 1 and when the Hebbian confidence is low, the probability of firing is close to 0. For instance, the input $[+1, +1, -1]$ represented by the decimal number 6 yields maximal confidence in excitation and the probability that the quantum neuron will fire is therefore 1, while the input $[-1, -1, +1]$ represented by the decimal number 1 yields maximal confidence in the inhibition and, thus, the probability of firing is 0. For other inputs, such as $[-1, +1, -1]$ and $[-1, +1, +1]$ with confidence levels...
Figure 5.7: The output of the classical Hebbian network (dashed line) vs. the network inputs and the normalized Hebbian weighted sum (solid line) vs. the network inputs. The diamond shaped points represent the probability of the quantum neuron to fire for each input.

4 and −4 respectively, the probability of firing is \( \sim 85.4\% \) and \( \sim 14.6\% \) respectively. The classification results for the example of Table 5.1 are given in Table 5.2.

The neuron was constructed in a way that ensures the correct answer after measurement of the output state if all input bits have maximal agreement on the desired output. In our example this happens only for the inputs \([-1, -1, +1]\) and \([+1, +1, -1]\). On the other hand, the classical neuron classifies all possible inputs without accounting for the disagreement between the output value and the different input bits as well as the different reactions for different points in the training set.

### 5.4 A Quantum Hebbian Network

In this section we consider the \(m\)-qubits Hebbian network, as shown in Fig. 5.8, comprising \(m\) quantum neurons from Sec. 5.3. The network is trained on \(l\) training patterns \(\{(x^i, y^i)\}_{i=1}^l\), where \(x^i \in \{\pm 1\}^n\) and \(y^i \in \{\pm 1\}^m\) for \(i = 1, \ldots, l\). The network can implement an associative memory that stores the \(l\) patterns in the network by computing the Hebbian weight matrix

\[
W = \sum_{i=1}^l y^i x^i T
\]  

and retrieving \(y^i\) from an input \(x^i\). 

61
Table 5.2: The results of the quantum Hebbian classification trained with the training set in Table 5.1.

| $x_1$ | $x_2$ | $x_3$ | $|y\rangle$ | $\langle y|0\rangle$ | $\langle y|1\rangle$ |
|-------|-------|-------|-------------|----------------|----------------|
| -1    | -1    | -1    | $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ | 50%            | 50%            |
| -1    | -1    | +1    | $|0\rangle$ | 100%           | 0%             |
| -1    | +1    | -1    | $\cos\left(\frac{3\pi}{8}\right)|0\rangle + \sin\left(\frac{3\pi}{8}\right)|1\rangle$ | 14.6%          | 85.4%          |
| -1    | +1    | +1    | $\cos\left(\frac{3\pi}{8}\right)|0\rangle + \sin\left(\frac{3\pi}{8}\right)|1\rangle$ | 85.4%          | 14.6%          |
| +1    | -1    | -1    | $\cos\left(\frac{3\pi}{8}\right)|0\rangle + \sin\left(\frac{3\pi}{8}\right)|1\rangle$ | 14.6%          | 85.4%          |
| +1    | +1    | -1    | $|1\rangle$ | 0%             | 100%           |
| +1    | +1    | +1    | $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ | 50%            | 50%            |

The network is applied on a register of $m$ qubits, representing the $m$-dimensional output initialized to $\bigotimes_{j=1}^{m}|+\rangle$. After training, the network performs the following operation when presented with an input $x$

$$\bigotimes_{j=1}^{m}|+\rangle \rightarrow |y\rangle$$

where $|y\rangle$ is the quantum assigned output to the test point $x$.

The $j$th neuron in the network applies the rotation $R(\theta_j)^{(W_{j,.}x)}$ on the input $x$, where $W_{j,.}$ is the $j$th row in the Hebbian weight matrix, $W_{j,i}$ is the $i$th column in that row, and

$$\theta_j = \frac{\pi}{4\sum_{i=1}^{n}|W_{j,i}|}$$

is the basic rotation step of the $j$th neuron. The rotation applied by the network is therefore

$$\bigotimes_{j=1}^{m} R(\theta_j)^{(W_{j,.}x)} = \bigotimes_{j=1}^{m} R(\theta_j)^{\sum_{i=1}^{n} W_{j,i}x_i}$$

Consider, for example, the set of training pairs

$$x^1 = [+1, -1, -1, +1], y^1 = [+1, -1, +1]$$
$$x^2 = [+1, -1, +1, -1], y^2 = [-1, -1, -1]$$
$$x^3 = [-1, +1, +1, +1], y^3 = [+1, +1, -1]$$

that yield the Hebbian weight matrix

$$\begin{pmatrix}
-1 & +1 & -1 & +3 \\
-3 & +3 & +1 & +1 \\
+1 & -1 & -3 & +1 
\end{pmatrix}$$

and the corresponding rotation angles

$$\theta_1 = \frac{\pi}{24}, \theta_2 = \frac{\pi}{32}, \theta_3 = \frac{\pi}{24}$$
Figure 5.8: A model of a quantum neural network with \( m \) outputs trained on an \( n \)-dimensional training set.

Introducing one of the stored patterns \( x^1, x^2, \) or \( x^3 \) to the network yields one of the results

\[
|y^{1'}\rangle = R(\theta_1)^2 \otimes R(-\theta_2)^6 \otimes R(\theta_3)^6 |++\rangle \\
\approx 0.49039 |001\rangle + 0.097545 |011\rangle + 0.84938 |101\rangle + 0.16895 |111\rangle \\
|y^{2'}\rangle = R(-\theta_1)^6 \otimes R(-\theta_2)^6 \otimes R(-\theta_3)^2 |++\rangle \\
\approx 0.84938 |000\rangle + 0.49039 |001\rangle + 0.16895 |010\rangle + 0.097545 |011\rangle \\
|y^{3'}\rangle = R(\theta_1)^4 \otimes R(\theta_2)^8 \otimes R(-\theta_3)^4 |++\rangle \\
\approx 0.25 |010\rangle + 0.066987 |011\rangle + 0.93301 |110\rangle + 0.25 |111\rangle \\
\]

respectively. The probabilities of measuring the correct outputs are therefore

\[
\langle y^1 | y^{1'} \rangle \approx 72.15\% \\
\langle y^2 | y^{2'} \rangle \approx 72.15\% \\
\langle y^3 | y^{3'} \rangle \approx 87.05\% \\
\]

(5.21)
Furthermore, when presented with the negation of a stored pattern $x = -x^i$, the network outputs $y^i$ with all its qubits rotated by an angle of $\frac{\pi}{2}$. As a result, the probability of measuring the negation of $y^i$ is the same as the probability of measuring $y^i$ if the network had been presented with $x^i$. For instance, presenting the network with the input $[-1, +1, -1, +1]$, which is the negation of $x^2$ yields the output $[+1, +1, +1]$, which is the negation of $y^2$ with probability $\sim 72.15\%$.

The proposed network also demonstrates some pattern correction abilities through the Hebbian interactions between inputs and outputs. This can be seen, for instance, when introducing $x = [+1, +1, +1, +1]$, which is a noisy version of $x^3$. This yields the output

$$|y'\rangle = R(\theta_1)^2 \otimes R(\theta_2)^2 \otimes R(-\theta_3)^2 |+++angle$$

and the probabilities of measuring the different basis states are shown in Fig. 5.9. The highest probability is achieved for measuring the basis state $|110\rangle$, which corresponds to the output of $x^3$.

![Figure 5.9: The probability of measuring a computational basis state at the output of the network trained on the training set in Eq. 5.18 vs. the decimal representation of the states when the network is presented with $[+1, +1, +1, +1]$.
](image)

The presented model embeds the Hebbian relations between the different neurons in the amplitude of the superposition of the computational basis states. Another realization can embed the Hebbian relations in the probability of measuring each computational basis state.
This can be done by choosing the basic rotation angle to be

\[ \theta_j = \frac{\pi}{4} \frac{1}{\sqrt{\sum_{i=1}^{n} |W_{j,i}|}} \]

and the rotation applied by each neuron for a given input vector \( x \) is

\[ R(\theta_j)[\sqrt{\langle W_j, x \rangle}] \]

Using the realization of the Hebbian relations as suggested in Eq. 5.24 and Eq. 5.25 for the retrieval of the stored patterns gives higher success probabilities of measuring the correct outputs as follows:

\[
\langle y^1 | y^{1'} \rangle \approx 74.19\% \\
\langle y^2 | y^{2'} \rangle \approx 74.19\% \\
\langle y^3 | y^{3'} \rangle \approx 95.70\%
\]  

(5.26)

In addition, to cope with the rounding error in Eq. 5.25, instead of applying the basic rotation angle \( \left[ \sqrt{\langle W_j, x \rangle} \right] \) times, one might consider a network of \( m \) neurons, in which each neuron performs only one rotation with respect to the presented test point by computing the overall rotation angle

\[ \theta'_j = \theta_j \sqrt{\langle W_j, x \rangle} \]

(5.27)

As a result, the network probabilities of successful retrieval of the stored patterns become

\[
\langle y^1 | y^{1'} \rangle \approx 88.39\% \\
\langle y^2 | y^{2'} \rangle \approx 88.39\% \\
\langle y^3 | y^{3'} \rangle \approx 95.91\% 
\]

(5.28)

and the error correction probability achieved for the input \( x = [+1, +1, +1, +1] \) is also improved as shown in Fig. 5.10.

We have shown that the presented algorithm demonstrates pattern retrieval and correction abilities.

The performance of the algorithm should be further studied to determine bounds on the success probability of pattern retrieval and correction and to further improve its performance.
Figure 5.10: The probability of measuring a computational basis state at the output of the network trained on the set in Eq. 5.18 vs. the decimal representation of the states when the network is presented with $[+1, +1, +1, +1]$ and when the square root of the Hebbian relations are embedded in the superposition.

### 5.5 Probabilistic Classification using Quantum Neurons

We now consider binary classification of $n$-dimensional strings with Hebbian quantum neurons. Suppose that we are given a training set $\{(x^i, y^i)\}_{i=1}^I$, where $x^i \in \{\pm 1\}^n$ and $y^i$ is one of two classes represented by the states $|0\rangle$ and $|1\rangle$.

A single quantum neuron trained to capture the Hebbian relations between the inputs and the output on the training set defines a probabilistic classifier. The classification result is a superposition of the two classes with amplitudes that express the amount of belief in each class.

Consider the following simple 3-dimensional training set

$$
x^1 = [-1, -1, -1]^T \quad y^1 = -1
$$

$$
x^2 = [-1, -1, +1]^T \quad y^1 = +1
$$

$$
x^3 = [-1, +1, -1]^T \quad y^1 = +1
$$

$$
x^4 = [-1, +1, +1]^T \quad y^1 = +1
$$

$$
x^5 = [+1, -1, +1]^T \quad y^1 = +1
$$

$$
x^6 = [+1, +1, -1]^T \quad y^1 = -1
$$

and let the negative class be represented by $|0\rangle$ and the positive by $|1\rangle$. Fig. 5.11 depicts the
training points and their class labels. Now consider the quantum neuron that should classify the test points

\[
x_{1\text{tst}} = [+1, -1, -1]^T
\]

\[
x_{2\text{tst}} = [+1, +1, +1]^T
\]

(5.35) (5.36)

Figure 5.11: A 3 dimensional training set with binary classification.

The quantum neuron applies the rotations \( R\left(-\frac{\pi}{24}\right)^6 \) and \( R\left(\frac{\pi}{24}\right)^2 \) on the initial state \(|+\rangle\) for the test points \( x_{1\text{tst}} \) and \( x_{2\text{tst}} \) respectively. This yields the outputs \(|0\rangle\) and \( \frac{1}{2} |0\rangle + \frac{\sqrt{3}}{2} |1\rangle \) respectively, which assigns the first test point to the negative class with certainty and the second test point to the positive class with probability 75%.

### 5.6 Conclusions

In this chapter we defined two models of quantum neurons. The first has \( n + 1 \) qubits while the second has a single qubit. We devised a set of quantum operators that realize Hebbian relations between inputs and output. We developed a Hebbian algorithm for the training of the second model. We showed that a network of such neurons can realize probabilistic learning.
Chapter 6

Conclusions

We addressed the problem of finding quantum computational analogues to models of artificial neural networks. We presented several kinds of models that could each be an appropriate quantum analogue for an artificial neural network under some assumptions. We showed that each of these models is appropriate for consideration for a set of specific tasks. The variety of models and their corresponding learning algorithms manifest several improvements over their classical counterparts. The improvements can be embodied in the time complexity, the storage capacity, or the probabilistic nature of the models.

We presented a model of associative memory in which the interactions between neurons is based on Grover’s operator and a set of algorithms that implement pattern completion and correction. We introduced the concept of memory as a quantum operator, thus avoiding the dependence of the initial state of the system on the memory set. We showed that the model can be used to control the basin of attraction of the network. We proved that the model improves the capacity of the associative memory without additional bits that are used in sparse encoding techniques. Furthermore, we showed that the probability of spurious memories vanishes as the number of bits grows.

We presented two models of a quantum neuron. The quantum neuron was shown to exploit quantum superposition for a probabilistic model of firing. A Hebbian classical training algorithm was devised for the second model of a quantum neuron to embody the Hebbian relation between its inputs and output. The quantum neuron is constructed from a small number of one-qubit quantum gates. The neurons were trained classically to determine the quantum gates that they apply as well as their number. A neural network of such neurons was shown to demonstrate probabilistic learning capabilities. We showed that such neural networks demonstrate error correction capabilities and probabilistic classification. In the future, bounds on the learning error should be determined as well as further applications for the Hebbian quantum neuron.
We believe that this work provided evidence that quantum computation can contribute to the improvement of the artificial neural networks. We expect that, in the future, a wider variety of quantum neural network models will be found and that they will present a substantial advancement for the field of artificial neural networks.
Bibliography


74


חישוב עברי קוונטי

בשני העשור الأوוריים החל בברית מוס.HttpStatus פרו Variation על בגרות נורוגיה קוונטי, ואית השיעור לגב יכלו השוורות של נורוגיה, עניין לא עוד מודל אניברצל לשני קוגוורי עדין או הוכחת הקדושות של מודלים אלה. בביהודה או מתמטים מוס keyboardType קוונטי של שLabour זה נאנת מודלים נוספים שמשתתפות נוירוניות קרויו, בסדר זה, השוורות העדיבות והאנלורימה של היותן של מודלים שונים קוגונטיים בעברם של אנרוקר דמי

לא זמנים של מודלים קוונטיים של משות תעשיות. אף מבריאת פעוט המופות והיא אסцитאטיים המבוסס על אנטריפור גרובר, על מוכחיים שמהודל מסוגל להתחמץ זרבות שמספר או אספרי קרבונטיים (הנוירוגיה), בנסף בצדי השלחן זרבות ותיקה זרבות שגויים, בין מוכחיים שמהודל זה בעלא אנון מפגשות הרבד והסתובבות ל马来 זרבות מניירפים בזורופ 진행 פלפס.

בנסף או של מגרים לדיום של מודלים של נורוגיה קוונטיים, או ב Drawable מתנות נוספים שזמות לקוררטז של מודלים המבטים סידור את הנקראים של הקבוצות של נורוגיה אלทะเลות נורוגיהقودל של זכר או אסцитאטיים וסיוון הנסטורעדו, או הנקראים הקבוצות של סופרפוריצי, של השולות הקבוצות של הנקראים והאנרוקר דמי קבוצת האימונים.
The quantum computer

The quantum computer is a versatile and powerful tool in quantum mechanics, enabling the simulation and computation of quantum systems. It involves concepts from quantum mechanics and quantum computing. For example, the example of Deutsch's problem, which was solved using quantum computers in a highly efficient manner, was first introduced in 1985. Another significant example is Shor's algorithm, which demonstrates the speed-up of quantum computing over classical computing for certain problems, such as factoring large numbers. Grover's algorithm, introduced in 1996, provides a quadratic speed-up for searching unsorted databases, which is another important application of quantum computing.

In quantum mechanics, the state of a quantum system is described by a wave function, which is a complex-valued function of position and momentum. The wave function is normalized, meaning its squared absolute value represents the probability density of finding the particle at a given point. The evolution of the quantum state is governed by the Schrödinger equation, which is a fundamental equation of quantum mechanics.

The quantum state of a system is represented by a vector in a Hilbert space, and the evolution of the state is described by a unitary operator. The measurement of a quantum system collapses its state, and the outcome of a measurement is determined by the probability amplitude of the state.

The key difference between quantum and classical computing lies in the ability of quantum systems to perform certain calculations exponentially faster than their classical counterparts. This is due to the principles of superposition and entanglement, which allow quantum computers to explore multiple solutions simultaneously.
希シェב עטור קורנייetto טחטזה מחקר חהז יתישית הימ valueForKey את המחקר בﴩוחת עצביות
מלאכתיתות את תורח החושוב הקורניי. בשלי העשויים האזורית, רבים חקירים בריכ
בתוחח הרשתות העצביות את אמונת ששים שחשויבים קורנ 의해 תעלה בתכונת
ﴩוחת אללה לחבות בפשור ביצועיםוקו.

תרשוח עצביות מלאכתיתות

תרשוח עצביות מלאכתיתות או מערוך חישוב דינמי, המשלבת ייחודה חישוב פשטו
יתştır הנקראות גורניי, בכישוריות מוקבילות. התוור המ𪾢מותי הוא נعظيم למדל את א
העצב הביוולוגי יכי לסיית בהاستعراض התהליכים המنهارטיים במעום, או התפרח_bmorderid
מתימטי מופשי.

תרשוח עצביות תנייה מערוכות המרכבות מספר של גורניי ומאמנויות למעCompanyId
אופטימיזציה שווה. תרשוח עצביות ידוועה למצבונים ממשלים ל דיגו תbove, שער
פונקציית הירפוע אוניסטים אטומיץ. במשך השמיש פורחה אגונית יתיוידמה למידה שום בﴩוח
עצביות, האמפשיהם את ימאו hustל תמלולת מוס(detail)

הלמוד ההביאני (Hebbian) של הגרט מתיימשם של חק הב, הקובע שאן גורניי
שפיעלתו דומה במשך גוז חלקים לקורא להשקט הס ธันוטים ביניהם. עקרון זה היה שיומש ביהור
בתיית הרשתות העצביות המלאכתיתות, ובשנת 1982 הפג שמיי (Hopfield) רשת עצביות
דיגים ראש למדת את המלולת לפיח המשמשת הפרותר ביוית אופטימיותו הזכר
אפורטסאציו. הרשת היצנתיינת תינוק משואות, אך קיומל התאשוש והשהור של
(Spurious Memories) נдержан תורח התנע ליעוף יינמל התאשוש של רשת הפשיו
המידה בוי תוי ומוכת, כמיה, הוא סובל ממסרה ביש גורונת מודיימי
(Sparse Encoding) (Kanerva), שיכל לאחסן מספר
אקרסנגיאלי של גורונת, אך אלה השתמשו מספר גורונת שויוא אקרסנגיאלי בים
הכינים.
המחקר נועד בהנחיית פרופסור יורם ברעם בכولادה למדעי המחשב.

אני מצה לiodeת ותי, מושידו המдеся ולתכנון על התמיכת הכיסוף הנדרשת בשתיוותי.
חישוב עצבי קוונטי
והכרון אסוציאטיבי

היבר על מחקר

לשם מייליק חלקי של הדרישות לקבלת התואר
דוקטור לפילוסופיה

תאמר סלמא

הוות לשטט הטכנולוגי – מרכז טכנולוגי לישראל
תמונה תשע"ב
וולי 2012
חישוב עצבי קוונטי
הכרון אסוציאטיבי

תامر סלמאן